



UNIVERSITAT  
POLITÈCNICA  
DE VALÈNCIA

*Interactivity, Adaptation and  
Multimodality in Neural  
Sequence-to-sequence Learning*

PHD DISSERTATION

Álvaro Peris

Supervised by Prof. Francisco Casacuberta

December, 2019



# Copyright Note

I hereby declare that this document describes my original work, developed under the supervision of Prof. Francisco Casacuberta.

**Section 5.4** describes an user evaluation of an adaptive translation system. This was done in collaboration with Miguel Domingo and the company Pangeanic, with funding from the Spanish Center for Technological and Industrial Development (Centro para el Desarrollo Tecnológico Industrial). This company has experience in providing high-quality translations and they have on staff professional translators. Miguel Domingo and the team from Pangeanic developed a plugin for communicating the adaptive systems under study with their production platform. My part consisted in advising and supervising the experimental conditions, giving support for the development of the adaptive systems, processing and analyzing the results obtained.

Most of **Chapter 6** is the result of a collaboration with Marc Bolaños, supervised by Prof. Petia Radeva, from Universitat de Barcelona/CVC. This collaboration was supported by the R-MIPRCV network, under grant TIN2014-54728-REDC. They are experts in computer vision, while we had experience in text generation. In the early stages of this collaboration, the different roles were clear: they were in charge of extracting features from the visual objects while we were focused on generating text from these features. However, as the collaboration advanced, these frontiers became more diffuse and both Marc Bolaños and I acquired experience in both fields. We tightly cooperated in the design and implementation the captioning systems, as well as in writing of research papers derived from this work.





# Agradecimientos

Estando al final del camino que es una tesis doctoral, echo la vista atrás y considero que lo he disfrutado mucho. El que esto sea así, se debe fundamentalmente a las personas con las que he tenido la suerte de estar rodeado durante estos años y a las que tengo que agradecer su participación, en un grado o en otro, en este proceso.

Como no podía ser de otra manera, comienzo por la persona que más influencia ha tenido en ello: mi director, Paco. Desde que en quinto de carrera me transmitió su fascinación por el mundo de las redes neuronales, ha sido un guía excepcional. Además de su sapiencia en lo científico, siempre ha depositado en mí una confianza que agradeceré infinitamente, dándome la oportunidad y la libertad de investigar lo que yo considerara; permitiendo que me involucrara en otros temas más allá de la investigación pura y haciendo por mí todo lo que estaba en su mano.

También han tenido un gran impacto en esta tesis Marc y Petia. Considero que ha sido una colaboración que me ha enriquecido enormemente y el (breve) tiempo que estuve con ellos en Barcelona me sentí como uno más. El hecho de trabajar con Petia, además de enseñarme mucho, me ha resultado muy divertido e interesante. ¿Y qué voy a decir de Marc? A golpe de burrería con Theano y de horas de cabezazos por Skype, considero que aprendimos mucho juntos.

Trabajar con los compañeros del PRHLT ha sido muy enriquecedor. Además de sus conocimientos y conversaciones interesantes, siempre ha habido risas. Recordaré con cariño las comidas pantagruélicas de Económicas. Tengo que agradecer a Dani Ortiz su tutela en los primeros momentos, cuando yo aún estaba haciendo el PFC. Su introducción en los detalles de la traducción automática y sus consejos sobre cómo

---

escribir documentos científicos fueron una gran ayuda. También quiero agradecer especialmente a los compis del grupo de traducción, en especial a Mara y Miguel.

Fuera del ámbito académico, tengo la suerte de tener que agradecer a mucha gente, por lo que voy a agruparlos en conjuntos no disjuntos y a ser conciso: a los colegas, a la peña del basket, a la de los viajes, a la de la música, al resto de triple P, a los innobles... En definitiva, a todos aquellos que me han hecho disfrutar estos años.

A Marina, quien me hace muy feliz y que tiene una paciencia infinita en lo referente a mis historias.

Y cómo no, tengo que dar las gracias a mi familia: titos (¡y Tata!), primicas y Eduardo. En estos momentos, me acuerdo mucho de mis dos abuelos, que no han visto este proceso, pero que han contribuido al mismo más de lo que pudiera parecer. Y, por supuesto, a mis dos abuelas, quienes, aunque no sepan muy bien lo que hago, siempre me han apoyado.

Dejo para el final a las dos personas que más han contribuido en mi felicidad personal y en que haya podido dedicarme a lo que me gusta. Han sido un apoyo incondicional y el que ahora pueda estar escribiendo estas líneas se lo debo en gran parte a ellos. Obviamente, me refiero a mis padres. Sin ellos, esto habría sido imposible.

¡Muchas gracias a todos!

Valencia, finales de septiembre de 2019.

# Abstract

The sequence-to-sequence problem consists of transforming an input sequence into an output sequence. A variety of problems can be posed in these terms, including machine translation, speech recognition or multimedia captioning. In the last years, the application of deep neural networks has revolutionized these fields, achieving impressive advances. Despite these improvements, the output of the automatic systems is still far from perfect. For achieving high-quality predictions, fully-automatic systems require to be supervised by a human agent, who corrects the errors. This is a common procedure in the translation industry. This thesis is mainly framed into the machine translation problem, tackled using fully neural systems. Our main objective is to develop more efficient neural machine translation systems, that allow for a more productive usage and deployment of the technology. To this end, we base our contributions on two main cornerstones: how to better use the system, and how to better leverage the data generated as it is used.

In the first case, we apply the so-called interactive-predictive framework to neural machine translation. This embeds the human agent and the system into a cooperative correction process, that seeks to reduce the human effort spent for obtaining high-quality translations. We develop different interactive protocols for the neural machine translation technology, namely, a prefix-based and a segment-based protocol. They are implemented by modifying the search space of the model. Moreover, we introduce mechanisms for achieving a fine-grained interaction while maintaining the decoding speed of the system. We carried out a wide range of experiments that show the potential of our contributions. The previous state of the art is improved by a large margin and the current systems are able to react better to the human interactions.

---

Next, we study how to improve a neural system using the data generated as a byproduct of this correction process. To this end, we rely on two main learning paradigms: online and active learning. Under the first one, the system is updated on the fly, as soon as a sentence is corrected. Hence, the system is continuously learning from the corrections, avoiding previous errors and specializing towards a given user or domain. A large experimentation tested the adaptive systems under different conditions and domains, demonstrating the capabilities of adaptive systems. Moreover, we also carried out a human evaluation of the system, involving professional users. They were very pleased with the adaptive system, and worked more efficiently using it. The second paradigm, active learning, is devised for the translation of huge amounts of data, which makes the correction of all of them prohibitively expensive. In this scenario, the system selects samples that should be supervised, and leaves the rest automatically translated. Applying this framework, we obtained reductions of approximately a quarter of the effort required for reaching a desired translation quality. The neural approach also obtained large improvements compared with previous translation technologies.

Finally, we address another challenging problem: visual captioning. It consists of generating a description in natural language from a visual object, namely an image or a video. We follow the sequence-to-sequence framework, under a multimodal perspective. We start by tackling the task of generating captions of videos from a general domain. Next, we move on to a more specific case: describing events from egocentric images, acquired along the day. Since these events are consecutive, we aim to extract inter-eventual relationships, for generating more informed captions. To this end, we propose a context-augmented system, able to consider the previous events while analyzing the current one. The results show that the context-aware model improved the generation quality with respect to a regular one. As final point, we apply the interactive-predictive protocol to these multimodal captioning systems. As in the machine translation case, this protocol diminished the effort required for correcting the outputs of an automatic system.

# Resumen

El problema conocido como *de secuencia a secuencia* consiste en transformar una secuencia de entrada en una secuencia de salida. Bajo esta perspectiva se puede atacar una amplia cantidad de problemas, entre los cuales destacan la traducción automática, el reconocimiento automático del habla o la descripción automática de objetos multimedia. En los últimos años, la aplicación de redes neuronales profundas ha revolucionado esta disciplina, y se han logrado avances notables. Sin embargo, y a pesar de estas mejoras, los sistemas automáticos todavía producen predicciones que distan mucho de ser perfectas. Para obtener predicciones de gran calidad, los sistemas automáticos se utilizan con la supervisión de un ser humano, quien corrige los errores. Esta forma de trabajar es muy común en la industria de la traducción. Esta tesis se centra principalmente en el problema de la traducción del lenguaje natural, el cual se ataca usando modelos enteramente neuronales. Nuestro objetivo principal es desarrollar sistemas de traducción neuronal más eficientes. Para ello, nuestras contribuciones se asientan sobre dos pilares fundamentales: cómo utilizar el sistema de una forma más eficiente y cómo aprovechar datos generados durante la fase de explotación del mismo.

En el primer caso, aplicamos el marco teórico conocido como predicción interactiva a la traducción automática neuronal. Este proceso consiste en integrar usuario y sistema en un proceso de corrección cooperativo, con el objetivo de reducir el esfuerzo humano empleado en obtener traducciones de alta calidad. Desarrollamos distintos protocolos de interacción para dicha tecnología, aplicando interacción basada en prefijos y en segmentos. Estos protocolos se implementan básicamente modificando el proceso de búsqueda del sistema. Además, ideamos mecanismos para obtener una interacción con el sistema más precisa, pero manteniendo la velocidad de generación del mismo. Llevamos a cabo una extensa experimentación, que muestra el potencial de

---

estas técnicas: superamos el estado del arte anterior, obtenido mediante tecnologías clásicas, por un gran margen y observamos que nuestros sistemas reaccionan mejor a las interacciones humanas.

A continuación, estudiamos cómo mejorar un sistema neuronal mediante los datos generados como subproducto de este proceso de corrección. Para ello, nos basamos en dos paradigmas del aprendizaje automático: el aprendizaje muestra a muestra y el aprendizaje activo. En el primer caso, el sistema se actualiza al vuelo, inmediatamente después de que el usuario corrige una frase. Por lo tanto, el sistema aprende de una manera continua a partir de correcciones, evitando cometer errores previos y especializándose en un usuario o dominio concretos. Evaluamos estos sistemas en una gran cantidad de situaciones y dominios diferentes, que demuestran el potencial que tienen los sistemas adaptativos. También llevamos a cabo una evaluación humana, con traductores profesionales. Éstos quedaron muy satisfechos con el sistema adaptativo. Además, fueron más eficientes cuando lo usaron, si lo comparamos con el uso de un sistema estático. En lo referente al segundo paradigma, el aprendizaje activo, lo aplicamos para el escenario en el que se deban traducir grandes cantidades de frases, siendo inviable la supervisión de todas ellas. En este caso, el sistema selecciona aquellas muestras que vale la pena supervisar, traduciendo el resto automáticamente. Aplicando este protocolo, redujimos de aproximadamente un cuarto el esfuerzo humano necesario para llegar a cierta calidad de traducción. Además, también superamos el estado del arte anterior por un margen considerable.

Finalmente, atacamos el complejo problema de la descripción de objetos multimedia. Este problema consiste en describir en lenguaje natural un objeto visual, una imagen o un vídeo. Comenzamos con la tarea de descripción de vídeos pertenecientes a un dominio general. A continuación, nos movemos a un caso más específico: la descripción de eventos a partir de imágenes egocéntricas, capturadas a lo largo de un día. Buscamos extraer relaciones entre eventos para generar descripciones más informadas, desarrollando un sistema capaz de analizar un mayor contexto. El modelo con contexto extendido genera descripciones de mayor calidad que un modelo normal. Por último, aplicamos la predicción interactiva a estos sistemas de descripción multimodal, observando también una disminución del esfuerzo necesario para corregir las salidas de un sistema automático.

# Resum

El problema conegut com a *de seqüència a seqüència* consisteix en transformar una seqüència d'entrada en una seqüència d'eixida. Seguint aquesta perspectiva, es pot atacar una àmplia quantitat de problemes, entre els quals destaquen la traducció automàtica, el reconeixement automàtic de la parla o la descripció automàtica d'objectes multimèdia. En els últims anys, l'aplicació de xarxes neuronals profundes ha revolucionat aquesta disciplina, i s'han aconseguit progressos notables. No obstant això, els sistemes automàtics encara produeixen prediccions que disten molt de ser perfectes. Per a obtenir prediccions de gran qualitat, els sistemes automàtics són utilitzats amb la supervisió d'un ésser humà, qui corregeix els errors. Aquesta forma de treballar és molt comú a la indústria de la traducció. Aquesta tesi se centra principalment en el problema de la traducció de llenguatge natural, el qual s'ataca emprant models enterament neuronals. El nostre objectiu principal és desenvolupar sistemes de traducció neuronal més eficients. Per a aquesta tasca, les nostres contribucions s'assenten sobre dos pilars fonamentals: com utilitzar el sistema d'una manera més eficient i com aprofitar dades generades durant la fase d'exploració d'aquest.

En el primer cas, apliquem el marc teòric conegut com a predicció interactiva a la traducció automàtica neuronal. Aquest procés consisteix en integrar usuari i sistema en un procés de correcció cooperatiu, amb l'objectiu de reduir l'esforç humà emprat per obtenir traduccions d'alta qualitat. Desenvolupem diferents protocols d'interacció per a aquesta tecnologia, aplicant interacció basada en prefixos i en segments. Aquests protocols s'implementen bàsicament modificant el procés de cerca del sistema. A més a més, busquem mecanismes per a obtenir una interacció amb el sistema més precisa, mantenint la velocitat de generació. Duem a terme una extensa experimentació, que mostra el potencial d'aquestes tècniques: superem l'estat de l'art

---

anterior, obtingut mitjançant tecnologies clàssiques, per un gran marge i observem que els nostres sistemes reaccionen millor a les interaccions humanes.

A continuació, estudiem com millorar un sistema neuronal mitjançant les dades generades com a subproducte d'aquest procés de correcció. Per a això, ens basem en dos paradigmes de l'aprenentatge automàtic: l'aprenentatge mostra a mostra i l'aprenentatge actiu. En el primer cas, el sistema s'actualitza immediatament després que l'usuari corregeix una frase. Per tant, el sistema aprèn d'una manera contínua a partir de correccions, evitant cometre errors previs i especialitzant-se en un usuari o domini concrets. Avaluem aquests sistemes en una gran quantitat de situacions i per a dominis diferents, que demostrin el potencial que tenen els sistemes adaptatius. També duem a terme una avaluació humana, amb traductors professionals. Aquests van quedar molt satisfets amb el sistema adaptatiu. A més, van ser més eficients quan ho van usar, si ho comparem amb l'ús d'un sistema estàtic. Pel que fa al segon paradigma, l'aprenentatge actiu, l'apliquem per a l'escenari en el qual han de traduir-se grans quantitats de frases, i la supervisió de totes elles és inviable. En aquest cas, el sistema selecciona les mostres que paga la pena supervisar, traduint la resta automàticament. Aplicant aquest protocol, reduïrem en aproximadament un quart l'esforç necessari per a arribar a certa qualitat de traducció. A més a més, també superem l'estat de l'art anterior per un marge considerable.

Finalment, ataquem el complex problema de la descripció d'objectes multimèdia seguint la mateixa perspectiva de seqüència a seqüència. Aquest problema consisteix en descriure, en llenguatge natural, un objecte visual, és a dir, una imatge o un vídeo. Comencem amb la tasca de descripció de vídeos d'un domini general, la qual ataquem de forma similar al problema de la traducció. A continuació, ens movem a un cas més específic: la descripció d'esdeveniments a partir d'imatges egocèntriques, capturades al llarg d'un dia. Com que aquests esdeveniments són consecutius, busquem extraure relacions entre ells per a generar descripcions més informades. Per a això, desenvolupem un sistema capaç d'analitzar un major context per considerar els esdeveniments previs mentre s'analitza l'actual. Els resultats mostren que el model amb context estés genera descripcions de major qualitat que el model bàsic. Finalment, apliquem la predicció interactiva a aquests sistemes de descripció multimodal. De la mateixa forma que al cas de la traducció automàtica, aquest protocol disminueix l'esforç necessari per a corregir les eixides d'un sistema automàtic.



# Preface

The pattern recognition discipline, based on machine learning methods, is nowadays experiencing an unprecedented expansion. Several fields, such as language processing or computer vision, have been revolutionized by the application of deep neural networks, fed with large amounts of data. The recent improvements achieved in these areas have moved the systems from research prototypes towards end-products, used by a broad audience. This thesis tackles a particular pattern recognition problem, known as sequence-to-sequence learning, which consists of the transduction of an input sequence into an output sequence. Currently, the most successful methods to tackle this problem rely on the aforementioned neural architectures. A variety of tasks can be addressed under this sequence-to-sequence perspective. We are particularly interested in the machine translation (MT) task, consisting of the automatic translation of sentences from a source language into a target language. The introduction of the so-called neural machine translation (NMT) broke through several performance barriers.

Despite the advances achieved in this field, the MT problem is still not solved and automatic systems still make errors. A common usage of MT systems in the industry involves the supervision of the automatic translations by a human expert, who corrects the errors made by the system. In this thesis, we focus on the implementation of a specific supervision paradigm called interactive-predictive MT with neural sequence-to-sequence models. Moreover, new data are continuously generated as a byproduct of this error-correction process. We study how to leverage these data, to continuously improve the system.

In addition, the neural sequence-to-sequence framework can be applied to other tasks. In this thesis, we tackle a multimodal task, namely, the captioning of visual

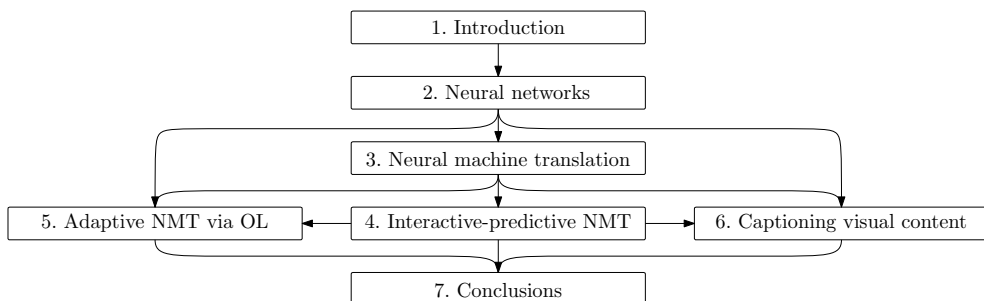
---

content. This involves the generation of descriptions in the natural language from a visual object, an image or a video. After tackling the video captioning task in general, we focus on a specific subtask within this field: captioning of sequences of egocentric images captured along one day. We finally apply the interactive-predictive framework to these multimodal tasks.

We divide the scientific goals of this thesis into three main groups:

1. **Interactive-predictive neural machine translation.** We develop the application of the interactive-predictive framework to NMT. We describe alternative protocols of interaction, aiming to provide flexibility to the process.
2. **Adaptive neural machine translation.** We study the creation of adaptive NMT systems, able to exploit the corrected sentences, in a post-editing or interactive-predictive MT scenario. The systems are updated on-the-fly, as soon as the sentences are corrected, via online learning (OL) techniques. We study different methods and conduct evaluations on a simulated workbench and on a real scenario, involving professional post-editors. In addition, we also study the application of active learning techniques to the translation of large data collections with NMT.
3. **Captioning visual content.** We tackle multimodal sequence-to-sequence tasks, focusing on video captioning. After building a video captioning system, we develop a solution to generate captions from sequences of egocentric images, continuously acquired during a day. Hence, these sequences are interrelated. We develop a model to specifically model these relationships. Finally, we show that the interactive-predictive protocol, originally developed for NMT, can be successfully applied in this scenario.

This dissertation is structured in seven chapters, related in the following way:



The content of each chapter is as follows:

---

**Chapter 1** frames the scope of this thesis, introducing the pattern recognition field and, more specifically, the MT field. It reviews the different historical approaches devised to tackle this problem. Moreover, it sets the experimental framework followed in this thesis and the main scientific objectives.

**Chapter 2** describes the mathematical model that represents the core of the thesis: neural networks. It addresses the parameter estimation process, describes different neural architectures and a number of techniques used along the thesis to improve the generalization capability of the model.

**Chapter 3** introduces the neural machine translation technology, describing the most common architectures and decoding process. Moreover, it reviews different aspects relating the NMT field that nowadays receive the attention of the research community. It also compares NMT in the different translation tasks that will be tackled in the thesis.

**Chapter 4** introduces the interactive-predictive pattern recognition field, that aims to minimize the effort spent by the user while supervising an automatic system. It proposes the application of this theoretical framework to the neural technology, introducing alternative interaction protocols. After that, these interactive-predictive neural systems are evaluated.

**Chapter 5** describes the adaptation of NMT systems via online learning techniques. After receiving a corrected sample, the system can be updated to include this new knowledge. Here are described the methods to perform this adaptation and introduces two novel alternatives. In addition, an active learning framework for neural systems is proposed, useful for a situation that requires the translation of large amounts of data. All these scenarios are thoroughly evaluated in a variety of conditions, including a user evaluation involving professional post-editors.

**Chapter 6** departs from the MT problem to tackle different multimodal sequence-to-sequence tasks. More precisely, it is focused on the generation textual descriptions of videos. These techniques are also applied to the captioning of daily events, captured with an egocentric camera. Finally, the interactive-predictive framework described in **Chapter 4** is applied to these multimodal systems.

**Chapter 7** draws the main conclusions of the thesis, describing the scientific contributions and publications derived from it and traces several lines of future research.

These chapters are complemented by two appendices. **Appendix A** describes NMT-Keras, an open-source library developed to build neural models, that has been used to carry out most of the experiments described in the thesis. In **Appendix B** we provide the results of a survey carried out in the scope of **Chapter 5**.



# Notation

Symbol	Description
$\mathcal{A}, \dots, \mathcal{Z}$	Sets.
$\mathbf{A}, \dots, \mathbf{Z}$	Matrices.
$\mathbf{a}, \dots, \mathbf{z}$	Vectors.
$[\cdot; \cdot]$	Vectorial concatenation.
$a_1, \dots, a_T$	Sequence of length $T$ .
$\mathbf{a}_1^T$	
$\mathbf{a}_1, \dots, \mathbf{a}_T$	Sequence of vectors of length $T$ .
$\mathbf{a}_1^T$	
$a^{(s)}$	$s$ -th element from a set.
$\text{Pr}(\cdot)$	Probability distribution with no model assumptions.
$\text{p}(\cdot)$	Probability distribution with model assumptions.
$\text{p}(\cdot; \Theta)$	Probability distribution parameterized in terms of $\Theta$ .
$\Theta$	Parameters of a model.
$ \cdot $	Cardinality of a set, sequence or tensor.
$\odot$	Element-wise product.
$\ell$	Loss function.
$\nabla_{\Theta} \ell$	Gradient of the function $\ell$ with respect to the parameters $\Theta$ .



# Contents

Preface	xi
1 Introduction	1
1.1 Pattern recognition and probabilistic modeling	2
1.2 Machine translation	6
1.3 Statistical machine translation	7
1.4 Experimental framework	10
1.5 Scientific goals	16
1.6 Summary	18
2 Neural networks	19
2.1 From linear classifiers to multi-layer perceptrons	20
2.2 Parameter estimation: gradient descent	23
2.3 Neural network architectures	27
2.4 Improving generalization in neural networks	39
2.5 Summary	41

3	Neural machine translation	43
3.1	RNN-based NMT	45
3.2	Transformer: attention-based NMT	49
3.3	NMT decoding	52
3.4	Dealing with the vocabulary restriction	53
3.5	Machine translation results	55
3.6	Summary	61
4	Interactive-predictive neural machine translation	63
4.1	Interactive-predictive machine translation	64
4.2	Probabilistic framework	67
4.3	Interactive-predictive neural machine translation	71
4.4	Interactive-predictive neural machine translation results	77
4.5	Summary	89
5	Adaptive neural machine translation via online learning	91
5.1	Applications of online learning in machine translation	93
5.2	Online learning for NMT post-editing	95
5.3	Results on NMT adaptation via online learning	101
5.4	A user evaluation of machine translation post-editing with online learning	123
5.5	Active learning for the interactive-predictive translation of large data streams	130
5.6	Summary	139
6	Captioning visual content	141
6.1	Video captioning	143
6.2	Egocentric captioning	151
6.3	Interactive-predictive captioning	171
6.4	Summary	174
7	Conclusions	177
7.1	Scientific contributions	177
7.2	Dissemination of the work	180



7.3 Future work. . . . .	183
A The NMT-KERAS toolkit	187
A.1 Design . . . . .	188
A.2 Features. . . . .	190
A.3 Interactive-predictive and adaptive pattern recognition . . . . .	192
A.4 Additional applications. . . . .	195
A.5 Summary . . . . .	195
B Human post-editors survey	197
B.1 Initial questionnaire . . . . .	197
B.2 Final questionnaire. . . . .	198
List of Figures	203
List of Tables	207
Bibliography	209



# Chapter 1

## Introduction

Language is one of the characteristic traits of humankind. The ability to communicate, from concise to abstract thoughts, has allowed the development of societies and the advances of sciences. Consequently, language itself is a wide and complex communication system, thoroughly studied since the dawn of history. The *natural language processing* (NLP) field refers to the computerized study of natural language. Its ultimate goal is to create systems for understanding and generating natural languages at the human level. One of the most challenging tasks within NLP is *machine translation* (MT). It refers to the automatic translation of sentences from a natural language to another. This problem attracts the attention of industry and research, for breaking down the barriers generated by different languages.

In the last thirty years, the field of MT has been greatly influenced by artificial intelligence techniques. More precisely, the employment of statistical models to solve the MT problem brought impressive advances. This is known as *statistical machine translation* (SMT). Moreover, in the last few years, a novel approach to SMT has been developed: the so-called *neural machine translation* (NMT). It relies on highly expressive models, called *neural networks*, to perform the translations. The NMT technology achieves very fluent translations and it has been widely adopted by most translation stakeholders (e.g. [Crego et al., 2016](#); [Wu et al., 2016](#)).

Moreover, not only MT has benefited from this revolution: neural networks are widely used in most NLP problems, such as speech recognition and synthesis ([Chan et al., 2016](#)), handwritten text recognition ([Graves et al., 2009](#)), and language analysis

(Devlin et al., 2018). Neural networks are also the predominant model in other disciplines like computer vision (Krizhevsky et al., 2012), system control (Zissis et al., 2015) or capital forecasting (French, 2017), among others.

Despite these impressive advances, MT is still far from being a solved problem. The systems are extremely sensitive to noisy samples, domain or user mismatch and lack of data. However, MT is commonly used in industry, generating translation hypotheses, which are corrected by a human agent. This process, known as *post-editing*, is more efficient than building the translations from scratch in a number of scenarios (Arenas, 2008; Hu and Cadwell, 2016).

In this thesis, we aim to increase the productivity of this process. To this end, we apply the so-called *interactive-predictive machine translation* (IMT) framework (Foster et al., 1997) to the NMT technology. This consists of a tight integration of system and user, fostering cooperation and with the goal of obtaining high-quality translations while minimizing the human effort spent in the process. Moreover, as a consequence of this method of work, new data are being continuously generated. We devise strategies to profit from these data, following an *online learning* (OL) paradigm, adapting the NMT systems on-the-fly, towards a given user or domain, using the successive corrections. Finally, we tackle other scenarios, involving multimodal signals. More precisely, the generation of language from sequences of images. We also study the integration of the interactive-predictive framework to these multimodal systems.

This introductory chapter presents the pattern recognition field, the formal framework that sustains this thesis. Next, the MT field is briefly reviewed from a historical perspective, and the different paradigms devised to tackle this problem are briefly reviewed. Next, we set up the experimental framework that is followed throughout the dissertation. We conclude the chapter by tracing the scientific goals of this thesis.

## 1.1 Pattern recognition and probabilistic modeling

Pattern recognition is a discipline framed in the artificial intelligence field, devoted to finding regularities in data in an automatic way (Bishop, 2006). Most approaches to pattern recognition make use of machine learning techniques, building a mathematical model of the reality for a given task. This model is intended to learn properties on some observed data and generalize them, to perform predictions on new, unseen samples. To that end, the employment of statistical inference and probabilistic modeling techniques become fundamental.

We seek a function  $f$  to make predictions about samples from an input domain ( $\mathcal{X}$ ). Our goal is to assign to an input sample  $x \in \mathcal{X}$  a prediction  $\hat{y} \in \mathcal{Y}$ , where  $\mathcal{Y}$  is an output domain. Hence,  $f : \mathcal{X} \rightarrow \mathcal{Y}$ , is a predictor function:  $\hat{y} = f(x)$ .

Under a probabilistic perspective, this function represents an a posteriori probability distribution: Our desired output is the element from  $\mathcal{Y}$  with the highest probability, conditioned to  $x$ . This expression is known as the optimal Bayes decision rule (Bishop, 2006), as in Eq. (1.1):

$$\hat{y} = \arg \max_{y \in \mathcal{Y}} \Pr(y | x) \quad (1.1)$$

However, the true distribution from this expression ( $\Pr(y | x)$ ) is unknown. Hence, we need to approximate it. Thus, we rely on a statistical model, parameterized by  $\Theta$ , as in Eq. (1.2):

$$\hat{y} = \arg \max_{y \in \mathcal{Y}} p(y | x; \Theta) \quad (1.2)$$

This expression summarizes the three main challenges of probabilistic modeling:

1. **Model definition:** How to define mathematical models able to approximate well the true probability distribution  $\Pr(y | x)$ .
2. **Parameter estimation:** Once the model has been defined, we need to obtain the parameters that better approximate the true distribution. Such parameters are usually estimated from a set of data samples, in a process known as (stochastic) training. The goal is to obtain an optimal set of parameters  $\hat{\Theta}$  that maximize a parameter-dependent objective criterion  $\ell_{\Theta}$  on a given data set ( $\mathcal{S}$ ), as in Eq. (1.3):

$$\hat{\Theta} = \arg \max_{\Theta} \ell_{\Theta}(\mathcal{S}) \quad (1.3)$$

3. **Search problem:** Once the parameters have been estimated, we need to obtain the best prediction, searching for  $\hat{y}$  in the output domain space that maximizes the probability; i.e., solving the  $\arg \max$  operation in Eq. (1.2). This process is known as decoding and it can represent a hard problem if  $\mathcal{Y}$  is a large space.

### 1.1.1 Taxonomy of pattern recognition approaches

Statistical pattern recognition systems can be categorized according to several facets. Although the frontiers of these classifications are somewhat diffuse, we differentiate three main criteria: the domain of the predictions, the function approximated by the models and the training paradigm used to estimate the parameters of the model.

#### According to the output domain

Depending on the output domain ( $\mathcal{Y}$ ), we face two main types of problems:

**Unstructured prediction:** The output is an atomic element, typically a single value ( $f : \mathcal{X} \rightarrow \mathbb{R}$ ) or a vector of dimension  $d$  ( $f : \mathcal{X} \rightarrow \mathbb{R}^d$ ). Depending on the form of this value, we can divide the systems into two subcategories:

- **Regression:** The output domain is arbitrary, usually a real value. Hence, we generally understand as a regression problem the prediction of a certain value.
- **Classification:** Specific case of regression in which the output domain is finite and known. The classification problem consists of assigning to an input object one or more classes, from a finite, and typically small, set of classes.

**Structured prediction:** Refers to problems in which the output has a structure ( $f : \mathcal{X} \rightarrow \mathcal{Y}$ , where  $\mathcal{Y}$  is a structured domain). As structure we understand a dependency across the elements of the object. Typical structured objects include sequences, trees or graphs. We can also differentiate between structured classification or structured regression.

Within the scope of this thesis, an especially interesting subclass of structured prediction problems is called sequence to sequence. The goal is to predict an output sequence given an input sequence. Several NLP problems can be formulated as sequence-to-sequence tasks, such as machine translation, captioning, parsing, summarization or speech recognition.

#### According to the type of models

Depending on the function approximated, we find two families of models:

**Discriminative models:** they directly approximate the conditional probability from Eq. (1.1), predicting the target variable ( $y$ ) given the observation ( $x$ ). Among

the most common discriminative models are neural networks and conditional random fields.

**Generative models:** they approximate a joint probability distribution ( $\Pr(x, y)$ ). Eq. (1.1) can be rewritten in terms of the joint distribution as:

$$\hat{y} = \arg \max_{y \in \mathcal{Y}} \frac{\Pr(x, y)}{\Pr(x)}$$

Note that in this expression, since the denominator is independent of the maximization, it can be removed. This decomposition is known as noisy channel modeling (Eq. (1.4), Shannon, 1948):

$$\hat{y} = \arg \max_{y \in \mathcal{Y}} \Pr(x | y) \Pr(y) \tag{1.4}$$

### According to the learning paradigm

The parameters of the model are usually estimated in a process known as learning or training. The training is performed on a collection of data samples  $\mathcal{S}$ . According to the characteristics of these data, we find different learning paradigms (Bishop, 2006):

**Supervised learning:** At training time, we have available the input objects and their corresponding labels. Hence, the training data is a set with  $S$  object-target pairs:  $\{(x^{(s)}, y^{(s)})\}_{s=1}^S$ , where  $(x^{(s)}, y^{(s)})$  represents the  $s$ -th training sample. This is by far the most extended paradigm, and most mature applications rely on supervised learning techniques. However, to collect labeled samples is an expensive task and this may condition the size of such annotated datasets.

**Unsupervised learning:** The training data only contains information from the input objects:  $\{x^{(s)}\}_{s=1}^S$ . Although unsupervised learning presents several difficulties, its main advantage is that unlabeled data is more abundant and cheaper to obtain than annotated data. With the huge amount of information available nowadays, this paradigm is taking off recently.

**Other paradigms:** Alternative approaches stand between supervised and unsupervised learning. Among them, it is worth mentioning semi-supervised learning (Chapelle et al., 2006), in which the training data contains labeled and unlabeled samples; active learning (AL, Cohn et al., 1994), in which a human oracle is asked to label relevant samples; or reinforcement learning (Sutton and Barto, 1998), in which the system receives a weak feedback (not the ground-truth label) from its predictions.

In this thesis, we rely on the supervised learning paradigm, although we also explore the active learning scenario (see [Section 5.5.1](#)).

## 1.2 Machine translation

Automatically translating a language into another is a dream pursued by humankind from long ago. According to [Hutchins \(2004\)](#), back in the 17th century, philosophers such as Descartes and Leibniz, sought for universal or logical communication codes. These ideas can be seen as the forerunners of constructed auxiliary languages (e.g. Esperanto) and of digital translation programs. During the 18th and 19th centuries, proliferated the so-called universal languages. The firsts machines developed to automatically translate languages did not appear until the 20th century. In 1949, [Weaver \(1949\)](#) set the basis for using computers to perform translations. He relied on ideas from information theory ([Shannon, 1948](#)) developed during the World War II. During the 1950s, there was an excessive optimism with respect to MT. The results, however, were unsatisfactory, the progress, slow and the funding devoted to MT greatly decreased.

During the 1970s and 1980s, MT research was focused on the so-called rule-based systems. In the late 1980s, as the computational capacity increased, a new family of methods arose: the so-called corpus-based MT systems. They relied on statistical methods ([Brown et al., 1990, 1993](#)), and their capabilities and potential were rapidly acknowledged by the scientific community. From here, the development of statistical machine translation was greatly boosted during 20 years, reaching its peak with phrase-based statistical machine translation (PB-SMT). At the end of the first decade of the 21st century, PB-SMT systems were arguably considered the state-of-the-art and the translation industry was steadily including them into their working pipelines ([Wendt, 2010; Federico et al., 2014](#)).

But this would not last for long: framed in the third wave of neural networks, the novel NMT approach was developed ([Kalchbrenner and Blunsom, 2013; Cho et al., 2014; Sutskever et al., 2014](#)). From here, NMT has deeply penetrated in the research community ([Bojar et al., 2017a, 2018](#)) and in the industry ([Crego et al., 2016; Wu et al., 2016](#)). The advent of the NMT has also opened new research directions, questions and challenges.



### 1.3 Statistical machine translation

The statistical approach to MT is a corpus-based technique, that studies the statistical patterns from large amounts of text. The general framework of SMT is based on the statistical inference theory presented in [Section 1.1](#).

SMT assumes that we can compute a translation probability  $\Pr(y_1^I | x_1^J)$  between a sentence  $y_1^I = y_1, \dots, y_I$  from a target language  $\mathcal{Y}$  and a sentence  $x_1^J = x_1, \dots, x_J$  in the source language  $\mathcal{X}$ . The goal of the SMT system is, given the source sentence  $x_1^J$ , to determine and find the target sentence with the highest probability  $\hat{y}_1^I$ . This can be computed according to [Eq. \(1.5\)](#):

$$\hat{y}_1^I = \arg \max_{I, y_1^I} \Pr(y_1^I | x_1^J) \quad (1.5)$$

Since the true distribution is unknown, it is approximated using statistical models, with parameters  $\Theta$ , as in [Eq. \(1.6\)](#):

$$\hat{y}_1^I = \arg \max_{I, y_1^I} p(y_1^I | x_1^J; \Theta) \quad (1.6)$$

This expression encapsulates the three main challenges of pattern recognition ([Section 1.1](#)) under the SMT prism:

1. **Model definition:** Development of models capable to approximate well the translation probability distribution  $\Pr(y_1^I | x_1^J)$ .
2. **Parameter estimation:** Once the model has been defined, its parameters need to be estimated, typically from data. These data collections are usually parallel corpora: sentence-aligned documents of translated sentences.
3. **Search problem:** Once the parameters have been estimated, the translations are obtained by searching for the target language string with the highest probability. This is also known as decoding. This is a difficult problem ([Udupa and Maji, 2006](#)) that requires a fast solution, for practical considerations. Most systems tackle the problem via suboptimal but fast search algorithms.

### 1.3.1 Classical approaches to SMT

Eq. (1.6) is hard to compute, and the first SMT works (Brown et al., 1990) tackled it under a generative perspective, following the noisy channel framework (Eq. (1.4)). This required to estimate two separate models: the language model and the (inverse) translation model.

Classical language models were based on the concept of  $n$ -grams. An  $n$ -gram is a sequence of  $n$  consecutive tokens. Assuming Markovian properties, a given token depends only on its  $n - 1$  preceding tokens. The estimation of its probability is done by counting occurrences on a training corpus. In the last years, the most successful language models are based on neural networks (Bengio et al., 2003; Mikolov et al., 2010).

The first statistical translation models were based on word alignments (Brown et al., 1990, 1993). An alignment is a correspondence which indicates, for each word of the source sentence, the word from the target sentence from which it arose. Brown et al. (1993) developed the so-called IBM models, defining five models of increasing complexity, intended to be trained sequentially.

A central issue suffered by the IBM models is that they are unable to capture the context, as they operate at a word level. Several works aimed to augment the context of word-based alignment models. The most adopted solution were the so-called phrase-based SMT (Zens et al., 2002): an extension of the single word alignment to consecutive sequences of words, known as phrases.

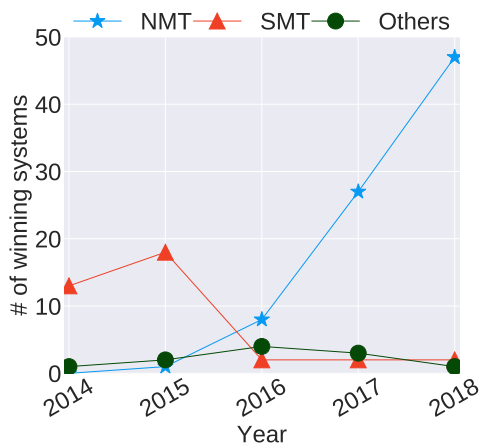
**The log-linear model of PB-SMT** Another important advance of PB-SMT models was to tackle the problem following a discriminative perspective (Section 1.1.1), directly modeling  $\Pr(y_1^I \mid x_1^J)$  from Eq. (1.5). This was done with the so-called log-linear model (Och and Ney, 2002; Koehn, 2010b): a weighted log-linear combination of feature functions, estimated independently. Among the more common functions included in the log-linear model we find a (target) language model, bidirectional translation models and a reordering model, among others (Koehn, 2010b).

Once the feature functions have been estimated, it is necessary to estimate the weights of the log-linear combination. This process aims to find the log-linear combination weights that optimize an arbitrary criterion, typically a translation quality metric (see Section 1.4.1).

### 1.3.2 Neural machine translation

The first ideas of using neural networks to generate translations date from long ago (Allen, 1987; Chrisman, 1991; Castaño and Casacuberta, 1997; Forcada and Neco, 1997). These systems were hard to scale to real tasks and these ideas were not further explored.

The first successful attempts to a purely NMT system are relatively recent. Almost simultaneously, several independent works tackled the problem as a sequence-to-sequence transduction task (Graves, 2013), using a neural encoder–decoder model (Kalchbrenner and Blunsom, 2013; Cho et al., 2014; Sutskever et al., 2014). These works proposed to perform the translation using solely a large neural network. With the inclusion of the so-called attention mechanism (Bahdanau et al., 2015), the NMT performance was greatly boosted, almost reaching the state of the art.



**Figure 1.1:** Technology of the top-performing systems in the constrained translation tasks from WMT. The “SMT” tag includes hierarchical and PB-SMT systems, which may also use neural features. The “others” category includes hybrid, rule-based and combination of MT systems. Data collected from [www.statmt.org](http://www.statmt.org).

From here, NMT had a meteoric trajectory. The shared tasks from the Conference on Machine Translation (WMT) are a good indicator of this evolution: NMT debuted in the shared tasks in 2015 (Jean et al., 2015b). Since then, NMT has rapidly outperformed the rest of technologies. Fig. 1.1 shows the number of the winning systems in the last WMT shared tasks, according to each technology. In addition to exhibiting the superior performance of NMT, this figure also illustrates the increasing popularity of NMT, in terms of number of participants in the WMT shared tasks.

NMT tackles the problem of SMT by using a single, large neural network, whose all components are jointly trained. More precisely, the three challenges of pattern recognition (Section 1.3) are addressed as follows:

1. **Model definition:** The model of NMT systems is a large neural network, that directly approximates the probability distribution  $\Pr(y_1^I \mid x_1^J)$ . Most NMT systems follow an encoder–decoder architecture: the source sentence is represented in a numerical way by means of an encoder neural network. Another neural network (decoder) takes this representation and transforms it into a sentence in

the target language. Refer to [Sections 3.1](#) and [3.2](#) for a detailed description of these neural architectures.

2. **Parameter estimation:** All the parameters of the model are estimated typically by means of gradient descent, usually under a maximum likelihood approach. This represents a key difference between PB-SMT systems and NMT: while PB-SMT consists of multiple decoupled components, trained independently and combined by means of the log-linear model, the parameters of the NMT model are jointly estimated. [Section 2.2.2](#) describes the training methods for neural networks.
3. **Search problem:** Most NMT systems use the beam search method to find the best translations. This procedure is detailed in [Section 3.3](#).

## 1.4 Experimental framework

We describe now the experimental framework followed in this thesis. We start by describing the evaluation metrics employed to evaluate our proposals. Next, we introduce the user protocol defined when working with interactive-predictive systems. Next, we describe the significance tests, used to determine whether the systems present statistically significant differences.

### 1.4.1 Evaluation of machine translation systems

The assessment of MT is an extremely hard task. First, there is no consensus on what a *good* translation means. The goodness of a translation depends on several facets, which are not always measurable, and often depend on the final use-case of the system.

#### Evaluating the translation quality

The most reliable assessment methods of MT require a human evaluation of the translations. Unfortunately, this process is very costly and becomes unaffordable while developing MT systems. Therefore, it is necessary to devise methods to automatically evaluate MT systems. Although less reliable than human evaluations, these methods are inexpensive and become fundamental for the advances of the field. Most automatic metrics compare the output of the system to one or more reference translations, automatically computing a score. This score is intended to correlate well with a human judgment.

Since the earliest times of MT (Pierce and Carroll, 1966), a body of metrics have been proposed. Among them, the BLEU score (bilingual evaluation understudy, Papineni et al., 2002) is the most widely used. Nonetheless it is also accepted that BLEU suffers from several limitations when correlating with human judgments (Turian et al., 2003; Tatsumi, 2009) and it can be fooled with bad translations (Smith et al., 2016). Other metrics that are widely employed in the literature are such as TER (translation error rate, Snover et al., 2006) or METEOR (Lavie and Denkowski, 2009).

Despite these efforts, the automatic assessment still remains an open problem. In most works, as well as in this thesis, the translation quality is primarily assessed by means of TER and BLEU. In the case of multimedia captioning, it is common to evaluate the systems also with METEOR and CIDEr (Vedantam et al., 2015). These metrics are briefly described below.

**TER:** TER is defined as the minimum number of word edit operations that must be made in order to transform the translation hypothesis into the reference translation. The number of edit operations is normalized by the number of words in the reference sentence. The edit operations considered are insertion, substitution, deletion and swapping groups of words. The number of edit operations is obtained by dynamic programming. For the sake of readability, we will show TER scores multiplied  $\times 100$ .

**BLEU:** BLEU aims to model the correspondence between the output from a MT system and the one produced by a human. The BLEU score is based on the  $n$ -gram precision between the hypothesis and a reference. It also evaluates whether a translation hypothesis has the adequate length. While long hypotheses are naturally penalized by  $n$ -gram precision, a BLEU introduces a brevity penalty for shorter translations. The final score is computed as a weighted geometric mean of the  $n$ -gram precision, penalized by the brevity penalty. The maximum order of the  $n$ -grams is set to 4. We will show BLEU scores as percentages. Note that BLEU is a precision-based metric, hence, the higher the better. This is contrary to error-based metrics (such as TER), in which lower scores are better.

**METEOR:** BLEU only considers  $n$ -gram precision, ignoring the recall component. Moreover, it lacks an explicit word matching. METEOR aims to mitigate these issues. It is an alignment-based metric, that computes all valid alignments between the hypothesis and the reference(s). These alignments are computed using a stemmer, and synonym and paraphrase databases. Therefore, this is a language-dependent metric. Once the set of alignments is computed, the metric computes the unigram precision and recall. In addition, it also computes an alignment penalty for unmatched unigrams. The METEOR score is a harmonic mean of unigram precision and recall, modified by the alignment penalty.

**CIDEr:** This metric was designed to evaluate captioning systems, which typically have multiple references. Given a caption and a set of references, CIDEr computes the number of matching  $n$ -grams of the caption across all the references, but it penalizes the frequent  $n$ -grams appearing in the set of references. By construction, CIDEr ranges from 0 (lowest quality) to 10 (highest quality).

## Evaluating the human effort

The ultimate goal of IMT is to reduce the human effort required to correct MT outputs. Therefore, we need to assess, additionally to the translation quality, this human effort spent during the interactive-predictive translation process. As explained before, we followed Zaidan and Callison-Burch (2010) and used TER as a representation of human-targeted TER, considering the reference sentences as human post-edited versions of the MT hypotheses. This gives us a broad approximation of the effort required to post-edit a translation hypothesis.

To estimate the effort spent in the process of interactive-predictive translation, we followed the classical evaluation framework of IMT, which estimates the human effort as the number of actions performed by the human agent in order to obtain the desired translations. Our protocol supports two different input devices: keyboard and mouse. We estimate the human effort according to the actions performed with such devices.

In Chapter 4 we distinguished between word-level and character-level interaction. To fairly assess each approach, we need to use metrics that capture the interactions at the corresponding level. Therefore, we evaluate word-level systems with word-level metrics:

**Word Stroke Ratio (WSR, Tomás and Casacuberta, 2006):** number of word corrections that the user had to make in order to obtain the desired translation, divided by the total number of words in the final sentence. It is assumed that the cost of correcting a word is constant, regardless of its length.

**Mouse Action Ratio (MAR, Barrachina et al., 2009):** measures the effort made by the user with the mouse during the interactive-predictive process. It is defined as the number of mouse actions made by the user, divided by the total number of characters in the final hypothesis. We count mouse “clicks” as mouse actions. Correcting a wrong word requires to select the word and accounts for one mouse action. If the user is correcting contiguous words, no mouse action is performed. Validating a segment requires two mouse actions: clicking at the beginning and at the end of the segment. A single mouse action is enough for validating one-word segments. We add an additional action per sentence, accounting for the final validation of a hypothesis.

On the other hand, character-level systems are evaluated according to:

**Keystroke and mouse-action ratio (KSMR, Barrachina et al., 2009):** it is defined as the number of keystrokes plus number of mouse actions required in the IMT process, divided by the number of characters of the reference. If the user is correcting contiguous characters, no mouse action is needed. An additional mouse action that accounts for the acceptance of a hypothesis is added to each sentence.

### Simulation protocol

A direct evaluation of an IMT system requires to conduct experiments with human experts. Unfortunately, this is excessively costly. Hence, automatic evaluation methodologies have been developed in the literature (Tomás and Casacuberta, 2006; Barrachina et al., 2009; González-Rubio et al., 2013), to simulate the behavior of a real user. We followed such protocols to evaluate our proposals.

In the simulation, it is assumed that the reference sentences from our parallel corpora are the outputs desired by the user. This is a pessimistic assumption, since the user can find appropriate different translations for the same sentence. We consider only one translation to be the correct one, as is normal in MT evaluation.

We defined two simulated scenarios, accounting for both interactive-predictive protocols: prefix-based and segment-based. In the first one, the simulated user searches for the first wrong element from the translation hypothesis and positions the mouse pointer on it. Once the pointer is positioned, the user introduces the correction. These actions correspond to a mouse-action and a keystroke, respectively. The system then, produces an alternative hypothesis, considering the validated prefix. This process continues until the hypothesis produced by the system is the one desired by the user, i.e., the reference sentence.

In the segment-based interactivity, the simulated user selects the correct segments from the hypothesis. Each segment selection corresponds to mouse-action. We assume, without loss of generality, that word corrections are made from the left to the right, and they correspond to a keystroke. The system generates an alternative translation hypothesis when the correction is inserted. Such hypothesis accounts both for the word correction and the validated segments from previous interactions.

In the case of adaptive systems, we applied OL techniques using the corrected sentences, performing the adaptation after each sample. That is, before starting the translation of the next sentence, the models are updated according to the previous sample in both the post-editing and IMT scenarios.

### 1.4.2 Hypothesis testing

In order to determine whether a given system is better than another, a common practice is to perform statistical hypothesis tests. In statistical hypothesis testing, a result has statistical significance if it is unlikely to have occurred given a null hypothesis. Therefore, we need to compute the probability level of the null hypothesis, namely the  $p$ -value. If the  $p$ -value is lower than a predefined confidence level, we can reject the null hypothesis. As null hypothesis, we state that “*The outputs of two systems do not differ with respect to a given metric of interest*”.

In order to determine whether the differences between two systems are statistically significant or not, we use paired approximate randomization tests (Noreen, 1989). The underlying idea is that, under the null hypothesis, the systems are not different: any measure of quality for a given output could have been produced by both systems. The idea of paired approximate randomization is to shuffle, with the same probability, the measures of both systems. This randomization is repeated a number of times. The significance levels are computed as the percentage of trials in which the statistic on the shuffled data is greater than or equal to the test statistic computed on the test data. In this thesis, we set the number of repetitions  $R = 10,000$  and the confidence level for rejecting the null hypothesis was set to 0.95.

This test is similar to bootstrap resampling (Koehn, 2004), which is widely employed by the MT community. The main difference is that bootstrap resampling allows replacement during the shuffling process. This causes bootstrap resampling to be more prone to making false positive errors than approximate randomization (Riezler and Maxwell, 2005).

### 1.4.3 Machine translation tasks

We evaluate our proposals for MT in four tasks, of different nature and magnitude. For each task, we consider three language pairs, translating in both directions. We preprocessed our corpora following Edunov et al. (2018): lowercasing and tokenizing them with the scripts from Moses (Koehn et al., 2007). To train the neural systems, we applied joint byte pair encoding (Sennrich et al., 2016, see Section 3.4.1), with 32,000 merge operations.

The translations tasks, whose main features are described in Table 1.1, are the following:

**XRCE:** user manuals from Xerox printers (SchlumbergerSema S.A. et al., 2001). This corpus has been extensively used in the IMT literature (Barrachina et al., 2009; Ortiz-Martínez, 2016). It is a small corpus and, due to its domain, it contains short sentences with rigid and repetitive structures.



**Table 1.1:** Main figures of the XRCE, TED, UFAL and Europarl corpora.  $|S|$ ,  $|T|$  and  $|V|$  account for number of sentences, number of tokens and vocabulary size, respectively.  $k$  and  $M$  stand for thousands and millions. Numbers computed after applying BPE.

		Training			Development			Test		
		$ S $	$ T $	$ V $	$ S $	$ T $	$ V $	$ S $	$ T $	$ V $
XRCE	De	50k	541k	20k	964	11k	1k	995	12k	2k
	En		590k	10k		11k	1k		12k	2k
	Fr	52k	678k	12k	994	12k	2k	984	12k	2k
	En		617k	12k		11k	2k		11k	2k
	Es	56k	753k	13k	1,025	16k	2k	1,125	10k	2k
	En		667k	11k		14k	2k		9k	2k
TED	De	133k	2.7M	25k	883	21k	5k	1,565	33k	6k
	En		2.7M	19k		21k	4k		32k	4k
	Fr	107k	2.3M	23k	934	22k	4k	1,664	35k	5k
	En		2.1M	19k		20k	3k		33k	4k
	Zh	107k	2.0M	24k	934	34k	4k	1,664	33k	4k
	En		2.1M	18k		23k	3k		32k	4k
UFAL	De	3.0M	130M	31k	500	13k	4k	1,000	28k	4k
	En		127M	25k		12k	4k		25k	4k
	Fr	2.8M	134M	29k	500	14k	3k	1,000	29k	5k
	En		118M	24k		11k	4k		24k	4k
	Es	780k	13M	29k	500	13k	3k	1,000	26k	5k
	En		12M	22k		11k	3k		24k	4k
Europarl	De	1.9M	54M	27k	3,003	91k	10k	3,000	78k	12k
	En		55M	18k		84k	8k		75k	8k
	Fr	2.0M	63M	26k	3,003	92k	10k	3,000	84k	10k
	En		57M	21k		83k	9k		73k	8k
	Es	1.8M	50M	25k	3,003	91k	10k	3,000	82k	10k
	En		48M	29k		83k	8k		74k	8k

**TED:** transcriptions of TED talks (Mauro et al., 2012). It has also been used in several IMT and online learning works. We used the standard `dev2010` and `tst2010` partitions for development and test, respectively. In this task, we tackle the translation of Chinese. We split the Chinese into sequences of words with the tool developed by Tseng et al. (2005).

**UFAL:** data crawled from several medical collections, collected during the European project *Health in my Language* (Bojar et al., 2017b). The development and test data come from the *KConnect* project (Libovický et al., 2016).

**Europarl:** proceedings from the European Parliament (Koehn, 2005). We used the release 7. For the sake of comparison with previous works on IMT, we used the `newstest2012` and the `newstest2013` as development and test partitions respectively. Note that these partitions relate to news, and are slightly out of the domain of Europarl training corpus.

## 1.5 Scientific goals

This thesis is framed within the recently emerged NMT technology. We aim to push forward this field, by building flexible, interactive, adaptive translation systems, making a better profit from the data and also tackling other tasks, following the NMT spirit. The scientific goals of the thesis are described in the following sections.

### 1.5.1 Interactive-predictive neural machine translation

The interactive-predictive approach to MT aims to obtain high-quality translations while diminishing the human effort spent in the process. Although NMT conveys tremendous advances in the field, the systems still make errors. The application of the interactive-predictive protocol to NMT would enable a more efficient use of this technology in the post-editing field.

In [Chapter 4](#) we develop the interactive-predictive protocol for the NMT framework. We present novel interactive-predictive NMT (INMT) systems, able to interact with the user via prefixes or validating segments from translations. We also describe several refinements of the systems, aiming to enhance the user experience and productivity.

### 1.5.2 Adaptive neural machine translation

As the natural consequence of the post-editing or IMT processes, new data are continuously generated. These data represent valuable resources: they are domain-specific samples which, under certain conditions, may be expensive to collect. In addition, they are tailored to the user style of translation or to its preferences. Moreover, and since they are generated as a byproduct of the correction protocol, they have no additional production costs. Hence, the exploitation of these data is very interesting for the translation industry. We are interested in a particular case from this scenario, in which the system is updated as soon as a sentence is corrected. Therefore, the

corrections provided by the user are taken into account by an adaptive NMT system immediately, hopefully improving the translation quality provided and diminishing the post-editing effort on a continuous manner.

Moreover, industry has also the demand of translating huge amounts of sentences on a regular basis. In this case, the manual revision of all translation hypotheses is excessively expensive. Under these conditions, the active learning paradigm becomes very suitable. Under an AL framework, the system selects those samples that are considered worth being reviewed by the human agent. Once corrected, the system will be updated with these samples. Therefore, an important aspect of the AL paradigm consists of the development of selection functions, that sample the most useful sentences to be corrected by the user.

**Chapter 5** explores these scenarios. We firstly study how to update the NMT systems under these conditions. In addition to classical methods to incrementally update NMT systems, we propose two novel alternatives, that aim to improve some of the potential issues suffered by classical update rules. Moreover, we develop an AL framework for NMT. We propose alternative criteria for selecting whether a sentence should be reviewed by a human agent or not.

### 1.5.3 Captioning visual content

MT is just one of the many problems that can be tackled following the sequence-to-sequence paradigm. Among the other problems, image or video captioning results particularly interesting. These problems consist of generating a description of the content of a given image or video. It can be seen as an *image to text* translation. The development of such systems has interesting applications. One of them consists of captioning daily events, captured with an egocentric device. The construction of such life-logging systems is useful for treating and mitigating mild cognitive impairment (Sellen et al., 2007). In addition, these captioning systems, like MT, are not perfect. This opens the application of the interactive-predictive protocol, aiming to correct the outputs in a more efficient way.

**Chapter 6** is devoted to the development of these multimodal systems. We start by tackling a general-domain video captioning problem. Next, we specifically tackle the egocentric captioning problem from a collection of daily events. We hypothesize that there exists a relationship between these events that can be exploited by a system capable to deal with extended contexts. We propose multimodal captioning systems to tackle these problems. Finally, we study the application of the interactive-predictive framework developed in **Chapter 4** to these multimodal systems.

### 1.5.4 Dissemination of the work

An important aspect of the scientific progress is its dissemination. All the software developed in this thesis is freely available as open-source repositories, with MIT license. [Section 7.2.4](#) and [Appendix A](#) describe the software released as a byproduct of this thesis.

## 1.6 Summary

In this chapter we presented the statistical approach to the pattern recognition field, describing the main problems to solve and categorizing the different approaches to them. We briefly described the phrase-based technology and introduced the NMT framework. Hence, after this introduction, we are able to frame our task of interest: NMT is an instance of the sequence-to-sequence problem, a subcategory of the structured prediction field. We will tackle it using neural networks, a discriminative model that approximates the conditional translation probability. This model is typically trained on a supervised manner, from bilingual data collections.

We also introduced the general experimental framework used to assess our proposals. We explained the evaluation metrics, the hypothesis testing followed and the translation tasks that will be studied along the thesis. The chapter concluded by defining the scientific goals intended to be addressed in this thesis.

## Chapter 2

# Neural networks

As stated in the previous chapter, we will tackle the sequence-to-sequence problem using neural networks. These are mathematical models that obtain a mapping between an input and an output representation space. Neural networks consist of multiple simple processors, also called artificial neurons or units, with weighted connections between them. These models are loosely inspired in the neurons that constitute the biological brain, following the mathematical foundations of how information is represented in biological systems.

From a historical perspective, the expectations on neural networks can be seen as a roller coaster of hype and moderate disappointments. Their popularity had three different waves or epochs of great hope (Goodfellow et al., 2016). The first one started with the development of the theories of biological learning and the aim of mathematically modeling them (McCulloch and Pitts, 1943). The first algorithms that automatically learned the parameters of neural networks were developed in this first wave (Rosenblatt, 1958; Widrow, 1960).

The second wave of neural networks dates from the 1980s, and it was called “connectionism”. Major milestones were accomplished during this epoch, such as the popular back-propagation algorithm (Rumelhart et al., 1986), the identification of training difficulties of neural networks (Hochreiter, 1991; Bengio et al., 1994) and the introduction of key architectures, such as *convolutional neural networks* (ConvNets; LeCun et al., 1989a) or *recurrent neural networks* (RNN; Elman, 1990; Jordan, 1990; Hochreiter and Schmidhuber, 1997). Moreover, by this time neural networks at-

tracted the attention of statisticians, that provided a probabilistic interpretation of neural networks (Bishop, 1995; Ney, 1995). However, to train complex networks was a hard task then, mostly due to hardware and data limitations. Therefore, research on connectionist models decreased.

The third and current wave of neural networks started in mid 2000s, when researchers were able to properly train complex neural networks, leveraging huge amounts of data and exploiting the increase of the computational capabilities, using specialized hardware, such as graphics processing units (GPU). During this third wave, the term *deep learning* was coined (LeCun et al., 2015) to describe neural networks composed of multiple hidden layers. Nowadays, deep learning models constitute the top-performing approach for computer vision (Krizhevsky et al., 2012; Szegedy et al., 2017) and NLP, including MT (Bojar et al., 2018) or speech recognition (Zeghidour et al., 2018). Moreover, deep learning methods are being effectively introduced in other scientific areas such as chemistry (Kang and Cho, 2018) or medicine and biology (Ching et al., 2018).

In this chapter, we review the main aspects regarding neural networks that will be used to build our systems in the following chapters. We describe the neural network architectures that will be used to build our sequence-to-sequence models in Chapters 3 and 6. In addition, we describe the parameter estimation problem and some techniques that improve the generalization capabilities of these models.

## 2.1 From linear classifiers to multi-layer perceptrons

Recall from Section 1.1 that we are interested in obtaining a discriminant function  $f : \mathcal{X} \rightarrow \mathcal{Y}$  that maps an object from an arbitrary representation space  $\mathcal{X}$  into its corresponding class, from a finite space of classes  $\mathcal{Y}$ . Each object  $\mathbf{x}$  from  $\mathcal{X}$  is represented by a feature vector of size  $m$  ( $\mathbf{x} \in \mathbb{R}^m$ ).

A linear predictor function  $a : \mathcal{X} \rightarrow \mathbb{R}$  produces a linear combination of an input feature vector and a set of parameters  $\Theta$ . This set of parameters, also known as weights, contains a vector  $\theta \in \mathbb{R}^m$  and an independent term,  $\theta_0 \in \mathbb{R}$ , as in Eq. (2.1):

$$a(\mathbf{x}; \Theta) = \theta \mathbf{x} + \theta_0 \tag{2.1}$$

However, when using linear functions we are limited to inducing linear relationships. To obtain more expressive models, we can apply a non-linear function  $g : \mathbb{R} \rightarrow \mathbb{R}$  to the linear model. This function is called *activation function* and its arguments are known as activations. Therefore, a *non-linear predictor function* is defined as in Eq. (2.2):

$$g(a(\mathbf{x}; \Theta)) = g(\boldsymbol{\theta}\mathbf{x} + \theta_0) \quad (2.2)$$

We can express our desired discriminant function  $f$  in terms of  $|\mathcal{Y}|$  non-linear predictor functions, as in Eq. (2.3):

$$f(\mathbf{x}; \Theta) = \arg \max_{1 \leq c \leq |\mathcal{Y}|} g(\boldsymbol{\theta}_c \mathbf{x} + \theta_{c0}) \quad (2.3)$$

where each predictor function has  $(\boldsymbol{\theta}_c, \theta_{c0})$  as parameters, for  $1 \leq c \leq |\mathcal{Y}|$ .  $\Theta$  is a set encapsulating the parameters of all predictors that conform the discriminant function.

Discriminant functions are the atomic elements upon which neural networks are built. Under a connectionist perspective, each predictor function is called neuron or unit.

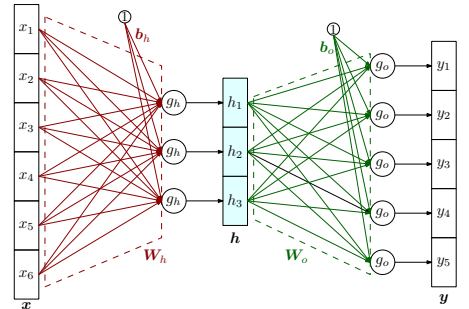
### 2.1.1 Multilayer perceptron

A *multi-layer perceptron* (MLP) applies several discriminant functions, organizing them as a chain: the units from an MLP can be seen as layers, in which the output of a given layer is the input to the next one.

An MLP has three types of layers: input, output and hidden layers. Input and output layers are used to introduce data to the system and to produce the output. The hidden layers consist of several discriminant functions, which apply Eq. (2.2). Therefore, an MLP with a single hidden layer of  $h$  units applies Eq. (2.2) to its inputs ( $\mathbf{x} \in \mathbb{R}^m$ ). We can stack the parameters of each predictor function into a hidden weight matrix  $\mathbf{W}_h \in \mathbb{R}^{h \times m}$  and a bias vector  $\mathbf{b}_h \in \mathbb{R}^h$ , and the output of this layer can be seen as a vector  $\mathbf{h} \in \mathbb{R}^h$ . This vector is called the hidden state and it is computed according to Eq. (2.4):

$$\mathbf{h} = g_h(\mathbf{W}_h \mathbf{x} + \mathbf{b}_h) \quad (2.4)$$

where  $g_h$  is the hidden layer activation function, applied element-wise. This hidden state is the input to the next layer, which produces an  $r$ -dimensional output  $\mathbf{y} \in \mathbb{R}^r$  by applying another set of predictor functions, as in Eq. (2.5):



**Figure 2.1:** MLP with a single hidden layer.  $\mathbf{x}$ ,  $\mathbf{y}$  and  $\mathbf{h}$  denote the input and output vectors and the hidden state, respectively. The parameters of the model ( $\Theta$ ) are  $\mathbf{W}_h$ ,  $\mathbf{b}_h$ ,  $\mathbf{W}_o$  and  $\mathbf{b}_o$ .

$$\mathbf{y} = g_o(\mathbf{W}_o \mathbf{h} + \mathbf{b}_o) \quad (2.5)$$

where, analogously to the previous layer,  $\mathbf{W}_o \in \mathbb{R}^{r \times h}$  and  $\mathbf{b}_o \in \mathbb{R}^r$  are the parameters to estimate and  $g_o$  is the output layer activation function. Fig. 2.1 shows the architecture of this single-layered MLP.

### 2.1.2 Activation functions

A key aspect of neural networks is the activation functions. Some activation functions make neural networks be universal approximators (Cybenko, 1989). That means, they are able to approximate any function arbitrarily well. Moreover, note that, as neural networks are primarily optimized via gradient-based methods (Section 2.2.2), these activation functions must be differentiable.

Coping with these properties, some activation functions have been extensively used in the neural networks literature and can be considered as “classical” activation functions. These functions, together with their derivatives, are shown in Table 2.1.

**Table 2.1:** Activation functions. \* denotes non-derivable functions.

Name	Activation function	Derivative
Rectified linear unit (ReLU)	$\text{ReLU}(x) = \max(0, x)$	$\text{ReLU}'(x) = \begin{cases} 1 & \text{if } x > 0 \\ 0 & \text{if } x < 0 \\ * & \text{if } x = 0 \end{cases}$
Logistic sigmoid	$\sigma(x) = \frac{1}{1 + \exp(-x)}$	$\sigma'(x) = \sigma(x)(1 - \sigma(x))$
Hyperbolic tangent	$\tanh(x) = \frac{\exp(x) - \exp(-x)}{\exp(x) + \exp(-x)}$	$\tanh'(x) = 1 - (\tanh(x))^2$
Softmax	$\varphi(\mathbf{x}) = \frac{\exp x_i}{\sum_{k=1}^{ \mathbf{x} } \exp x_k}$ , for $1 \leq i \leq  \mathbf{x} $	$\varphi'(\mathbf{x}) = \varphi(\mathbf{x})(\mathbf{1} - \varphi(\mathbf{x}))$

Most of these functions are applied element-wise to each activation. The softmax function, however, is defined in a vectorial domain ( $\mathbb{R}^{|\mathbf{x}|} \rightarrow \mathbb{R}^{|\mathbf{x}|}$ ). This is a normalization function, which ensures that all elements of its output space are between 0 and 1 and the sum of these elements is 1. These are the requirements of discrete probability distributions. Therefore the softmax function is usually applied as the output function of probabilistic models.



## 2.2 Parameter estimation: gradient descent

The training of neural networks involves an optimization problem: we want to find the parameters of the network  $\Theta$  that minimize an objective function  $\ell$ .

### 2.2.1 Objective functions

The choice of the objective function to optimize depends on the problem to tackle. In this thesis, we are interested in probabilistic modeling. We therefore employ neural networks to approximate probability distributions. This is usually done by applying the softmax function (Table 2.1) as the last activation function of the model. When dealing with such probabilistic networks, we generally are interested in measuring the differences between two probability distributions: the one produced by the model and the one given by the training data. The cross-entropy ( $H$ ) between two probability distributions,  $\text{Pr}_1$  and  $\text{Pr}_2$ , quantifies this difference, following Eq. (2.6):

$$H(\text{Pr}_1, \text{Pr}_2) = - \sum_k \text{Pr}_1(k) \log \text{Pr}_2(k) \quad (2.6)$$

Cross-entropy minimization is the most common optimization criterion for neural networks under a probabilistic framework. To achieve this, the output space  $\mathcal{Y}$  is encoded following a one-hot scheme: let  $K$  be the dimension of  $\mathcal{Y}$ . Each target label is represented by a binary vector  $\mathbf{y} \in \{0, 1\}^K$ . A sample belonging to the class  $k$  has all its elements set to 0 except for that representing the class, which is set to 1 at the  $k$ -th position. This codification defines the desired probability distribution for each target sample. Hence, maximizing this criterion is equivalent to optimizing the probability.

Next, let  $\hat{\mathbf{y}} \in \mathbb{R}^K = f(\mathbf{x}; \Theta)$  be the output produced by a model  $f$  with parameters  $\Theta$  whose last activation function is a softmax. Thus,  $\hat{\mathbf{y}}$  can be interpreted as the probability distribution provided by the model. Now, we can compare this probability distribution to the desired distribution, defined by the target sample ( $\mathbf{y}$ ).

By setting the label distribution  $\mathbf{y}$  as  $\text{Pr}_1$  and the model distribution  $\hat{\mathbf{y}}$  as  $\text{Pr}_2$ , we yield Eq. (2.7), the cross-entropy loss function:

$$\ell(\mathbf{y}, \hat{\mathbf{y}}) = - \sum_{k=1}^K y_k \log \hat{y}_k \quad (2.7)$$

where  $y_k$  and  $\hat{y}_k$  denote the  $k$ -th elements of the vectors  $\mathbf{y}$  and  $\hat{\mathbf{y}}$ , respectively.

## 2.2.2 Stochastic gradient descent

*Stochastic gradient descent* (SGD, Robbins and Monro, 1951) is the most common method used to estimate the parameters  $\Theta$  of the network. SGD is a particular case of the gradient descent optimization method, in which the training samples are chosen from a dataset made up of  $S$  observations:  $\mathcal{S} = \{(\mathbf{x}^{(s)}, \mathbf{y}^{(s)})\}_{s=1}^S$ , where each  $\mathbf{x}^{(s)}$  is an input sample and  $\mathbf{y}^{(s)}$  is the corresponding output sample. SGD tackles the problem of obtaining the parameters  $\hat{\Theta}$  that minimize the loss function  $\ell$  on the training corpus, as in Eq. (2.8):

$$\hat{\Theta} = \arg \min_{\Theta} \frac{1}{S} \sum_{s=1}^S \ell(\mathbf{y}^{(s)}, \hat{\mathbf{y}}^{(s)}) \quad (2.8)$$

where, as in the previous section,  $\hat{\mathbf{y}}^{(s)} = f(\mathbf{x}^{(s)}; \Theta)$  is the output computed by the network with parameters  $\Theta$  for the input sample  $\mathbf{x}^{(s)}$ .

SGD minimizes  $\ell$  by iteratively taking steps toward the negative gradient of the loss function with respect to the parameters of the current point. Therefore, let  $\Theta_n$  be the set of parameters estimated at the  $n$ -th training step. The initial parameters  $\Theta_0$  are initialized arbitrarily. SGD computes an update of the weights ( $\Delta\Theta_n$ ) which is applied to  $\Theta_n$ , as in Eq. (2.9):

$$\frac{nm}{den} \Theta_{n+1} = \Theta_n + \Delta\Theta_n \quad (2.9)$$

This update is defined as the minus gradient of the loss function with respect to the model parameters at current step ( $\mathbf{g}_n$ ), and it is usually modified by a factor  $\rho$ , called the *learning rate*, which controls the magnitude of the updates (Eq. (2.10)):

$$\Delta\Theta_n = -\rho \mathbf{g}_n \quad (2.10)$$

A gradient is the vector field of the partial derivatives of the elements of a vector. We explicitly write the gradient of the function  $\ell(\mathbf{y}^{(s)}, \hat{\mathbf{y}}^{(s)})$  with respect to the parameters  $\Theta_n$  as  $\nabla_{\Theta_n} \ell(\mathbf{y}^{(s)}, f(\mathbf{x}^{(s)}; \Theta_n))$ . This gradient is defined as:

$$\mathbf{g}_n = \nabla_{\Theta_n} \ell(\mathbf{y}^{(s)}, f(\mathbf{x}^{(s)}; \Theta_n)) = \left( \frac{\partial \ell(\mathbf{y}^{(s)}, f(\mathbf{x}^{(s)}; \Theta_n))}{\partial \theta_1}, \dots, \frac{\partial \ell(\mathbf{y}^{(s)}, f(\mathbf{x}^{(s)}; \Theta_n))}{\partial \theta_W} \right)$$

where  $\theta_1, \dots, \theta_W$  are the  $W$  elements composing  $\Theta_n$ .

SGD computes the gradients for each individual observation from the training set. This can be extended to the so-called mini-batch SGD, which consists of applying the

gradient descent method to mini-batches of samples. A mini-batch is a small subset of samples. We can obtain an unbiased estimate of the gradient by taking the average gradient of a mini-batch of samples drawn independent and identically distributed from the training dataset (Goodfellow et al., 2016). This allows the method to provide a more accurate estimate of the gradient of the training set than pure SGD. Moreover, the computations for all samples in the mini-batch can be parallelized and exploited by specialized multi-core hardware, such as GPUs (see Section 2.2.6).

### 2.2.3 Alternative update rules for SGD

The SGD method presented above presents some practical shortcomings that make its application difficult:

1. It can be a very slow process, even using mini-batch SGD and specialized hardware.
2. The choice of the learning rate is critical.
3. SGD finds local minima and it is important to initialize the network parameters to appropriate values.

These issues have been thoroughly studied in the literature, since they represent important bottlenecks of the application of SGD. The first two problems are commonly addressed by extending the plain SGD algorithm, usually via alternative update rules. The latter problem involves the development of adequate initialization strategies of the weights (see Section 2.2.4).

Among the plain SGD extensions, we find the inclusion of a momentum term (Polyak, 1964; Nesterov, 1983; Rumelhart et al., 1986) and adaptive algorithms that aim to compute a parameter-dependent learning rate. Adagrad (Duchi et al., 2011), Adadelta (Zeiler, 2012) or Adam (Kingma and Ba, 2014) are very popular update rules, that accelerate the training convergence. Table 2.2 provides a compact recap of these alternative updates.

### 2.2.4 Parameter initialization

The initialization of neural networks has an important impact on the training process: different initial points can make a model converge, affect its convergence speed and reach better generalization capabilities. Nowadays, the initialization strategies are mostly heuristic.

The most popular initialization strategy for the hidden layers draws values from a uniform probability distribution (Glorot and Bengio, 2010). We will use this strategy

**Table 2.2:** Alternative update rules proposed in the literature. The update ( $\Delta\Theta_n$ ) is the modification applied to the weights on each training step. The auxiliary column denotes additional computations and accumulators used by the different methods.  $\epsilon$  denotes a small value, to ensure numerical stability.

Optimizer	Update ( $\Delta\Theta_n$ )	Auxiliary
SGD (Robbins and Monro, 1951)	$-\rho\mathbf{g}_n$	–
SGD with momentum (Rumelhart et al., 1986)	$\mu\mathbf{v}_{n-1} - \rho\mathbf{g}_n$	$\mathbf{v}_{n-1} = \Delta\Theta_{n-1}$
Adadelta (Zeiler, 2012)	$-\rho\frac{\sqrt{\mathbf{d}_{n-1}+\epsilon}}{\sqrt{\mathbf{v}_n+\epsilon}} \odot \mathbf{g}_n$	$\mathbf{v}_n = \beta\mathbf{v}_{n-1} + (1-\beta)\mathbf{g}_n^2$ $\mathbf{d}_n = \beta\mathbf{d}_{n-1} + (1-\beta)\Delta\Theta_n^2$
Adam (Kingma and Ba, 2014)	$-\rho\frac{\frac{\mathbf{m}_n}{1-\beta_1^n}}{\sqrt{\frac{\mathbf{v}_n}{1-\beta_2^n}+\epsilon}}$	$\mathbf{m}_n = \beta_1\mathbf{m}_{n-1} + (1-\beta_1)\mathbf{g}_n$ $\mathbf{v}_n = \beta_2\mathbf{v}_{n-1} + (1-\beta_2)\mathbf{g}_n^2$

to initialize our weights in Chapters 3 and 6. The recurrent weights of RNNs (Section 2.3.1) are initialized to random orthogonal matrices, which makes the learning speed invariant to the depth of the model (Saxe et al., 2013).

### 2.2.5 The back-propagation algorithm

The most common method for obtaining the gradients used by SGD is the back-propagation algorithm (Rumelhart et al., 1986). The main idea is to compute the gradients of the loss function in the output layer of a model. Next, by repeatedly applying the chain rule of calculus, the error made by the model is propagated to the lower layers, through the gradient of the loss function with respect to these lower layers. The algorithm has two main stages:

**Forward-pass:** A training sample  $\mathbf{x}$  is introduced to the network and passed through all its layers, obtaining a predicted output  $\hat{\mathbf{y}}$ . This prediction is compared to the desired output  $\mathbf{y}$ , by means of the loss function  $\ell(\mathbf{y}, \hat{\mathbf{y}})$ .

**Backward-pass:** Starting from the output layer, the gradients of the loss with respect to the parameters of each layer are computed. Applying the chain rule, the gradients are propagated to lower level layers, until reaching the first one.

### 2.2.6 Defining neural networks with computational graphs

In practice, the implementation of neural networks can be done using a computational graph. Most of the frameworks used to deploy modern neural models rely on the construction of these computational graphs. The graphs define a set of nodes, which indicate variables. The variables represent data (e.g. vectors, matrices, tensors). The framework also defines operations, as functions between variables. Therefore, if a given variable is the output of an operation performed on another variable, an edge is created, connecting both variables. Once a graph has been declared, the variables are replaced by the numeric values of the task at hand.

Computational graphs allow us to efficiently apply automatic differentiation techniques (see e.g. [Baydin et al., 2018](#)) to obtain the gradients of the network. This eases and speeds up the development and deployment of complex models. Most frameworks allow the automatic differentiation of neural networks. These frameworks can be classified into two major programming paradigms: declarative and imperative. Under the first paradigm, the user declares *what* must be done; while in the second one, the user specifies *how* the computations will be executed.

In declarative frameworks, the user first defines a symbolic computation graph. This graph is interpreted by the framework during execution. The graph can be efficiently differentiated via symbolic differentiation ([Baydin et al., 2018](#)). This strategy allows us to apply optimization techniques to the computation graph, which speed the computations. Among the most popular frameworks, Theano ([Bergstra et al., 2010](#); [Theano Development Team, 2016](#)) pioneered the exploitation of this approach. Theano is a Python library which allows us to declare computational graphs, which are compiled to generate GPU-optimized code. Tensorflow ([Abadi et al., 2016](#)) was designed following the spirit of Theano, with the objective of deploying of large-scale systems.

The imperative paradigm is gaining increasing attraction. Since under this paradigm the user provides to the framework the exact computations to perform, this allows for more intuitive programs. Among the most popular tools, we find (py)Torch ([Collobert et al., 2011](#); [Paszke et al., 2017](#)).

## 2.3 Neural network architectures

In addition to the already described MLP, a large number of neural network architectures have been designed. A given architecture becomes more adequate than another depending on the task to tackle. In this section, we review the main architectures, in which the systems built in this thesis are based. We describe recurrent and convolutional neural networks, with proven capabilities for sequence modeling and image

analysis. Moreover, we also describe the mechanism used to map a discrete space into a continuous one: the word embeddings.

### 2.3.1 Recurrent neural networks

Recurrent neural networks feature connections forming a directed cycle. This creates a recurrent hidden state of size  $h$  ( $\mathbf{h} \in \mathbb{R}^h$ ) that allows the network to effectively model discrete-temporal sequences. RNNs process a sequence of  $T$   $m$ -dimensional input vectors  $\mathbf{x}_1^T = \mathbf{x}_1, \dots, \mathbf{x}_T$ , with each  $\mathbf{x} \in \mathbb{R}^m$ , to obtain an output sequence of  $r$ -dimensional vectors  $\mathbf{y}_1^T = \mathbf{y}_1, \dots, \mathbf{y}_T$ , with each  $\mathbf{y} \in \mathbb{R}^r$ . To do this, different RNN architectures have been proposed in the literature (Elman, 1990; Jordan, 1990; Hochreiter and Schmidhuber, 1997; Cho et al., 2014). The simplest RNN architecture is defined by Eq. (2.11):

$$\begin{aligned} \mathbf{h}_t &= g(\mathbf{U}\mathbf{x}_t + \mathbf{W}\mathbf{h}_{t-1} + \mathbf{b}) \\ \mathbf{y}_t &= \mathbf{h}_t \end{aligned} \tag{2.11}$$

where  $\mathbf{U} \in \mathbb{R}^{h \times m}$  and  $\mathbf{W} \in \mathbb{R}^{h \times h}$  are the input-to-hidden and hidden-to-hidden weight matrices to estimate, while  $\mathbf{b} \in \mathbb{R}^h$  is the bias term. As in the regular MLP (Eq. (2.4)),  $g$  is an activation function applied element-wise. Note that in this architecture, the output produced by the RNN is the hidden state itself. Hence,  $h \equiv r$ . Unless specified, in the rest of this document, we will employ RNN architectures that compute its outputs as their hidden states.

### Training RNNs

To train RNNs, it is usually employed an extension of the back-propagation algorithm, called back-propagation-through-time (Rumelhart et al., 1986; Werbos, 1990; Williams and Peng, 1990). The idea is to unfold the network over time: an unfolded RNN can be seen as a feedforward network with as many layers as unfolding steps taken. The back-propagation-through-time algorithm applies standard back-propagation (Section 2.2.5) to this unfolded version of the RNN.

Depending on the unfolding strategy, the algorithm has several variants (Williams and Peng, 1990). The most popular one—and that used in this thesis—performs the forward pass through all the elements in the sequence, accumulating inputs, states and target vectors during this interval. Next, a single backward pass is performed with the accumulated values. Since only a single forward and backward pass are accumulated, the complexity of this method is linear with respect to the length of the sequence.

In addition to back-propagation-through-time, alternative algorithms for training RNNs have been proposed. Autoregressive RNNs can be trained using the “teacher forcing scheme” (Williams and Zipser, 1989), in which the ground truth samples are fed back into the model. Originally, the technique was described for RNNs having output-to-hidden recurrence and it was an alternative to the back-propagation-through-time method. However, in models with output-to-hidden recurrence and hidden-to-hidden recurrence, teacher forcing can be combined with back-propagation-through-time.

### Weaknesses of simple RNN

Despite being powerful sequence modelers, RNNs have two main drawbacks:

1. The input sequence is only scanned in one direction, usually from left to right (or from the past to the future, if we consider temporal sequences). Hence, information flowing from the right to the left is not considered. In order to capture both left and right contexts, Schuster and Paliwal (1997) proposed the bidirectional RNNs (Section 2.3.1).
2. The simple RNN architecture suffers from the vanishing gradient problem (Bengio et al., 1994) (or its exploding version), which makes the network training difficult when modeling long-term relationships (see Section 2.3.1).

### Bidirectional recurrent neural networks

A *bidirectional recurrent neural network* (BRNN) consists of two independent recurrent layers: the so-called forward layer, which processes the input sequence in the regular direction (from 1 to  $T$ ); and the backward layer, which processes the sequence reversed in time (from  $T$  to 1), as in Eq. (2.12):

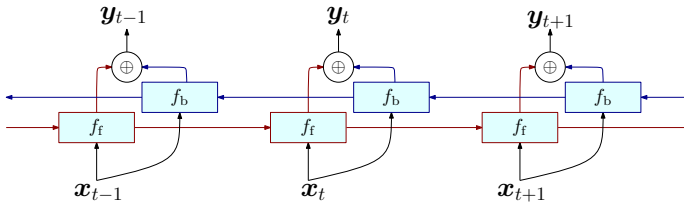
$$\begin{aligned} \mathbf{h}_t^f &= f_f(\mathbf{x}_t, \mathbf{h}_{t-1}^f) \\ \mathbf{h}_t^b &= f_b(\mathbf{x}_t, \mathbf{h}_{t+1}^b) \end{aligned} \tag{2.12}$$

where  $\mathbf{h}_t^f$  and  $\mathbf{h}_t^b$  denote the forward and backward hidden states, respectively;  $f_f$  and  $f_b$  refer to the unidirectional RNN function (e.g. Eq. (2.11) or Section 2.3.1), for forward and backward layers, respectively.

These recurrent layers are combined according to a combination operator  $\oplus$ , to obtain an output  $\mathbf{y}_t$ , with information from both directions, following Eq. (2.13):

$$\mathbf{y}_t = \mathbf{h}_t^f \oplus \mathbf{h}_t^b \tag{2.13}$$

The most common combination strategy is to concatenate forward and backward states, although others can also be used (e.g. summation or averaging). Since the recurrent layers have no interaction between them, bidirectional RNNs can be trained using similar algorithms as those used for unidirectional RNNs. Fig. 2.2 shows a BRNN, unfolded in time.



**Figure 2.2:** Bidirectional recurrent neural network, unfolded in time. It consists of two independent recurrent layers, one of them analyzing the input sequence forward in time ( $f^f$ ) and the other one ( $f^b$ ), analyzing it backward. Each RNN has independent states, which are merged through with the combination operator ( $\oplus$ ). The output of the network is the result of this combination of inner layers.

## Vanishing and exploding gradients

One of the biggest difficulties of the training of very deep neural networks are the vanishing and exploding gradient problems (Hochreiter, 1991; Bengio et al., 1994). They refer to the accumulation of gradients of these deep networks during an update. Note that, since the back-propagation-through-time unfolds the RNN for the complete sequence, RNNs are greatly affected by these problems. The vanishing gradient problem refers to the situation in which the norm of the gradients is close to 0: in this case, as the gradients are propagated (multiplied) to previous time-steps, they diminish exponentially fast to norm 0. Hence, the model is unable to learn the long-term dependencies from the sequence. Analogously, if the norm of the propagated gradients is large, they grow exponentially, causing numerical instability and learning issues. This is known as the exploding gradient problem.

The typical sigmoid activation functions ( $\sigma$ ,  $\tanh$ ) are prone to suffer from this problem: their derivatives are bell-shaped and their values are close zero. Other activation functions like ReLU present fewer vanishing gradient issues and are commonly used in deep (non-recurrent) neural architectures (e.g. Krizhevsky et al., 2012; Szegedy et al., 2015).

In the case of RNNs, alternative recurrent functions have been developed. They can be seen as complex activation functions, taking the form of gated cells. These gates decide the amount of information that flows through the cell. By construction, these activation functions have a derivative of 1, and the vanishing and exploding



gradient problems are (partially) alleviated. These units are necessary to successfully model distant dependencies.

In this thesis, we employ the popular *long short-term memory* (LSTM, Hochreiter and Schmidhuber, 1997) units, for all our recurrent models, which is described below. In addition to LSTMs, it is worth also mentioning the gated recurrent unit (GRU, Cho et al., 2014), which can be seen as a simplified version of LSTM units, and it is a very common alternative to LSTMs.

To address the exploding gradients problem, in addition to the aforementioned functions, other solutions involve clipping the norm of the gradients to a maximum predefined value during training and applying weight regularization (Section 2.4, Pascanu et al., 2013).

### LSTM: Long short-term memory units

LSTM units arguably are the most popular recurrent gated architecture. Since their inception (Hochreiter and Schmidhuber, 1997), subsequent works introduced alternative variants (see e.g., Greff et al., 2017). In this thesis, we use the LSTM variant introduced by Gers et al. (2000), which is described below. Fig. 2.3 shows an illustration of this LSTM cell.

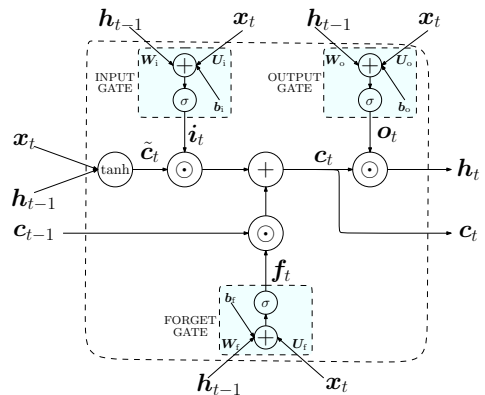
In addition to the hidden state,  $\mathbf{h}_t \in \mathbb{R}^h$ , LSTM networks maintain an additional state, called memory:  $\mathbf{c}_t \in \mathbb{R}^h$ . The hidden state is computed as the memory state, regulated by an output gate  $\mathbf{o}_t \in \mathbb{R}^h$ , as in Eq. (2.14):

$$\mathbf{h}_t = \mathbf{o}_t \odot \mathbf{c}_t \quad (2.14)$$

where  $\odot$  denotes the element-wise multiplication.

The memory state is computed as the addition of the memory state of the previous time-step, modulated by a forget gate  $\mathbf{f}_t \in \mathbb{R}^h$ , plus an updated memory state for the current time-step ( $\tilde{\mathbf{c}}_t \in \mathbb{R}^h$ ), modulated by an input gate  $\mathbf{i}_t \in \mathbb{R}^h$ , as in Eq. (2.15):

$$\mathbf{c}_t = \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \tilde{\mathbf{c}}_t \quad (2.15)$$



**Figure 2.3:** LSTM cell. The output depends on the previous hidden and memory states ( $\mathbf{h}_{t-1}$ ,  $\mathbf{c}_{t-1}$ ) and the current input  $\mathbf{x}_t$ . Input, output and forget gates modulate the information that flows through the unit.

The updated memory state is computed as in Eq. (2.16):

$$\tilde{\mathbf{c}}_t = \tanh(\mathbf{U}_c \mathbf{x}_t + \mathbf{W}_c \mathbf{h}_t + \mathbf{b}_c) \quad (2.16)$$

where  $\mathbf{W}_c \in \mathbb{R}^{h \times h}$ ,  $\mathbf{U}_c \in \mathbb{R}^{h \times m}$  and  $\mathbf{b}_c \in \mathbb{R}^h$  are the weight matrices for the previous hidden state, input of the layer and the bias term, respectively.

The input, forget and output gates are computed according to Eq. (2.17):

$$\begin{aligned} \mathbf{f}_t &= \sigma(\mathbf{U}_f \mathbf{x}_t + \mathbf{W}_f \mathbf{h}_t + \mathbf{b}_f) \\ \mathbf{i}_t &= \sigma(\mathbf{U}_i \mathbf{x}_t + \mathbf{W}_i \mathbf{h}_t + \mathbf{b}_i) \\ \mathbf{o}_t &= \sigma(\mathbf{U}_o \mathbf{x}_t + \mathbf{W}_o \mathbf{h}_t + \mathbf{b}_o) \end{aligned} \quad (2.17)$$

where  $\mathbf{W}_f, \mathbf{W}_i, \mathbf{W}_o \in \mathbb{R}^{h \times h}$ ,  $\mathbf{U}_f, \mathbf{U}_i, \mathbf{U}_o \in \mathbb{R}^{h \times m}$  and  $\mathbf{b}_f, \mathbf{b}_i, \mathbf{b}_o \in \mathbb{R}^h$  are the parameters to estimate.

### 2.3.2 Attention mechanisms

*Attention mechanisms* are essential components of modern neural architectures, especially relevant for sequence-to-sequence encoder–decoder models, as the ones used for NMT (see next chapter). They were motivated by the fact that encoder–decoder approaches based on RNNs (Cho et al., 2014; Sutskever et al., 2014) used the last hidden state of the encoder as the input of the decoder, i.e. a fixed-size representation of the input sequence, which had to contain the whole information from the input sequence. To encapsulate this information from a long sequence into a fixed-size representation required large models, which were hard to train, and even so, the performance dropped when dealing with long sequences. Attention mechanisms constituted an effective method to deal with such long sequences: they computed a dynamic representation of the input sequence at each decoding time-step, based on the current state of decoding. This allowed the model to *focus* on the most useful parts of the input sequence, depending on the current decoding state.

Generalizing this concept, an attention mechanism is a function that computes a contextual representation of a complex input object, in our case, a sequence of  $T$  elements. The final representation is conditioned by a state vector, called *query*, and it is used to weight the input sequence. Both objects are related by means of an attention function ( $a$ ), that computes a compatibility score for each one of the elements from the input sequence, with respect to the query. These scores are then used to compute a weighted average of the  $T$  elements of the sequence. This final representation is called the *context vector* ( $\mathbf{z}$ ). Under the scope of attention mechanisms, the object of interest (the input sequence) is represented by  $T$  vector pairs, called *keys* and *values*.

The keys are used as arguments of the function  $a$ , that obtains the attention scores. These scores will weight the values, yielding the desired context vector.

More formally, let  $\mathbf{q} \in \mathbb{R}^q$  be a query vector of size  $q$ , and  $\mathbf{k}_1^T$  and  $\mathbf{v}_1^T$  be the  $T$  key-value vectors representing the input sequence, of dimensions  $k$  and  $v$ , respectively. An attention mechanism can be seen as a two-step procedure. First, we apply an attention function  $a$  to each one of the elements from the set of keys, to score it with respect to the query. This provides a vector of  $T$  scores, which is normalized via the softmax function ( $\varphi$ ), yielding a vector of attention weights. In the second step, the attention weights are used to compute the context vector  $\mathbf{z}$ , weighting the values, as in Eq. (2.18):

$$\mathbf{z} = \varphi(a(\mathbf{q}, \mathbf{k}_1^T))\mathbf{v}_1^T \quad (2.18)$$

### Attention functions

Alternative attention mechanisms have been developed, basically by modifying the attention function  $a$ . The original function, proposed by Bahdanau et al. (2015), was a single-layered MLP of size  $d_a$ . This function is applied element-wise to the  $T$  vectors from the key matrix  $\mathbf{k}_t \in \mathbb{R}^k, \forall \mathbf{k}_t \in \mathbf{k}_1^T$ , as in Eq. (2.19):

$$a(\mathbf{q}, \mathbf{k}_1^T) = \mathbf{v}^\top \tanh(\mathbf{W}\mathbf{q} + \mathbf{U}\mathbf{k}_t + \mathbf{b}) \quad (2.19)$$

where  $\mathbf{v} \in \mathbb{R}^{d_a}$ ,  $\mathbf{W} \in \mathbb{R}^{d_a \times q}$ ,  $\mathbf{U} \in \mathbb{R}^{d_a \times k}$  and  $\mathbf{b} \in \mathbb{R}^{d_a}$  are the weights to estimate.

Among the alternative attention functions proposed, it is worth highlighting the “dot-product” attention function (Luong et al., 2015a). It is also applied element-wise to  $\mathbf{K}, \forall \mathbf{k}_t \in \mathbf{k}_1^T$ , as in Eq. (2.20):

$$a(\mathbf{q}, \mathbf{k}_1^T) = \mathbf{q}\mathbf{k}_t^\top \quad (2.20)$$

This alignment function is computationally cheaper than the one proposed by Eq. (2.19), but it requires the same dimensions of queries and keys ( $q$  and  $k$ ). An issue suffered by this function is that, when using high-dimensional vectors, the magnitude of this dot product is increased. This causes the gradient of the softmax function to be extremely low (Vaswani et al., 2017). Therefore, this attention function performs worse with high-dimensional vectors than the additive one (Britz et al., 2017). In order to make this model more invariant to such dimensional issues, Vaswani et al. (2017) proposed the “scaled dot product” attention function (Eq. (2.21)), again  $\forall \mathbf{k}_t \in \mathbf{k}_1^T$ :

$$a(\mathbf{q}, \mathbf{k}_1^T) = \frac{\mathbf{q}\mathbf{k}_1^T}{\sqrt{q}}, \forall \mathbf{k}_1 \in \mathbf{k}_1^T \quad (2.21)$$

where the division by  $\sqrt{q}$  is applied element-wise.

### Recurrent units with attention

A common usage of attention mechanisms is to embed them in the decoder of RNN-based encoder–decoder models (as those described in [Chapters 3](#) and [6](#)). The attention mechanism serves as a bridge between the encoder and the decoder RNNs. An RNN with attention introduces the context vector computed by the attention mechanism ( $\mathbf{z}_t$ ) as an additional input, as in [Eq. \(2.22\)](#):

$$\begin{aligned} \mathbf{h}_t &= g(\mathbf{U}\mathbf{x}_t + \mathbf{W}\mathbf{h}_{t-1} + \mathbf{A}\mathbf{z}_t + \mathbf{b}) \\ \mathbf{y}_t &= \mathbf{h}_t \end{aligned} \quad (2.22)$$

where  $\mathbf{U} \in \mathbb{R}^{h \times m}$ ,  $\mathbf{W} \in \mathbb{R}^{h \times h}$ ,  $\mathbf{A} \in \mathbb{R}^{h \times z}$ ,  $\mathbf{b} \in \mathbb{R}^h$  are the weights to estimate.

When introduced in RNNs, the context vector is typically computed by applying the attention mechanism using as query the hidden state from the decoder at the previous time-step ( $\mathbf{h}_{t-1}$ ). The keys and values ( $\mathbf{k}_1^T$ ,  $\mathbf{v}_1^T$ ) come from the encoder, as in [Eq. \(2.23\)](#):

$$\mathbf{z}_t = \varphi(a(\mathbf{h}_{t-1}, \mathbf{k}_1^T))\mathbf{v}_1^T \quad (2.23)$$

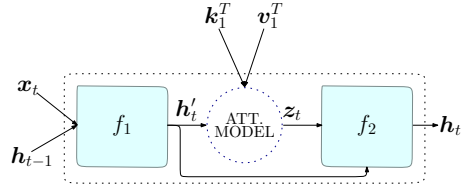
Analogously to simple RNNs, the attention mechanism can also be introduced to LSTM units following [Eq. \(2.24\)](#):

$$\begin{aligned} \mathbf{h}_t &= \mathbf{o}_t \odot \mathbf{c}_t \\ \mathbf{c}_t &= \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \tilde{\mathbf{c}}_t \\ \tilde{\mathbf{c}}_t &= \tanh(\mathbf{U}_c\mathbf{x}_t + \mathbf{W}_c\mathbf{h}_t + \mathbf{A}_c\mathbf{z}_t + \mathbf{b}_c) \\ \mathbf{f}_t &= \sigma(\mathbf{U}_f\mathbf{x}_t + \mathbf{W}_f\mathbf{h}_t + \mathbf{A}_f\mathbf{z}_t + \mathbf{b}_f) \\ \mathbf{i}_t &= \sigma(\mathbf{U}_i\mathbf{x}_t + \mathbf{W}_i\mathbf{h}_t + \mathbf{A}_i\mathbf{z}_t + \mathbf{b}_i) \\ \mathbf{o}_t &= \sigma(\mathbf{U}_o\mathbf{x}_t + \mathbf{W}_o\mathbf{h}_t + \mathbf{A}_o\mathbf{z}_t + \mathbf{b}_o) \end{aligned} \quad (2.24)$$

where  $\mathbf{U}_c, \mathbf{U}_f, \mathbf{U}_i, \mathbf{U}_o \in \mathbb{R}^{h \times m}$ ,  $\mathbf{W}_c, \mathbf{W}_f, \mathbf{W}_i, \mathbf{W}_o \in \mathbb{R}^{h \times h}$  and  $\mathbf{A}_c, \mathbf{A}_f, \mathbf{A}_i, \mathbf{A}_o \in \mathbb{R}^{h \times z}$ ,  $\mathbf{b}_c, \mathbf{b}_f, \mathbf{b}_i, \mathbf{b}_o \in \mathbb{R}^h$  are the weights to estimate. The context vector is obtained as in the previous case ([Eq. \(2.23\)](#)).

### Conditional recurrent units

A well-performing alternative to integrate RNNs with attention mechanisms is the so-called conditional units (Senrich et al., 2017; Peris and Casacuberta, 2019a). They consist of several RNN transition blocks, with an attention mechanism in between. Most recurrent decoders used in the following chapters of the thesis (Chapters 3 to 6) will follow this architecture. Fig. 2.4 shows an illustration of a conditional cell.



**Figure 2.4:** Conditional RNN unit. It consists of a stacked application of RNN units, with an attention mechanism in between.

The first block of a conditional RNN applies the corresponding recurrent function ( $f_1$ , typically a gated unit) to its input ( $\mathbf{x}_t$ ), computing an intermediate hidden state  $\mathbf{h}'_t$ , as in Eq. (2.25):

$$\mathbf{h}'_t = f_1(\mathbf{x}_t, \mathbf{h}_{t-1}) \quad (2.25)$$

This intermediate representation are the queries of an attention mechanism, which obtains the context vector  $\mathbf{z}_t$  similarly as in the previous section (Eq. (2.26)):

$$\mathbf{z}_t = \varphi(a(\mathbf{h}'_t, \mathbf{k}_1^T))\mathbf{v}_1^T \quad (2.26)$$

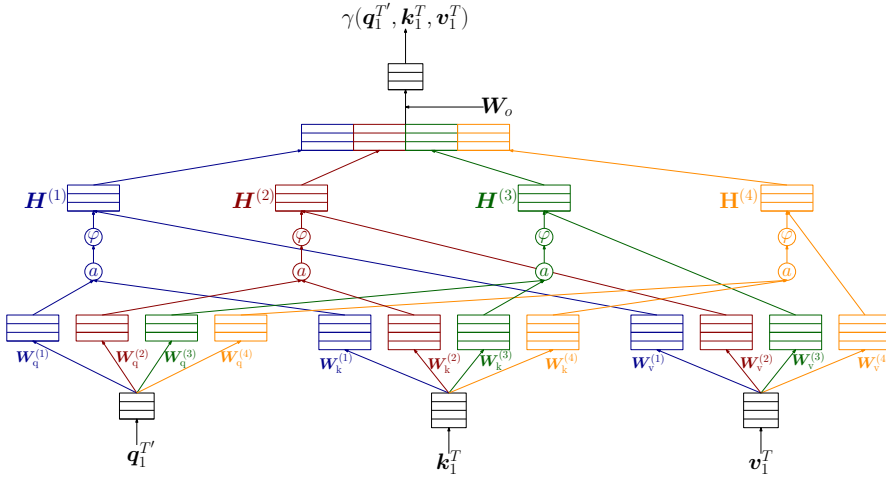
This context vector is the input to the second RNN block, which also takes into account the intermediate representation  $\mathbf{h}'_t$  and applies the recurrent function  $f_2$ , as in Eq. (2.27):

$$\mathbf{h}_t = f_2(\mathbf{z}_t, \mathbf{h}'_t) \quad (2.27)$$

### Multi-head attention

In addition to its usage in combination with RNNs, attention mechanisms can act as sequence modelers by themselves. They can account for inter-sequence (between two different sequences) and intra-sequence (between elements of the same sequence) relationships (Parikh et al., 2016; Lin et al., 2017). This latter case is called self-attention and it has been exploited by non-recurrent architectures for sequence-to-sequence learning (Vaswani et al., 2017). Under this scope, to make the attention mechanisms more expressive, Vaswani et al. (2017) introduced the so-called multi-

head attention. This is an extension of the regular attention mechanism, that allows to learn representations of different sub-spaces at different positions. Multi-head attention is the main component of the *Transformer* model (Vaswani et al., 2017), a common and well-performing sequence-to-sequence architecture, that will be used in following chapters (Chapters 3 to 6). Refer to Section 3.2 for a description of this architecture.



**Figure 2.5:** Multi-head attention mechanism. This example shows stacks of  $T = 4$ ,  $T' = 3$  elements and  $H = 4$  parallel heads. Elements belonging to each one of the heads share color. In a first stage, the elements from the input sequences  $\mathbf{q}_1^{T'}$ ,  $\mathbf{k}_1^T$  and  $\mathbf{v}_1^T$  are linearly projected  $H$  times. Next, each one of the projections pass through the attention mechanism, yielding  $H$  different heads. These heads are concatenated, obtaining a matrix of  $T$  rows and  $Hd_m$  columns. The result of the multi-head attention is computed as a linear projection of the concatenated heads.

Considering that multi-head attention is devised for non-recurrent architectures, there are no dependencies between the different query vectors. Therefore, the operations can be parallelized by stacking  $T'$  query vectors as a sequence of queries ( $\mathbf{q}_1^{T'}$ , with each  $\mathbf{q}_{t'} \in \mathbb{R}^q$ , for  $1 \leq t' \leq T'$ ).

The multi-head attention model, illustrated in Fig. 2.5, computes this attention in parallel, over  $H$  different, learned projections of size  $d_m$  of the queries, keys and values. These projections, for  $1 \leq h \leq H$  are computed following Eq. (2.28):

$$\begin{aligned}
 \bar{\mathbf{q}}_1^{T'}{}^{(h)} &= \mathbf{W}_q^{(h)} \mathbf{q}_{t'}, \text{ for } 1 \leq t' \leq T' \\
 \bar{\mathbf{k}}_1^T{}^{(h)} &= \mathbf{W}_k^{(h)} \mathbf{k}_t, \text{ for } 1 \leq t \leq T \\
 \bar{\mathbf{v}}_1^T{}^{(h)} &= \mathbf{W}_v^{(h)} \mathbf{v}_t, \text{ for } 1 \leq t \leq T
 \end{aligned} \tag{2.28}$$

where each  $\mathbf{W}_Q^{(h)} \in \mathbb{R}^{d_m \times q}$ ,  $\mathbf{W}_K^{(h)} \in \mathbb{R}^{d_m \times k}$  and  $\mathbf{W}_V^{(h)} \in \mathbb{R}^{d_m \times v}$  are the trainable matrices. After projecting queries, keys and values, the multi-head attention model applies an attention mechanism in parallel all the elements of these sequences, according to Eq. (2.29), which is analogous to Eq. (2.18):

$$\mathbf{H}_h = \varphi(a'(\bar{\mathbf{q}}_1^{T'(h)}, \bar{\mathbf{k}}_1^{T'(h)})\bar{\mathbf{v}}_1^{T'(h)}) \quad (2.29)$$

where  $a' : (T' \times q, T' \times k) \rightarrow T' \times T'$  denotes an attention function that is applied in parallel, computing a sequence of scores for each one of the elements of the sequence  $\bar{\mathbf{q}}_1^{T'(h)}$ . Following Eq. (2.18), these scores are normalized applying the softmax function ( $\varphi$ ) to each one of the vectors in the sequence, yielding a sequence of attention weights  $\alpha_1^{T'}$ , where each  $\alpha_{t'} \in \mathbb{R}^{T'}$ , for  $1 \leq t' \leq T'$ . These weights are used to weight the sequence of projected values. These attended representations  $\mathbf{H}_h \in \mathbb{R}^{T' \times d_m}$ , for  $1 \leq h \leq H$ , are called heads and the typical attention function used for multi-head attention is the scaled dot product (Eq. (2.21), Vaswani et al., 2017).

The heads are concatenated into a matrix ( $[\mathbf{H}_1; \dots; \mathbf{H}_H] \in \mathbb{R}^{T' \times H d_m}$ ) and projected into an output space of  $o$  dimensions by means of a trainable matrix  $\mathbf{W}_o \in \mathbb{R}^{H d_m \times o}$ . Therefore, the multi-head attention function is defined as a function  $\gamma : (\mathbb{R}^{T' \times q}, \mathbb{R}^{T' \times k}, \mathbb{R}^{T' \times v}) \rightarrow \mathbb{R}^{T' \times o}$ , as in Eq. (2.30):

$$\gamma(\mathbf{q}_1^{T'}, \mathbf{k}_1^{T'}, \mathbf{v}_1^{T'}) = [\mathbf{H}_1; \dots; \mathbf{H}_H] \mathbf{W}_o \quad (2.30)$$

where each head is computed as aforementioned described. Note that performing most of these operations can be done in parallel. Hence, the computation of multi-head attention is very fast on specialized hardware, such as GPUs.

### 2.3.3 Convolutional neural networks

ConvNets are a class of neural networks consisting of a stacked application of convolutional and pooling operations. ConvNets have a crucial importance in computer vision, excelling for object detection and classification. Since in Chapter 6 we tackle image-related tasks, it is convenient the processing of the images with ConvNets. The predecessors of modern ConvNets can be traced back to Fukushima (1980); LeCun et al. (1989b); Waibel et al. (1989). These ideas were further developed for document-level recognition, resulting in the *LeNet* architecture, which are the base of modern ConvNets (LeCun et al., 1998). These networks consist of a cascaded application of convolutional layers and pooling operations. A convolutional layer consists of a set of weights, also called kernels or filters, which are applied to its input to obtain a feature map.

More formally, let  $\mathbf{x}_1, \dots, \mathbf{x}_T$ , with each  $\mathbf{x} \in \mathbb{R}^m$ , be inputs vectors of a convolutional layer. The convolutional layer obtains a new feature  $c_t$  by applying the filter  $\mathbf{W} \in \mathbb{R}^{h \times m}$ , to windows of  $h$  input elements, called strides, as in Eq. (2.31):

$$c_t = g(\mathbf{W}\mathbf{x}_t^{t+h-1} + b) \quad (2.31)$$

where  $g$  is an activation function and  $b$  is a bias term. These filters are applied through the full input, producing a feature map  $\mathbf{c} = c_1, \dots, c_{T-h+1}$ . Multiple filters with different strides are typically applied in a convolutional layer, obtaining several feature maps ( $\mathbf{c}_1, \dots, \mathbf{c}_n$ ). Each feature map usually has its weights tied.

After this convolutional layer, it is usual to apply a pooling layer, which condenses the information from the feature maps into a more compact representation. Common pooling operators consist of taking the maximum or the mean value of the feature maps.

It is common to stack a number of convolution plus pooling layers, to compute different representations of the input object. The stack of convolutions is followed by a number of fully-connected layers (an MLP), which perform the task at hand, e.g. object classification (Krizhevsky et al., 2012) or text classification (Kim, 2014).

Moreover, since the ConvNets implicitly learn representations of the inputs, this can be leveraged for other tasks. ConvNets are very effective feature extractors. The representations computed by the fully-connected layers or by the last convolutional layer are rich, and can be effectively exploited by other models, for example to tackle image and video captioning, as done in Chapter 6.

### 2.3.4 Word embeddings

One of the critical ideas that allowed the successful application of neural models to NLP tasks was the development of word embeddings. A word embedding is a mapping from a discrete vocabulary space into a dense, real-valued vector of low dimensionality: they bridge the discrete space of words and the continuous representation handled by neural networks.

The origins of this idea can be tracked to Hinton (1986) and Elman (1990, 1991). In the field of MT, Casañ and Castaño (1999) proposed a neural translator with hand-crafted distributed representation of words. The full exploitation of distributed representations of words was achieved by one of the seminal works on neural language models (Bengio et al., 2003), who proposed to project each word to the continuous space and perform the probability estimation on this continuous space. Each word from the vocabulary was represented by a different row in a matrix. This matrix



was estimated together with the rest of parameters of the network, in an end-to-end training pipeline: the embeddings are a byproduct of the training process.

From here, research on word embeddings attracted a large interest (e.g. Mikolov et al., 2013a,b; Pennington et al., 2014), because they serve as basis for a wide variety of NLP tasks. In the last year, the development more robust word representations is one of the most popular research topics in the NLP community (Peters et al., 2018; Devlin et al., 2018).

## 2.4 Improving generalization in neural networks

The goal of machine learning models is to approximate a function (Section 2.2.2). The fitting of a model measures how well it approximates this target function. A properly-trained model, which approximates well the target function on the training data, should also be able to work well on unseen data. Regularization applied to machine learning refers to any method intended to lower the generalization error of a given model, but not necessarily its training error. Neural networks, and more specifically deep networks, are very expressive models, able to approximate complex functions. However, this capacity also makes them prone to memorize the training data, incurring into overfitting issues. When working with neural networks, to apply regularization techniques becomes mandatory, in order to reduce the overfitting of the model. In this thesis, we made use of the following regularization techniques:

- **Early stopping:** it is a training strategy used to find the point of the training process in which the model starts to overfit the training data. To this end, we compute the error on a validation dataset. Early stopping assumes that if the error made by the model in the development set does not improve after a given number of consecutive evaluations it will no longer improve, because the model is overfitting the training data. Therefore, we have reached our desired point. We will make use of early stopping when training all of our systems Section 3.5.1 and Chapter 6.
- **Weight decay:** The generalization capability of a model relies on a balance between its expressiveness and the amount of training data available: having a large number of training samples allows the usage of more expressive models (Vapnik and Chervonenkis, 1971; Bishop, 2006). In contrast, if there are few training samples, the model will tend to overfit the training data, obtaining a poor generalization error. The expressiveness of the network can be constrained by limiting the growth of the weights, unless necessary. This can be done by adding a regularization term to the training objective during the optimization process. Weight decay introduces the magnitude of some parameters of the

model into the objective function to minimize (Krogh and Hertz, 1992). Refer to [Section 3.5.1](#) for details on the application of weight decay in our systems.

- **Noise injection:** Following the spirit of weight decay, an alternative regularizing method consists of injecting some noise (Bishop, 1995). This noise can be applied to the network components (weights and activations), to the input objects and to the outputs of the network. In this thesis, we will apply Gaussian additive noise during training. This technique adds noise drawn from a zero-centered Gaussian distribution to the network parameters during the training phase.
- **Label smoothing** When training models, we also apply label smoothing (Szegedy et al., 2016) to add some noise in the target labels. This makes the model to be less confident on the training data. In our sequence-to-sequence systems, the target samples are codified as binary vectors, with all elements set to zero but the one indicating the label of the sample set to one. Label smoothing subtracts a small value from the label and distributes it among the rest of the vector elements.
- **Dropout:** Dropout (Srivastava et al., 2014) is a regularization strategy that randomly masks some units during training, with a given probability  $p$ . Dropout reduces overfitting, as the method prevents the collaboration between weights to memorize training samples. Dropout is implemented as a masking layer, sampled at every step from a Bernoulli distribution with probability  $p$  during training time. At prediction time, all units are present, with their weights scaled down by a factor of  $p$ . Again, refer to [Section 3.5.1](#) and [Chapter 6](#) for details on the application of dropout in our systems.
- **Batch normalization:** in standard neural network architectures, the inputs of a given layer are conditioned by the outputs of all preceding layers. As the training is done via SGD with back-propagation, when the parameters of a layer are modified, the inputs of the following layers are also affected. As a network becomes deeper, the variations on the distributions that are input to the layers become larger. This effect is known as internal “covariate shift” (Ioffe and Szegedy, 2015) and makes the learning process of deep neural networks difficult and unstable. The idea of batch normalization is to reduce the internal covariate shift, by normalizing the outputs of a layer before being input to the following one (Ioffe and Szegedy, 2015). Therefore, the input distributions are more consistent, which makes the training process converge faster and be more robust to initialization and learning rate choices. This normalization is performed layer-wise, according to mini-batch statistics.
- **Layer normalization:** A generalization of batch normalization is layer normalization (Ba et al., 2016). Instead of computing the normalization statistics

across mini-batches, they are computed for the activations of a layer. Hence, the normalization is performed feature-wise. Hence, layer normalization is independent from the mini-batch size and it can be applied in online learning regimes, as those described in [Chapter 5](#).

## 2.5 Summary

In this chapter we introduced artificial neural networks and reviewed its main features. We addressed the parameter estimation problem via gradient descent, using the back-propagation algorithm to efficiently obtain the derivatives of the loss function with respect to the parameters of the network. We also enumerated several update rules, which aim to accelerate the training convergence.

We revisited the most relevant neural network architectures to sequence-to-sequence modeling. We put a special emphasis on RNNs and attention mechanisms, as these represent the core of the sequence-to-sequence models. We also introduced word embeddings and ConvNets. Finally, we also described several techniques that will help us to improve the generalization capabilities of our models.

All these techniques and neural architectures will be used in the following chapters to build sequence-to-sequence systems, that allow us to tackle the MT ([Chapter 3](#)) and the multimodal captioning ([Chapter 6](#)) problems.



## Chapter 3

# Neural machine translation

As we introduced in [Chapter 1](#), the ideas of tackling MT with neural networks date back to the second wave of neural networks ([Allen, 1987](#); [Chrisman, 1991](#); [Castaño and Casacuberta, 1997](#); [Forcada and Neco, 1997](#)). These pioneering works addressed the translation problem under constrained conditions, due to the computational limitations existing at that time. Although the results were promising, these early models suffered serious problems to tackle more realistic tasks: they did not escalate well, requiring large network sizes and prohibitive learning times.

These scaling issues were partially addressed by [Bengio et al. \(2003\)](#), as explained in [Section 2.3.4](#), who introduced word embeddings in a neural language model. From here, and backed by the advent of a higher computational capability, neural models were used in a wide variety of tasks, such as language modeling ([Schwenk, 2007](#); [Mikolov et al., 2010](#)), handwritten text recognition ([Graves et al., 2009](#)) or automatic speech recognition ([Graves et al., 2013](#)).

In the MT field, these neural models were initially introduced into the PB-SMT pipeline, as additional features of the log-linear model ([Devlin et al., 2014](#); [Sundermeyer et al., 2014](#)). But the major paradigm shift came from the introduction of NMT ([Kalchbrenner and Blunsom, 2013](#); [Cho et al., 2014](#); [Sutskever et al., 2014](#)). In NMT, translations are obtained solely by large neural networks, usually relying on an encoder-decoder framework: the encoder reads a source sentence and obtains a representation of it. Given this representation, the decoder generates the corresponding translation. The neural models are trained in an end-to-end way. This is opposed

to the PB-SMT approach, which is made up of multiple decoupled models, trained independently.

In this chapter we describe the main components of NMT technology: training objective, architectural choices, decoding process and word segmentation. To this end, we heavily rely on the concepts explained in [Chapter 2](#). Next, we describe the systems that are the base of the following chapters ([Chapters 4](#) and [5](#)) and present the performance of them in our MT tasks (see [Section 1.4.3](#)). For the sake of clarity, in the following sections, expressions relating to the source sequence are denoted by the index  $j$ , while those referring to the target sequence are indexed by  $i$ .

### Parameter estimation

NMT follows the probabilistic framework to address the translation problem. Recall from [Section 1.3](#) that the goal of SMT is to obtain, given a sentence  $x_1^J$  in the source language, its most likely translation  $\hat{y}_1^I$ , in the target language. This was defined by [Eq. \(1.5\)](#) as:

$$\hat{y}_1^I = \arg \max_{I, y_1^I} \Pr(y_1^I | x_1^J)$$

Applying the chain rule of the probability, this expression can be factorized as shown in [Eq. \(3.1\)](#)

$$\hat{y}_1^I = \arg \max_{I, y_1^I} \prod_{i=1}^I \Pr(y_i | y_1^{i-1}, x_1^J) \quad (3.1)$$

This conditional probability can be directly modeled by a neural model with parameters  $\Theta$ . Taking logarithms for the sake of numerical stability, we reach [Eq. \(3.2\)](#):

$$\hat{y}_1^I \approx \arg \max_{I, y_1^I} \sum_{i=1}^I \log p(y_i | y_1^{i-1}, x_1^J; \Theta) \quad (3.2)$$

The parameters  $\Theta$  are usually estimated on a parallel corpus  $\mathcal{S} = \{(x^{(s)}, y^{(s)})\}_{s=1}^S$ , consisting of  $S$  sentence pairs. The training objective is to find the set of parameters  $\hat{\Theta}$  that minimizes the minus log-likelihood on this training set, as in [Eq. \(3.3\)](#):

$$\hat{\Theta} = \arg \min_{\Theta} \sum_{s=1}^S \sum_{i=1}^{I^{(s)}} -\log p(y_i^{(s)} | y_1^{i-1(s)}, x^{(s)}; \Theta) \quad (3.3)$$

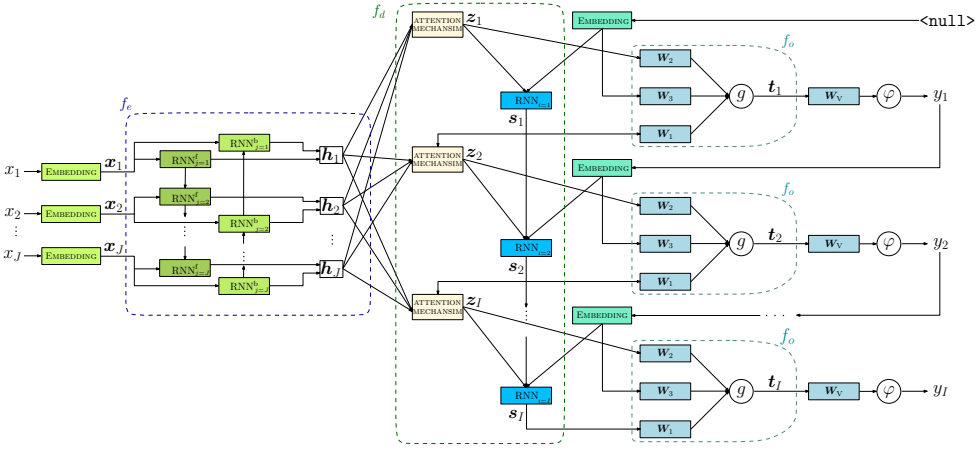
where  $I^{(s)}$  is the length of the  $s$ -th target sentence and  $y_1^{i-1(s)}$  denotes the  $s$ -th target sentence up to the position  $i - 1$ . To minimize this cost function is equivalent to maximize the likelihood. As discussed in [Section 2.2.1](#), this is achieved by neural networks using the cross-entropy loss function ([Eq. \(2.7\)](#)). This is, by far, the most common NMT training procedure, although alternative estimation methods have been proposed, with the goal of directly optimizing translation metrics ([Shen et al., 2016](#)) or regularizing this maximum likelihood objective ([Edunov et al., 2018](#)).

The optimization process is carried out by gradient descent ([Section 2.2.2](#)). Since in [Eq. \(3.1\)](#) we have a recurrence on the outputs, it is usual to follow the teacher forcing training scheme. This is done at training time by introducing the ground truth samples as an additional input of the model, but shifted one time-step to the right.

### 3.1 RNN-based NMT

MT is a paradigmatic example of a sequence-to-sequence problem. RNNs ([Section 2.3.1](#)) are devised to model sequences, therefore, the first well-performing NMT systems were based on RNN encoder–decoder models ([Cho et al., 2014](#); [Sutskever et al., 2014](#)). The addition of attention mechanisms into the NMT model ([Bahdanau et al., 2015](#)) allowed the system to focus on different parts of the input sentence, providing better performance when dealing with long sequences.

This attention-based RNN system has been the basis of many following works, that aimed to improve the attention mechanism ([Luong et al., 2015a](#)), the recurrent units ([Tu et al., 2017](#)) or increasing the depth of the model ([Wu et al., 2016](#)). Despite all these proposals, the original attention-based RNN remained without dramatic changes as the predominant model for NMT, until the advent of the Transformer model ([Vaswani et al., 2017](#), see [Section 3.2](#)). In the following sections, we will review the main components of this attention RNN-based NMT model. The full picture of RNN-based NMT is shown in [Fig. 3.1](#).



**Figure 3.1:** RNN-based encoder–decoder with attention.  $x_1, \dots, x_J$  is the sequence of source words, projected into the continuous space by means of an embedding matrix. This sequence of embeddings are processed by an encoder  $f_e$ : a bidirectional RNN, yielding a sequence of annotations ( $\mathbf{h}_1^J$ ). This sequence is the input of the decoder RNN  $f_d$ , consisting of a decoder RNN with an attention mechanism, followed by a deep output function ( $f_o$ ), and a fully-connected output layer ( $\mathbf{W}_V$ ). Finally, the softmax function ( $\varphi$ ) is used to obtain the probabilities of the target words.

**Encoder** The input of the system is a sequence of tokens  $x_1^J = x_1, \dots, x_J$ , each of them belonging to a finite vocabulary  $\mathcal{X}$ . Each element is codified with a unique index, from 1 to  $|\mathcal{X}|$ . Then, each word  $x_j$  is projected into a continuous space, by means of an embedding matrix, as in Eq. (3.4):

$$\mathbf{x}_j = \mathbf{E}_s(x_j) \quad (3.4)$$

where  $\mathbf{E}_s \in \mathbb{R}^{e \times |\mathcal{X}|}$  is the embedding matrix of the source language,  $e$  is the embedding size and  $\mathbf{E}_s(x_j)$  denotes the row of the embedding matrix corresponding to the element  $x_j$ .

The sequence of embeddings  $\mathbf{x}_1, \dots, \mathbf{x}_J$  is processed by an encoder RNN ( $f_e$ ), usually with LSTM (Section 2.3.1) or GRU cells. Since we have access to the complete input sequence, this encoder RNN is typically bidirectional (Section 2.3.1) The combination function of forward and backward layers is usually the concatenation of their hidden states. Therefore, from the sequence of embeddings is obtained a sequence of states, which model the dependencies across the sequence. These states are called annotations, computed as in Eq. (3.5):



$$\mathbf{h}_1^J = f_e(\mathbf{x}_1^J) \quad (3.5)$$

where  $\mathbf{h}_1^J$  is a sequence of  $J$  annotations, in which each element  $\mathbf{h}_j \in \mathbf{h}_1^J$ ,  $1 \leq j \leq J$ , can be seen as a representation of size  $k$  of the elements around the position  $j$  of the source sequence. This encoder RNN can be made of several stacked layers (Wu et al., 2016; Barone et al., 2017). If the encoder is a deep network,  $\mathbf{h}_1^J$  is made of the hidden states from the top layer in the stack.

**Decoder** The decoder consists of an RNN with attention mechanism followed by a deep output layer. It models the conditional translation probability following the factorization from Eq. (3.1). This can be done by performing the recurrence over the sequence of previously generated tokens, providing the RNN of autoregressive capabilities. These tokens are introduced to the decoder RNN via their embedding, following Eq. (3.4). As in the encoder, it is common to use (deep) LSTM or GRU architectures; or their conditional alternative (Section 2.3.2). The attention mechanism is applied as explained in Section 2.3.2.

This attention mechanism bridges the sequence of annotations computed by the encoder together with the hidden state of the decoder RNN. At each decoding step  $i$ , the attention mechanism computes a context vector  $\mathbf{z}_i$  as described in Eq. (2.18):

$$\mathbf{z}_i = \varphi(a(\mathbf{s}_{i-1}, \mathbf{h}_1^J))\mathbf{h}_1^J$$

where  $\varphi(a(\mathbf{s}_{i-1}, \mathbf{h}_1^J))$  computes the attention weights of the annotations at the  $i$ -th decoding step. The attention function  $a$  is usually the additive attention (Eq. (2.19)) or the dot-product attention (Eq. (2.20)).

Therefore, at the  $i$ -th decoding time-step, the decoder computes a hidden state  $\mathbf{s}_i$  considering the context vector ( $\mathbf{z}_i$ ) computed by the attention mechanism, the word embedding of the previously generated token ( $\mathbf{E}_t(y_{i-1})$ ) and the previous hidden state of the decoder  $\mathbf{s}_{i-1}$ , following Eq. (3.6):

$$\mathbf{s}_i = f_d(\mathbf{E}_t(y_{i-1}), \mathbf{s}_{i-1}, \mathbf{z}_i) \quad (3.6)$$

where  $f_d$  is the recurrent function with attention (e.g. Eq. (2.24)),  $\mathbf{s}_i \in \mathbb{R}^q$  is the hidden state of the decoder RNN, of dimension  $q$ ,  $\mathbf{E}_t \in \mathbb{R}^{d \times |\mathcal{Y}|}$  is the embedding matrix of the target language,  $\mathcal{Y}$  being the finite target vocabulary and  $d$  the dimension of the target word embedding.

The first state of the decoder is usually initialized according to some information from the encoder, according to a function  $f_i$ , as in Eq. (3.7):

$$\mathbf{s}_0 = f_i(\mathbf{h}_1^J) \quad (3.7)$$

Popular initialization strategies define  $f_i$  as an MLP, with an average representation of the annotations (Xu et al., 2015; Sennrich et al., 2017) or the last state of the backward encoder RNN (Bahdanau et al., 2015) as input.

The output state of the decoder RNN ( $\mathbf{s}_i$ ) is combined together with the context vector  $\mathbf{z}_i$  computed by the attention mechanism and the embedding of the previously generated word  $\mathbf{E}_t(y_{i-1})$  in a deep output layer (Pascanu et al., 2014), which applies the function  $f_o$  to obtain an  $l$ -sized intermediate representation  $\mathbf{t}_i \in \mathbb{R}^l$ , as in Eq. (3.8):

$$\mathbf{t}_i = f_o(\mathbf{s}_i, \mathbf{z}_i, \mathbf{E}_t(y_{i-1})) \quad (3.8)$$

$f_o$  applies the non-linear function  $g$  (typically tanh) applied to a combination of linear projections of its inputs, following Eq. (3.9):

$$f_o(\mathbf{s}_i, \mathbf{z}_i, \mathbf{E}(y_{i-1})) = g(\mathbf{W}_1 \mathbf{s}_i + \mathbf{W}_2 \mathbf{z}_i + \mathbf{W}_3 \mathbf{E}_t(y_{i-1}) + \mathbf{b}) \quad (3.9)$$

where  $\mathbf{W}_1 \in \mathbb{R}^{l \times q}$ ,  $\mathbf{W}_2 \in \mathbb{R}^{l \times k}$ ,  $\mathbf{W}_3 \in \mathbb{R}^{l \times d}$  and  $\mathbf{b} \in \mathbb{R}^l$  are trainable weights.

This intermediate representation is projected to the space of the target vocabulary, by means of a vocabulary-sized fully-connected layer. Finally, to obtain a probability distribution over the target vocabulary  $\mathbf{p}_i$ , we apply the softmax function, as in Eq. (3.10):

$$\mathbf{p}_i = \varphi(\mathbf{W}_V \mathbf{t}_i + \mathbf{b}_V) \quad (3.10)$$

where  $\mathbf{W}_V \in \mathbb{R}^{|\mathcal{Y}| \times l}$  and  $\mathbf{b}_V \in \mathbb{R}^{|\mathcal{Y}|}$  are the weights to learn.

The probability distribution  $\mathbf{p}_i$  corresponds to the one defined by Eq. (3.2): the probability of the token  $y$  at the  $i$ -th time-step is given its the corresponding position in  $\mathbf{p}_i$  (Eq. (3.11)):

$$p(y_i = y \mid y_1^{i-1}, x_1^J; \Theta) = \bar{\mathbf{y}}^\top \mathbf{p}_i \quad (3.11)$$

being  $\bar{\mathbf{y}} \in [0, 1]^{|\mathcal{Y}|}$  the one-hot codification of the token  $y$ .

## 3.2 Transformer: attention-based NMT

While RNNs seem the natural choice for sequence modeling, they cannot be parallelized: to compute the current state, it is necessary to process the previous ones. Aiming at overcoming this limitation, alternative architectures have been developed. Among them, it can be highlighted the Transformer model (Vaswani et al., 2017), described in the following. ConvNets have also been used for NMT, to encode sentences (Kalchbrenner and Blunsom, 2013) and also as fully convolutional NMT systems, with attention mechanisms (Gehring et al., 2017). Other works departed from the encoder–decoder framework, tackling the MT task as a 2-dimensional problem, employing multidimensional LSTM networks Bahar et al. (2018) or ConvNets Elbayad et al. (2018) to relate source and target sequences.

The Transformer model is based on the application of attention mechanisms, which compute different representations of the source and target sequences. Since the recurrences are removed from the model, training can be parallelized to a greater extent than RNN-based models. Moreover, it is capable of modeling large contexts more easily than RNNs (Agrawal et al., 2018), and perform better in multilingual settings (Lakew et al., 2018). The Transformer model is rapidly gaining popularity among the MT community, arguably replacing the RNN-based system as the standard model (see Bojar et al., 2018). On the other hand, Transformers also suffer from weaknesses: they are extremely sensitive to hyperparameters, which makes it hard to find working configurations and they require even larger amounts of data to yield a good performance, compared to RNN-based systems (see Section 3.5.3).

As RNN-based NMT systems, the Transformer follows the encoder–decoder approach: an encoder computes a representation of the source sequence and the decoder generates the translated sentence from this representation. For the sake of simplicity, all subcomponents of the Transformer produce outputs of dimension  $d_m$ .

One of the key aspects of the Transformer design is the way in which these representations are obtained. In addition to the inter-sequence attention mechanism (as the one present in RNN-based systems), the Transformer computes an intra-sequence attention, also called self-attention, on its input and output sequences. The model can relate different positions of a given sequence to compute a representation of it. Therefore, in the Transformer model, the attention is not only used as a connection between encoder and decoder, but also as a way of building internal representations.

**Introducing positional information** The Transformer has the same inputs and outputs as an RNN-based NMT system: the sequence of elements of the source sentence and the sequence of elements of the target sequence shifted one time-step to the right, following the teacher forcing training scheme.

The elements from the discrete vocabulary spaces are projected into a continuous space via embedding matrices. Applying Eq. (3.4) to the input sequence  $x_1^J$ , we obtain a sequence of  $J$  embeddings of dimension  $d_m$ :  $\mathbf{x}_1, \dots, \mathbf{x}_J$ . However, since the recurrence is dropped in the model, it is necessary to inject positional information into the sequence representation. This can be done via positional encodings (Gehring et al., 2017). The Transformer model uses fixed positional encodings: a sequence of vectors  $\mathbf{e}_1, \dots, \mathbf{e}_J$  that introduce positional information to the sequence. Each positional encoding vector is a  $d_m$ -dimensional vector, constructed using sinusoidal signals, according to its position within the sequence. Therefore, each element from  $\mathbf{e}_j$  (for  $1 \leq j \leq J$ ) is defined according to Eq. (3.12):

$$e_{j,k} = \begin{cases} \sin(j/10000^{2k/d_m}) & \text{if } k \text{ is even} \\ \cos(j/10000^{2k/d_m}) & \text{if } k \text{ is odd} \end{cases}, \text{ for } 0 \leq k \leq d_m \quad (3.12)$$

These positional information is added to the regular embeddings, to obtain a sequence of position-aware embeddings, as in Eq. (3.13):

$$\bar{\mathbf{x}}_1, \dots, \bar{\mathbf{x}}_J = (\mathbf{x}_1 + \mathbf{e}_1), \dots, (\mathbf{x}_J + \mathbf{e}_J) \quad (3.13)$$

**Encoder** The encoder is a stack of  $N$  layers, all of them following the same structure: a multi-head attention mechanism (Section 2.3.2) followed by a feed-forward network. All sublayers have residual connections. The result of each residual connection is normalized via layer normalization (Section 2.4). To regularize the model, dropout is applied to the output of each layer, prior to the normalization. For the sake of simplicity in the notation, we omit the layer normalization and dropout operations in this section. Moreover, in the rest of the section, we will denote the  $n$ -th layer of a stack with the superscript  $[n]$ .

Therefore, the input of the  $n$ -th layer of the encoder is a sequence of  $J$  inputs of  $d_m$  dimensions:  $\mathbf{h}_1^{J[n]}$ . Each encoder layer computes an output sequence  $\mathbf{h}_1^{J[n+1]}$  of the same dimensions. Hence, for  $0 \leq n < N$ , the encoder is defined by Eq. (3.14):

$$\mathbf{h}_1^{J[n+1]} = f_{\text{F}}^{[n]}(\mathbf{h}_1^{J[n]} + \gamma^{[n]}(\mathbf{h}_1^{J[n]}, \mathbf{h}_1^{J[n]}, \mathbf{h}_1^{J[n]})) + \mathbf{h}_1^{J[n]} + \gamma^{[n]}(\mathbf{h}_1^{J[n]}, \mathbf{h}_1^{J[n]}, \mathbf{h}_1^{J[n]}) \quad (3.14)$$

where  $\gamma^{[n]}$  is the multi-head attention defined in Eq. (2.30) and  $f_{\text{F}}$  is a 2-layered feed-forward network, with a ReLU and a linear activation, as in Eq. (3.15):

$$f_{\text{F}}^{[n]}(\mathbf{h}_1^{J[n]}) = \text{ReLU}(\mathbf{h}_j^{[n]} \mathbf{W}_1^{[n]} + \mathbf{b}_1^{[n]}) \mathbf{W}_2^{[n]} + \mathbf{b}_2^{[n]}, \forall \mathbf{h}_j^{[n]} \in \mathbf{h}_1^{J[n]} \quad (3.15)$$

where  $\mathbf{W}_1^{[n]} \in \mathbb{R}^{d_m \times d_{\text{F}}}$ ,  $\mathbf{b}_1^{[n]} \in \mathbb{R}^{d_{\text{F}}}$ ,  $\mathbf{W}_2^{[n]} \in \mathbb{R}^{d_{\text{F}} \times d_m}$  and  $\mathbf{b}_2^{[n]} \in \mathbb{R}^{d_m}$  are the parameters to learn.

The inputs to the first layer of the encoder stack ( $\mathbf{h}_1^{J(0)}$ ) are the position-aware embeddings computed by Eq. (3.13). These inputs are also regularized via dropout. Note that the encoder applies the attention over the same sequence. Therefore, it is a self-attention system, which computes representations at an intra-sequence level.

**Decoder** The decoder of the Transformer model is another stack of  $M$  layers. Each layer can be separated in two different parts: The first one is devoted to encoding the sequence of shifted outputs, by applying self-attention in a similar way as done by the encoder. The second part bridges together the representations of both self-attention modules, performing inter-sequence attention and generating the target sequence.

The positional information is injected to the sequence of shifted output embeddings, as described in Section 3.2. This produces a sequence  $\bar{\mathbf{a}}_1^I = \bar{\mathbf{a}}_1, \dots, \bar{\mathbf{a}}_I$  of  $I$  embeddings of the shifted outputs. Similarly as in the encoder, the Transformer decoder applies a self-attention mechanism ( $\gamma_1$ ) to this sequence. To prevent the decoder to look into future elements of the sequence, this attention is masked. Following the same notation than in the previous section, the self-attended representation ( $\mathbf{a}_1^{I[m+1]}$ , with each  $\mathbf{a}_i \in \mathbb{R}^{d_m}$ ) at the  $m$ -th decoding layer (for  $0 \leq m < M$ ) is computed as in Eq. (3.16):

$$\mathbf{a}_1^{I[m+1]} = \mathbf{a}_1^{I[m]} + \gamma_1^{[m]}(\mathbf{a}_1^{I[m]}, \mathbf{a}_1^{I[m]}, \mathbf{a}_1^{I[m]}) \quad (3.16)$$

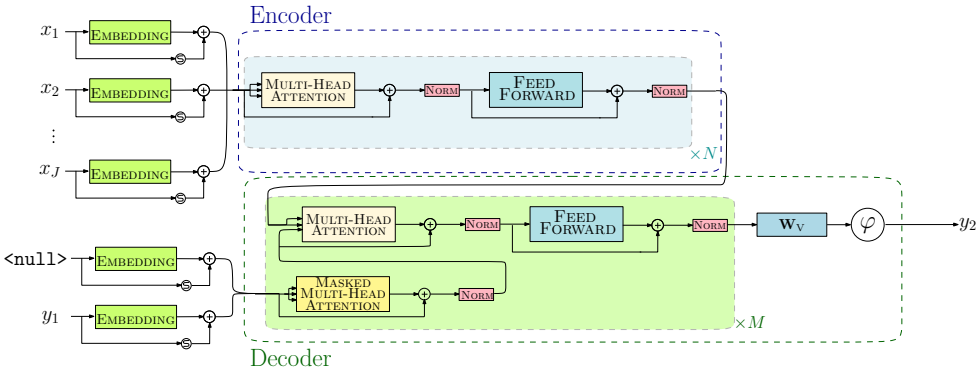
The second part of the decoder, that bridges together the representations obtained from the input and the shifted output sequences, must compute an inter-sequence attention. This is done by applying a inter-sequential multi-head attention mechanism ( $\gamma_2$ ) in which keys and values come from the source sequence ( $\mathbf{h}_1^{J[N]}$ ) and queries come from the target sequence ( $\mathbf{a}_1^{I[m]}$ ). Similarly to the encoder, the multi-head attention mechanism is followed by a feed-forward network, to compute an output sequence of  $I$  elements ( $\mathbf{z}_1^{I[m+1]}$ ) at each decoder layer, for  $0 \leq m < M$ . This output representation is the input of the following layer in the decoder stack, as in Eq. (3.17):

$$\mathbf{z}_1^{I[m+1]} = f_{\text{F}}^{[m]}(\mathbf{a}_1^{I[m]} + \gamma_2^{[m]}(\mathbf{a}_1^{I[m]}, \mathbf{h}_1^{J[N]}, \mathbf{h}_1^{J[N]})) + \mathbf{a}_1^{I[m]} + \gamma_2^{[m]}(\mathbf{a}_1^{I[m]}, \mathbf{h}_1^{J[N]}, \mathbf{h}_1^{J[N]}) \quad (3.17)$$

Following the stack of layers of the decoder, we apply the same fully-connected layer with a softmax activation than for RNN-based NMT (Eq. (3.10) and Eq. (3.11)) to the outputs of the decoder ( $z_1^I [M]$ ), yielding Eq. (3.18):

$$p_i = \varphi(\mathbf{W}_V z_i^{[M]} + \mathbf{b}_V) \quad (3.18)$$

where  $\mathbf{W}_V \in \mathbb{R}^{|\mathcal{Y}| \times d_m}$  and  $\mathbf{b}_V \in \mathbb{R}^{|\mathcal{Y}|}$  are the weights to learn. Fig. 3.2 shows an illustration of the Transformer model.



**Figure 3.2:** The Transformer model. As in the RNN-based NMT model, the input of the system is a sequence of words  $x_1, \dots, x_J$ , projected into the continuous space via an embedding matrix. To have a notion of sequentiality, these embeddings are augmented with positional information. The encoder is a stack of  $N$  layers. Each layer features a multi-head attention mechanism followed by a feed-forward layer. The decoder is another stack of  $M$  layers. Previous words are encoded similarly as input words, but using a masked multi-head attention mechanism. Next, input and output representations are combined through another multi-head attention mechanism and feed-forward layers. The representation of the last decoder layer is projected to the target language vocabulary space. Finally, a softmax function computes the probabilities in this space.

### 3.3 NMT decoding

Now, we pay attention to the search problem: given a source sentence  $x_1^I$  and an NMT model with parameters  $\Theta$ , how to obtain the sentence in the target language with the highest probability ( $\hat{y}_1^I$ ). Most search methods exploit the factorization of the conditional probability shown by Eq. (3.2):

$$\hat{y}_1^I = \arg \max_{I, y_1^I} \sum_{i=1}^I \log p(y_i | y_1^{i-1}, x_1^J; \Theta)$$

The search is generally an incremental process: starting from a null-hypothesis (a hypothesis with no words), they iteratively construct partial hypotheses by adding words to previous hypotheses. This null-hypothesis is initialized with the special beginning-of-sentence token (`<bos>`). The process is repeated until completing hypotheses. A hypothesis is considered to be completed if its last word is an end-of-sentence token (`<eos>`). Therefore, at the end of the process, we obtain a search tree, in which each level represents a decoding time-step and each path from the root (`<bos>`) to a leaf (marked with `<eos>`) represents a complete translation hypothesis and has associated a score (typically, the cumulated log-probability). Hence, the goal is to find the path with highest score from the root node to a leaf.

However, this search tree has a branching factor of the size of the target vocabulary. To build and explore the complete tree becomes computationally unaffordable, in terms of time and memory. Therefore, we need efficient procedures to approximate the search. The most popular heuristics applied to tackle this problem is the so-called beam search method (Lowerre, 1976). Beam search limits the branching factor of the tree to a maximum predefined value, called size of the beam ( $b$ ). At each level of the tree, beam search expands each partial hypothesis with all possible words in the target vocabulary. Next, the set of expanded hypotheses is pruned, keeping only the  $b$  with the highest score. If a complete hypothesis is generated, it is removed from the partial hypotheses set and it is stored into a set of completed hypotheses. The beam size is then decreased by one. This process is repeated until the size of the beam reaches zero. Finally, the method returns the most probable hypothesis from the set of complete hypotheses.

While beam search constitutes a well-performing trade off between computational complexity and its search exhaustiveness, it has several limitations: it suffers from the *length* and *label biases*, which harm the performance of the method (Bottou, 1991; Koehn and Knowles, 2017). Moreover, when used for MT, the beam search is unaware of the coverage of the translations, which may prevent the translation of several parts of the source sentence. Overcoming these limitations while keeping the method efficient is an active research field (Tu et al., 2016; Wu et al., 2016).

### 3.4 Dealing with the vocabulary restriction

A limiting aspect of NMT is the size of the vocabularies: MT is an open-vocabulary task, while the NMT models require finite vocabularies. As mentioned in the previous section, each element from the source and target vocabularies are mapped into a unique index and projected to the continuous space via embedding matrices, which are proportional to the vocabulary size. Moreover, the output layer requires us to compute a normalization through the full target language vocabulary. This makes it impractical to use models with very large vocabularies. Moreover, new words can

appear while using the system and the model should be able to tackle these unknown words.

The first works on NMT (Kalchbrenner and Blunsom, 2013; Sutskever et al., 2014; Cho et al., 2014) relied on the usage of short-lists: the vocabulary was fixed beforehand, typically to the  $K$  most common words of the training set. The rest of words, were mapped to an special token, denoting the unknown word. While this eased the training of the model, it could introduce performance issues, especially if a source sentence contained many out-of-vocabulary words.

This problem was partially alleviated by taking advantage of the attention mechanism, substituting an unknown word by the source word with the highest attention (Jean et al., 2015a; Luong et al., 2015b). Although some authors (Jean et al., 2015a) devised a mechanism to effectively train on large vocabularies, this was a problematic condition suffered by NMT.

### 3.4.1 Subword NMT

A more interesting alternative was introduced by Sennrich et al. (2016): instead of translating sequences of words, Sennrich et al. (2016) proposed to translate sequences of subwords. The idea is to use a compression algorithm to encode words as sequences of smaller units. The BPE algorithm (Gage, 1994) is very adequate to this purpose.

BPE is an iterative data compression algorithm, in which the most common pair of bytes are merged and substituted by a single, unused byte. Applied to the word segmentation problem, BPE starts from a corpus split in single characters, with a special end-of-word symbol. Iteratively, it merges the two most common consecutive symbols, into a new, unused symbol. This is done until reaching a stopping criterion, typically a predefined number of merges. At the end of this process, the vocabulary of the compressed corpus is equal to the number of merge operations, plus the size of the initial vocabulary (characters). Since the end-of-word symbol is kept during the process, it is trivial to revert the encoding with a replace operation. Therefore, this compression technique obtains different granularities of the representation of words: rare words will be represented as sequences of subwords, while most common words will tend to be represented closer as a word level.

BPE has become a standard in the NMT field: most systems make use of BPE or similar approaches (e.g. Ataman and Federico, 2018; Wu et al., 2016). Although this segmentation is effective in many cases, tackling morphologically rich languages requires less arbitrary segmentation strategies (Passban et al., 2018).

It is usual to apply BPE jointly to source and target corpora (if they share alphabet). Our NMT systems follow this setup (see Section 3.5.1). This prevents



segmentation inconsistencies across the languages and ensures the same segmentation of the same words (e.g. proper nouns).

## 3.5 Machine translation results

Once we reviewed the main features of NMT, we describe the NMT systems used in [Chapters 4](#) and [5](#) and evaluate them in tasks described in [Section 1.4.3](#). This evaluation will be the starting point in [Chapters 4](#) and [5](#). We also compare them with a well-known PB-SMT system (Moses, [Koehn et al., 2007](#)). This evaluation allows us to determine the performance of our NMT systems on a standard MT scenario and compare them to PB-SMT systems. We automatically evaluate the translation quality of the different systems, according to TER and BLEU, as described in [Section 1.4.1](#).

### 3.5.1 NMT systems

All NMT systems were built using our in-house developed toolkit, NMT-Keras ([Peris and Casacuberta, 2018a](#)). Refer to [Appendix A](#) for a more detailed description of the software. This toolkit can construct neural systems using Theano or Tensorflow. We used the first library as backend of the toolkit, as we found it to be faster than Tensorflow, especially for recurrent models.

We tested two architectures for the NMT system: RNN-based ([Section 3.1](#)) and Transformer ([Section 3.2](#)). All systems were trained via mini-batch SGD ([Section 2.2.2](#)), using the Adam update rule ([Kingma and Ba, 2014](#)), with a learning rate of 0.0002 ([Wu et al., 2016](#)). To avoid the exploding gradient problem ([Section 2.3.1](#)), we clipped the  $L_2$  norm of the gradients to 10. The training batch size was 50. The decoding method was beam search ([Section 3.3](#)), using a beam size of 6.

We applied uniform label smoothing ([Section 2.4](#)), with  $\epsilon = 0.1$ ; and Gaussian noise ([Section 2.4](#)) to the weights during training (with a magnitude of 0.01). We used early stopping ([Section 2.4](#)), based on BLEU, computed each 3,750 training steps and with a patience of 10. During training, we restricted the length of the input and output sequences to 60 elements.

## RNN-based NMT

The RNN-based NMT system featured a single-layered bidirectional LSTM encoder, using concatenation as fusion operator. The decoder function was another single-layered RNN with conditional LSTM units, with an additive attention mechanism (Eq. (2.19)). The initial hidden and memory states of the decoder were initialized according to two independent MLPs with tanh activation functions from the average of the annotations computed by the encoder, as done by Xu et al. (2015). Following Britz et al. (2017), the dimension of all layers, including word embeddings, were set to 512. We applied layer normalization, weight decay ( $\lambda = 10^{-4}$ ) and dropout ( $p = 0.1$ ) to all non-recurrent connections.

## Transformer

The Transformer model (Section 3.2) followed the configuration defined by Vaswani et al. (2017) as *base* model: The size of the word embeddings and  $d_m$  were set to 512. Each multi-head attention layer had  $H = 8$  heads. Each parallel projection was of size  $d_m/H$  (i.e. 64). The hidden and output dimensions of the feed-forward layers were 2,048 and 512, respectively. In contrast to Vaswani et al. (2017), we used different embedding matrices for each language. The dropout probability was set for all layers to  $p = 0.1$ . The embeddings were scaled by a factor of  $\sqrt{d_m}$ . For tasks with sufficient training data (500,000 training samples or more), the encoder and decoder stacked 6 layers, while in tasks without these amounts of data, the stacks were of 4 layers.

### 3.5.2 PB-SMT system

Regarding the configuration of the PB-SMT system, we used the latest stable version of Moses (4.0) with its standard setup: the phrase table was obtained using symmetrised word alignments, computed by mGIZA++ (Och and Ney, 2002). The feature functions included in the log-linear were the phrase translation table, word penalty, unknown word penalty, phrase penalty, lexical reordering model, distortion model and the language model. This language model was a 5-gram, smoothed with the technique proposed by Kneser and Ney (1995). The tuning of the weights from the log-linear model was made through minimum error rate training (Och, 2003) on the development sets.

### 3.5.3 Machine translation results

We evaluate the different MT systems and technologies on all the tasks and language combinations introduced in [Section 1.4.3](#).

#### XRCE

The XRCE corpus represents a challenging task for NMT systems, due to its reduced dimensions. [Table 3.1](#) shows the translation performance, in terms of TER and BLEU, of PB-SMT, RNN-based and Transformer-based NMT.

**Table 3.1:** Translation quality for the XRCE task in terms of TER [↓] and BLEU [↑] for RNN-based, Transformer (Trans.) and PB-SMT systems. Paired approximate randomization test was applied to all systems. Systems significantly better are indicated with †, when comparing PB-SMT and RNN; with ‡ when comparing PB-SMT and Transformer; and with \* when comparing RNN and Transformer.

		TER [↓]			BLEU [↑]		
		RNN	Trans.	PB-SMT	RNN	Trans.	PB-SMT
XRCE	En→De	63.0*	64.3	64.4	25.4†*	23.2	22.4
	De→En	51.1	54.9	50.1‡	36.2*	31.3	36.8‡
	En→Fr	51.9*	57.2	50.2†‡	38.0*	32.2	37.9‡
	Fr→En	50.9*	55.7	46.5†‡	36.6*	30.2	37.4‡
	En→Es	27.5	28.3	24.7†‡	63.5*	60.5	64.0‡
	Es→En	28.6*	32.1	29.0‡	59.0†*	53.9	55.8‡

In terms of TER, NMT systems were outperformed by PB-SMT for most language pairs. Without the exception of the En→De translation, PB-SMT achieved better results than the Transformer model by large margins. These differences were smaller when comparing PB-SMT against RNN-based NMT, although they still favored PB-SMT. Consequently, RNN-based NMT performed better than the Transformer model in all cases. These differences were statistically significant except for two language pairs. In terms of BLEU, the differences were lower. On the one hand, PB-SMT and RNN-based NMT performed similarly for most cases. There were two language combinations (En→De and Es→En) for which NMT significantly outperformed PB-SMT. In the rest of combinations, the differences were small and non-significant. On the other hand, the Transformer model obtained bad results, for almost every pair: its performance was clearly below PB-SMT and RNN-based NMT, yielding degradations up to 6 points.

We find two reasons for the poor performance of the Transformer model: it is very sensitive to hyperparameters and to data scarcity ([Popel and Bojar, 2018](#)). We are

now dealing with a low-resource task and it is hard to find the optimal hyperparameter configuration. Moreover, this task relates a very structured domain, as are printer manuals, that contains a number of well-defined structures (page headers and footers, sections, etc). This type of structures are well handled by  $n$ -grams and phrase-tables and hence, PB-SMT systems perform very well in this situation.

## TED

For the TED task, RNN-based models clearly obtained the best results, achieving large improvements with respect to the PB-SMT system and the Transformer model for every language pair (Table 3.2). Compared with PB-SMT systems, we observed regular improvements, from 2 to more than 6 points of TER and BLEU. The differences with respect to the Transformer model were slightly smaller, but also consistent. The Transformer model was placed in between the PB-SMT and the RNN-based NMT systems: For most tasks, it significantly outperformed PB-SMT, but was surpassed by RNN-based NMT. This is related to what we observed in the previous task: the TED corpus is still small, and the Transformer suffered from this data scarcity. In addition, note that the sentences from a TED talk are less structured than in the XRCE domain. Hence, PB-SMT systems did not perform so well in this domain, and they were overcome by both neural systems.

**Table 3.2:** Translation quality for the TED task in terms of TER [ $\downarrow$ ] and BLEU [ $\uparrow$ ] for RNN-based, Transformer (Trans.) and PB-SMT systems. Paired approximate randomization test was applied to all systems. Systems significantly better are indicated with  $\dagger$ , when comparing PB-SMT and RNN; with  $\ddagger$  when comparing PB-SMT and Transformer; and with  $*$  when comparing RNN and Transformer.

		TER [ $\downarrow$ ]			BLEU [ $\uparrow$ ]		
		RNN	Trans.	PB-SMT	RNN	Trans.	PB-SMT
TED	En→De	54.8 $\dagger$	57.1 $\ddagger$	59.3	25.6 $\dagger*$	23.1 $\ddagger$	19.4
	De→En	49.1 $\dagger*$	52.5 $\ddagger$	53.5	30.2 $\dagger*$	27.1 $\ddagger$	25.3
	En→Fr	49.6 $\dagger*$	52.5 $\ddagger$	52.5	33.5 $\dagger*$	30.6 $\ddagger$	27.6
	Fr→En	46.6 $\dagger*$	49.5 $\ddagger$	50.2	32.4 $\dagger*$	30.1 $\ddagger$	29.9
	En→Zh	76.7 $\dagger$	77.6 $\ddagger$	83.2	9.3 $\dagger*$	8.2	8.7 $\ddagger$
	Zh→En	75.7 $\dagger*$	76.7 $\ddagger$	77.5	13.7 $\dagger*$	11.5	11.0

The largest differences in terms of BLEU and TER were found when translating to complex languages, namely, German and Chinese. NMT succeeded at capturing the complex relationships existing in the sequences from these languages. Moreover, when translating from German, the differences were also high, denoting that NMT was also able to better capture the relationships among the source sentences.

## UFAL

We move now to a task with more data. In this case, neural systems clearly outperformed PB-SMT for the UFAL task (Table 3.3). For every language combination the neural systems worked better than PB-SMT, and we found especially large improvements on the En→De and En→Es directions.

**Table 3.3:** Translation quality for the UFAL task in terms of TER [↓] and BLEU [↑] for RNN-based, Transformer (Trans.) and PB-SMT systems. Paired approximate randomization test was applied to all systems. Systems significantly better are indicated with †, when comparing PB-SMT and RNN; with ‡ when comparing PB-SMT and Transformer; and with \* when comparing RNN and Transformer.

		TER [↓]			BLEU [↑]		
		RNN	Trans.	PB-SMT	RNN	Trans.	PB-SMT
UFAL	En→De	55.6 <sup>†</sup>	55.4 <sup>‡</sup>	62.9	23.7 <sup>†</sup>	24.5 <sup>‡</sup>	18.2
	De→En	50.8	44.1 <sup>‡*</sup>	51.0	29.8	34.8 <sup>‡*</sup>	29.7
	En→Fr	46.1 <sup>†</sup>	45.9 <sup>‡</sup>	47.9	37.2 <sup>†</sup>	37.8 <sup>‡*</sup>	35.0
	Fr→En	42.8	42.3	42.9	37.6 <sup>†</sup>	38.0 <sup>‡*</sup>	36.2
	En→Es	40.5 <sup>†</sup>	40.8 <sup>‡</sup>	45.7	40.7 <sup>†*</sup>	39.1 <sup>‡</sup>	33.1
	Es→En	35.4 <sup>†*</sup>	36.4 <sup>‡</sup>	40.4	44.4 <sup>†*</sup>	43.1 <sup>‡</sup>	38.7

Comparing neural models, we observed again that the Transformer model worked better for German. RNNs worked better for the case in which we had less data (Spanish and English combinations): while for German and French there were available around 3 million of parallel segments, for Spanish, there were only 780,000. These differences may affect the performance of the neural systems: the Transformer model required more data to work properly. RNNs were able to obtain a good performance with the limited amount of data from the Spanish part of the corpus.

## Europarl

Similarly as for the UFAL corpus, neural models generally outperformed PB-SMT in terms of BLEU (Table 3.4). The largest differences were found for language combinations involving German (up to 3.6 points). The differences in combinations that involve French and Spanish were smaller. Regarding the comparison between RNNs and Transformer, the latter model worked slightly better. For all combinations but Es→En, the Transformer obtained significant improvements with respect to the RNN-based system.

**Table 3.4:** Translation quality for the Europarl task in terms of TER [ $\downarrow$ ] and BLEU [ $\uparrow$ ] for RNN-based, Transformer (Trans.) and PB-SMT systems. Paired approximate randomization test was applied to all systems. Systems significantly better are indicated with  $\dagger$ , when comparing PB-SMT and RNN; with  $\ddagger$  when comparing PB-SMT and Transformer; and with  $*$  when comparing RNN and Transformer.

		TER [ $\downarrow$ ]			BLEU [ $\uparrow$ ]		
		RNN	Trans.	PB-SMT	RNN	Trans.	PB-SMT
Europarl	En $\rightarrow$ De	67.0	63.8 $\ddagger^*$	68.2 $\dagger$	18.4 $\dagger$	19.1 $\ddagger^*$	14.9
	De $\rightarrow$ En	62.9	60.7 $*$	60.9 $\dagger$	21.2 $\dagger$	22.3 $\ddagger^*$	20.4
	En $\rightarrow$ Fr	60.5	57.4 $\ddagger^*$	58.4 $\dagger$	24.6	26.6 $\ddagger^*$	24.4
	Fr $\rightarrow$ En	62.2	62.7	61.0 $\dagger\ddagger$	22.8 $\dagger$	23.5 $\ddagger$	20.9
	En $\rightarrow$ Es	58.6	55.4 $\ddagger^*$	56.8 $\dagger$	25.0 $\dagger$	26.1 $\ddagger^*$	24.6
	Es $\rightarrow$ En	58.4	58.1 $*$	55.4 $\dagger\ddagger$	25.6 $*$	25.1	25.6

In terms of TER, the gap between PB-SMT and NMT was reduced. In several cases, PB-SMT systems were able to significantly outperform NMT. These differences were especially large when translating into English.

### 3.5.4 Discussion

From the set of experiments carried out in the previous section, we observed several behaviors that are consistent with the rest of the literature on NMT. NMT systems generally outperform PB-SMT on most tasks. PB-SMT worked better for tasks with scarce resources and highly structured (e.g. XRCE). But, as the amount of training data was increased, NMT clearly outperformed PB-SMT systems. The Transformer model was more sensitive to this lack of data. For tasks with scarce data (XRCE, TED) it usually performed significantly worse than RNN-based systems.

NMT systems are able to model better complex relationships than PB-SMT. The largest differences between NMT and PB-SMT systems were found when working with distant language pairs (English $\leftrightarrow$ Chinese) and with morphologically complex languages (German). With the exception of the XRCE task, neural systems always outperformed PB-SMT when using German as source or target language. Provided that there are enough data, the Transformer NMT system modeled better German than the RNN-based system. We observed significant improvements for language combinations involving German for the UFAL and Europarl tasks.

We also found the Transformer to be more sensitive to hyperparameters than RNN-based NMT. While we found the standard Transformer hyperparameters (Vaswani et al., 2017) worked well in tasks with enough training data, this is not the case of scarce-resource scenarios. The choice of hyperparameters for the Transformer model

is critical, and small variations cause the model to not converge, thus requiring a wide search of hyperparameters. Hence, using this model out of the well-known scenarios may be adventurous and computationally expensive, having a negative impact in the environment (Strubell et al., 2019). We need to consider all these aspects when deploying a NMT system. Finally, and as expected, the Transformer model was significantly faster to train than RNN-based systems, as it does not require recurrences.

## 3.6 Summary

In this chapter, we thoroughly described the NMT technology: the main neural architectures, the decoding process and the usage of subwords. These systems are the basis upon which we build the interactive-predictive and adaptive NMT in the following chapters (Chapters 4 and 5), hence it is important to thoroughly study them.

We evaluated our NMT systems in a variety of translation tasks and languages. We compared the two main architectures described in Sections 3.1 and 3.2, namely an RNN-based and a Transformer system. We found that the Transformer model was harder to optimize, due to its sensitivity to hyperparameters: finding a good set of hyperparameters for all tasks is hard. On the other hand, RNN-based systems performed well in all cases, requiring less hyperparameter tuning.

We also compared our neural systems to classical PB-SMT. The results showed that NMT generally worked better than classical PB-SMT systems. The differences were especially large when we have available enough training data. NMT also performed better when dealing with morphologically complex languages (such as Chinese or German). It is also worth remarking the case of a highly structured task with scarce training resources (XRCE), in which PB-SMT still performed better than NMT. Finally, we found that the Transformer model suffered in this scenario. This model requires either larger training datasets or hyperparameter tuning to fully exploit its potential.





## Chapter 4

# Interactive-predictive neural machine translation

Nowadays, NMT is predominant approach to MT, as discussed in the introduction of this thesis (see [Section 1.3.2](#)). In the previous chapter, we described this technology and showed that in most scenarios NMT outperformed PB-SMT systems. Despite these advances, the MT problem is still far to be solved ([Koehn and Knowles, 2017](#)). Automatic systems produce wrong translations, that may be intolerable for some users or domains. For example, translations of medical records must be accurate and error-free. The translation problem has several subtleties, that make it hard for machines to tackle it: ambiguity, discourse adequacy, anaphora resolution, domain-specific meanings, stylistic forms, etc. Automatic MT systems usually fail to solve these translation aspects ([Toral et al., 2018](#); [Toral and Way, 2018](#)).

In scenarios that require high-quality translations, the outputs of the MT systems are usually revised by a human agent, who corrects the errors made by the MT system. This process is known as post-editing. As the MT systems are continuously improving their capabilities, translation post-editing has acquired a major relevance in the translation market, allowing to achieve a higher productivity, compared to translating from scratch ([Arenas, 2008, 2009](#); [Green et al., 2013a](#)). Effective translation post-editing methods are required by the translation industry and, indeed, they are already provided by many agencies ([Hu and Cadwell, 2016](#)).

However, post-editing is a decoupled strategy in which the computer proposes a translation and the human agent fixes it, working independently. Higher efficiency rates can be achieved if human and system collaborate on a joint strategy. Looking for this human–computer collaboration, Foster et al. (1997) introduced the so-called interactive-predictive machine translation. This approach is an iterative prediction–correction process: each time the user corrects a word, the system reacts offering a new translation hypothesis, expected to be better than the previous one.

This alternative process to correct the output of a system has been under development since its inception, for more than twenty years. Nowadays, interactive-predictive MT has been consolidated and it is integrated into several computed assisted translation tools, such as Lilt<sup>1</sup>.

This chapter presents an interactive-predictive NMT system. We start by reviewing the related research done in the field. Next, we formally present the interactive-predictive MT framework, considering two main interaction protocols. Then, we instantiate the NMT technology in this general framework, describing the modifications that should be done to an NMT system to fit it into the interactive-predictive framework and proposing several extensions to enhance the system.

## 4.1 Interactive-predictive machine translation

The collaboration between human and system has been under study from long ago. In early works (Kay, 1980; Slocum, 1985; Whitelock et al., 1986), the users typically solved several types of ambiguities (lexical, syntactical or semantic). The users mainly interacted with the source text, providing its correct meaning. The aim of these interactive systems was making the disambiguation procedure more comfortable and efficient.

An important breakthrough was performed by Foster et al. (1997), who criticized previous approaches, arguing that the focus of human–machine interaction should be the target text. They introduced the concept of human-targeted MT, which differentiated from previous approaches by introducing the interaction during the generation of the target text. This allowed a deeper embedding of the interaction process and this approach was exploited by the (at that time emerging) corpus-based MT systems.

Following this rationale, the human-targeted interactive-predictive MT advanced, supported by the TransType (Langlais et al., 2002), TransType2 (Macklovitch, 2006) and CasMaCAT (Alabau et al., 2013) projects. Barrachina et al. (2009) formalized the interactive-predictive translation framework under a statistical point of view, yielding

---

<sup>1</sup><https://lilt.com>

the interactive-predictive approach to MT. Under the scope of these projects, major milestones were achieved, pushing the MT systems to new frontiers.

Regarding interactive-predictive MT, the generation of the suffixes for PB-SMT systems received much attention from the research community. This process was usually carried out by searching on the search graphs used by the PB-SMT systems (Bender et al., 2005; Barrachina et al., 2009). Alternative search strategies for better exploiting these search graphs were explored by Ortiz-Martínez (2011); Vakil and Khadivi (2012); Cai et al. (2013); Koehn et al. (2014); Azadi and Khadivi (2015). Other works studied suggesting to the user more than one translation hypothesis (Koehn, 2010a; Torregrosa et al., 2014). Additional novelties came from profiting from the usage of the mouse, for validating a prefix and suggesting a new suffix each time the user clicked into a position to type a word correction (Sanchis-Trilles et al., 2008). The addition of confidence measures aided the user to validate correct prefixes (González-Rubio et al., 2010a,b).

### **Integrating IMT into different technologies**

The IMT framework was introduced during the peak of PB-SMT systems. Therefore, most work regarding this topic was based on this technology. However, IMT systems have been deployed for other MT technologies. González-Rubio et al. (2013) presented a IMT system based on a hierarchical translation model. Green et al. (2014) investigated the interactive use of translation memories. Pérez-Ortiz et al. (2014) and Torregrosa et al. (2017) built technology-agnostic IMT systems, which treated the underlying MT system as a black-box.

Finally, given the recent success of NMT, this technology has also been adapted to fit into the interactive-predictive framework. To the best of our knowledge, the first works on INMT were simultaneously proposed by Knowles and Koehn (2016); Wuebker et al. (2016) and Peris et al. (2017c). Hokamp and Liu (2017) proposed a constrained search algorithm useful for INMT which was very similar to the one proposed by Peris et al. (2017c) and described in Section 4.3.2. Hasler et al. (2018); Post and Vilar (2018) developed efficient algorithms for adding constraints to the search. These methods can be used to build interactive-predictive NMT systems.

## Beyond prefix-based constraints

Most works on IMT follow the prefix-based interactive-predictive protocol, in which the user is forced to follow a strict left-to-right interaction. Some researchers spent a significant effort to overcome this tight constraint. [González-Rubio et al. \(2016\)](#) allowed the selection of correct segments from translation hypotheses, which must remain fixed along the IMT process. This procedure was extended by [Domingo et al. \(2016\)](#) and [Domingo et al. \(2018\)](#). We also integrated this protocol for NMT ([Section 4.3.2](#), [Peris et al., 2017c](#)).

Related to this, we find the so-called active interaction framework. In this paradigm, the system asks the user to only correct certain parts of the hypothesis, typically those with the least confidence to be properly translated ([González-Rubio et al., 2010a,b](#); [Lam et al., 2018](#)).

Other authors ([Marie and Max, 2015](#)) proposed a system based on touch interactions, which allowed the user to select the correct parts of a hypothesis. Extending this work, [Cheng et al. \(2016\)](#) developed a pick-revise procedure for IMT, consisting of the selection by the user of the most critical part of a hypothesis and its correction. This pick-revise framework has also been applied to NMT systems ([Hokamp and Liu, 2017](#); [Post and Vilar, 2018](#)).

It is worth pointing out a major difference between approaches taken by these later works and the one taken in this thesis and by other authors ([Barrachina et al., 2009](#); [González-Rubio et al., 2016](#); [Ortiz-Martínez, 2016](#); [Peris et al., 2017c](#)). When using an IMT system, we demand perfect translations for a given sentence (as in a full post-editing setup). Therefore, our goal is to diminish the human effort spent in the process to reach high-quality translations. On the other hand, the pick-revise framework (followed by [Cheng et al. \(2016\)](#); [Hokamp and Liu \(2017\)](#); [Post and Vilar \(2018\)](#)) accepts some translation errors, sacrificing the final quality at the expense of less human effort (as in light post-editing). Thus, their goal is to improve translation quality with few interactions.

## Multimodal interaction

The most common input of the feedback signal is the keyboard and mouse. However, the human-computer interaction can be done through different modalities. This is known as multimodal interaction. Among the modalities investigated for IMT, we can highlight the interaction through handwritten strokes ([Alabau et al., 2014](#)) or speech ([Alabau et al., 2011](#)).

Moreover, the interactive-predictive approach can be generalized to applications beyond IMT. The framework is applicable to the transcription of handwritten text

documents (Toselli et al., 2007; Martín-Albo et al., 2013), ancient manuscripts (Graneli et al., 2016), music (Inesta and Pérez-Sancho, 2013), layout detection (Quirós et al., 2017), parsing (Toselli et al., 2011), speech transcription (Rodríguez et al., 2007), image retrieval (Segarra et al., 2011) or multimodal captioning (Peris and Casacuberta, 2019b).

## 4.2 Probabilistic framework

The IMT framework relies on the statistical formalization of the MT problem (Section 1.3). Recall from this chapter that the goal is to find the best translation  $\hat{y}_1^{\hat{I}} = \hat{y}_1, \dots, \hat{y}_{\hat{I}}$  of length  $\hat{I}$ , given a source sentence  $x_1^J = x_1, \dots, x_J$  of length  $J$ , as described in Eq. (1.5).

Under the interactive-predictive paradigm, the static post-editing stage shifts to an iterative human–computer collaboration process. The user interacts with the system by means of a feedback signal  $f$ . The system suggests then an alternative translation hypothesis  $\tilde{y}_1^{\tilde{I}} = \tilde{y}_1, \dots, \tilde{y}_{\tilde{I}}$ , which takes into account the feedback. This new translation is obtained by integrating the feedback signal into the previous expression, yielding Eq. (4.1):

$$\tilde{y}_1^{\tilde{I}} = \arg \max_{I, y_1^I} \Pr(y_1^I \mid x_1^J, f) \quad (4.1)$$

Depending on the meaning conveyed by  $f$ , alternative interactive-predictive protocols can be defined. In the following, we describe two protocols: prefix-based and segment-based.

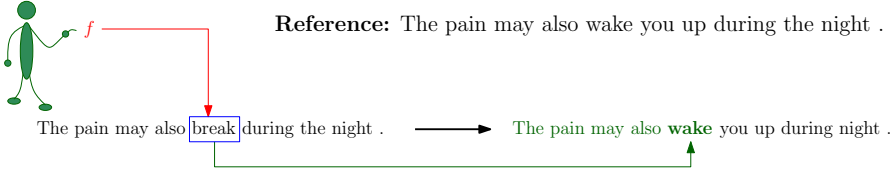
### 4.2.1 Prefix-based interactive-predictive machine translation

Prefix-based IMT was the first interaction protocol and it arguably is the most natural way of work. We describe this protocol for left-to-right writing languages, but this can be trivially extended to languages with right-to-left writing directions (e.g. Arabic or Hebrew).

The protocol starts by the MT system proposing an initial translation  $\hat{y}_1^{\hat{I}} = \hat{y}_1, \dots, \hat{y}_{\hat{I}}$  of the source sentence  $x_1^J$ , following Eq. (1.5). Next, the user then searches, from left to right, the first wrong word  $\hat{y}_i$  from the hypothesis and provides the correct word. Hence, with this correction, the system receives a feedback signal with the form  $f = \hat{y}_i$ , where  $\hat{y}_i$  is the corrected word at the position  $i$  in the target hypothesis. But this feedback signal conveys a two-fold meaning: it states that the  $i$ -th target word must be  $\hat{y}_i$ , but it also validates the hypothesis up to this position ( $\hat{y}_1^{i-1}$ ). Taking this

into account, we can rewrite  $f$  as  $f = \hat{y}_1^i$ , where  $\hat{y}_1^i$  is the validated prefix together with the corrected word.

At the next iteration, the system must generate the best suffix  $\tilde{y}_{i+1}^{\bar{I}}$  to build a new translation  $\tilde{y} = \hat{y}_1^i, \tilde{y}_{i+1}^{\bar{I}}$ . This process is repeated until the user accepts the complete hypothesis of the system. Fig. 4.1 represents this protocol.



**Figure 4.1:** Single iteration of prefix-based IMT. The user wants to translate the French sentence “La douleur peut également vous réveiller pendant la nuit .” into English. The user corrects the first wrong word from the hypothesis provided by the system, introducing the word “wake” at position 5. Next, the system generates a new hypothesis, that contains the validated prefix together with the corrected word. Note that, although the system generates a partially correct suffix, in this new hypothesis a new error is also introduced (“during night” instead of “during the night”). This behavior is intended to be solved with the segment-based approach (Section 4.2.2).

Introducing this feedback into Eq. (4.1), we obtain Eq. (4.2):

$$\tilde{y}_1^{\bar{I}} = \arg \max_{I, y_1^I} \Pr(y_1^I | x_1^J, f = \hat{y}_1^i) \quad (4.2)$$

We can rewrite the system hypothesis, splitting it into the validated prefix and the suffix, obtaining Eq. (4.3):

$$\tilde{y}_1^{\bar{I}} = \arg \max_{I, y_1^I} \Pr(\hat{y}_1^i, y_{i+1}^I | x_1^J, f = \hat{y}_1^i) \quad (4.3)$$

Since the validated prefix must remain fixed, we can remove it from this expression. We are thus interested in generating the most probable suffix  $\tilde{y}_{i+1}^{\bar{I}}$  to a prefix validated by the user (Eq. (4.4), Barrachina et al., 2009):

$$\tilde{y}_{i+1}^{\bar{I}} = \arg \max_{I, y_{i+1}^I} \Pr(\hat{y}_1^i, y_{i+1}^I | x_1^J) \quad (4.4)$$

Note that this equation is a prefix-constrained version of Eq. (1.5). Therefore, at each iteration, the process consists of a regular search in the translations space, but constrained by the validated prefix  $\hat{y}_1^i$ .

#### 4.2.2 Segment-based interactive-predictive machine translation

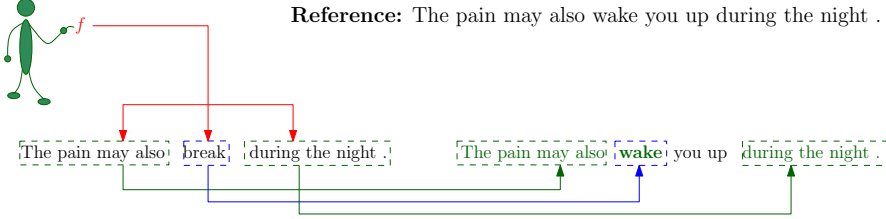
Although simple and intuitive, the prefix-based protocol suffered from two main issues: first, it was quite restrictive. The human translator was forced to always follow the left-to-right validation direction. This could be unnatural for the users or even inadequate in many cases. And second, the IMT system could produce worse suffixes, which also should be corrected by the user. Apart from increasing the human effort of the process, this introduced an annoying behavior: if the system modifies parts of the hypothesis that were correct although non-validated, the user can feel exasperated, because such new errors that must be corrected in the upcoming iterations were already solved. This is illustrated in Fig. 4.9: the initial hypothesis provided by the system contained the correct suffix *during the night*. However, this suffix was not validated, because the prefix-based protocol requires to correct a previous error. As this error is corrected, the system generates the suffix *during night*, which is incorrect. The user must then introduce a new correction, to a hypothesis that was correct in the previous iteration. This results annoying and cumbersome to the user. Aiming to overcome these handicaps, González-Rubio et al. (2016) introduced the so-called segment-based interaction.

In the segment-based IMT protocol, the collaboration between human and a computer is extended. Now, the user is allowed to validate *segments* of the hypothesis, in addition to correcting wrong words. We define the segments as non-overlapping subsequences of words that must appear in future hypotheses. Fig. 4.2 represents the segment-based interaction for the previous example. In this case, in addition to the word correction, the user also validated the segment *during the night*, which must remain in successive hypotheses. This prevents the system from degrading correct parts of the hypothesis.

As before, the process starts with the MT system proposing an initial translation  $\hat{y}_1^{\hat{f}} = \hat{y}_1, \dots, \hat{y}_{\hat{f}}$ . The user then validates  $K$  ( $0 \leq K \leq I$ ) correct segments from  $\hat{y}_1^{\hat{f}}$  and introduces a word correction. With these actions, we build a feedback signal composed of multiple elements:  $f_1^K = f_1, \dots, f_K$ , where each  $f_k$  represents a validated segment of  $w_k$  words, i.e.  $f_k = \hat{y}_{i'}^{i'+w_k}$ . The word correction introduced by the user is included as a one-word segment. Note that the prefix-based approach is a particular case of the segment-based one, in which the sequence of validated segments only contains the validated prefix (including the corrected word).

At the next iteration, the system must generate a sequence of non-validated segments  $\tilde{g}_1^K = \tilde{g}_1, \dots, \tilde{g}_K$  that fills  $f_1^K$  to conform a new translation  $\tilde{y} = f_1, \tilde{g}_1, \dots, f_K, \tilde{g}_K$ .

Once again, this process is repeated until the user accepts the complete suggestion of the system



**Figure 4.2:** Segment-based IMT iteration for the same example than in Fig. 4.1. In this case, the user validates two segments and introduces a word correction. The system generates a new hypothesis that contains the word correction and keeps the validated segments. The user feedback is  $f =$  “The pain may also”, “wake”, “during the night .”. The reaction of the system is to generate the sequence of non-validated segments  $\tilde{g} = \lambda$ , “you up”,  $\lambda$ ; being  $\lambda$  the empty string. The hypothesis offered by the system is the combination of the validated and non-validated segments.

Therefore, the goal now is to generate the translation subsequences that fill the validated segments. In our statistical framework, these translation segments are obtained as in Eq. (4.5):

$$\tilde{g}_1^K = \arg \max_{g_1^K} \Pr(g_1^K | x_1^J, f = f_1^K) \quad (4.5)$$

which can be rewritten as Eq. (4.6):

$$\tilde{g}_1^K = \arg \max_{g_1^K} \Pr(f_1, g_1, \dots, f_K, g_K | x_1^J) \quad (4.6)$$

This expression is similar to the prefix-based IMT equation (Eq. (4.4)). The search process in Eq. (4.4) is limited to the space of suffixes, constrained by  $\hat{y}_1^i$ ; while in Eq. (4.6) the search space is all possible substrings of the translations of  $x_1^J$ , constrained by the sequence of segments  $f_1, \dots, f_K$ .



### 4.3 Interactive-predictive neural machine translation

The addition of interactive-predictive mechanisms to NMT systems affects the search process: the search space must be constrained, in order to take into account the user feedback and generate compatible hypotheses. In addition, a crucial requirement of interactive-predictive systems is the response time: the system should react to the user interactions in real time, in order to provide an adequate user experience. According to Nielsen (1993), a delay of 0.1 seconds or less is unnoticeable and the user has the feeling of instant reactivity. A response time between 0.1 and 1 seconds will be noticed by the user, but its flow of thought would stay uninterrupted. Finally, for Nielsen (1993) the limit for the user to keep the attention on the system is about 10 seconds. Longer delays require feedback from the system. Therefore, in order to make our systems usable, we must maintain the response times as fast as possible.

In this section, we describe the modifications of the search process that interactive-predictive scenarios require. Following the statements from Section 4.2, we distinguish between prefix-based and segment-based interactive-predictive protocols.

Recall that, as presented in Chapter 3, NMT systems generate, at each output time-step  $i$ , a probability distribution over the target vocabulary  $\mathcal{Y}$ . This is done via a linear projection to the target vocabulary space and a softmax function (Eqs. (3.10) and (3.18)). A vector  $\mathbf{p}_i$  contains the probability distribution produced by the model at time-step  $i$ . Each element of  $\mathbf{p}_i$  corresponds to an element from the set  $\mathcal{Y}$ . Therefore, the probability expression of a word  $y \in \mathcal{Y}$  at time-step  $i$  is equivalent to a forced decoding strategy, as in Eq. (4.7):

$$p(y_i = y \mid y_1^{i-1}, x_1^J; \Theta) = \bar{\mathbf{y}}^\top \mathbf{p}_i \quad (4.7)$$

where  $\bar{\mathbf{y}} \in [0, 1]^{|\mathcal{Y}|}$  the one-hot codification of the word  $y$ .

#### 4.3.1 Prefix-based interactivity

In this protocol, the user corrects the left-most wrong word of the system hypothesis Section 4.2.1. Given a translation hypothesis  $\hat{y}_1^{\hat{f}} = \hat{y}_1, \dots, \hat{y}_{\hat{f}}$ , the feedback given to the system has the form  $f = \hat{y}_1^i$ , where  $\hat{y}_1^i$  is the validated prefix together with the corrected word. The inclusion of this feedback into the NMT system is natural because sentences are generated from left to right. Given a prefix  $\hat{y}_1^i$ , only a single path accounts for it. The branching of the search process starts once this path has been covered. Introducing the user feedback  $f = \hat{y}_1^i$ , Eq. (4.7) becomes Eq. (4.8):

$$p(y_{i'} \mid y_1^{i'-1}, x_1^J, f = \hat{y}_1^i; \Theta) = \begin{cases} \delta(y_{i'}, \hat{y}_{i'}^i), & \text{if } i' \leq i \\ \bar{\mathbf{y}}_{i'}^\top \mathbf{p}_{i'} & \text{otherwise} \end{cases} \quad (4.8)$$

where  $\delta(\cdot, \cdot)$  is the Kronecker delta:

$$\delta(y_{i'}, \hat{y}_{i'}) = \begin{cases} 1, & \text{if } y_{i'} \equiv \hat{y}_{i'} \\ 0, & \text{otherwise} \end{cases}$$

This can be seen as generating the most probable suffix given a validated prefix, and fits into the statistical framework deployed by Barrachina et al. (2009) and described in the previous section.

### 4.3.2 Segment-based interactivity

In the segment-based interactive-predictive protocol, the user can perform two actions: introduce a word correction and validate segments to keep in future iterations.

The feedback signal has now the form  $f_1^K = f_1, \dots, f_K$ , where  $f_1, \dots, f_K$  is a sequence of  $K$  non-overlapping segments validated by the user. The word correction introduced by the user is inserted as a one-word segment in  $f_1^K$ .

The system must generate a new hypothesis compatible with the feedback signal. To achieve this, the problem is reformulated as the generation of the optimal sequence of non-validated segments  $\tilde{g}_1^K = \tilde{g}_1, \dots, \tilde{g}_K$ , where each  $\tilde{g}_k$  is a subsequence of words in the target language. The goal is to obtain a sequence of non-validated segments such that its combination with the sequence of validated segments provide a better translation  $y'$  according to Eq. (4.6).

Unlike the prefix-based approach, the positions of the validated segments  $f_k$  in the next hypothesis are unknown beforehand: the user only validates segments of words, not positions in the hypothesis. Therefore, validated segments cannot be introduced in a rigid way as in Eq. (4.8). The search process must be constrained in a softer way.

We propose to allow the model to decide whether the search process should be constrained or unconstrained.  $f_1^K$  and  $\tilde{g}_1^K$  are non-overlapping sequences. Hence, the words produced by the system exclusively belong either to a validated or to a non-validated segment. We can differentiate the word generation process according to whether we are generating words belonging to a validated segment or to a non-validated one.

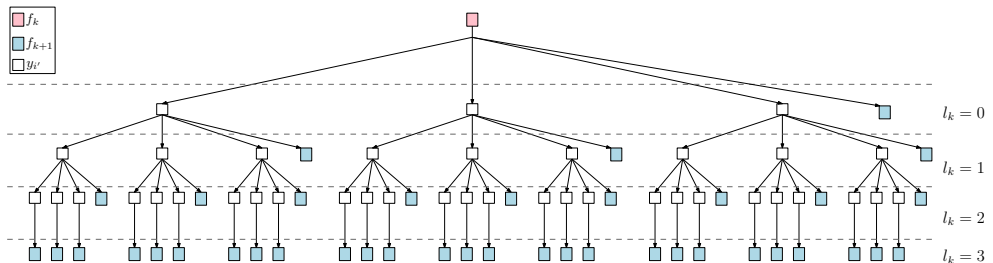
In the first case, the word generation is constrained to the words of the segment: Let  $f_k$  be the  $n$ -th validated segment and let  $i_k$  be the previous position in  $y$  where  $f_k$  should start. The word probability expression for the words belonging to this validated segment is defined as in Eq. (4.9):

$$p(y_{i_k+i'} | y_1^{i_k+i'-1}, x_1^J, f_1^K; \Theta) = \delta(y_{i_k+i'}, f_{ki'}), \quad 1 \leq i' \leq |f_k| \quad (4.9)$$

where  $|f_k|$  is the length of the validated segment  $f_k$  and  $f_{ki'}$  refers to the  $i'$ -th word of such segment.

In the second case, words belong to a non-validated segment ( $g_k$ ). Each alternative hypothesis  $y$  will (partially) have the form  $y = \dots, f_k, g_k, f_{k+1}, \dots$ , and our goal is to generate the most adequate segment. This is harder for the NMT system, because the length of this non-validated segment is unknown and needs to be estimated.

To that end, we “look ahead” and peek into the next decoding time-steps. Let  $l_k$  be the (unknown) length of this non-validated segment  $g_k$ , located between two validated segments,  $f_k$  and  $f_{k+1}$ . To estimate the optimal value of  $l_k$ , we expand the search several time-steps, until a maximum number  $L$ . After each partial hypothesis, we include the next validated segment, to compute its likelihood. Hence, we obtain an expanded search tree, in which the paths from the root to the leaves represent alternative non-validated segments ( $g_k$ ), with different values of  $l_k$  ( $0 \leq l_k \leq L$ ). We take the value of  $l_k$  that provides the most probable hypothesis, normalized by the length of the non-validated segment. Fig. 4.3 shows this branching procedure.



**Figure 4.3:** Non-validated segment length estimation for segment-based interaction. For this example, we set the maximum length of a non-validated segment to  $M = 3$  and we assume a beam size of 3. The pink item represents the last element from the previous validated segment ( $f_k$ ), blue items represent the first element from the next validated segment ( $f_{k+1}$ ). White items denote the words generated for each non-validated segment. Each path from the root of the tree to one of the leaves (blue items) represents a non-validated segment  $g_k$ . We are interested in the non-validated segment with the highest probability ( $\tilde{g}_k$ ). Once we obtained the most promising non-validated segments, we can reuse the computations made in this exploration to retrieve the one with the highest probability.

More formally, the value of  $l_k$  is estimated following Eq. (4.10):

$$\hat{l}_k = \arg \max_{0 \leq l_k \leq L} \frac{1}{l_k + 1} \sum_{i'=i_k+1}^{i_k+l_k+1} \log p(y_{i'} | y_1^{i'-1}, x_1^J; \Theta) \quad (4.10)$$

where, as before,  $i_k$  is the previous position in  $y$  where  $g_k$  should start, i.e. the last position of  $f_k$ . Note that this expression allows the model to generate non-validated

segments of length zero, resulting in the consecutive positioning of the validated segments  $f_k$  and  $f_{k+1}$ .

Once the length of the non-validated segment has been estimated, words belonging to that segment are generated following the regular procedure. We can reuse the computations made in Eq. (4.10) to evaluate this expression without additional cost, applying Eq. (4.11):

$$p(y_{i_k+i'} \mid y_1^{i_k+i'-1}, x_1^J, f_1^K; \Theta) = \mathbf{y}_{i_k+i'}^\top \mathbf{p}_{i_k+i'}, \quad 1 \leq i' \leq \hat{l}_k \quad (4.11)$$

This is a flexible strategy, that provides the system freedom to select the number of words to insert between validated segments. Hence, the system can generate the most convenient number of words between validated segments, according to its probabilistic model. In order to avoid the repetition of words, if the system started to generate words in a non-validated segment  $\tilde{g}_k$  that belongs to the next validated segment  $f_{k+1}$ , we include  $f_{k+1}$  into  $\tilde{g}_k$ , collapsing both segments.

When all validated segments have been already generated, the hypothesis is completed following the non-interactive process.

This search method is very similar to the *grid beam search* strategy developed by Hokamp and Liu (2017). The main difference is that Hokamp and Liu (2017) allow for an arbitrary reordering of the validated segments, while we require them to be supplied in order.

### 4.3.3 Character-level interaction

So far, we described interactive-predictive protocols and systems with word-level interactions. Nevertheless, it is interesting to allow the user to interact with the system at a character level. This makes it possible for a higher granularity and a more natural interaction with the system. Most of the existing IMT tools accept character-level interactions (e.g. CasMaCAT).

In the field of NMT, to perform translations at character level is a promising research direction. Character-based NMT directly allows us to perform the interactions at a character level. Unfortunately, the prohibitive decoding times of character-level NMT (Lee et al., 2017) prevent its direct usage in an interactive-predictive setup, which requires an almost instantaneous reactivity.

To overcome this limitation, we propose a simple, yet effective way of interacting with a word-level NMT system at character level (Peris and Casacuberta, 2019a). This strategy is also applicable to subword-level systems (Section 3.4.1). In a nutshell, the feedback signal will be introduced by the user at a character-level, while the system

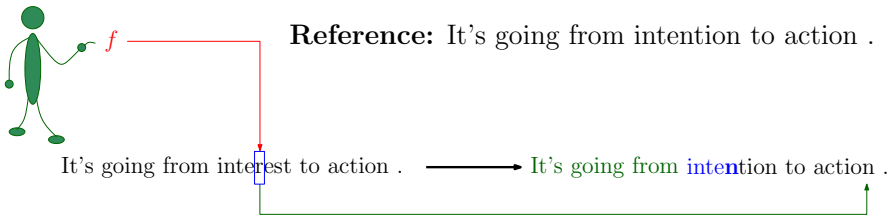
internally will still be working at a word-level. For the sake of simplicity, we describe this method for prefix-based interaction (Section 4.3.1), but it is also extensible to other protocols such as the segment-based or pick-revise frameworks.

As in the general IMT framework, the system must produce a hypothesis compatible with the user feedback. Following the prefix-based protocol, the user corrects hypotheses from left to right. But in this case, these corrections refer to a specific character from a given word. As in the word-level case, this feedback signal has a two meanings: the correct character is introduced to the system while inherently validating the hypothesis prefix, up to this character. The main difference is that in character-level interactions can be introduced in the middle of a word.

More formally, let us assume that the user introduces a correction in the  $u$ -th position of the  $i'$ -th word of the hypothesis. Therefore, the validated prefix are all words up to position  $i' - 1$  together with the validated part of the  $i'$ -th word:

$$f = (\hat{y}_0^{i'-1}, \hat{y}_{i'}^u)$$

where  $\hat{y}_0^{i'-1}$  is the sequence of validated words, up to word in position  $i' - 1$ ,  $\hat{y}_{i'}^u$  is the correct part of the word  $\hat{y}_{i'}$  together with the corrected character position  $u$ . Fig. 4.4 shows an example of this feedback signal.



**Figure 4.4:** Character-level interaction. In this case, the word “interest” is incorrect. The user introduces the character “n” in its corresponding position as feedback signal. This conveys a twofold meaning: first, it indicates to the system a correct prefix of three words (“It’s going from”, green in the image) that should be kept in future hypotheses. And second, it indicates that the fourth word must start with *inten-* (colored in blue). Considering this information, the system reacts and produces an alternative hypothesis, compatible with this feedback: “It’s going from intention to action .”. In this example, the feedback signal with the form  $f = (\hat{y}_0^{i'-1}, \hat{y}_{i'}^u)$  is instantiated as  $f = (\text{It’s going from}, \text{inten})$ .

To process this feedback, we need to generate a word constrained by the prefix  $\hat{y}_{i'}^u$ . This can be problematic because words are atomic elements for the NMT system. We tackle this issue creating a mask  $\mathbf{m}_u$  of the target vocabulary according to the user prefix  $\hat{y}_{i'}^u$ . Therefore,  $\mathbf{m}_u \in \{0, 1\}^{|\mathcal{V}|}$  is a vocabulary-sized binary vector, in which

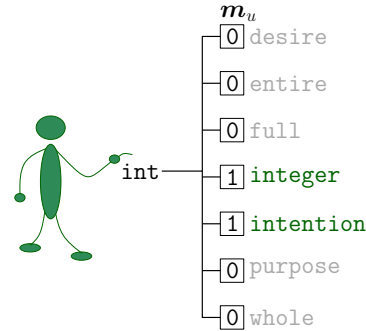
each position is set to 1 if the corresponding word in the vocabulary is compatible with the user prefix and to 0 otherwise.

If there are no compatible words with the validated prefix, we apply forced decoding to this prefix and continue the process with the unconstrained vocabulary. Hence, the output probability expression (Eq. (4.8)) is rewritten as Eq. (4.12):

$$p(y_i | y_0^{i-1}, x_1^J, f = (\hat{y}_1^{i'-1}, \hat{y}_{i'}^u); \Theta) = \begin{cases} \delta(y_i, \hat{y}_{i'}) & \text{if } i < i' \\ \bar{\mathbf{y}}_i^\top \mathbf{m}_u \odot \mathbf{p}_i & \text{if } i \equiv i' \\ \bar{\mathbf{y}}_i^\top \mathbf{p}_i & \text{otherwise} \end{cases} \quad (4.12)$$

This strategy is illustrated in Fig. 4.5. Following it, we get the benefits of character-level interaction while maintaining the decoding speed of (sub)word-level NMT. Moreover, since we keep the probabilities of each compatible word, is straightforward to implement additional features to the system, such as word completion. It is also remarkable that this vocabulary masking strategy can help the system to disambiguate words. Taking as example Fig. 4.5, if we filter the vocabulary, the system must choose between `integer` and `intention`. This reduces the possible ambiguity with other vocabulary words, such as `entire`, `full` or `whole`.

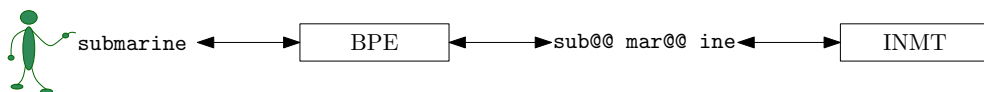
We tested this strategy in Section 4.4.2, with positive results: such a simple modification of the search process greatly reduces the effort required by the interactive-predictive systems, without significant computation overheads.



**Figure 4.5:** Constraining the vocabulary for character-level interaction. For this example, we assume a vocabulary of 7 words. The user introduces a valid prefix consisting of one or more characters (`int`). To predict the next word, the system computes a compatibility mask  $\mathbf{m}$ , which filters those words that are incompatible with the given prefix (in gray). In this case, the compatible words (in green) are `integer` and `intention`.

### 4.3.4 Dealing with unknown words

During the process of IMT, the user will likely introduce out-of-vocabulary words, that must be adequately managed by the system. The usage of word segmentation techniques such as BPE (Section 3.4.1) greatly reduce the number of unknown words. The INMT systems developed in this thesis feature a subword encoding/decoding layer: the user never revises hypotheses nor introduces feedback at a subword level. The feedback is always introduced to word-level hypothesis. Internally, feedback signals are converted to the subword level. Analogously, the NMT system generates the hypotheses at a subword level, but they are also converted into words. Fig. 4.6 exemplifies this.



**Figure 4.6:** INMT at subword level. The usage of subwords is transparent to the user. A BPE layer transforms the word-level feedback introduced by the user into subwords. Analogously, the output of the system is also converted from sub-words into words.

Despite the effectiveness of subwords, the unknown word problem is not solved and the user still can introduce out-of-vocabulary words. We address this situation similarly to the general NMT case: all out-of-vocabulary words are mapped to a special token (`<unk>`), which represents the unknown word. If the feedback provided by the user contains out-of-vocabulary words, we feed the NMT system with `<unk>` symbols. After obtaining a new hypothesis, we recover the original words introduced by the user by replacing the `<unk>` symbols produced by the system.

A future avenue of work is to explore better alternatives to deal with unknown words. An initial approach would consider the usage of a character-level NMT system as back-off system of the main one (working at word-level). The character-level system would only be used to generate unknown words. This hybrid approach has already been proposed for regular MT (Luong and Manning, 2016) and its extension to IMT would be a promising starting point.

## 4.4 Interactive-predictive neural machine translation results

We now show and discuss the results of our interactive-predictive NMT systems. We evaluate the NMT systems evaluated in the previous chapter (see Section 3.5) under the interactive-predictive framework, for all the tasks and language combinations described in Section 1.4.3. Recall that, as described in Section 1.4, we simulate the behavior of our users.

The results are organized according to the interaction level of the system: word (Section 4.4.1) or character (Section 4.4.2). Finally, we analyze additional aspects of them and raise several points of discussion in Section 4.4.3.

#### 4.4.1 Word-level interactive-predictive neural machine translation

We start by evaluating INMT systems at the word level. These systems react after the insertion of an entire word. We start by evaluating the classical prefix-based protocol (Section 4.3.1). Next, we assess the segment-based interactive-predictive protocol (Section 4.3.2). We are assessing interactions at the word level, therefore, the evaluation will be done using word-level metrics, namely, WSR and MAR (see Section 1.4.1).

Moreover, we compare the NMT technology to PB-SMT, for the prefix-based protocol. To that end, we followed the procedure described by Barrachina et al. (2009) to explore a word graph and generating the best suffix for a given prefix. For each sentence to translate, we generated a word graph with the PB system. The word graph was treated as a weighted finite-state automaton. We parsed the prefix over the word graph to find the best path that accounts for the prefix, going from the initial state to any other intermediate state. Finally, we obtained the corresponding translation for the best path from this intermediate state to the final state. The implementation of prefix-based IMT is consistent with Barrachina et al. (2009), but generating the word graphs with the current PB-SMT state-of-the-art Moses toolkit, described in Section 3.5.2.

#### Prefix-based interaction

Table 4.1 shows the results of the prefix-based interaction, for RNN-based and Transformer NMT systems, compared to PB-SMT systems. It is worth mentioning that, due to the different metrics and pre/post-processing techniques employed in the literature (e.g. corpus categorization), it is hard to directly compare the obtained results with other authors. Therefore, we computed the PB-SMT results by using our data with the software kindly provided by the authors of Barrachina et al. (2009). Hence, these results can be considered an updated version of Barrachina et al. (2009), using Moses v4.0, with our data and processing.

Regarding the results, both neural approaches, RNN-based and Transformer, performed similarly. In most cases, the differences in the results were negligible and were tightly related to differences on the translation quality of the systems (Section 3.5), rather than in the characteristics of each model.



**Table 4.1:** Results of prefix-based interaction for all tasks, measured in WSR and MAR. We compare RNN-based and Transformer (Trans.) NMT systems, and PB-SMT. Significance tests are computed between neural systems (RNN and Transformer). \* indicates significant improvements of a neural model with respect to the other.

		WSR [↓]			MAR [↓]		
		RNN	Trans.	PB-SMT	RNN	Trans.	PB-SMT
XRCE	En→De	55.1*	56.5	61.3	10.8*	11.2	13.8
	De→En	38.4*	42.2	60.9	9.4*	10.0	13.4
	En→Fr	45.7*	49.4	50.7	11.1*	12.0	13.8
	Fr→En	47.4*	51.8	49.8	10.6*	11.5	15.4
	En→Es	28.4*	32.1	27.4	7.3*	8.0	6.4
	Es→En	30.7*	37.4	26.2	7.2*	8.3	7.0
TED	En→De	45.7	45.8	64.4	9.5	9.6	16.0
	De→En	41.2*	42.6	61.1	10.3	10.6	17.7
	En→Fr	40.1*	42.2	58.1	9.6*	10.0	15.7
	Fr→En	41.1*	42.5	57.9	10.4*	10.7	17.2
	En→Zh	68.1	66.7*	93.1	28.9	29.6	65.9
	Zh→En	59.4	60.1	82.1	14.2	14.3	23.5
UFAL	En→De	53.1	51.3*	70.3	8.0	7.8*	12.4
	De→En	39.5*	40.6	61.2	7.5	7.6	13.4
	En→Fr	34.7	33.7*	51.7	6.6	6.4*	11.0
	Fr→En	37.6	37.4	49.0	7.2	7.1	10.9
	En→Es	35.3	35.4	50.9	6.6	6.6	11.1
	Es→En	32.1*	35.6	48.7	6.2*	6.7	10.8
Europarl	En→De	57.3	55.4*	75.4	10.4	10.1	16.1
	De→En	53.7	51.7*	70.1	11.4	11.0	18.2
	En→Fr	48.4	47.2*	72.0	10.2	9.9	17.0
	Fr→En	53.2	51.2*	74.4	11.4	11.0	19.0
	En→Es	51.1	44.8*	67.0	10.4	9.5*	17.1
	Es→En	50.6	50.8	60.4	10.9	10.9	15.8

Comparing the neural technology with PB-SMT systems, we observed a clear result: NMT consistently outperformed PB-SMT. In all tasks but XRCE, these differences were notoriously large, ranging effort reductions up to a 30%, in terms of WSR and MAR. In the case of the XRCE task, the performance gap between PB-SMT and NMT was lower and, for several language combinations, PB-SMT systems performed better than NMT. This behavior could be anticipated by observing the translation quality results for this task (Table 3.1): PB-SMT systems performed generally better than NMT for this small task. This is also reflected in the interactive-predictive protocol.

These large WSR and MAR differences are due to two the different nature of both approaches: neural models have, by construction, a naturally smooth behavior, in contrast to PB-SMT systems. NMT reacts better to the introduction of word corrections and presents a more solid recovery from unexpected feedback. These observations were also noticed by Knowles and Koehn (2016).

Regarding the introduction of word corrections, the modifications on the NMT search process required by interactive-predictive protocols are relatively small: when a correction is introduced into the system, only the prefix is constrained. After applying forced decoding, the system is free to complete the rest of the sentences, without additional constraints. On the other hand, PB-SMT relies on a search over a pruned word graph. This pruning can affect to potentially valid translations, that will not be reached. Therefore, modifications on the hypotheses are handled by INMT systems in a less constrained way.

Moreover, the insertion of an unexpected correction (e.g. an unknown word or a rare phrasal construction) can produce a complete failure of the PB-SMT system. If the user feedback leads to a specific state in the word graph that has no path to any final state, the system fails. An error correction method is used to overcome this issue (Barrachina et al., 2009; Ortiz-Martínez, 2011), but it may lead to successive wrong hypotheses. NMT systems handle the out-of-vocabulary problem by applying splitting unknown words, using BPE. Therefore, they are better prepared for this inconvenience.

### Segment-based interaction

We evaluate now the segment-based protocol, which is described in Section 4.3.2. Table 4.2 shows the segment-based interaction results, for both neural systems. Using this protocol, the typing effort is always diminished, at the expense of more mouse interactions: The WSR is reduced by a factor ranging from 7% up to 15%. On the other hand, MAR values are increased in all tasks.

This is an expected behavior: since the user validates segments using the mouse, its activity is higher. However, this avoids introducing errors in successive iterations on correct parts of the hypothesis. Therefore, this mouse activity increase may pay off from the user comfortability point of view. Nevertheless, due to the flexibility of the segment-based approach, users are free to choose how to use the system: a user may prefer to use more intensively the mouse, while another one may prefer to introduce more word corrections. Both user behaviors are supported by the protocol.

Despite the fact that the segment-based approach obtained better WSR than the prefix-based protocol for each task, the differences between them are rather small. This is probably due to the way in which the neural system generates the translations:

**Table 4.2:** Results of segment-based INMT for all tasks, in terms of WSR [↓] and MAR [↓]. We compare RNN-based and Transformer (Trans.) NMT systems. Significantly better results of a model with respect to the other are denoted by \*.

		WSR [↓]		MAR [↓]	
		RNN	Trans.	RNN	Trans.
XRCE	En→De	50.9*	54.7	14.9*	16.0
	De→En	35.1*	39.9	13.3*	14.1
	En→Fr	43.2*	48.1	14.6*	15.4
	Fr→En	45.1*	50.3	11.7*	12.6
	En→Es	22.7*	30.2	7.5*	12.7
	Es→En	29.1*	35.5	12.5*	13.2
TED	En→De	42.4	42.3	12.2	12.2
	De→En	39.2*	40.3	11.9	12.2
	En→Fr	37.5*	38.0	12.7*	12.9
	Fr→En	40.0	40.2	12.5	12.6
	En→Zh	58.4	56.6*	64.2	62.5*
	Zh→En	51.2	49.2*	21.2	20.4*
UFAL	En→De	51.2	50.4*	15.2	15.1
	De→En	35.9*	37.1	13.5	13.7
	En→Fr	27.9	26.8*	10.0	9.8*
	Fr→En	30.5	30.4	10.8	10.8
	En→Es	31.0	31.2	10.9	10.9
	Es→En	29.5*	33.1	10.3	10.5
Europarl	En→De	61.2	52.9*	19.9	15.9*
	De→En	57.8	55.5*	17.9	16.3*
	En→Fr	46.3	45.3*	14.3	14.1*
	Fr→En	48.5*	49.2	14.9*	15.1
	En→Es	48.8	40.2*	14.9	12.5*
	Es→En	48.6	48.9	14.1	14.1

since it follows a left-to-right direction, the prefix-based approach fits nicely into it. The inclusion of the segment-based feedback is more artificial. Although it helps the system, in light of the results, the benefits obtained are limited.

However, the segment-based framework allows the implementation of more complex user models. For instance, the user could remove words between validated segments, or drop all non-validated segments from a hypothesis, achieving the desired translation in fewer interactions. We think that higher gains could be obtained by sophisticating the user model.

#### 4.4.2 Character-level interactive-predictive neural machine translation

Next, we evaluate the introduction of character-level corrections in the prefix-based IMT protocol. Since we introduce feedback at this level, we assess the system using KSMR, which measures character-level effort.

**Table 4.3:** Effort required by interactive-predictive NMT systems, RNN and Transformer (Trans.), compared to the literature (PB-SMT, [Ortiz-Martínez, 2011, 2016](#)), in terms of KSMR (%). \* indicates significant improvements of RNN or Transformer for a given task and language pair. – indicates a missing result for this task and language combination.

		KSMR [↓]		
		RNN	Trans.	PB-SMT
XRCE	En→De	27.5*	30.4	39.1
	De→En	24.1*	27.3	39.5
	En→Fr	23.8*	27.7	34.4
	Fr→En	27.6*	31.2	37.0
	En→Es	13.9*	14.5	16.7
	Es→En	17.6	17.8	18.1
TED	En→De	26.7*	27.6	–
	De→En	25.9*	27.3	–
	En→Fr	24.0*	25.8	–
	Fr→En	26.9*	28.0	–
	En→Zh	60.2*	63.5	–
	Zh→En	39.9*	41.2	–
UFAL	En→De	23.8	21.1*	–
	De→En	27.9	19.0*	–
	En→Fr	19.0	15.9*	–
	Fr→En	19.2	17.7*	–
	En→Es	17.1	16.1*	–
	Es→En	16.1*	17.7	–
Europarl	En→De	30.6	29.5*	48.0
	De→En	33.2	25.4*	–
	En→Fr	30.1	29.4*	43.2
	Fr→En	33.1	31.5*	–
	En→Es	30.4	28.4*	45.9
	Es→En	29.1	29.5	–

[Table 4.3](#) shows the performance in KSMR of the INMT systems. We also compare these results with the best results found in the literature for each task and language combination ([Ortiz-Martínez, 2011, 2016](#)), which were PB-SMT systems.

Regarding the comparison of both neural architectures, we observed again a strong correlation on the amount of data used to train the system and their performance: Transformer models clearly outperformed RNN-based systems in those scenarios in which the training datasets were large (UFAL, Europarl). In the cases with few data (TED, XRCE), RNN-based systems worked better than Transformers. The differences were significant in most cases.

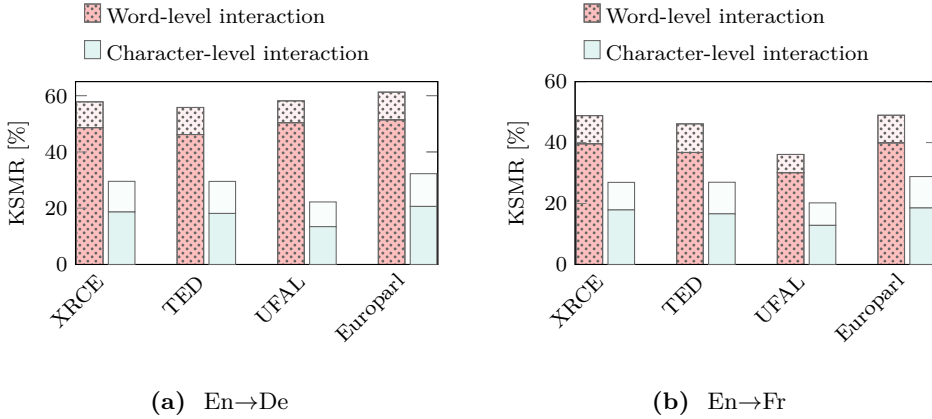
Comparing the neural systems to PB-SMT, the first technology performed substantially better. In several cases, the effort was greatly reduced, from a 20% to a 36% of relative improvement. Similarly to the observations in the word-based interaction, the performances of NMT and PB-SMT are close for the translation from/into Spanish in the XRCE task. We also observed that, again, the more training data is used to train an NMT system, the larger differences it achieves with respect to PB-SMT, as shown by the results on the Europarl corpus. It is also worth pointing out that, in terms of translation quality, our NMT systems outperformed the PB-SMT systems from [Ortiz-Martínez \(2016\)](#) by around 5 BLEU points for this task. These differences are also reflected in terms of the effort required. Finally, we observed that the differences between neural and PB-SMT systems were sharpened when translating from/into German. NMT systems were able to better model the complex structures of German. Moreover, neural models are able to properly leverage the vocabulary-masking strategy (see [Section 4.4.2](#)) and BPE, to better handle the user feedback, than PB-SMT systems. This explains the large differences in terms of effort regarding the German language.

### Evaluating the vocabulary-masking strategy

We introduced a simple yet effective way to perform character-level interactions on an NMT system that works at word (or subword) level ([Section 4.3.3](#)). A system with character-level interactions will potentially require fewer keystrokes than another based on word-level interactions, provided that it is able to correctly profit from the user feedback. On the other hand, the number of mouse actions may be increased in character-level systems, since the user can move the mouse along one word to correct it, spending more than one mouse action. In word-based interaction, words are treated as atomic units; therefore, the number of mouse actions required is potentially lower.

In order to assess the proposed character-based INMT systems, we measured the KSMR required by the same INMT system when performing interactions at either word or character level. Results are shown in [Fig. 4.7](#). From KSMR, we differentiated keystrokes and mouse actions.

According to [Fig. 4.7](#), to perform character-level interactions greatly diminished the number of keystrokes required. Reductions were around 50% in the case of French and even larger in the case of German (between 60% and 75%). This suggests that



**Figure 4.7:** KSMR of INMT systems of all tasks. We compare word-level interaction (dotted) to character-level interaction. From each bar, the upper (lighter) part represents the mouse action fraction of KSMR, and the lower part accounts for the keystrokes.

the system was able to correctly predict even the long and compounded words from German. As expected, character-based interaction slightly increased the amount of mouse actions required. Nevertheless, the increase in mouse actions was small.

Comparing both levels of interactions, conclusions are indisputable: character-level interaction is more effective than word-level, in terms of human effort required. Moreover, character-level interaction allows the user to have a more precise and natural control of the IMT process.

#### 4.4.3 Additional aspects regarding INMT

We analyze now several facets regarding the interactive-predictive framework. We study the response times of IMT systems and analyze examples of interactive-predictive sessions.

##### Response times

As discussed in [Section 4.3](#), the response time of an interactive-predictive system is an important aspect of it. The average response times for all systems are shown in [Table 4.4](#). In the neural systems, the response rates are adequate for an interactive-predictive scenario, when using a GPU<sup>2</sup>. According to [Nielsen \(1993\)](#), the user would perceive a small delay on the system response, but keeping the feeling of instant

<sup>2</sup>Experiments executed on a single GeForce GTX 1080 GPU.

reactivity. If computations were made on the CPU, the response time were increased by approximately 10 times. To deploy the interactive-predictive systems on a CPU, we should apply strategies yo scale up the system and enhancing the CPU response time (e.g. Devlin, 2017; Klein et al., 2017).

In the case of the PB-SMT systems, we measure two different scenarios, depending on whether the word graph is pre-computed or not. In the first case, response times are acceptable, allowing fluent interactivity. In the second case, response times are inadequate for the interactivity. Nevertheless, note that a comparison of the NMT system with the first scenario is unfair, because in the NMT system we do not assume any pre-computation. It should be investigated if advancing these computations may help to enhance the responsiveness of the NMT system.

**Table 4.4:** Average system response time (in seconds) per interaction, for each task. NMT-CPU refers to the execution of the RNN-based neural system in a CPU, while the column NMT-GPU show the times for the GPU execution. In the case of PB-SMT systems, we report the time of the sole exploration of the word graph (PB-SMT Search), together with the time required for both building and exploring the word graph (PB-SMT All). PB-SMT systems run on CPU. Best results for each task are bold-faced.

	NMT-GPU	NMT-CPU	PB-SMT All	PB-SMT Search
XRCE	<b>0.13</b>	1.30	1.13	0.24
TED	<b>0.21</b>	3.12	0.68	0.25
UFAL	<b>0.25</b>	2.03	1.23	0.42
Europarl	<b>0.12</b>	1.41	1.08	0.32

### Qualitative analysis: IMT session examples

Now, we qualitatively study the behavior of interactive-predictive systems, by analyzing an example from the UFAL dataset. We show the different IMT sessions, of neural and PB-SMT systems, from the translation of a French sentence into English. The source sentence is “*Ils és à fournir un échantillon d’ urine propre.*” and the desired translation is “*They will be asked to provide a clean catch urine sample.*”.

Fig. 4.8 shows the session of the PB-SMT system. First, the system generates an initial hypothesis. Following the protocol, at iteration **IT-1**, the user then corrects the first wrong word, introducing the word “*asked*” at position 4. Then, the system reacts and suggests a new suffix. Note that in this new hypothesis, the words “*asked*” and “*requested*” are consecutive in the new hypothesis: the PB-SMT system is unable of properly understand the correction provided by the user, because the feedback ‘*asked*’ and the word “*requested*” refer to the same. The next iteration starts then, with the user validating the next wrong word and the system generating a compatible

<b>Source:</b> Ils és seront invités à fournir un échantillon d’urine propre.		
<b>Target translation:</b> They will be asked to provide a clean catch urine sample.		
<b>IT-0</b>	MT	They will be invited to provide a panel of urine clean.
<b>IT-1</b>	User	<i>They will be <b>asked</b></i> invited to provide a panel of urine clean.
	MT	<i>They will be asked</i> requested to provide clean urine sample.
<b>IT-2</b>	User	<i>They will be asked <b>to</b></i> requested to provide clean urine sample.
	MT	<i>They will be asked to</i> provide a panel of urine clean.
<b>IT-3</b>	User	<i>They will be asked to provide a <b>clean</b></i> panel of urine clean.
	MT	<i>They will be asked to provide a clean</i> urine sample.
<b>IT-4</b>	User	<i>They will be asked to provide a clean <b>catch</b></i> urine sample.
	MT	<i>They will be asked to provide a clean catch</i> (urine sample).
<b>IT-5</b>	User	<i>They will be asked to provide a clean catch <b>urine</b></i> (urine sample).
	MT	<i>They will be asked to provide a clean catch urine.</i>
<b>IT-6</b>	User	<i>They will be asked to provide a clean catch urine <b>sample</b>.</i>
	MT	<i>They will be asked to provide a clean catch urine sample.</i>
<b>END</b>	User	<i>They will be asked to provide a clean catch urine sample.</i>

**Figure 4.8:** PB-SMT prefix-based IMT session to translate a sentence from French into English: given the input sentence, the user desires to obtain the target sentence. User corrections are in **bold**, while validated prefixes are in *italic*.

hypothesis. This is repeated until the desired translation is obtained. This IMT process required 6 word corrections.

Fig. 4.9 shows the same sentence, but using the RNN-based NMT system. The first hypothesis is similar to that provided by the PB-SMT system, and the first word correction introduced by the user is the same as in the previous case. But now, the NMT system was able to adapt the next hypothesis to the user feedback: The system processed the correction “*asked*” as a synonym for “*invited*”. Therefore, it is able to provide a coherent alternative. With this, the number required of iterations is reduced from 6 to 4, which accounts for a reduction of the 33% of the effort, compared to the PB-SMT system.

However, note that at **IT-1** of Fig. 4.9, the NMT system already produced the correct segment “urine sample.”. Nevertheless, this segment was lost at **IT-2**. The segment-based protocol aims to prevent this behavior and it is shown in Fig. 4.10. Now, at the first iteration, the user validates three segments and introduces a word cor-



---

<b>Source:</b> Ils é s seront invités à fournir un échantillon d'urine propre.		
<b>Target translation:</b> They will be asked to provide a clean catch urine sample.		
<b>IT-0</b>	<b>MT</b>	They will be invited to provide a clean urine sample.
<b>IT-1</b>	<b>User</b>	<i>They will be asked</i> invited to provide a clean urine sample.
	<b>MT</b>	<i>They will be asked</i> to provide a clean urine sample.
<b>IT-2</b>	<b>User</b>	<i>They will be asked to provide a clean catch</i> urine sample.
	<b>MT</b>	<i>They will be asked to provide a clean catch.</i>
<b>IT-3</b>	<b>User</b>	<i>They will be asked to provide a clean catch</i> <b>urine</b> .
	<b>MT</b>	<i>They will be asked to provide a clean catch urine.</i>
<b>IT-4</b>	<b>User</b>	<i>They will be asked to provide a clean catch urine</i> <b>sample</b> .
	<b>MT</b>	<i>They will be asked to provide a clean catch urine sample.</i>
<b>END</b>	<b>User</b>	<i>They will be asked to provide a clean catch urine sample.</i>

---

**Figure 4.9:** Real RNN-based prefix-based IMT session, considering the same sentence, protocol, and format as in Fig. 4.8, but using the NMT system.

rection (“asked”), which is included as a one-word validated segment. This feedback allowed the model to keep through the successive hypotheses the validated segment “urine sample.”. At the second iteration, since there are no new segments, the user only introduces a word correction. The system reacts then by providing the desired hypothesis. Using the segment-based approach, only two interactions are necessary, which accounts for a reduction of 66% and 50% of the number of interactions from prefix-based PB-SMT and NMT systems, respectively. However, these interactions are more costly in terms of mouse actions, because the user needs to select the validated segments.

---

<b>Source:</b> Ils es seront invités à fournir un échantillon d' urine propre.	
<b>Target translation:</b> They will be asked to provide a clean catch urine sample.	
<b>IT-0</b>	<b>MT</b> They will be invited to provide a clean urine sample.
<b>IT-1</b>	<b>User</b> <span style="border: 1px solid black; padding: 2px;">They will be</span> asked invited <span style="border: 1px solid black; padding: 2px;">to provide a clean</span> urine sample.
	<b>MT</b> <span style="border: 1px solid black; padding: 2px;">They will be</span> asked <span style="border: 1px solid black; padding: 2px;">to provide a clean</span> urine sample.
<b>IT-2</b>	<b>User</b> <span style="border: 1px solid black; padding: 2px;">They will be</span> asked <span style="border: 1px solid black; padding: 2px;">to provide a clean</span> catch <span style="border: 1px solid black; padding: 2px;">urine sample.</span>
	<b>MT</b> <span style="border: 1px solid black; padding: 2px;">They will be</span> asked <span style="border: 1px solid black; padding: 2px;">to provide a clean</span> catch <span style="border: 1px solid black; padding: 2px;">urine sample.</span>
<b>END</b>	<b>User</b> <i>They will be invited to provide a clean urine sample.</i>

---

**Figure 4.10:** Real neural segment-based IMT session. Same sentence, system, and format as in Fig. 4.9, but following the segment-based protocol. Boxed text represents validated segments. Now, in addition to correcting words, the user validates with the mouse correct segments in the hypotheses.

Finally, Fig. 4.11 shows the same example, for prefix-based interaction, but introducing the feedback at character level. As before, the user wants to introduce the word “asked”. To that end, in this case, only the first character is introduced. Only with this character, the system is able to properly generate the rest of the word. However, the generation of the word “catch” is more problematic. The system failed to propose the correct alternative, and the user had to type four out of the five characters of that word. Finally, at **IT-6**, the system also generated correctly the word “urine” from its initial character. In this example, the user only typed 6 characters. Compared to the system with word-level feedback (Fig. 4.9), which required the typing of 23, this evidences the enhancements brought by character-based feedback (quantitatively evaluated in Section 4.4.2).

Interestingly, even when failing in generating the words desired by the user, all hypotheses produced by the NMT system were fluent and coherent. The system is able to remove words when they are inappropriate, as in the hypothesis from **IT-4**. This is in contrast to PB-SMT system, which produces more artificial and unnatural hypotheses.

---

<b>Source:</b> Ils és seront invités à fournir un échantillon d' urine propre.		
<b>Target translation:</b> They will be asked to provide a clean catch urine sample.		
<b>IT-0</b>	<b>MT</b>	They will be invited to provide a clean urine sample.
<b>IT-1</b>	<b>User</b>	<i>They will be ainvited to provide a clean urine sample.</i>
	<b>MT</b>	<i>They will be asked to provide a clean urine sample.</i>
<b>IT-2</b>	<b>User</b>	<i>They will be asked to provide a clean curine sample.</i>
	<b>MT</b>	<i>They will be asked to provide a clean cholesterol sample.</i>
<b>IT-3</b>	<b>User</b>	<i>They will be asked to provide a clean ca sample.</i>
	<b>MT</b>	<i>They will be asked to provide a clean capillary sample.</i>
<b>IT-4</b>	<b>User</b>	<i>They will be asked to provide a clean catpillary sample.</i>
	<b>MT</b>	<i>They will be asked to provide a clean cateter.</i>
<b>IT-5</b>	<b>User</b>	<i>They will be asked to provide a clean catcheter.</i>
	<b>MT</b>	<i>They will be asked to provide a clean catch sample.</i>
<b>IT-6</b>	<b>User</b>	<i>They will be asked to provide a clean catch usample.</i>
	<b>MT</b>	<i>They will be asked to provide a clean catch urine sample.</i>
<b>END</b>	<b>User</b>	<i>They will be asked to provide a clean catch urine sample.</i>

---

**Figure 4.11:** Prefix-based IMT session, for an RNN-based NMT with character-level interaction, considering the same sentence, protocol, and format as in Fig. 4.8. In this case, the user corrects a character of the hypothesis and obtains an alternative.

## 4.5 Summary

In this chapter, we introduced an interactive-predictive machine translation approach, aiming to obtain high quality translations reducing the human effort spent in the process. After reviewing the most relevant works involving the topic, we presented the statistical framework on which IMT relies. Next, we developed neural interactive-predictive systems, which are compliant within such framework. This was achieved by constraining the search process of NMT decoders.

We described two main modifications of the system, which led to prefix-based and segment-based interactive-predictive systems. While the first protocol is natural for NMT systems, the second one is more challenging. We tackled the generation of non-validated segments via partial expansions of the search tree. This allows the model to search for the most suitable non-validated segments in a flexible and efficient way. We also addressed the possibility of performing character-level interactions on top

of a (sub)word-level system. This provides a more fine-grained interactivity, while keeping the decoding latency at an adequate speed.

Moreover, this framework can be applied beyond MT. It is generalizable to other tasks involving structured predictions. We explore these scenarios in [Section 6.3](#), where we apply the interactive-predictive framework to two multimodal sequence-to-sequence tasks, namely, image and video captioning. Additionally, we built a web-based demonstration for all these interactive-predictive systems<sup>3</sup>, that implements the prefix-based INMT protocol with character-level feedback. This demo is built upon the NMT-Keras package (see [Appendix A](#)) and it is also open-sourced.

We performed a wide evaluation of the interactive-predictive protocols described in this chapter. Compared to classical PB-SMT, our interactive-predictive NMT systems performed usually much better, in terms of the amount of actions required to obtain the desired translations. We conjecture that these enhancements are due to the flexibility of the neural models and their capability to adapt to the user feedback. The segment-based INMT protocol led to diminishing the typing effort required by the user, at the expense of an increase of the mouse actions. However, this is a more flexible protocol, that aims to offer more freedom to the user: the prefix-based protocol is a special case of this segment-based strategy. Hence, the user can still use the prefix-based protocol, if desired. Finally, we found that the introduction of feedback at character level although simple is effective: it approximately halved the KSMR required during the IMT process.

As final takeaway, we conclude the chapter by stating that neural IMT systems performed better than PB-SMT ones, and that performing character-level interactions is preferable to word-level interactions. However, all these assertions must be confirmed by future user studies.

---

<sup>3</sup> Accessible at: <http://casmacat.prhlt.upv.es/interactive-seq2seq>

# Adaptive neural machine translation via online learning

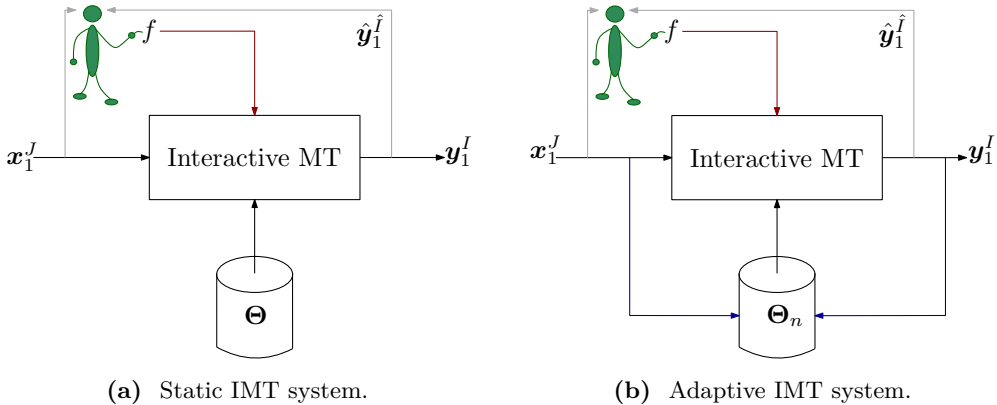
As discussed in the previous chapter, MT systems are not perfect and make mistakes. These errors are usually corrected in a post-editing phase or following the interactive-predictive MT protocol, as described in that chapter. Although efficient, this methodology can be improved in several ways: MT systems make the same mistakes over and over. Thus, users must perform repetitive corrections, which can lead to exasperation. Moreover, if a system is trained on a given domain, its performance will degrade when translating sentences from a different domain. Although this phenomenon is common to all corpus-based technologies, NMT systems are more sensitive to domain mismatch (Koehn and Knowles, 2017).

A vast amount of work is being devoted to solving these issues, which are central problems of the current research and the industrial exploitation of MT. This chapter is focused on profiting from the corrections made by the user in the post-editing or IMT processes, to improve the NMT systems. Both protocols generate new data—the corrected samples—which can be leveraged to train the system. These new data have valuable properties, that help to overcome the aforementioned problems: on the one hand, they represent in-domain data, which can help to adapt the system towards the domain of interest. On the other hand, these data contain corrections of the errors made by the system. Therefore, the system can learn from its mistakes, avoiding to make these errors again.

Moreover, we are interested in adapting the system in a continuous manner, modifying the system as soon as the corrected samples are available. This matches with the online learning paradigm, in which the training data is available sequentially and the models are updated after processing each sample (incremental learning). The typical post-editing or IMT workflow complies with these stages:

1. A source sentence comes to the system.
2. The MT system produces a translation hypothesis.
3. A human agent revises this hypothesis and corrects the errors made by the system or interactively translates the source sentence. This generates a corrected translation.
4. The corrected sample is used to adapt the system, updating the model parameters.

Following this procedure, we can build adaptive MT systems, able to take into account corrections made by the humans. Fig. 5.1 illustrates the adaptive IMT framework, compared to an static one.



**Figure 5.1:** Static and adaptive IMT systems. Figure adapted from [Ortiz-Martínez et al. \(2010\)](#). In both cases, the system produces a hypothesis  $\hat{y}_1^I$  for the source sentence  $x_1^J$ . This hypothesis is transformed, following the IMT procedure, yielding the correct translation  $y_1^I$ . After this process, in the static system (left), the MT model ( $\Theta$ ) remains unchanged; while in the adaptive system (right), the MT model is incrementally retrained with these new data, obtaining a dynamic model  $\Theta_n$ .

Continuous learning from MT post-edits or IMT includes techniques that can be leveraged to adapt a MT system to different domains, styles or users. They are orthogonal to other adaptation techniques, such as fine-tuning with in-domain data. This

chapter explores the application of OL techniques to NMT systems. First, we review the relevant literature regarding this topic. Next, we describe algorithms to perform continuous adaptation of NMT systems, proposing two novel algorithms and methods, aiming to obtain more effective systems. We also explore the application of the active learning paradigm to NMT systems. Finally, we conduct a set of experiments to assess our proposals.

## 5.1 Applications of online learning in machine translation

The application of online learning techniques to MT has been thoroughly explored in the literature. Most works aimed to adapt the MT system to a given domain or tailor it according to a given document. Classical applications include the application of OL techniques to adjust the weights of the log-linear model of PB-SMT: The margin-infuse relaxed algorithm (MIRA; Crammer and Singer, 2001) processes all samples one by one which is especially useful when dealing with a large number features. Therefore, it has been applied to tune PB-SMT with sparse features (Watanabe et al., 2007; Hasler et al., 2011; Chiang, 2012; Green et al., 2013b). The incremental estimation of the different models that conform PB-SMT systems via OL techniques has also been studied (Ortiz-Martínez et al., 2010; Ortiz-Martínez, 2016).

### Learning from post-edits

However, the most paradigmatic application of OL in the MT field is to profit from the post-edited sentences, as described above. Many advances in this direction were achieved during the CasMaCat (Alabau et al., 2013) and MateCat (Federico et al., 2014) projects. Martínez-Gómez et al. (2012) studied several OL algorithms to adjust the weights of a PB-SMT system during the post-editing phase. Bertoldi et al. (2013) proposed an adaptive PB-SMT system, based on cache components. Mathur et al. (2013) introduced additional features, which allowed to take into account the corrections made by the user. Closely related to this, Denkowski et al. (2014a) and Denkowski et al. (2014b) implemented dynamic translation and language models which, together with the tuning of weights from the log-linear model, provided a reduction of the human effort required to post-edit the outputs of a system. Sanchis-Trilles and Casacuberta (2015) studied the adaptation of the model weights by means of Bayesian learning. Lagarda et al. (2015) adapted a general PB-SMT system to a specific domain, during the post-editing stage.

This online learning scenario matches very well with the interactive-predictive framework. Consequently, their combination has also been studied. Nepveu et al. (2004) proposed adaptive language and translation models, devised for an IMT context. The IMT system introduced by Ortiz-Martínez et al. (2010) allowed the in-

cremental update of the underlying PB-SMT system and was further extended by [Ortiz-Martínez \(2016\)](#).

### Online learning in NMT

Few works studied the application of OL techniques to NMT in the post-editing scenario. [Turchi et al. \(2017\)](#) and [Peris et al. \(2017a\)](#) studied almost simultaneously and independently a similar scenario: an NMT system was refined with post-edited samples in order to perform domain adaptation. Moreover, [Peris et al. \(2017a\)](#) studied alternative training methods to perform this adaptation. [Kothur et al. \(2018\)](#) included a dictionary of translations, to deal with the novel words included in the new domain. [Wuebker et al. \(2018\)](#) proposed to apply sparse updates, to adapt the NMT system to different users.

Finally, other works aimed to tailor NMT systems towards a given user or domain, but not necessarily using online learning. [Vilar \(2018\)](#) leveraged a speaker adaptation method from speech systems ([Swietojanski et al., 2016](#)) to perform domain adaptation in MT. [Khayrallah et al. \(2018\)](#) proposed a regularized loss function, to adapt the system with few samples. [Michel and Neubig \(2018\)](#) adapted NMT to different speakers, by only modifying the output vocabulary layer of the model. NMT adaptation via reinforcement learning has also been recently studied with encouraging results ([Kreutzer et al., 2017](#); [Lam et al., 2018](#)).

### User studies on adaptive machine translation

Translation post-editing has been a widely adopted practice in the industry for a long time (e.g., [Vasconcellos and León, 1985](#)). As the MT technology advanced and was improved, the post-editing process gained more relevance and many user studies have demonstrated its capabilities ([Green et al., 2013a](#); [Bentivogli et al., 2016](#); [Castilho et al., 2017](#)).

User studies on online adaptation from post-edits have been conducted, mainly for phrase-based statistical machine translation systems ([Green et al., 2013b](#); [Denkowski et al., 2014a](#); [Alabau et al., 2016](#); [Bentivogli et al., 2016](#)). Regarding the NMT technology, several user studies have been recently conducted, analyzing different MT technologies (PB-SMT, NMT and rule-based MT [Koponen et al., 2019](#); [Jia et al., 2019](#)) or protocols (IMT versus regular post-editing, [Daems and Macken, 2019](#)). [Karimova et al. \(2018\)](#) showed savings in human effort, due to the effect of online learning. We also carried a user study which, in contrast to the aforementioned works, involved professional translators. In our case, the adaptive system also brought performance improvements and the users were pleased with this system (see [Section 5.4](#)).



## 5.2 Online learning for NMT post-editing

Typical training methods for NMT systems involve the minimization of a loss function, by means of stochastic gradient descent (Section 2.2.2). This procedure can be directly applied in an online learning setting (Murata, 1998). Hence, online learning in NMT systems can be performed using the same methods than in regular mini-batch training, but applied sample by sample.

Therefore, given the  $n$ -th training sample, consisting of a sequence of  $J$  input vectors  $\mathbf{x}_1^J = \mathbf{x}_1, \dots, \mathbf{x}_J$ , and a sequence of  $I$  reference vectors  $\mathbf{y}_1^I = \mathbf{y}_1, \dots, \mathbf{y}_I$ , an NMT system  $f$ , with parameters  $\Theta_n$ , produces a prediction from the input sequence  $\hat{\mathbf{y}}_1^I = f(\mathbf{x}_1^J; \Theta)$ . This prediction is compared to the reference, by means of the loss function  $\ell$ , typically the mean cross-entropy of the sequence (Eq. (2.6)), as in Eq. (5.1):

$$\ell(\mathbf{y}_1^I, \hat{\mathbf{y}}_1^I) = -\frac{1}{I} \sum_{i=1}^I \sum_{k=1}^K y_{i,k} \log \hat{y}_{i,k} \quad (5.1)$$

where  $y_{i,k}$  and  $\hat{y}_{i,k}$  denote the  $k$ -th elements of the  $i$ -th vectors from  $\mathbf{y}_1^I$  and  $\hat{\mathbf{y}}_1^I$ , respectively. Provided that the last activation of the NMT system is the softmax function and that the reference vectors follow a one-hot codification, this loss is equivalent to  $-\log p(\mathbf{y}_1^I | \mathbf{x}_1^J; \Theta_n)$  and its minimization matches with a maximum likelihood estimator (see Section 2.2.1).

To minimize this loss function, SGD computes an update of the parameters ( $\Delta\Theta_n$ ), to be added to the current parameters (Eq. (2.9)):

$$\Theta_{n+1} = \Theta_n + \Delta\Theta_n$$

This update follows the opposite direction of the gradient of the loss function with respect to the model parameters as in Eq. (2.10):

$$\Delta\Theta_n = -\rho \mathbf{g}_n$$

where  $\rho$  is a learning rate that controls the step size. Recall that a gradient is the vector field of partial derivatives of the elements of a vector:

$$\mathbf{g}_n = \nabla_{\Theta_n} \ell(\mathbf{y}_1^I, \hat{\mathbf{y}}_1^I) = \left( \frac{\partial \ell(\mathbf{y}_1^I, \hat{\mathbf{y}}_1^I)}{\partial \theta_1}, \dots, \frac{\partial \ell(\mathbf{y}_1^I, \hat{\mathbf{y}}_1^I)}{\partial \theta_W} \right)$$

where  $\theta_1, \dots, \theta_W$  are the elements of  $\Theta_n$ .

As described in [Section 2.2.3](#), a significant effort has been spent in the literature trying to minimize the critical importance of the learning rate choice. Adaptive SGD algorithms try to overcome this dependence by dynamically computing the learning rate.

However, we empirically found ([Section 5.3.2](#)) that these adaptive SGD algorithms do not completely alleviate the need of tuning the learning rate in our scenario of online learning for NMT. A correct choice of the learning rate becomes extremely important, even for adaptive SGD optimizers, to build solid adaptive NMT systems.

### 5.2.1 Hypergradient descent to stabilize the learning

An alternative to make the choice of the learning rate more robust, is to consider this selection as an optimization problem itself. Therefore, we can apply a gradient descent process to the learning rate, adapting it in a less heuristic way. This was devised long ago by [Almeida et al. \(1998\)](#), but remained practically unknown until its recent reintroduction by [Baydin et al. \(2017\)](#), who called it hypergradient descent.

Hypergradient descent methods differ from regular update rules ([Table 2.2](#)) in that they apply an additional gradient descent procedure on the learning rate. Therefore, SGD with hypergradient descent (SGD-HD) derives [Eq. \(2.10\)](#) with respect to the learning rate, in addition to the derivation with respect the model parameters. Hence, SGD-HD rewrites [Eq. \(2.10\)](#) as [Eq. \(5.2\)](#):

$$\Delta \Theta_n = -\rho_n \mathbf{g}_n \quad (5.2)$$

where  $\rho_n$  is the learning rate computed for the current update  $n$ .  $\rho_n$  is obtained after an additional gradient descent process, following [Eq. \(5.3\)](#):

$$\rho_n = \rho_{n-1} - \alpha \Delta \rho_n \quad (5.3)$$

where, analogously to [Eq. \(2.10\)](#),  $\alpha$  is a (hypergradient) learning rate. As in the regular SGD process,  $\Delta \rho_n$  is defined as the partial derivative of the loss function with respect to the learning rate  $\rho$  ([Eq. \(5.4\)](#), [Baydin et al., 2017](#)):

$$\Delta \rho_n = \frac{\partial \ell(\mathbf{y}_1^I, \hat{\mathbf{y}}_1^I)}{\partial \rho} \quad (5.4)$$

Putting together [Eqs. \(2.9\)](#) and [\(2.10\)](#),  $\Theta_n = \Theta_{n-1} - \rho \mathbf{g}_n$ . Therefore, applying the chain rule, [Eq. \(5.4\)](#) can be rewritten as in [Eq. \(5.5\)](#):

$$\frac{\partial \ell(\mathbf{y}, \hat{\mathbf{y}})}{\partial \rho} = \mathbf{g}_n \frac{\partial (\Theta_{n-1} - \rho \mathbf{g}_{n-1})}{\partial \rho} = \mathbf{g}_n (-\mathbf{g}_{n-1}) \quad (5.5)$$

A nice property of this derivation is that only requires to store the gradients of the past update, which is already computed in the regular SGD method. Therefore, the computational overhead of hypergradient descent is small with respect to original SGD.

It is worth mentioning that hypergradient descent can be applied not only to SGD, but also to the other update rules, such as those described in Table 2.2. Their hypergradient versions can be obtained following analogous reasoning and derivations than for the case of SGD.

### 5.2.2 Considering the hypotheses

The typical gradient descent method only takes into account the ground-truth sample to compute the updates done to the system. We hypothesize that it is interesting to also consider the hypothesis produced by the system to compute a more adequate update.

Therefore, we say that a model  $\Theta_n$  is *well-trained* if it assigns more probability to the reference sentence ( $\mathbf{y}_1^I$ ) than to any other translation hypothesis ( $\mathbf{h}_1^I$ ), satisfying Eq. (5.6):

$$p(\mathbf{y}_1^I | \mathbf{x}_1^J; \Theta_n) \geq p(\mathbf{h}_1^I | \mathbf{x}_1^J; \Theta_n) \quad (5.6)$$

Otherwise, the model is *ill-trained* and hence, it can be improved. Note that, since NMT systems are based on a suboptimal search method (typically beam search), a well-trained model can produce an incorrect hypothesis, but satisfy the restriction from Eq. (5.6). In this case, the system would incur in a search error, not a training one. However, these type of errors are very infrequent (in our experimentation, this occurred in less than 1% of the cases) and their resolution relies beyond the scope of this thesis. Note that, hypothesis and reference sequences can differ in length. We equalize them adding padding symbols, ensuring that both have the same length.

We propose two alternative methods to achieve this optimization. First, we propose an algorithm inspired by passive-aggressive (PA) techniques (Crammer and Singer, 2001; Crammer et al., 2006), which have been successfully applied to the optimization of PB-SMT systems (Hasler et al., 2011; Chiang, 2012; Martínez-Gómez et al., 2012). Second, we aim to fix some of the issues of this PA algorithm, proposing an alternative approach, inspired by the Minimax theorem (von Neumann, 1928).

### Passive-aggressive via subgradient techniques

Passive-aggressive methods (Crammer and Singer, 2001; Crammer et al., 2006) are a family of online learning algorithms, devised following the intuition of performing the minimum modifications of a wrong model (passiveness), in order to account for the true label of the current sample (aggressiveness).

Applied to NMT, we assume that our wrong model is an NMT system that does not satisfy Eq. (5.6). Our objective is to obtain a new set of parameters  $\Theta_{n+1}$  such that  $\Theta_{n+1}$  is close to  $\Theta_n$  (passiveness) but also satisfying our condition of correctness (aggressiveness). To that end, we express the correctness condition with the loss function defined in Eq. (5.7):

$$\ell_{\text{PA}}(\mathbf{x}_1^J, \mathbf{y}_1^I, \mathbf{h}_1^I, \Theta) = \log p(\mathbf{h}_1^I \mid \mathbf{x}_1^J; \Theta_n) - \log p(\mathbf{y}_1^I \mid \mathbf{x}_1^J; \Theta_n) \quad (5.7)$$

Assuming, as before, that the NMT system has a softmax output function and that  $\mathbf{h}_1^I$  and  $\mathbf{y}_1^I$  follow a one-hot codification, this loss function can be built from the cross-entropy loss ( $\ell$ , Eq. (5.1)):

$$\ell_{\text{PA}}(\mathbf{x}_1^J, \mathbf{y}_1^I, \mathbf{h}_1^I, \Theta) = \ell(\mathbf{y}_1^I, f(\mathbf{x}_1^J; \Theta)) - \ell(\mathbf{h}_1^I, f(\mathbf{x}_1^J; \Theta)) \quad (5.8)$$

Our objective can be formulated as a minimization problem with constraints (Eq. (5.9) Crammer et al., 2006):

$$\begin{aligned} \Theta_{n+1} &= \arg \min_{\Theta} \frac{1}{2} \|\Theta - \Theta_n\|^2 + C \xi \\ \text{s.t. } \ell_{\text{PA}}(\mathbf{x}_1^J, \mathbf{y}_1^I, \mathbf{h}_1^I, \Theta) &\leq \xi \text{ and } \xi \geq 0 \end{aligned} \quad (5.9)$$

where  $\xi \geq 0$  is a slack variable, included to provide more flexibility to the method and  $C$  is a parameter that controls the aggressiveness of the algorithm (Crammer et al., 2006).

From Eq. (5.9),  $\xi \geq \max(0, \ell_{\text{PA}}(\mathbf{x}_1^J, \mathbf{y}_1^I, \mathbf{h}_1^I, \Theta))$ . Considering these restrictions, Eq. (5.10) defines  $F_{\Theta}$  as a new function to optimize:

$$F_{\Theta}(\mathbf{x}_1^J, \mathbf{y}_1^I, \mathbf{h}_1^I, \Theta_n) = \frac{1}{2} \|\Theta - \Theta_n\|^2 + C \max(0, \ell_{\text{PA}}(\mathbf{x}_1^J, \mathbf{y}_1^I, \mathbf{h}_1^I, \Theta)) \quad (5.10)$$

And our end goal is to find the set of parameters that minimize this function, as in Eq. (5.11):

$$\Theta_{n+1} = \arg \min_{\Theta} F_{\Theta}(\mathbf{x}_1^J, \mathbf{y}_1^I, \mathbf{h}_1^I, \Theta_n) \quad (5.11)$$

This function is discontinuous, hence having non-differentiable regions. To solve this optimization, we rely on a subgradient method (Shor et al., 2003). This is an iterative algorithm, that aims to solve minimization problems with non-differentiable objective functions (as Eq. (5.11)). Hence, subgradient methods become useful in our scenario. A subgradient of a function  $f$  at a point  $x$ , denoted as  $\partial_x f$ , is any vector  $\mathbf{a}$  that satisfies  $f(y) \geq f(x) + \mathbf{a}^\top(y - x)$  for all  $y$  (Boyd et al., 2003). When  $f$  is differentiable, the only possible subgradient is the gradient itself. The subgradient optimization method resembles the general SGD method, but working with subgradients, as shown in Eq. (5.12):

$$\Theta^{(k+1)} = \Theta^{(k)} + \rho^{(k)} \mathbf{a}^{(k)} \quad (5.12)$$

where  $k$  denotes the iteration number,  $\mathbf{a}^{(k)}$  is any subgradient of the function to optimize with respect to  $\Theta^{(k)}$  at this iteration and  $\rho^{(k)}$  is a learning rate. We initialize this process to the current parameters of the model, setting  $\Theta_n$  as  $\Theta^{(0)}$ .

Going back to our optimization problem, we tackle it via the subgradient method. For the sake of simplicity, we set the maximum number of iterations to 1. The subgradient hence applies the weight update defined in Eq. (5.13):

$$\Theta_{n+1} = \Theta_n - \rho \partial_{\Theta^{(k)}} F_{\Theta^{(k)}}(\mathbf{x}_1^J, \mathbf{y}_1^I, \mathbf{h}_1^I, \Theta_n) = \quad (5.13)$$

$$= \Theta_n - \rho(\Theta^{(k)} - \Theta_n + \partial_{\Theta^{(k)}} C \ell_{\text{PA}}(\mathbf{x}_1^J, \mathbf{y}_1^I, \mathbf{h}_1^I, \Theta^{(k)})) \quad (5.14)$$

Considering that we perform a single iteration and that  $\Theta^{(0)} = \Theta_n$ , this expression can be simplified as in Eq. (5.15):

$$\Theta_{n+1} = \Theta_n - \rho C \partial_{\Theta_n} \ell_{\text{PA}}(\mathbf{x}_1^J, \mathbf{y}_1^I, \mathbf{h}_1^I, \Theta_n) \quad (5.15)$$

being  $\partial_{\Theta_n} \ell_{\text{PA}}(\mathbf{x}_1^J, \mathbf{y}_1^I, \mathbf{h}_1^I, \Theta_n)$  the subgradient of  $\ell$  with respect to  $\Theta_n$ , as in Eq. (5.16):

$$\partial_{\Theta_n} \ell_{\text{PA}}(\mathbf{x}_1^J, \mathbf{y}_1^I, \mathbf{h}_1^I, \Theta_n) = \begin{cases} \mathbf{g}_n & \text{if } \ell_{\text{PA}}(\mathbf{x}_1^J, \mathbf{y}_1^I, \mathbf{h}_1^I, \Theta_n) < 0 \\ 0 & \text{if } \ell_{\text{PA}}(\mathbf{x}_1^J, \mathbf{y}_1^I, \mathbf{h}_1^I, \Theta_n) > 0 \\ [0, \mathbf{g}_n] & \text{if } \ell_{\text{PA}}(\mathbf{x}_1^J, \mathbf{y}_1^I, \mathbf{h}_1^I, \Theta_n) = 0 \end{cases} \quad (5.16)$$

where, as previously,  $\mathbf{g}_n$  denotes the gradient of the loss function ( $\ell_{\text{PA}}$ ) with respect to the current parameters of the model ( $\Theta_n$ ). The latter case is solved arbitrarily by choosing  $\mathbf{g}_n$  as the subgradient. We call this passive-aggressive via subgradient methods update rule as PAS.

## A minimax algorithm for online NMT adaptation

A major issue suffered by this passive-aggressive rule is that it does not necessarily ensure that the model is improving  $\log p(\mathbf{y}_1^I | \mathbf{x}_1^J; \Theta_n)$ : a way to minimize Eq. (5.11) is by degrading both the reference and hypothesis probabilities.

The optimization process should modify the parameters of the model in a way that the probability of  $\mathbf{y}_1^I$  is higher than the probability of  $\mathbf{h}_1^I$ , while also increasing  $\log p(\mathbf{y}_1^I | \mathbf{x}_1^J; \Theta_n)$ . To that end, we propose an extension of the classical SGD method, for, on the one hand, moving the model parameters towards the direction of the gradient of the reference; and, on the other hand and only if necessary, moving the parameters in the opposite direction of the gradient of the incorrect hypothesis.

The method is detailed in Algorithm 5.1. Again, we assume that we have access to the hypothesis of the system ( $\mathbf{h}_1^I$ ) for a given source sentence and its reference translation ( $\mathbf{y}_1^I$ ). If hypothesis and reference differ (Line 4), we check whether the system is ill-trained (Line 5). If so, an iterative optimization process starts.

At each iteration, the model is updated following the regular SGD method (Lines 6 and 7): we compute an update ( $\Delta\Theta_k$ ) of the parameters as the gradient of the loss function of the reference sentence with respect to the current model parameters ( $\nabla\ell_{\Theta_k}(\mathbf{y}_1^I, f(\mathbf{x}_1^J; \Theta_k))$ ). Since we want to minimize this loss, we modify the parameters towards the negative direction of this gradient. Hence, this represents the regular SGD procedure for NMT.

Next, we check whether with this update we achieved a well-trained system. If not, we modify the system against the direction of the gradient provided by the hypothesis loss with respect to the (updated) parameters (Line 10). In this case, the model parameters are modified towards the (positive) gradient direction, yielding a gradient ascent behavior.

This loop will last until the system satisfies our desired condition (Eq. (5.6)) or it reaches a maximum number of iterations ( $M$ ). After this main loop, we return the updated parameters, to be used with the following samples.

Note that the choice of plain SGD as update rule is arbitrary. The algorithm is generalizable to other update rules (Table 2.2), or a combination of them (e.g. different optimizers for hypothesis and reference). This implies the substitution of Lines 7 and 11 in Algorithm 5.1 by the desired update rules.

**Algorithm 5.1:** Minimax gradient descent for NMT.

```

Input  :  $\ell, \ell'$  (objective functions),
            $\Theta_n$  (current NMT model),
            $\rho, \rho'$  (learning rates),
            $\mathbf{x}_1^J$  (source sentence),
            $\mathbf{h}_1^I$  (system hypothesis),
            $\mathbf{y}_1^I$  (reference sentence),
            $M$  (maximum updates allowed per sample)
Output :  $\Theta_{n+1}$  (updated NMT model)
1 begin
2    $k = 0$ 
3    $\Theta_k = \Theta_n$ 
4   if  $\mathbf{y}_1^I \neq \mathbf{h}_1^I$  then
5     while  $p(\mathbf{y}_1^I | \mathbf{x}_1^J; \Theta_k) \leq p(\mathbf{h}_1^I | \mathbf{x}_1^J; \Theta_k)$  do
6        $\Delta \Theta_k = \nabla_{\Theta_k} \ell(\mathbf{y}_1^I, f(\mathbf{x}_1^J; \Theta_k))$ 
7        $\Theta_{k+1} = \Theta_k - \rho \Delta \Theta_k$ 
8       if  $p(\mathbf{y}_1^I | \mathbf{x}_1^J; \Theta_{k+1}) \leq p(\mathbf{h}_1^I | \mathbf{x}_1^J; \Theta_{k+1})$  then
9          $k = k + 1$ 
10         $\Delta' \Theta_k = \nabla_{\Theta_k} \ell'(\mathbf{h}_1^I, f(\mathbf{x}_1^J; \Theta_k))$ 
11         $\Theta_{k+1} = \Theta_k + \rho' \Delta' \Theta_k$ 
12       $k = k + 1$ 
13      if  $k \geq M$  then
14        break
15   $\Theta_{n+1} = \Theta_k$ 
16  return  $\Theta_{n+1}$ 

```

### 5.3 Results on NMT adaptation via online learning

We now evaluate the online learning techniques applied to the systems already analyzed in the previous chapters. As described in [Section 5.2](#), the systems are updated on the fly, during the error correction process. This correction process can be done in a static post-editing way and under an IMT framework. We will evaluate both approaches. In light of the results obtained in [Section 4.4.2](#), we will perform the experiments with a prefix-based INMT system, with the feedback introduced at character level, as described in [Section 4.3.3](#).

Due to the high cost of involving human users into the experimentation process, this study will be conducted using simulated users. Following common practices (e.g., [Ortiz-Martínez, 2016](#)), we used the reference sentences of our datasets as the translations post-edited by the users. Moreover, [Section 5.4](#) presents the same experimental scenario, but with real, professional users.

In this section, the systems were deployed and evaluated according to the following steps: we start from our the systems we trained in previous chapters and, as before, our task is to translate the test set (Table 1.1). The difference is that now, after translating each sentence, we corrected it (via regular post-editing or INMT) and used the corrected sample to update the models, following the OL procedure. Hence, the systems were updated using the samples from the test documents. The evaluation was performed regularly: comparing system hypotheses with the references. The main difference is that adaptive systems will generate the hypotheses after being updated with previous references.

In order to develop an effective adaptation protocol via online learning, we need to answer several questions:

1. What is the potential effectiveness that the adaptation process can bring?
2. What is the best optimizer to use, including the most suitable update rule and the number of updates per sample to perform?
3. What are the different use-cases of an adaptive system and how does it behave in each use-case?

In the following sections, we will evaluate these aspects, for all tasks presented in Section 1.4.3. However, to manage the amount of language combinations, we will focus on the En→De and En→Fr language combinations.

### 5.3.1 On estimating the potential effectiveness of the adaptation

Since we face an adaptation scenario, the effectiveness of this process heavily depends on some characteristics of the data involved in the process. The adaptation process will be more effective if we deal with repetitive documents, because the knowledge from this document can be exploited more frequently than in non-repetitive documents. Moreover, we have to pay attention to the amount of novel information introduced by the data used for adaptation: the more novel knowledge this data includes, the more effective will be the adaptation. These two facets can be estimated beforehand, by measuring some features of the data:

**Restricted repetition rate (RRR, Ortiz-Martínez, 2016):** It is a measure of the repetitiveness of a document, built upon the repetition rate metric (Bertoldi and Federico, 2009). RRR is computed as the rate of non-singleton  $n$ -grams (with  $n$  from 1 to 4) that appear in the test document and were not present in the data used to train the original system.



**Unseen  $n$ -gram fraction (UNF, Ortiz-Martínez, 2016):** ratio of unseen  $n$ -grams from the test document in the original training set. As in RRR, there are considered  $n$ -grams from order 1 to 4.

According to what has been discussed before, the adaptation process will likely be more effective for documents tasks with high values of RRR and UNF (Ortiz-Martínez, 2016). Table 5.1 shows the values of these metrics for our tasks. In all cases, we found some repetitiveness and unseen  $n$ -grams. Hence, the system will likely benefit from an adequate adaptation process.

**Table 5.1:** Restricted repetition rate (RRR) and unseen  $n$ -gram fraction (UNF) of the XRCE, TED, UFAL and Europarl test partitions.

		RRR [%]	UNF [%]
XRCE	De	14.5	23.4
	En	15.5	11.5
	Fr	17.0	14.8
	En	16.7	15.4
TED	De	2.9	22.2
	En	4.2	13.9
	Fr	3.4	14.4
	En	4.4	14.2
UFAL	De	1.3	24.8
	En	3.1	13.7
	Fr	3.9	10.9
	En	3.4	14.6
Europarl	De	3.1	24.5
	En	3.1	18.6
	Fr	4.7	15.9
	En	4.2	18.4

Particularly relevant is the high RRR of the XRCE task. This task refers to printer manuals, and therefore contains very repetitive structures (e.g. page headers, similar usage instructions for different devices, etc). The UNF values were also high for this task, denoting that test documents contain novel information, profitable during the adaptation process. It is also worth mentioning that the UNF values for the German language were always higher than for other languages. This is unsurprising, because of the agglutinative characteristics of the German language, which makes the appearance of unseen structures more likely.

### 5.3.2 Choosing an optimizer

We now explore the behavior of the different optimizers in the OL scenario. First, we will study the differences in translation quality that the different SGD update rules provide, obtaining insights of their more suitable hyperparameters for each task. Next, we will apply several update rules to each training sample, intending to make the adaptation process more effective.

#### Determining the update rule

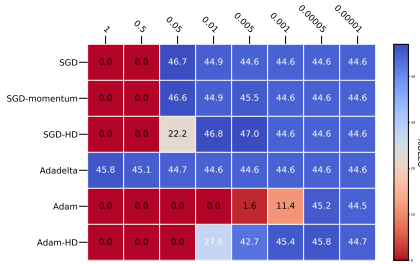
To evaluate the behavior of the different update rules, we performed a grid search over the validation set of each task, studying the effects of each one. We compared plain SGD, SGD with momentum, Adadelta, Adam and their hypergradient versions. We left the learning rate as the only hyperparameter to tune, fixing the rest to their defaults. We explored learning rates in the values:  $b \cdot 10^e$ ,  $b \in \{1, 5\}$ ,  $e \in \{1, -1, -2, -3, -4, -5, -6\}$ . In the case of hypergradient descent algorithms, we also tuned the hypergradient learning rate, exploring the values  $10^e$ ,  $e \in \{-3, -4, -5\}$ . For most configurations, the best hypergradient learning rate was  $10^{-4}$ .

**Fig. 5.2** condenses the exploration for one language pair (En→De) for all tasks and for the RNN-based model. The results were alike for the rest of tasks and are not shown for the sake of readability. We observed several behaviors across all the tasks.

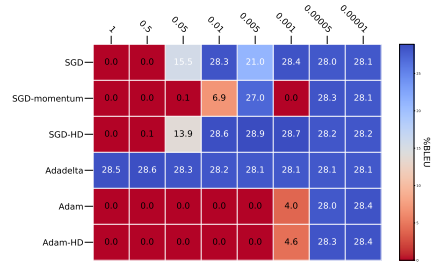
As expected, the choice of the learning rate was important: an excessively large learning rate broke the system, while small learning rates left the model parameters unchanged. The application of a momentum term to SGD had little effect.

Algorithms that aim to reduce the importance of the learning rate (Adadelta and hypergradient descent ones) yielded the best results for all tasks. More interestingly is that, in most cases, they worked well using learning rates of 0.005 in the case of SGD-HD and 1.0 in the case of Adadelta. This stability is important for situations without a development corpus. This is because, by construction, these algorithms make the choice of the learning rate less critical than other optimizers, performing well without requiring an excessive tuning.

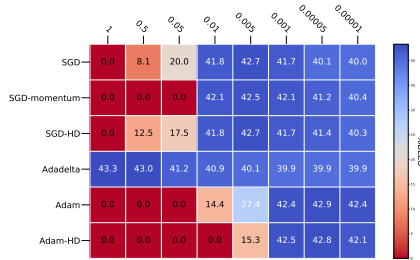
Finally, we found that Adam was excessively aggressive to be useful for OL in NMT. It required very small learning rates, otherwise, the model was completely distorted. And even with such small learning rates, it always performed worse than Adadelta or SGD-HD.



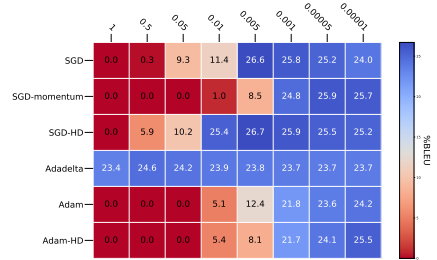
(a) XRCE En→Fr. BLEU without adaptation: 44.6.



(b) TED En→Fr. BLEU without adaptation: 28.1.



(c) UFAL En→Fr. BLEU without adaptation: 39.9.



(d) Europarl En→Fr. BLEU without adaptation: 23.7.

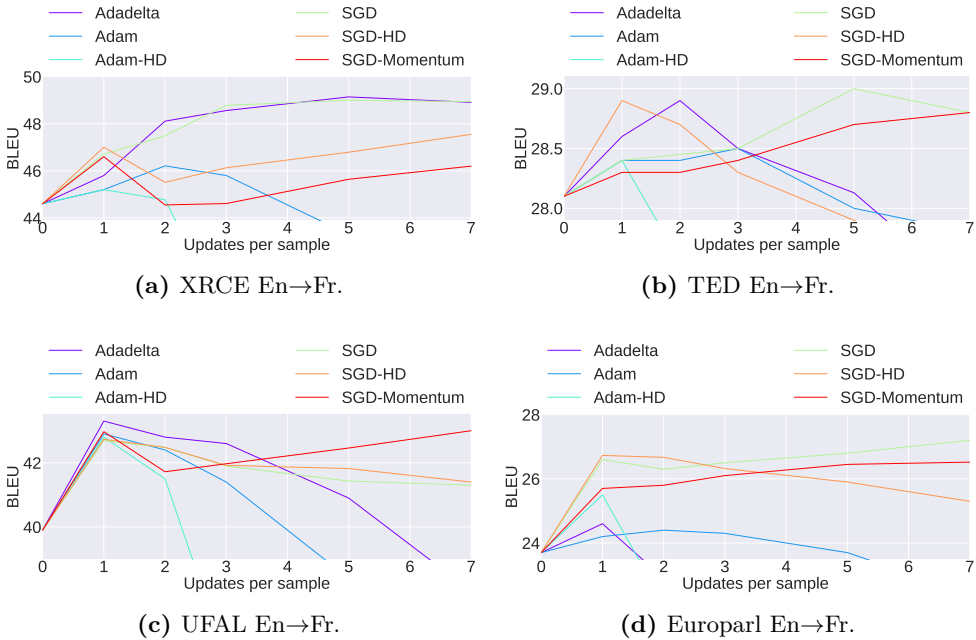
**Figure 5.2:** Translation quality (BLEU), on the development set, depending on the optimizer chosen (rows) and the learning rate (columns).

### On performing multiple updates per sample

We now study the behavior of the different optimizers when applying more than one update per sample. To that end, we followed a similar procedure than in the previous section, evaluating all the alternatives. We profited from the exploration we already carried out to choose the best hyperparameters for each optimizer and language pair. However, since we are performing more updates per sample, we also explored the application of smaller learning rates, to perform less aggressive updates. These smaller learning rates were chosen around the top-performing ones from Fig. 5.2.

In Fig. 5.3, we show the evolution of the translation quality according to the number of updates performed per sample. For comparison, we show the same tasks and language pairs than as Fig. 5.3.

All tasks but XRCE follow similar trends: compared to static systems (0 updates per sample), there are notable improvements when applying a single update per sample. But in most cases, the application of more updates has small effect on the final translation quality: those optimizers that performed worse in the single up-



**Figure 5.3:** Translation quality (BLEU), on the development set, depending on the number of updates per sample.

date regime, are eventually able to reach the performance of the better optimizers, when applying multiple updates. For instance, in the TED task, both SGD and SGD-Momentum reach similar performances than SGD-HD when applying 5 or 7 updates per sample. Similar behaviors hold for the UFAL and Europarl tasks.

On the other hand, the XRCE task has a completely different behavior: for this document, the more updates applied the better. This is probably due to the repetitive nature of this test set (see Section 5.3.1). Since this is a very repetitive document, the application of multiple updates per sentence makes the model overfit to this domain, which is beneficial.

Interestingly, performing multiple updates per sample with Adam and Adam-HD is harmful: in all cases, the translation quality was greatly degraded. This could be forecasted from the exploration carried out in the previous section, in which those optimizers were shown to be fragile in an online learning regime, requiring carefully tuned hyperparameters. The application of such optimizers with slightly different settings (multiple updates per sample) makes these hyperparameters be no longer adequate and, therefore, disrupting the translation system.

**Table 5.2:** Best online optimizer for each task.

		Algorithm	$\rho$	Updates
XRCE	En→De	Adadelta	0.5	5
	En→Fr	Adadelta	0.5	5
TED	En→De	SGD-HD	0.001	1
	En→Fr	SGD	0.005	5
UFAL	En→De	Adadelta	1.0	1
	En→Fr	Adadelta	1.0	1
Europarl	En→De	Adadelta	0.5	1
	En→Fr	SGD-HD	0.005	1

Finally, [Table 5.2](#) shows the best configuration found for each task and language pair. Adadelta and SGD-HD outperformed other algorithms. Moreover, the differences between them were small. Therefore, we can conclude that these are the most suitable optimizers for OL in NMT.

### 5.3.3 Alternative algorithms for online learning

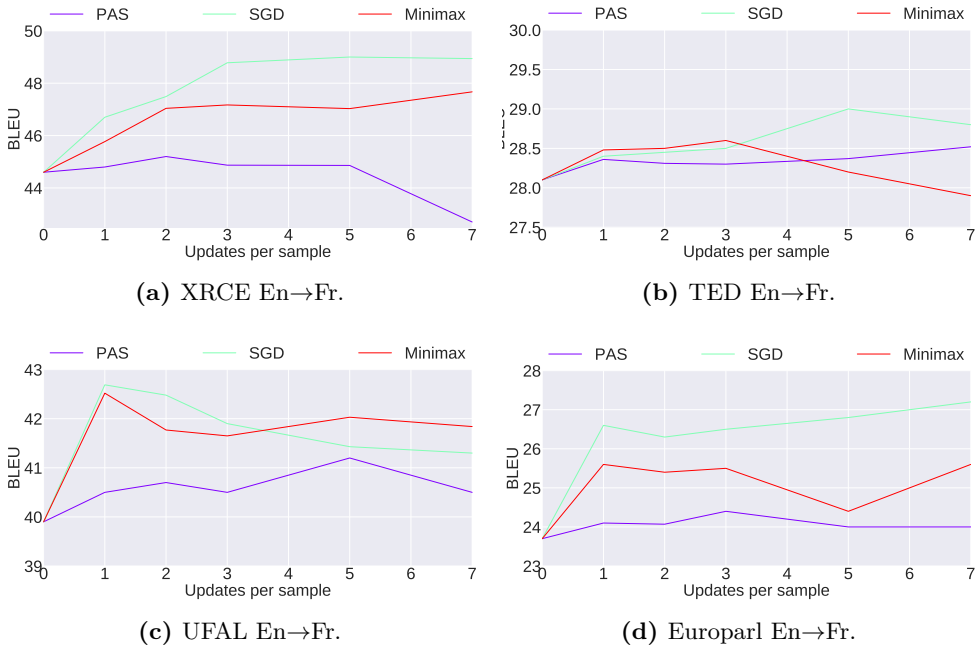
We now evaluate the adaptation algorithms proposed in [Section 5.2.2](#), that considered the hypotheses for computing the update rule, namely, PAS and Minimax.

In [Fig. 5.4](#), we show the evolution of the translation quality according to the number of updates performed per sample, for the PAS and Minimax methods, compared to SGD. We show the same tasks and language pairs than in [Fig. 5.2](#).

We observed a similar behavior in all tasks: systems featuring online learning improved with respect to offline systems (0 updates per sample). However, both PAS and Minimax methods were unable to outperform the plain SGD method, in terms of translation quality. The Minimax method behaved consistently better than the PAS method and it was more stable. Its performance was close to SGD, although it was unable to beat it. The PAS method performed clearly worse than the other methods.

Hence, we failed in our attempts to introduce the hypotheses generated by the system into the loss function. A major cause for that is the independence (up to some point) of the network objective function (cross-entropy of the target sentence given the source) with respect to the end objective of the system (translation quality, evaluated in terms of BLEU or TER).

To overcome this issue, alternative training methods aim to directly optimize the evaluation metric, rather than a neural loss function. Among them, minimum risk training has proven to be very effective for PB-SMT ([Och, 2003](#); [Chiang, 2012](#))



**Figure 5.4:** Translation quality (BLEU), on the development set, depending on the number of updates per sample for the PAS and Minimax algorithms, compared to regular SGD.

and also for NMT (Shen et al., 2016). Tightly related to this, the application of the reinforcement learning paradigm can also be used to overcome this independence. The application of reinforcement learning to sequence-to-sequence problems is currently a hot research topic (Ranzato et al., 2015; Xu et al., 2015; Shen et al., 2016). And more specifically, its application to NMT has provided positive results (Wu et al., 2016; Kreutzer et al., 2017; Gu et al., 2017; Edunov et al., 2018). In light of these results, we should move towards these paradigms to build more effective adaptive systems in the future.

### 5.3.4 Evaluating adaptive systems under different conditions

Once we evaluated the different optimizers, we move to the third research question raised at the beginning of this section (Item 3). To this end, we analyzed the capabilities and limitations of adaptive NMT systems under three different experimental conditions, varying the amount and type of training data available. In each case, we compared the performance of adaptive to static systems, in translation post-editing and in IMT. The evaluation was always carried out on the test set of each task.

Hence, OL techniques were also applied exclusively on this test data. The three cases we studied are:

1. **Exclusive use of in-domain data:** To train the NMT systems we only use task-specific data. This is the more standard setup and it has been followed in [Sections 3.5](#) and [4.4](#).
2. **Lack of in-domain data:** We assume that we only have available the test document to translate and a system already trained on a general dataset.
3. **Fine-tuning a general system:** As in the previous case, we have a system trained on a general domain. In addition, we use the training set from each task to fine-tune this general system.

### Scenario #1: Availability of in-domain data

In our first experimental scenario, we assume that we have enough in-domain data to train a system. Therefore, we follow the traditional pipeline in MT and the one followed in [Sections 3.5](#) and [4.4](#): we trained translation systems using corpora from a given domain and translated documents belonging to the same domain. In this case, online learning techniques aim to refine each system to the test documents.

We took advantage of the exploration carried out in [Section 5.3.2](#) to choose, for each task and language pair, the most suitable algorithm and hyperparameters in the online adaptation setup.

### Translation post-editing with online learning

[Tables 5.3](#) and [5.4](#) show the effect of including OL in the post-editing process, to RNN-based and Transformer systems, respectively. In almost every case, the effort required, measured in terms of TER, is improved, yielding significant reductions in most cases. BLEU is consequently improved, following the TER trend.

The largest improvements were obtained in the XRCE task, with gains ranging from 5.4 to 8.4 TER points. This was due to the high repetitiveness of the task (measured in terms of RRR) and to the large amount of new information introduced by the test document (UNF in [Table 5.1](#)). This makes the XRCE task especially adequate for incremental adaptation. We also observed significant gains in the UFAL and Europarl tasks.

The effects of adapting via OL were slightly smaller in the TED task. Although the adaptive systems performed usually better than static ones, in most cases the differences were small and non-significant in terms of TER. In terms of BLEU, these dif-

**Table 5.3:** Translation results of static (RNN) and adaptive (OL-RNN) RNN-based NMT systems, in terms of TER and BLEU, for all tasks. Significant improvements of adaptive systems are denoted by \*.

		TER [↓]		BLEU [↑]	
		RNN	OL-RNN	RNN	OL-RNN
XRCE	En→De	63.0	55.2*	25.4	34.6*
	En→Fr	51.9	43.5*	38.0	46.2*
TED	En→De	54.8	54.4	25.6	26.6*
	En→Fr	49.6	49.6	33.5	34.0*
UFAL	En→De	55.6	55.6	23.7	24.7*
	En→Fr	46.1	42.9*	37.2	40.8*
Europarl	En→De	67.0	63.3*	18.4	19.4*
	En→Fr	60.5	58.2*	24.6	25.7*

ferences were more consistent and adaptive systems significantly outperformed static ones in all cases but one.

**Table 5.4:** Translation results of static (Trans.) and adaptive (OL-Trans.) Transformer NMT systems, in terms of TER and BLEU, for all tasks. Significant improvements of adaptive systems are denoted by \*.

		TER [↓]		BLEU [↑]	
		Trans.	OL-Trans.	Trans.	OL-Trans.
XRCE	En→De	64.3	57.9*	23.2	30.9*
	En→Fr	57.2	51.8*	32.2	38.4*
TED	En→De	57.1	56.5	23.1	23.5*
	En→Fr	52.5	52.5	30.6	31.0
UFAL	En→De	55.4	53.2*	24.0	25.9*
	En→Fr	45.9	40.5*	37.8	43.2*
Europarl	En→De	63.8	63.2*	19.1	19.4*
	En→Fr	57.4	56.4*	26.6	27.4*

Compared to the existing literature, [Ortiz-Martínez \(2016\)](#) applied OL techniques to a PB-SMT system, for the XRCE and Europarl tasks and using the same data splits. In the first case, they also obtained large improvements in terms of BLEU (11.5 and 8.4, for En→De and En→Fr, respectively). Nevertheless, their baseline systems performed worse than ours. Including OL, they achieved a similar performance for the XRCE task to our online NMT systems. In the Europarl case, they improved their



static system by 1.0 and 1.4 BLEU points, for En→De and En→Fr. But again, their static systems were much worse than ours (13.1 and 21.2, for En→De and En→Fr, respectively).

### Interactive-predictive machine translation with online learning

Next, we move towards the deployment of adaptive, interactive-predictive NMT systems. We used the same configuration as in MT post-editing. Table 5.5 shows the effect of adding OL to the systems. The static version of these INMT systems were already discussed in Section 4.4.2. Moreover, we show other results obtained in the literature from Ortiz-Martínez (2016).

**Table 5.5:** INMT results, in terms of KSMR for all tasks for RNN-based and Transformer (Trans.) models. Significance tests were computed for static and adaptive (OL-) systems. Significant improvements of adaptive systems are denoted by \*. We also show the best results found in the literature, PB-SMT systems from Ortiz-Martínez (2016).

		KSMR [↓]				
		RNN	OL-RNN	Trans.	OL-Trans.	Literature
XRCE	En→De	27.5	24.6*	30.4	27.9*	37.0
	En→Fr	23.8	19.7*	27.7	24.0*	30.3
TED	En→De	26.7	25.9	27.6	27.1	–
	En→Fr	24.0	23.6	25.8	25.4	–
UFAL	En→De	23.8	20.9*	21.1	20.2*	–
	En→Fr	19.0	15.5*	15.9	14.7*	–
Europarl	En→De	30.6	28.8*	29.5	28.8*	48.0
	En→Fr	30.1	28.2*	29.4	27.5*	43.2

We found that adaptive INMT systems consistently outperformed static ones. As for translation post-editing, these differences were significant in all tasks but TED. Again, the XRCE task is the most benefited by OL, but we also obtained especially good results in the Europarl corpora. Besides their RRR and UNF values, it should also be noticed that the Europarl test documents had more samples than others. Therefore, the INMT system benefited from a longer adaptation process.

Compared to the literature (Ortiz-Martínez, 2016), we obtained similar gains in terms of KSMR for the XRCE task (around 5 KSMR points). In the case of Europarl, we obtained higher KSMR decreases: 3.9/1.8 against 1.2/1.2, for the En→De and En→Fr language pairs, respectively. Moreover, the large advantage in KSMR that INMT systems had with respect PB-SMT models is maintained in the online version.

## Scenario #2: Lack of in-domain data

In this second scenario, we assume that we have no in-domain training data available. This can be the case of a system trained with data from a general domain, but having to translate documents from a different (and potentially unknown) domain. We take advantage of OL to perform domain adaptation on-the-fly, from the general to the test domain. We expect to obtain better system hypotheses online learning processes goes on. The refinement of the system will hopefully entail a decrease of the human effort required to post-edit the upcoming samples.

As general systems, we took those trained on the Europarl corpus. In order to work with the same vocabulary, we applied the same BPE segmentation to all in-domain sets. Table 5.6 shows the vocabulary coverage of the general NMT systems and each one of the in-domain documents, together with the RRR and UNF metrics for each task. All values were computed according to the BPE version of each test document.

**Table 5.6:** Vocabulary coverage after applying BPE ( $C$ ), restricted repetition rate (RRR) and unseen  $n$ -gram fraction (UNF) with respect to the out-of-domain corpus (Europarl).

		Training	Development	Test		
		$C$	$C$	$C$	RRR [%]	UNF [%]
XRCE	De	98.7	99.8	99.5	20.6	38.8
	En	99.7	99.9	99.9	24.5	24.8
	Fr	98.2	97.5	97.4	24.9	38.4
	En	98.5	97.2	97.2	23.8	31.1
TED	De	98.1	99.8	99.9	3.2	19.2
	En	99.2	99.9	99.9	6.2	14.5
	Fr	97.9	98.7	99.0	4.6	11.1
	En	98.3	98.7	98.8	6.1	13.4
UFAL	De	92.2	99.8	99.7	2.6	39.2
	En	85.9	99.8	99.8	6.5	32.3
	Fr	91.3	99.7	99.7	8.1	25.5
	En	92.5	99.7	99.8	6.5	32.0

The vocabulary coverage was very high for all tasks (in all cases over 97%), showing that BPE can effectively leverage vocabulary differences among domains. The UNF values were increased with respect to the original training corpora for the XRCE and UFAL tasks (Table 5.1). This is unsurprising: as we work with different but quite specialized domains (technical and medical, respectively), we now have more  $n$ -grams from our test documents which were unseen in the Europarl training data. On the

other hand, it is also worth pointing out that in the case of the TED task, the UNF values were similar to those shown in [Table 5.1](#). This indicates that the TED task contained structures ( $n$ -grams) that also appeared in the Europarl corpus. This is plausible, as the TED corpus contains talks that use a general language, which was also contained in the Europarl corpus, and has less domain-specific terms than the XRCE and UFAL tasks. The RRR values also increased in all cases. This indicates that domain-specific structures, not present in the out-of-domain dataset, were repeated along the in-domain document. Hence, learning these unseen and repetitive structures will have an important impact on the quality of the adapted system.

As we assumed no in-domain data and a potentially unknown domain, we lacked development sets for this task. Hence, we used the same algorithm and learning rate for all tasks. We took advantage of the exploration carried out in [Section 5.3.2](#). Following [Table 5.2](#), we applied Adadelta with a learning rate of 0.5. Since now we are adapting a general system, we want to make the adaptation process more aggressive. Therefore, we applied three updates per sample.

### Translation post-editing with online learning

First, we compare the effort required for post-editing the outputs of the neural system. [Table 5.7](#) shows the results of translation quality, in terms of TER and BLEU, for static and adaptive RNN-based NMT systems. The results for the Transformer model are shown in [Table 5.8](#). As expected, the translation quality was lower than in the previous scenario.

**Table 5.7:** Translation quality for all tasks in terms of TER and BLEU. We compare static (RNN) and adaptive (OL-RNN) RNN-based NMT systems, exclusively trained on the general domain (Europarl). Significant improvements of adaptive systems are denoted by \*.

		TER [↓]		BLEU [↑]	
		RNN	OL-RNN	RNN	OL-RNN
XRCE	En→De	86.4	71.3*	6.3	16.7*
	En→Fr	87.0	66.4*	12.8	22.0*
TED	En→De	64.0	56.8*	20.4	24.6*
	En→Fr	59.4	55.5*	26.3	29.6*
UFAL	En→De	71.3	61.8*	14.8	19.8*
	En→Fr	55.2	52.0*	29.0	31.5*

The degradation of the systems was especially severe for the XRCE task, for both the RNN and Transformer. This was mostly due to the features of this corpus. The XRCE task relates to printer manuals and contains many short sentences, referring to

**Table 5.8:** Translation quality for all tasks in terms of TER and BLEU. We compare static (Trans.) and adaptive (OL-Trans.) Transformer systems exclusively trained on the general domain (Europarl). Significant improvements of adaptive systems are denoted by \*.

		TER [↓]		BLEU [↑]	
		Trans.	OL-Trans.	Trans.	OL-Trans.
XRCE	En→De	86.0	78.0*	6.4	10.5*
	En→Fr	78.0	68.1*	14.5	20.5*
TED	En→De	61.3	55.8*	21.2	25.2*
	En→Fr	57.0	53.6*	27.2	31.6*
UFAL	En→De	67.5	62.4*	15.9	18.7*
	En→Fr	53.1	49.8*	29.1	33.2*

technical details. Additionally, such manuals usually have some formatting templates. Since the system had never faced such templates, it made many mistakes. Moreover, note that the TER values were extremely high in this task. This phenomenon, also observed by [Chinea-Rios et al. \(2017\)](#) in a similar case, is due to the translation of short sentences with an NMT system trained on long sentences from a different domain (Europarl). Therefore, the system usually generated excessively long hypotheses. Therefore, in order to match the reference, the TER metric must perform many delete operations. To address this problem via search heuristics, we restricted the length of the output sequence to be three times the length of the input sequence. However, even though this heuristic was effective, the performance of the system was heavily affected.

On the other hand, the UFAL and TED corpora are closer to Europarl. Although the domains are different (medical and a variety of talks), their style, constructions and template of are similar to Europarl. Therefore, the differences in terms of translation quality were smaller. To illustrate this, [Fig. 5.5](#) shows examples of common sentences from all tasks.

As expected, the development of adaptive NMT systems greatly improved the quality of the systems. The improvements brought by continuous learning to the XRCE task were very large: TER was improved by 15.1 and 20.6 points in the case of RNN-based systems and by 8.0 and 9.9 points in the case of the Transformer. BLEU was also largely improved by 10.4 and 9.2 in the case of the RNN system and by 4.1 and 6.0 in the case of the Transformer. These large improvements were due to the aforementioned structure features of this text. Since the text was extracted from printer manuals, the restricted repetition rate was extremely high: more than 25% in all cases (see [Table 5.6](#)).

XRCE	<i>* press "select" to save the setting. * press "output" to select "on".</i>
TED	<i>Everybody talks about happiness these days. I'm going to talk today about energy and climate.</i>
UFAL	<i>It's a long , hollow tube at the end of your digestive tract where your body makes and stores stool. We are also studying how their work affects the quality of their lives.</i>
Europarl	<i>Resumption of the session I declare resumed the session of the European Parliament adjourned on Friday 17 December 1999, and I would like once again to wish you a happy new year in the hope that you enjoyed a pleasant festive period.</i>

**Figure 5.5:** Examples of the XRCE, UFAL and TED tasks, in English. All sentences belong to the corresponding test set. Sentences from the XRCE corpus are short and highly structured, while the other tasks have a more natural style.

OL is more effective in texts with high RRR, since upcoming events have already been seen. This effectiveness is boosted by our experimental conditions: we were translating with a general NMT system. Therefore, the system were prone to make the same errors over and over again. As we introduced continuous learning in the system, it rapidly adapted to the XRCE features. Since the XRCE test set is quite repetitive, the OL-NMT avoided to make the same error again, which had a great impact in effort reduction and translation quality. Again, we refer the reader to [Fig. 5.5](#) for a qualitative insight of this phenomenon.

Moreover, during the adaptation process, the general system learned to produce shorter sentences. In the case of the XRCE task, for En→Fr, the unadapted RNN system generated sentences of an average length of 14.1 words. The average sentence length of the adapted system was 11.8, much closer to the desired average length (reference sentences had an average of 12.1 words). This phenomenon occurs in all the configurations involving the XRCE task and explains such large TER improvements.

OL was also effective for the rest of tasks, obtaining consistent improvements for both RNN and Transformer, ranging from 3.2 to 9.1 TER points and 2.8 and 5.0 BLEU points. It is worth noting that in several cases (the TED task and UFAL En→De), a system exclusively trained on out-of-domain data and fine-tuned via OL was able to outperform a PB-SMT system trained on in-domain data ([Tables 3.2](#) and [3.3](#)). The other OL systems also behaved well, achieving performance close to the systems trained on in-domain data. These results demonstrate that online learning is a powerful method when developing translation systems with scarce data resources.

This experiment is comparable to the *a posteriori* adaptation strategy developed by [Turchi et al. \(2017\)](#). They also adapted a general model to a given domain by means of incremental learning on post-edited samples, obtaining significant BLEU improvements.

## Interactive-predictive machine translation with online learning

Next, we study the effectiveness of the general NMT system in the interactive-predictive framework and the effect of OL-based adaptation in terms of effort reduction. Table 5.9 shows the INMT results of adaptive systems and static ones.

**Table 5.9:** Human effort required by INMT systems—RNN and Transformer (Trans.)—with online learning (OL-) and without adaptation, in terms of KSMR. All the systems exclusively trained on the general domain (Europarl). Significant improvements of adaptive systems are denoted by \*.

		KSMR [↓]			
		RNN	OL-RNN	Trans.	OL-Trans.
XRCE	En→De	49.2	36.7*	51.8	42.9*
	En→Fr	50.1	38.3*	54.0	41.6*
TED	En→De	28.9	26.0*	27.9	26.0*
	En→Fr	28.4	26.7*	26.5	25.8*
UFAL	En→De	31.0	26.9*	30.8	28.1*
	En→Fr	27.4	24.9*	25.4	22.8*

The performance drop of the general INMT system followed the trend discussed in the previous section: in tasks with domains close to the general corpus, the performance of general INMT systems was close to an in-domain system (e.g. TED). But, if the domain of the document is far from the general corpus (XRCE), the human effort required rose dramatically.

The introduction of OL into the interactive-predictive systems had a similar effect to that observed in terms of translation quality: we obtained significant KSMR reductions for all tasks. The greatest improvements were again obtained in the XRCE task, due to the aforementioned reasons (highest RRR and shorter sentences). In the case of the TED task, online learning overcame the gap between training a specific system or using the general one.

### Scenario #3: Fine-tuning a general system

In our last experimental setup, we hybridized scenarios #1 and #2: we have available in-domain and out-of-domain data. Thus, we started from a general NMT system, trained on an out-of-domain corpus, and fine-tuned it with the in-domain training data. Finally, we followed the refinement procedure via OL, as in previous scenarios. We study if OL can bring enhancements to an already fine-tuned system, and if so, to what extent.

Again, we used the Europarl corpus as out-of-domain. We followed the same segmentation strategy than in Section 5.3.4, applying it also to the training set. Table 5.6 shows the vocabulary coverage of the training sets, generally high. Only in the UFAL corpus was the coverage slightly lower (from 85.9% to 92.5%).

Once we had our system trained on Europarl, we continued the training on each in-domain training set. For this retraining, we kept the hyperparameters used to train the original NMT system: Adam with  $\rho = 0.0002$ . Following Wu et al. (2016), we also tested simple SGD with learning rate annealing, but we obtained poorer results. We early-stopped the training following the same criterion as in the general case (Section 3.5.1), but evaluating each 1,500 updates and setting the patience to 10 evaluations. For the adaptation via OL, we followed the configurations described in Table 5.2.

### Translation post-editing with online learning

As in the previous experiments, we start by evaluating the impact of online learning in terms of translation quality. Differences between static and adaptive systems are shown in Tables 5.10 and 5.11, for the RNN-based and Transformer systems, respectively.

**Table 5.10:** Translation quality for all tasks in terms of TER and BLEU. We compare static (RNN) and adaptive (OL-RNN) RNN-based NMT systems. All NMT systems have been pre-trained on Europarl data and fine-tuned with the training data from each task. Significant improvements of adaptive systems are denoted by \*.

		TER [↓]		BLEU [↑]	
		RNN	OL-RNN	RNN	OL-RNN
XRCE	En→De	58.0	48.0*	28.3	39.4*
	En→Fr	48.5	41.8*	40.8	47.8*
TED	En→De	53.8	53.2	27.1	27.2
	En→Fr	48.3	47.5	35.4	36.4
UFAL	En→De	60.6	56.4*	22.1	24.3*
	En→Fr	48.2	41.4*	36.2	42.0*

The fine-tuned systems performed better than those exclusively trained on the in-domain data (scenario #1, Section 5.3.4) in those cases with scarce in-domain data, namely the XRCE and TED tasks. In these cases, the TER and BLEU were significantly improved when fine-tuning a general system. We found consistent improvements of more than 5 TER points. On the other hand, if we had available a large amount of in-domain data, the fine-tuning effectiveness is diluted. This is the case of

the UFAL task: as this is a large in-domain corpus, to pre-train with Europarl had minor effects on the fine-tuned systems, in which the translation quality was slightly degraded.

The addition of OL to the NMT systems improved the performance with respect to static systems, in concordance to the previous scenarios. In the XRCE and UFAL tasks, the improvements were large, either for the RNN-based and the Transformer systems: From 2.8 to 10 TER points and from 1.5 to more than 11 BLEU points. According to their RRR and UNF metrics (Table 5.6), this was expected, because those are the corpora the highest RRR. The TED task also benefited from OL, but to a lower extent, yielding non-significant improvements.

**Table 5.11:** Translation quality for all tasks in terms of TER and BLEU. We compare static (Trans.) and adaptive (OL-Trans) Transformer-based NMT systems. All NMT systems have been pre-trained on Europarl data and fine-tuned with the training data from each task. Significant improvements of adaptive systems are denoted by \*.

		TER [↓]		BLEU [↑]	
		Trans.	OL-Trans.	Trans.	OL-Trans.
XRCE	En→De	58.4	49.8*	27.9	36.4*
	En→Fr	48.1	41.1*	41.1	47.9*
TED	En→De	52.5	51.2	29.0	30.1
	En→Fr	47.0	45.8	37.1	38.1
UFAL	En→De	59.0	56.2*	22.7	24.6*
	En→Fr	47.4	41.5*	36.4	41.7*

## Interactive-predictive machine translation with online learning

Table 5.12 shows the effort required in an IMT scenario. Compared to those systems exclusively trained on the in-domain data (Table 5.5), we observed the same phenomenon as in the previous section: the usage of out-of-domain data was especially effective in tasks with scarce in-domain data. Fine-tuned systems performed clearly better in all cases but UFAL (En→De).

The largest improvements were obtained in the XRCE and TED tasks, with less training data. In these cases, the enhancements ranged from 2.0 to 7.5 KSMR points. As in terms of translation quality, fine-tuning had a minor effect on the UFAL task, as the in-domain corpus is large enough to build a good INMT system. While the fine-tuned RNN-based system achieved similar results than in Section 5.3.4, in the case of the Transformer model, the performance was slightly hurt. In addition to the aforementioned regarding the amount of training data, we must take into account that



**Table 5.12:** Human effort required by INMT systems—RNN and Transformer (Trans.)—with online learning (OL-) and without adaptation, in terms of KSMR. All NMT systems have been pre-trained on Europarl data and fine-tuned with the training data from each task. Significant improvements of adaptive systems are denoted by \*.

		KSMR [↓]			
		RNN	OL-RNN	Trans.	OL-Trans.
XRCE	En→De	23.8	19.8*	22.9	20.0*
	En→Fr	23.3	17.3*	21.8	18.2*
TED	En→De	23.9	23.7	23.1	22.3
	En→Fr	22.0	21.3	20.5	19.8
UFAL	En→De	24.2	22.8*	24.8	23.4*
	En→Fr	17.9	16.5*	17.0	15.9*

the training of the Transformer model is more sensitive to hyperparameters (Popel and Bojar, 2018). Therefore, we suspect that a better choice of hyperparameters during the fine-tuning process could avoid this degradation.

### 5.3.5 Further analyses

We analyze other aspects of the proposed adaptive INMT systems, such as response times of OL. In addition, in order to obtain additional insights of the adaptation via OL in NMT, we study the evolution of the adaptation process. Finally, we show an example of an INMT session, adapted with OL.

#### Temporal costs

As discussed in Section 4.3, interactive-predictive systems require adequate response times. The interaction response times were already presented in Table 4.4. Thereby, we need to evaluate now cost of updating the system. Table 5.13 shows the learning times for each task<sup>1</sup>. These values refer to the first scenario (Section 5.3.4). In the other scenarios we used as NMT system the one trained on Europarl. Therefore, this is the reference for those cases.

The learning times kept constant, regardless the task. The Transformer model performed slightly quicker updates, as it avoids of the recurrence. All learning times were around 0.1 seconds, therefore, differences between adaptive and static systems were almost unnoticeable in terms of usability.

<sup>1</sup>Experiments executed on a single GeForce GTX 1080 GPU.

**Table 5.13:** Average learning time, in seconds, for all tasks and NMT systems, RNN and Transformer.

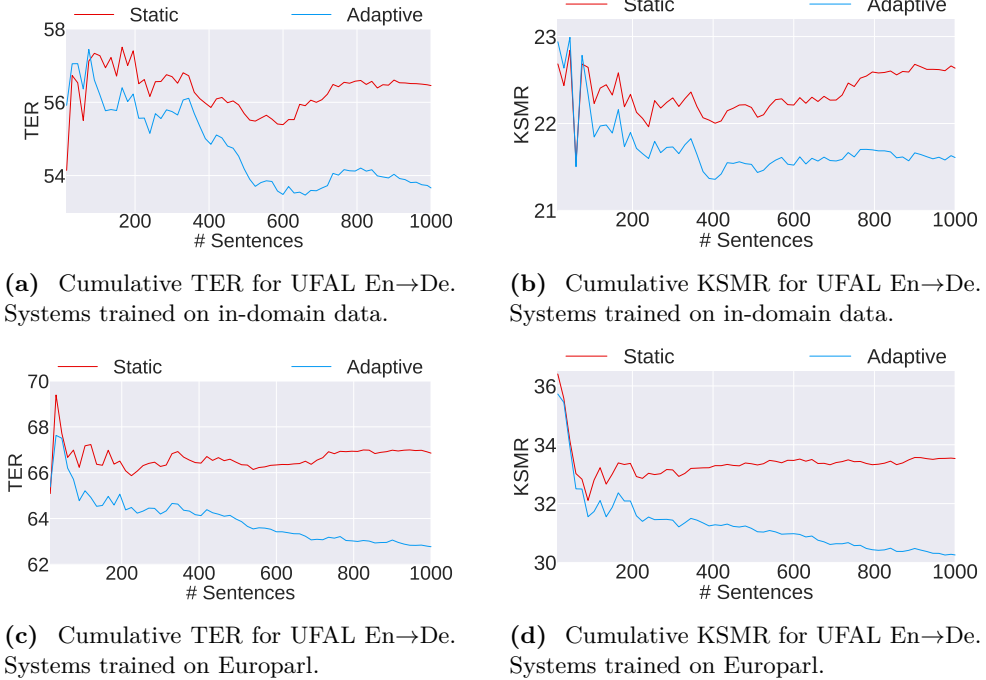
		RNN	Transformer
XRCE	En→De	0.09	0.06
	En→Fr	0.08	0.05
TED	En→De	0.12	0.09
	En→Fr	0.12	0.08
UFAL	En→De	0.15	0.11
	En→Fr	0.15	0.10
Europarl	En→De	0.14	0.09
	En→Fr	0.14	0.09

### On the impact of online learning

We deepen our investigation into the effects of continuous learning in NMT, comparing adaptive versus non-adaptive systems in translation post-editing and IMT. Since we want to reduce the effort required by the user, we are interested in TER and KSMR. We measured cumulative TER and KSMR, as the post-editing and IMT processes advanced. We report results (Fig. 5.6) from the UFAL En→De task, for the systems trained on in-domain and out-of domain data (Section 5.3.4 and Section 5.3.4, respectively).

As shown in Fig. 5.6a and Fig. 5.6b, the adaptive systems were able to rapidly take advantage of the post-edited samples. With approximately 100 samples, TER and KSMR were considerably lowered; with 600 sentences, the differences were large. From here, the systems had a performance ceiling. Nevertheless, if we attend to the static system, we observe that from sentence 600, the task becomes more difficult, and the TER and KSMR were increased. OL prevented some of this rise, stabilizing the performance of the systems.

OL applied to systems exclusively trained on out-of-domain data (Fig. 5.6c and Fig. 5.6d), improved the performance of the systems. Both TER and KSMR followed a continuous drop. Although expected, this behavior confirms that the systems could be enhanced to a greater extent by means of continuous learning if we had more data.



**Figure 5.6:** Cumulative TER and KSMR of static (solid lines) and adaptive (dashed lines) NMT systems for the UFAL En→De task. Plots Fig. 5.6a and Fig. 5.6b refer to systems trained on in-domain data, while results of Fig. 5.6c and Fig. 5.6d were obtained with a system trained only on out-of-domain data.

### Qualitative analysis

We show an example of a real INMT session, using static and online systems. The system is the RNN-based NMT one, trained only with in-domain data (scenario #1). The sentence is the same as the example from Section 4.4.3, but in this case, for the En→Fr direction.

The source sentence was “*They will be asked to provide a clean catch urine sample .*” and the desired translation was “*Ils és à fournir un échantillon d’ urine propre.*”. The static NMT system proposed the translation “*Ils és seront demandé de fournir un échantillon d’ urine propre .*”, which contains several mistakes. Fig. 5.7 shows the corresponding INMT session to this example. In this case, 6 iterations were required, in order to match the desired translation.

<b>Source:</b>		They will be asked to provide a clean catch urine sample .
<b>Target translation:</b>		Ils <b>és</b> seront demandé de fournir un échantillon d' urine propre .
<b>IT-0</b>	<b>MT</b>	Ils <b>és</b> seront demandé de fournir un échantillon d' urine propre .
<b>IT-1</b>	<b>User</b>	<i>Ils <b>és</b> seront</i> demandé de fournir un échantillon d' urine propre .
	<b>MT</b>	<i>Ils <b>és</b> seront</i> initiés pour fournir un échantillon d' urine propre .
<b>IT-2</b>	<b>User</b>	<i>Ils <b>és</b> seront</i> invités pour fournir un échantillon d' urine propre .
	<b>MT</b>	<i>Ils <b>és</b> seront</i> invité de fournir un échantillon d' urine propre .
<b>IT-3</b>	<b>User</b>	<i>Ils <b>és</b> seront</i> invités de fournir un échantillon d' urine propre .
	<b>MT</b>	<i>Ils <b>és</b> seront</i> invités de fournir un échantillon d' urine propre .
<b>IT-4</b>	<b>User</b>	<i>Ils <b>és</b> seront</i> invités à de fournir un échantillon d' urine propre .
	<b>MT</b>	<i>Ils <b>és</b> seront</i> invités à pour obtenir un échantillon d' urine propre .
<b>IT-5</b>	<b>User</b>	<i>Ils <b>és</b> seront</i> invités à f pour obtenir un échantillon d' urine propre .
	<b>MT</b>	<i>Ils <b>és</b> seront</i> invités à fournissant un échantillon d' urine propre .
<b>IT-6</b>	<b>User</b>	<i>Ils <b>és</b> seront</i> invités à <i>fournir</i> ssant un échantillon d' urine propre .
	<b>MT</b>	<i>Ils <b>és</b> seront</i> invités à <i>fournir</i> un échantillon d' urine propre .
<b>END</b>	<b>User</b>	<i>Ils <b>és</b> seront</i> invités à <i>fournir un échantillon d' urine propre</i> .

**Figure 5.7:** Real INMT session from the UFAL task (scenario #1). **IT-** refers to the number of iteration of the process, the **MT** row refers to the INMT hypothesis in the current iteration and in the **User** row is shown the feedback introduced by the user: the correct character (in bold). We color in green the prefix that the user has inherently validated while introducing the correction. 13 user actions are required, involving 6 keystrokes and 7 mouse actions (counting final hypothesis acceptance). This represents a KSMR of 19.7%.

Fig. 5.8 shows the same INMT session, but for an adaptive NMT system. Previously to this sample, the system was already adapted with 976 sentences. The initial hypothesis was slightly different than the one proposed by the static system: “*Ils **és** seront invités pour obtenir un échantillon d' urine d' arrêt propre .*”. The word “*invités*” was correctly translated, but in this case, the system introduced the erroneous clause “*d' arrêt*”. However, the adaptive system reacted better to the user feedback: in the second iteration, the system correctly predicted the word “*fournir*” with a single keystroke, while the static system was unable to correctly predict this word. Finally, the erroneous clause introduced in the initial hypothesis is removed as the user introduced the feedback belonging to the word “*propre*” at the second iteration.

<b>Source:</b>		They will be asked to provide a clean catch urine sample .
<b>Target translation:</b>		Ils és seront demandé de fournir un échantillon d' urine propre .
<b>IT-0</b>	<b>MT</b>	Ils és seront invités pour obtenir un échantillon d' urine d' arrêt propre .
<b>IT-1</b>	<b>User</b>	<i>Ils és seront invités</i> á pour obtenir un échantillon d' urine d' arrêt propre .
	<b>MT</b>	<i>Ils és seront invités</i> à fournir un échantillon d' urine d' arrêt propre .
<b>IT-2</b>	<b>User</b>	<i>Ils és seront invités</i> à fournir un échantillon d' urine pd' arrêt propre .
	<b>MT</b>	<i>Ils és seront invités</i> à fournir un échantillon d' urine propre .
<b>END</b>	<b>User</b>	<i>Ils és seront invités</i> à fournir un échantillon d' urine propre .

**Figure 5.8:** Same IMT session and notation as Fig. 5.7 but with an adaptive INMT system. Only 2 keystrokes and are 3 mouse actions are now required (KSMR=7.6%).

## 5.4 A user evaluation of machine translation post-editing with online learning

We now evaluate the adaptive NMT system in a real-life scenario. This evaluation was done in collaboration with the translation company Pangeanic, with funding from the Spanish Center for Technological and Industrial Development (Centro para el Desarrollo Tecnológico Industrial). This company has experience in providing high-quality translations and regularly relies on the use of translation post-editing. The results shown in this section were obtained with the collaboration of Miguel Domingo and the team from Pangeanic.

### 5.4.1 NMT systems

We used an RNN-based NMT system, as described in Section 3.1. In this case, the system was built using the OpenNMT-py toolkit (Klein et al., 2017). The system was trained following the configuration described in Section 3.5.1, but using regular LSTM units, instead of their conditional version. Following a similar strategy as in Section 5.3.2, we built the adaptive systems according to the results obtained on a development set: for each post-edited sample, we applied two vanilla SGD updates, with a fixed learning rate of 0.05.

## 5.4.2 Translation environment

The experiment was conducted using SDL Trados Studio as the translation environment. This software is widely used in the translation industry, and all the participants use it in their daily work. Fig. 5.9 shows a screenshot of the SDL Trados Studio interface.

Our NMT system was deployed as a server, which delivered the translations to SDL Trados Studio and performed the adaptation using the post-edits. This system is compatible with all OpenNMT-py models and it is publicly available<sup>2</sup>. We also developed a plugin that connected SDL Trados Studio with our systems.

The NMT system was deployed in a CPU server, equipped with an Intel(R) Xeon(R) CPU E5-2686 v4 at 2.30GHz and 16GB of RAM. On average, generating a translation took the system 0.23 seconds and each update took 0.45 seconds. These low latencies allow a correct usage of the system, as the flow of thoughts of the user remains uninterrupted (Nielsen, 1993).

The study involved three professional translators, with an average of four years experience, who regularly make use of MT in their workflow. Translators did not know whether the experiment they performed featured a static or an adaptive system.

## 5.4.3 Tasks and evaluation

We evaluated our systems on a real task from the production scenario of the translation company. The task was a small corpus belonging to a medico-technical domain (description of medical equipments), and was conformed by two documents of 1.7 and 2.7 thousand words respectively. The translation direction was from English to Spanish. Since we lacked an in-domain corpus, we trained a general system with the data from the translation task from WMT'13 (Bojar et al., 2013), made of 15 million parallel segments. Next, we applied the data selection technique described by Biçici and Yuret (2015) to select related instances from our general corpus, UFAL and a technological<sup>3</sup> one. We selected 8 million additional segments, which were used to fine-tune the general system.

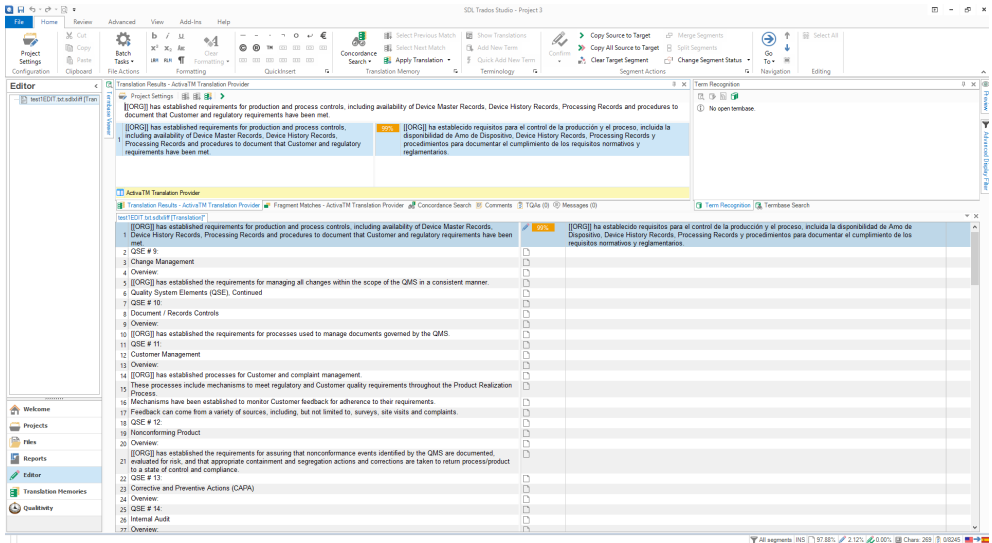
The effects of adaptivity were assessed in terms of hBLEU and hTER. These metrics refer to the BLEU and TER scores, but using the human post-edits as references. Since we computed per-sentence BLEU scores, we used exponential BLEU smoothing (method 3 from Chen and Cherry, 2014).

---

<sup>2</sup><https://github.com/midobal/OpenNMT-py/tree/OnlineLearning>

<sup>3</sup><https://metashare.metanet4u.eu/go2/qt1leapcorpus>

## 5.4. A user evaluation of machine translation post-editing with online learning



**Figure 5.9:** SDL Trados Studio user interface. From top to bottom, the first row and the leftmost column correspond to the user menus. On the next row, the middle column contains information about the segment that is being translated: on the left, the source sentence and, on the right, the MT translation. The right column displays, in case of being used, the content of the terminological dictionary. Finally, the document that is being translated appears on the bottom row: on the left, the original document and, on the right, the user post-edits.

### 5.4.4 Adaptation with simulated users

Before conducting the experimentation with human post-editors, we evaluate the system under simulated conditions, following the experimental framework described in [Section 5.3](#).

Table 5.14 shows the results in terms of translation quality of a static system, compared with an adaptive one, updated using the reference samples. These reference sentences were obtained in a previous translation job, performed by different translators of those involved in our study. The results obtained on this setup support the usefulness of the adaptation via online learning: in all cases, the adaptive system achieved better TER and BLEU than the static one. These differences were statistically significant in all cases but one. We observed important gains in terms of TER (5.5 and 1.1 points), which suggests a lower human effort required to post-edit these samples.

We also performed the adaptation on a larger document (1,500 sentences), belonging to the same domain. The adaptation to this one was even more effective: we observed gains of 10.4 TER points and 13.6 BLEU points.

#### 5.4.5 Adaptation with human post-editors

Once we tested our system in a simulated environment, we moved on to the experimentation with human post-editors. Three professional translators were involved in the experiment. For the adaptive test, all post-editors started the task with the same system, which was adapted to each user using their own post-edits. Therefore, at the end of the online learning process, each post-editor obtained a tailored system. For the static experiment, the initial NMT system remained fixed along the complete process.

In order to avoid the influence of translating the same text multiple times, each participant post-edited a different test set under each scenario (static and adaptive), as shown in Table 5.15.

The main results of this experiment are shown in Table 5.16. These numbers are averages over the results obtained by the different post-editors. The large reduction of post-editing time per sentence for the set T1 is especially relevant (an average of

**Table 5.14:** Results of the simulated experiments. TER and BLEU were computed considering the reference sentences. \* indicates statistically significant differences between the static and the adaptive systems.

Test	System	TER [↓]	BLEU [↑]
T1	Static	54.0	26.9
	Adaptive	48.5*	32.0*
T2	Static	56.1	23.4
	Adaptive	55.0	26.3*

**Table 5.15:** Distribution of users (1, 2 and 3), test sets (T1 and T2) and scenarios (static and adaptive) for the post-editing experimentation.

User	Static	Adaptive
User 1	T1	T2
User 2	T2	T1
User 3	T1	T2



**Table 5.16:** Results of the user experiments. Static systems stand for conventional post-editing, without adaptation. Adaptive systems refer to post-editing in an environment with online learning. Time corresponds to the average post-editing time per sentence, in seconds. hTER and hBLEU refer to the TER and BLEU of the system hypothesis computed against the post-edited sentences. \* indicates statistically significant differences between the static and the adaptive systems.

Split	System	Time (s)	hTER [↓]	hBLEU [↑]
T1	Static	37.9	39.5	47.3
	Adaptive	30.4	34.2	55.1*
T2	Static	45.8	38.4	45.7
	Adaptive	45.1	34.2*	50.5*

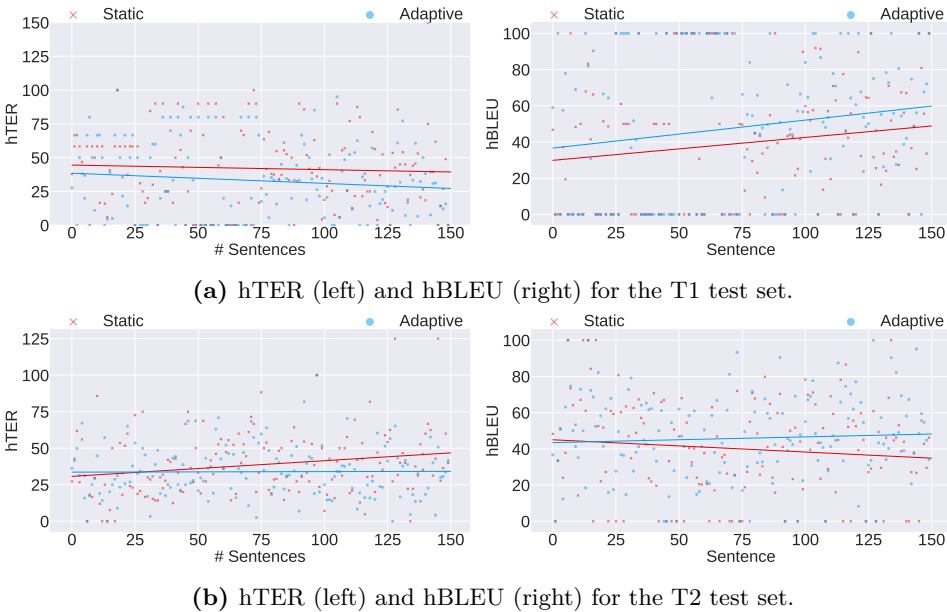
7.5 seconds per sentence). In the test set T2, the post-editing time of the adaptive system was also slightly lower than the static system one, but only by 0.7 seconds. In terms of translation quality, adaptive systems performed much better than static ones, as reflected by the significant improvements in terms of hTER (5.3 and 4.2 points) and hBLEU (7.8 and 4.8 points). These results show that adaptive systems generated more correct translations, as they required less post-edits from the user.

In order to gain additional insights of the adaptation process, we studied the evolution of hTER and hBLEU during the post-editing process. Fig. 5.10 compares these metrics, computed for each sentence from both test sets for static and adaptive systems. Moreover, to observe the evolution of such points, we computed linear fits of the scores of each system via the least squares method.

In Fig. 5.10a, we observe that for the test split T1, the adaptive system consistently produced slightly better hypotheses than the static one, but there was no clear evidence on the effects of online learning. Both systems behaved similarly: the hTER and hBLEU values were gradually increased, which suggests either that the test document was increasingly easier to translate or that the user felt more comfortable with the style and translations provided by the system. Therefore, they applied less post-edits to the final sentences.

In the case of T2 (Fig. 5.10b), we observe a degradation on hTER and hBLEU of the static system, as the post-editing process advances. This degradation was prevented by the adaptive system, in which hTER and hBLEU are even slightly improved. The effects of the adaptation are noticeable from the 30th sentence onwards.

Finally, it is interesting to compare the simulated experiment against this one. We observed that, in terms of automatic metrics, the system yielded much better results when evaluating against post-edits, rather than against reference sentences (compare



**Figure 5.10:** hTER and hBLEU per sentence of static and adaptive systems for both test sets (T1 and T2). Recall that hTER is an error metric, hence, the lower, the better. Individual sentence scores are plotted for each system, static (red crosses) and adaptive (blue dots). We also show a linear fit of the scores of each system, in dashed red and solid blue lines, for static and adaptive systems, respectively.

the “Static” rows from [Table 5.14](#) and [Table 5.14](#), respectively). This suggests that the translation hypotheses provided by the system were useful to the human users, as they produced similar post-edited samples. It is also worth pointing out that the adaptation process was, in most cases, slightly less effective in the simulated experiment.

#### 5.4.6 User perceptions and opinions

After finishing each experiment, the participants answered a questionnaire regarding the post-editing task they had just performed. In this survey, we asked the users about their satisfaction on the translations they produced, whether they preferred to perform post-editing or translating from scratch and their opinions on the automatic translations provided, in terms of grammar, style and overall quality. We also requested them to give their feedback on the task, as an open-answer question. This survey, together with the user answers is shown in [Appendix B](#).

The users were generally satisfied with the translations they generated. In all cases, they preferred to perform this translation task via post-editing rather than translating from scratch. Two of them preferred to perform this translation from scratch in less than around 25% of the sentences. The other post-editor preferred to translate from scratch around 50% of the sentences. In all cases, they are keen to perform translation post-editing in the future. These perceptions on the MT utility are slightly better than those reported by [Daems and Macken \(2019\)](#). We believe that these differences are due to the background in translation post-editing that our users had: they perform translation post-editing as their regular way of work; therefore, their perceptions toward this methodology are generally favorable.

Regarding the translation quality offered by the NMT system, their general opinion is that the system produced translations of average quality. The strongest attribute of the translations was their grammatical accuracy. The style and overall quality was perceived in some cases below the average, depending on the user and the experimental condition.

Once they finished both experiments, we also asked the users to identify the adaptive systems. All users guessed correctly which one was the adaptive one. Therefore, the influence of adaptability was noticeable by the user.

Regarding their general opinions, they all observed how corrections applied on one segment were generally reflected in the following segments, especially corrections related to product names, grammatical structures and lexical aspects. This mostly reduced upcoming corrections to changes in the style. Overall, their perception was that the static system produced less fluent translations, and that MT was very good in most cases, but useless in others.

The post-editors reported a couple of minor issues regarding the NMT system: in a few cases, they noticed that a domain-specific term was “forgotten” by the system, being wrongly translated. In addition, the users noticed in some cases, the occurrence of some made-up words (e.g., “absolvido”). This problem was probably caused by an incorrect learning of a new word segmentation, provided by BPE. It can be seen as a catastrophic forgetting issue ([Thompson et al., 2019](#)), in which the performance of a model is degraded on a general domain during the adaptation process. This causes the appearance of made-up words. [Thompson et al. \(2019\)](#) recently proposed alternative methods to mitigate this forgetting issues. We should test if these methods address our problem of made-up words, to deploy effective and adaptive translation systems.

## 5.5 Active learning for the interactive-predictive translation of large data streams

Once we studied interactive, adaptive NMT systems, we move on to a new scenario: the translation of large datasets. The translation industry is a high-demanding field. Large amounts of data must be translated on a regular basis. As stated in the previous sections, MT greatly boosts the productivity of translation agencies (Arenas, 2008). However, MT systems require human supervision—translation post-editing or interactive-predictive machine translation—to produce high-quality translations. This represents a great effort, as it needs expert human supervisors.

The requirements of the translation industry have increased in the last years (Dranch et al., 2018). We live in a global world, in which large amounts of data must be periodically translated. This is the case of the European Parliament, whose proceedings must be regularly translated; or the Project Syndicate<sup>4</sup> platform, which translates editorials from newspapers to several languages. In these scenarios, the sentences to be translated can be seen as unbounded streams of data (Levenberg et al., 2010).

When dealing with such massive volumes of data, it is prohibitively expensive to manually revise all the translations. Therefore, it is mandatory to spare human effort, at the expense of some translation quality. Hence, when facing this situation, we have a twofold objective: on the one hand, we aim to obtain translations with the highest possible quality. On the other hand, we are constrained by the amount of human effort spent in the supervision and correction process of the translations proposed by an MT system.

The active learning framework is well-suited to these objectives. Active learning is a machine learning discipline, based on the idea that an algorithm can obtain better results with few training samples if it is able to properly select the data from which it learns (Cohn et al., 1994). A common protocol is the so-called *stream-based* AL (Settles, 2009), in which the active learner (the model) selects the samples to be labeled by an oracle (the human) from a continuous stream of samples.

This stream-based AL protocol fits the aforementioned requirements. The application of AL techniques to MT involves asking a human oracle to supervise a fraction of the incoming data (Bloodgood and Callison-Burch, 2010). Once the human has revised these samples, they are used to improve the MT system, via incremental (González-Rubio et al., 2012) or batch learning (Dara et al., 2014). Therefore, a key element of AL is the so-called sampling strategy, which determines the sentences that should be corrected by the human.

---

<sup>4</sup>[www.project-syndicate.org](http://www.project-syndicate.org)

In this section, we explore the application of AL techniques to the translation of unbounded data streams with NMT systems. We apply AL techniques to select the instances to be revised by a human oracle. The correction process is done by means of an INMT, as shown in [Chapter 4](#). The supervised samples will be used by the NMT system to incrementally improve its models, as in [Section 5.2](#).

### 5.5.1 Active learning in machine translation

The translation of large volumes of data is a very appropriate scenario for the AL framework ([Cohn et al., 1994](#); [Olsson, 2009](#); [Settles, 2009](#)). The application of AL to PB-SMT has been studied for pool-based ([Haffari et al., 2009](#); [Bloodgood and Callison-Burch, 2010](#)) and stream-based ([González-Rubio et al., 2011](#)) setups. Later works ([González-Rubio et al., 2012](#); [González-Rubio and Casacuberta, 2014](#)), combined AL together with IMT, showing that AL can effectively reduce the human effort required to reach a certain translation quality. However, to our knowledge, a study on the use of AL for NMT in a scenario of translation of unbounded data streams was still missing.

When dealing with potentially unbounded datasets, it becomes prohibitively expensive to manually supervise all the translations. The goal of IMT is to obtain high quality translations, while minimizing the required human effort. This requires to revise all translation hypotheses, correcting the wrong ones. This process can be prohibitively expensive. Aiming to address this problem, in the AL framework, a sampling strategy selects a subset of sentences worth being supervised by the user. Once corrected, the MT system adapts its models with these samples.

The AL protocol applied to unbounded data streams is detailed in [Algorithm 5.2](#) ([González-Rubio et al., 2012](#)): first, we retrieve from the data stream  $\mathcal{S}$  a block  $\mathcal{B}$  of consecutive sentences, with the function `getBlockFromStream( $\mathcal{S}$ )`. According to the `sampling( $\mathcal{B}, \varepsilon$ )` function, we select from  $\mathcal{B}$  a subset  $\mathcal{V}$  of  $\varepsilon$  instances worth being supervised by the user. Upcoming sections ([Section 5.5.2](#)) present deeper insights on these sampling functions.

The sampled sentences are interactively translated as described in [Section 4.3](#). This process is implemented by the function `INMT( $\Theta_n, x_1^J, \hat{y}_1^I$ )`. Note that, instead of applying the interactive-predictive protocol, the user could simply post-edit the selected samples. Once the user translates via INMT a source sentence  $x_1^J$ , a correct translation  $y_1^I$  is obtained. Then, we use the pair  $(x_1^J, y_1^I)$  to update the parameters  $\Theta_n$  of the NMT model, as described in [Section 5.2](#). This is done with the function `update( $\Theta_n, (x_1^J, y_1^I)$ )`. Therefore, the NMT system is incrementally adapted with new data.

Those sentences considered unworthy to be supervised are automatically translated, with the function  $\text{translate}(\Theta_n, x_1^J)$ , following the process described in Section 3.3. Once we finished the translation of the current block  $\mathcal{B}$ , we start the process again.

**Algorithm 5.2:** Active learning for unbounded data streams with interactive-predictive neural machine translation.

```

input  :  $\Theta_0$  (Initial NMT model),
           $\mathcal{S}$  (stream of source sentences),
           $\varepsilon$  (effort level desired)
auxiliar :  $\mathcal{B}$  (block of source sentences)
            $\mathcal{V} \subseteq \mathcal{B}$  (sentences to be supervised by the user)

1 begin
2    $n = 0$ 
3   repeat
4      $\mathcal{B} = \text{getBlockFromStream}(\mathcal{S})$ 
5      $\mathcal{V} = \text{sampling}(\mathcal{B}, \varepsilon)$ 
6     foreach  $x_1^J \in \mathcal{B}$  do
7        $\hat{y}_1^I = \text{translate}(\Theta_n, x_1^J)$ 
8       if  $x_1^J \in \mathcal{V}$  then
9          $y_1^I = \text{INMT}(\Theta_n, x_1^J, \hat{y}_1^I)$ 
10         $\Theta_{n+1} = \text{update}(\Theta_n, (x_1^J, y_1^I))$ 
11        output( $y_1^I$ )
12         $n = n + 1$ 
13      else
14        output( $\hat{y}_1^I$ )
15  until  $\mathcal{S} \neq \emptyset$ ;

```

### 5.5.2 Sentence sampling strategies

One of the key elements of AL is to have a meaningful strategy to obtain the most useful samples to be supervised by the human agent. This requires an evaluation of the informativeness of unlabeled samples. The sampling strategies used in this thesis belong to two major frameworks: uncertainty sampling (Lewis and Catlett, 1994) and query-by-committee (Seung et al., 1992).

As baseline, we use a random sampling strategy: sentences are randomly selected from the data stream  $\mathcal{S}$ . Although simple, this strategy usually works well in practice. In the rest of this section, we describe the sampling strategies used and developed in this thesis.

## Uncertainty sampling

The idea behind this family of methods is to select those instances for which the model has the least confidence to be properly translated. Therefore, all techniques compute, for each sample, an uncertainty score. The selected sentences will be those with the highest scores. We describe three different strategies based on uncertainty: quality estimation (González-Rubio et al., 2012), coverage (Peris and Casacuberta, 2018b) and attention distraction (Peris and Casacuberta, 2018b) sampling.

## Quality estimation sampling

A common and effective way to measure the uncertainty of a MT system is to use confidence estimation (Gandrabur and Foster, 2003; Blatz et al., 2004; Ueffing and Ney, 2007). The idea is to estimate the quality of a translation according to confidence scores of the words.

More specifically, given a source sentence  $x_1^J$  and a translation hypothesis  $y_1^I$ , a word confidence score ( $C_w$ ) as computed as (Eq. (5.17) Ueffing and Ney, 2005):

$$C_w(x_1^J, y_i) = \max_{0 \leq j \leq J} p(y_i | x_j) \quad (5.17)$$

where  $p(y_i | x_j)$  is the translation probability of  $y_i$  and  $x_j$ , given by an IBM Model 2 (Brown et al., 1993).  $x_0$  denotes the empty source word. The choice of the IBM Model 2 is twofold: on the one hand, it is a very fast method, which only requires to query in a dictionary. We are in an interactive-predictive framework, therefore speed becomes a crucial requirement. On the other hand, its performance is close to more complex methods (Blatz et al., 2004; Dyer et al., 2013).

Following González-Rubio et al. (2012), the uncertainty score for the quality estimation sampling is defined as in Eq. (5.18):

$$C_{\text{qe}}(x_1^J, y_1^I) = 1 - \frac{|\{y_i \in y_1^I | C_w(x_1^J, y_i) > \tau_w\}|}{I} \quad (5.18)$$

where  $\tau_w$  is a word confidence threshold, adjusted according to a development corpus.

### Coverage sampling

One of the main issues suffered by NMT systems is the lack of coverage: the NMT system may not translate all words from a source sentence (Tu et al., 2016).

We propose to use the translation coverage as a measure of the uncertainty suffered by the NMT system when translating a sentence. Therefore, we modify the coverage penalty (Wu et al., 2016), to obtain a coverage-based uncertainty score (Eq. (5.19) Peris and Casacuberta, 2018b):

$$C_{\text{cov}}(x_1^J, y_1^I) = \frac{\sum_{j=1}^J \log(\min(\sum_{i=1}^I \alpha_{i,j}, 1))}{J} \quad (5.19)$$

where  $\alpha_{i,j}$  is the weight provided by the attention mechanism to the  $i$ -th target word and the  $j$ -th source word.

### Attention distraction sampling

When generating a target word, an attentional NMT system should attend on meaningful parts of the source sentence. If the system is translating an uncertain sample, its attention mechanism will be *distracted*. That means, dispersed throughout the source sequence. A sample with a great distraction will feature an attention probability distribution with heavy tails (e.g. a uniform distribution). Therefore, in the attention distraction sampling strategy (Peris and Casacuberta, 2018b), the sentences to select will be those with highest attention distraction.

To compute a distraction score, we compute the kurtosis of the weights given by the attention model for each target word  $y_i$ :

$$\text{Kurt}(y_i) = \frac{\frac{1}{J} \sum_{j=1}^J (\alpha_{i,j} - \frac{1}{J})^4}{(\frac{1}{J} \sum_{j=1}^J (\alpha_{i,j} - \frac{1}{J})^2)^2}$$

being, as above,  $\alpha_{i,j}$  the weight assigned by the attention model to the  $j$ -th source word when decoding the  $i$ -th target word. Note that, by construction of the attention model,  $\frac{1}{J}$  is equivalent to the mean of the attention weights of the word  $y_i$ .

Since we want to obtain samples with heavy tails, we average the minus kurtosis values for all words in the target sentence, obtaining the attention distraction score  $C_{\text{ad}}$  (Eq. (5.20) Peris and Casacuberta, 2018b):

$$C_{\text{ad}}(x_1^J, y_1^I) = \frac{\sum_{i=1}^I -\text{Kurt}(y_i)}{I} \quad (5.20)$$



## Query-by-committee

This framework maintains a committee of models, each one able to vote for the sentences to be selected. The query-by-committee (QBC) method selects the samples with the largest disagreement among the members of the committee. The level of disagreement of a sample  $x_1^J$  measured according to the vote-entropy function (Eq. (5.21) Dagan and Engelson, 1995):

$$C_{\text{qbc}}(x_1^J, y_1^I) = -\frac{\#V(x_1^J, y_1^I)}{|\mathcal{C}|} + \log \frac{\#V(x_1^J, y_1^I)}{|\mathcal{C}|} \quad (5.21)$$

where  $\#V(x_1^J, y_1^I)$  is the number of members of the committee that voted  $(x_1^J, y_1^I)$  to be supervised and  $|\mathcal{C}|$  is the number of members of the committee. If  $\#V(x_1^J, y_1^I)$  is zero, we set the value of  $C_{\text{qbc}}(x_1^J, y_1^I)$  to  $-\infty$ .

### 5.5.3 Evaluation of active learning to translate data streams

Our last adaptation scenario regards the active learning framework described in Section 5.5.1. A system under an AL setup involves two main facets to evaluate: the improvement on the quality of the system and the amount of human effort required to achieve such quality. As in the previous sections, we measure the translation quality in terms of BLEU and the human effort in KSMR (Section 1.4.1). The human follows the simulated interaction protocol described in Section 1.4.1.

Algorithm 5.2 involves two main corpora: one that was used to train the original NMT system and a large stream of source sentences. To ensure a fair comparison with the latter works of AL applied to IMT (González-Rubio and Casacuberta, 2014), we used the same datasets: our training data was the Europarl corpus (Koehn, 2005), with the development set provided at the 2006 WMT (Koehn and Monz, 2006). As test set, we used the News Commentary corpus (Callison-Burch et al., 2007). This test set is suitable to our problem at hand because first, it contains data from different domains (politics, economics and science), which represent challenging out-of-domain samples, but account for a real-life situation in a translation agency; and second, it is large enough to properly simulate long-term evolution of unbounded data streams. All data are publicly available. We conducted the experimentation in the Spanish to English language direction.

The NMT system under test was the RNN-based system described in Section 3.5.1. We incrementally update the system (Line 10 in Algorithm 5.2), with vanilla SGD, with a learning rate of 0.0005. We chose this configuration according to an exploration on the validation set.

The rest of experimental details were set according to previous works. The blocks retrieved from the data stream contained 500 samples (according to [González-Rubio et al. \(2012\)](#), the performance is similar regardless of the block size). For the quality estimation method, the IBM Model 2 was obtained with `fast_align` ([Dyer et al., 2013](#)) and  $\tau_w$  was set to 0.4 ([González-Rubio et al., 2010a](#)).

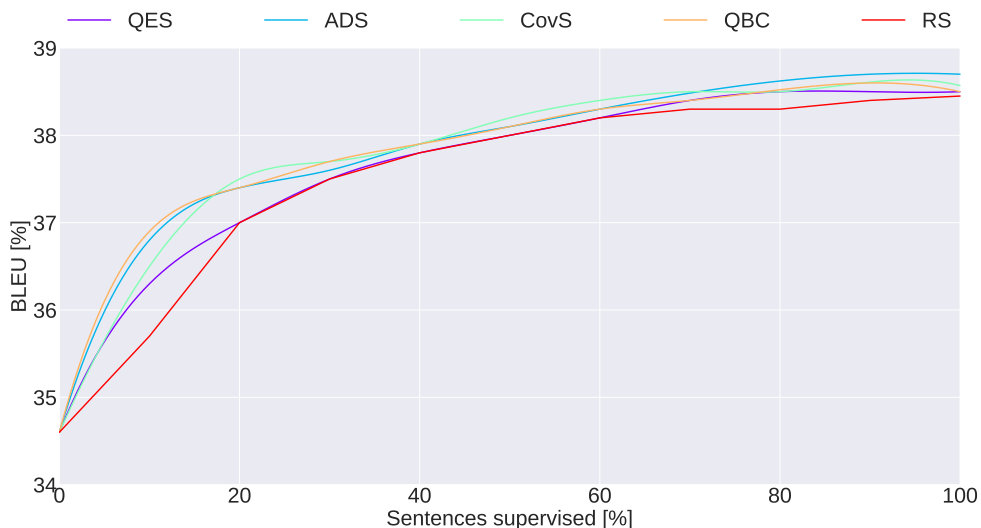
We study and compare the AL framework for all our sampling strategies: quality estimation sampling (QES), coverage sampling (CovS), attention distraction sampling (ADS), random sampling (RS) and query-by-committee (QBC). The committee was composed by the four uncertainty sampling strategies, namely QES, CovS, ADS and RS. The inclusion of the latter into the committee can be seen as a way of introducing some noise, aiming to prevent overfitting.

### Active learning evaluation

First, we evaluated the effectiveness of the application of AL in the NMT system, in terms of translation quality. [Fig. 5.11](#) shows the BLEU score of the initial hypotheses proposed by the NMT system ([Line 7](#) in [Algorithm 5.2](#)), as a function of the percentage of sentences supervised by the user ( $\varepsilon$  in [Algorithm 5.2](#)), i.e. the percentage of sentences used to adapt the system. The BLEU score of a static system without AL was 34.6. Applying AL, we obtained improvements up to 4.1 BLEU points.

As expected, the addition of the new knowledge had a larger impact when applied to a non-adapted system. Once the system becomes more specialized, a larger amount of data was required to further improve.

The sampling strategies helped the system to learn faster. Taking RS as a baseline, the learning curves of the other techniques were better, especially when using few (up to a 30%) data for fine-tuning the system. The strategies that achieved a fastest adaptation were those involving the attention mechanism (ADS, CovS and QBC). This indicates that the system is learning from the most useful data. The QES and RS required more supervised data to achieve comparable BLEU results. When supervising high percentages of the data, we observed BLEU differences. This is due to the ordering in which the selected sentences were presented to the learner. The sampling strategies performed a sort of curriculum learning ([Bengio et al., 2009](#)).



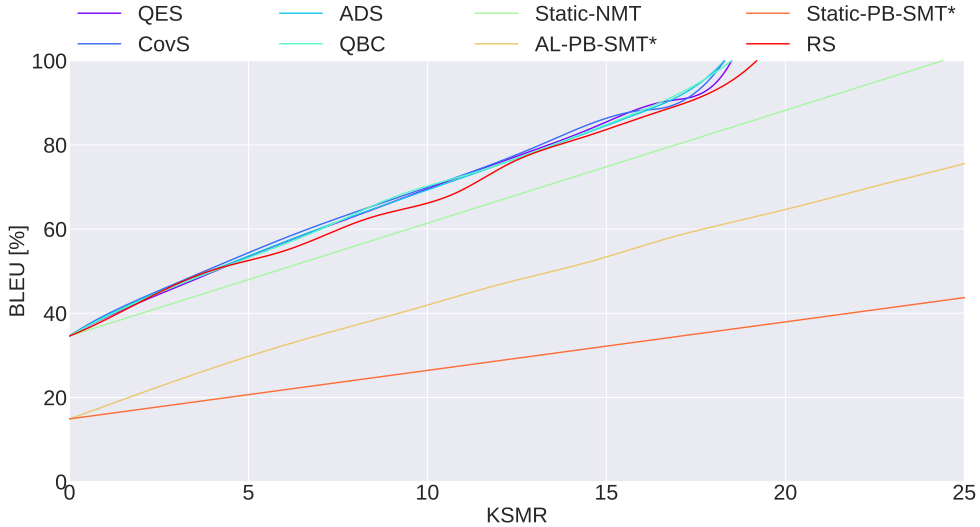
**Figure 5.11:** BLEU of the initial hypotheses proposed by the NMT system as a function of the amount of data used to adapt it. The percentage of sentences supervised refers to the value of  $\varepsilon$  with respect to the block size.

### Introducing the human into the loop

From the point of view of the user, it is important to assess not only the quality of the MT system, but also the effort spent to obtain such quality. Fig. 5.12 relates both, showing the amount of effort required to reach a certain translation quality. We compared the results of a system with AL to the same NMT system without AL and with two other PB-SMT systems, with and without AL, from González-Rubio and Casacuberta (2014).

Results in Fig. 5.12 show consistent positive results of the AL framework. In all cases, AL reduced the human effort required to obtain a certain translation quality. Compared to a static NMT system, approximately 25% of the human effort can be spent using AL techniques.

Regarding the different sampling strategies, all of them behaved similarly. They provided consistent and stable improvements, regardless the level of effort desired ( $\varepsilon$ ). This indicates that, although the BLEU of the system may vary (Fig. 5.11), this had small impact on the effort required for correcting the samples. All sampling strategies outperformed the random baseline, which had a more unstable behavior.



**Figure 5.12:** Translation quality (BLEU) as a function of the human effort (KSMR) required. Static-NMT relates to the same NMT system without AL. \* denotes systems from González-Rubio and Casacuberta (2014): Static-PB-SMT is a PB-SMT system without AL and AL-PB-SMT is the coverage augmentation PB-SMT system.

Compared to classical PB-SMT systems, NMT performed surprisingly well. Even the NMT system without AL largely outperformed the best AL-PB-SMT system. This is due to several reasons: on the one hand, the initial NMT system was much better than the original PB-SMT system (34.6 vs. 14.9 BLEU points). Part of this large difference were presumably due to the BPE used in NMT: the data stream contained sentences from different domains, but they can be effectively encoded into known sequences via BPE. The PB-SMT system was unable to handle well such unseen sentences. On the other hand, INMT systems usually respond much better to the human feedback than interactive-predictive PB-SMT systems (Knowles and Koehn, 2016; Peris et al., 2017c). Therefore, the differences between PB-SMT and NMT were enlarged even more.

Finally, it should be noted that all our sampling strategies can be computed speedily. They involve analysis of the NMT attention weights, which are computed as a byproduct of the decoding process; or queries to a dictionary (in the case of QES). The update of NMT system is also fast, taking approximately 0.1 seconds. This makes AL suitable for a real-time scenario.

## 5.6 Summary

This chapter presented adaptive NMT via online learning. Under this framework, an NMT system is incrementally adapted on the fly, with the new data generated on a post-editing or IMT process. We studied the implementation of OL strategies for NMT. Since we are working with neural networks, this incremental retraining can be done by applying the same training algorithms than in regular (mini)batch training, based on gradient descent. We also advocated that the usage of hypergradient descent algorithms may become useful in this scenario, as they provide robustness to the process. Moreover, we proposed two novel OL adaptation methods, inspired by the classical passive-aggressive and minimax algorithms. The experimental results for these methods were, however, negative. They were overcome by the regular methods.

We performed a thoughtful evaluation of these adaptive systems, studying its application in three different scenarios. All of them referred to plausible situations in the translation industry and relate to the amount of training data available: we may have enough in-domain data to properly train an NMT system, to also have an out-of-domain corpus to provide additional knowledge to the NMT or we can suffer from lack of in-domain data.

We conducted a wide experimentation, relating two language pairs in five different domains, for each one of the proposed scenarios. The results were conclusive: online learning techniques were able to bring significant improvements to static systems in almost every case. Adaptive NMT systems produced better translation hypotheses and reduced the human effort required to correct their outputs. The magnitude of such enhancements were task-dependent, according to the properties of the text to translate.

The application of online learning to INMT systems reduced even more the human effort required to correct translation hypotheses. Moreover, the computational overhead of OL was small, making suitable the use of adaptive NMT systems in an interactive-predictive scenario. We also compared our system to the state-of-the-art in IMT, based on PB-SMT models. Neural systems beat PB-SMT by a large margin in terms of the human effort required.

In addition, we performed an evaluation of adaptive NMT in a real post-editing scenario. The experiment involved three professional translators, who regularly make use of MT post-editing. We observed reductions in post-editing times and significant improvements in terms of (h)TER and (h)BLEU, thanks to online learning adaptation. The users were very pleased with the system: they noticed that corrections applied on a given segment generally were reflected on the successive ones, making the post-editing process more effective and less tedious.

Finally, we studied the application of these adaptive, interactive-predictive NMT systems to the translation of large amounts of data. We assumed that these data are available in form of a continuous stream of sentences. To deal with this, we studied the application of an active learning scenario to an INMT or post-editing frameworks. The core idea was to supervise the most useful samples from a potentially unbounded data stream, while automatically translating the rest of sentences. We developed two novel sampling strategies, in addition to other well-established methods that can be directly applied in this framework.

We evaluated the capabilities and usefulness of this AL framework by simulating real-life scenario, involving the aforementioned large data streams. AL was able to enhance the performance of the NMT system in terms of BLEU. Moreover, we obtained consistent reductions of approximately a 25% of the effort required to reach a desired translation quality. Finally, it is worth noting that NMT outperformed classical PB-SMT systems by a large margin.

## Chapter 6

# Captioning visual content

So far, we have discussed different aspects of MT. However, the neural sequence-to-sequence framework described in [Chapters 2](#) and [3](#) can be applied beyond MT. This is the case of automatic captioning multimedia content. It represents an interesting but difficult task that bridges together the fields of computer vision and NLP. The problem is defined as the automatic description in a natural language of visual instances, typically images or videos. This could lead to multiple applications (e.g. video indexing and retrieval, movie description for multimedia applications or for blind people or human-robot interaction).

Tightly related to this topic is the egocentric vision field ([Doherty et al., 2013](#); [Betancourt et al., 2015](#); [Bolaños et al., 2017](#)). The goal of egocentric vision is to analyze the visual information provided by wearable cameras, which have the capability to acquire images from a first person point-of-view. The analysis of these images provides information about the behavior of the user, useful for several complementary topics like social interactions ([Aghaei et al., 2018](#)), scene understanding ([Singh et al., 2016](#)), time-space-based localization ([Yao et al., 2018](#)), action ([Fathi et al., 2011](#); [Posas et al., 2018](#)) or activity recognition ([Iwashita et al., 2014](#); [Cartas et al., 2017](#)), or nutritional habits analysis ([Bolaños et al., 2018b](#)), among others. Thus, enabling us to understand the whole story and behavior of the users behind the pictures (i.e. automatic storytelling) followed by inferring their actions and habits could lead to a better quality of life for them. Considering the sheer amount of data that wearable cameras provide, there is a need to create automatic algorithms to analyze and sum-

marize them. In this chapter, we focus on the specific topic of creating automatic diaries of the life of the user by means of textual descriptions.

Among the potential health-related applications of automatic diary construction, we find particularly interesting the treatment of patients with dementia. As proven by Spector et al. (2003); Sellen et al. (2007), the daily review of egocentric pictures taken by the patients, can help them to partially recover their cognitive capabilities. Therefore, the automatic generation of additional information, the comparison with those provided by the users, or the automatic indexation and retrieval of these data, would provide novel tools to improve future cognitive frameworks for memory enhancement.

But prior to tackling the egocentric captioning problem, we focus first on the regular video captioning task. We investigate methods and mechanisms to build video captioning systems. Provided that a video is a sequence of images, we can tackle this problem as a sequence-to-sequence task, like in NMT (Chapter 3). After this, we will tackle our problem of egocentric captioning.

These problems present several properties that make them especially difficult. Besides the significant amount of image information to analyze, videos may have a variable number of images and can be described with sentences of different length. Furthermore, the descriptions of videos are high-level summaries that not necessarily need to be expressed in terms of the objects, actions and scenes observed in the images. There are many open research questions in this field requiring a deep understanding of video. These include how to efficiently extract important elements from the images (e.g. objects, scenes, actions), to define the local (e.g. fine-grained motion) and global spatial-temporal information, determine the salient content worth describing, and generating the final video description. All these specific questions need the joint attention of the computer vision and NLP communities in order to be solved.

This chapter covers the multimedia captioning topic. We start by proposing a video captioning system, that bridges bidirectional LSTM networks (BLSTM) together with ConvNets, to obtain rich representations of the video frames. This is described and evaluated in Section 6.1. Next, we move towards the egocentric daily captioning problem. We propose a system with extended, multimodal context to relate the events occurred during a day. We evaluate our method on the first dataset designed to caption egocentric events. The experimentation conducted reveals that the system is able to improve the performance when analyzing larger contexts. This is described and discussed in Section 6.2. Finally, we study the application of the interactive-predictive framework described in Chapter 4 to these multimodal systems. We conduct experiments on several datasets and different tasks, using the RNN-based and Transformer architectures. We show that the interactive-predictive framework



can be successfully applied to these multimodal scenarios. This is described in [Section 6.3](#).

[Sections 6.1](#) and [6.2](#) are the result of collaborative work with Marc Bolaños and Prof. Petia Radeva, from Universitat de Barcelona/Universitat Autònoma de Barcelona. They are computer vision experts while we had experience in NMT. Therefore, our backgrounds were ideal to tackle these challenges together. As computer vision experts, the feature extraction from the images was mostly done by them. Moreover, they collected and annotated the dataset presented in [Section 6.2.5](#). We focused on the textual part of the problem, proposing to apply the models from NMT to this task and bridging both worlds in early prototypes. After that, we tightly cooperated in the design and implementation of our captioning systems, as well as in the experimentation, evaluation and discussion regarding all of them. [Section 6.3](#) refers completely to my original work.

## 6.1 Video captioning

Video captioning can be seen as an instantiation of a sequence-to-sequence problem: we must generate a sequence of words given a sequence of images (the video frames). Hence, the encoder–decoder ideas—typically used in NMT ([Chapter 3](#))—can be directly applied to this task. The main difference with NMT is that in the encoding step, instead of processing natural text, we must deal with the visual information from video frames.

Early captioning work tackled the problem using primitive image recognizers to analyze the visual features and designing hand-crafted, rule-based text generators ([Farhadi et al., 2010](#)). These proposals usually detected a set of relevant concepts in an encoding step and generated an associated sentence according to these concepts ([Krishnamoorthy et al., 2013](#); [Rohrbach et al., 2013](#)). This line of research led to complex ontology-based ([Yao et al., 2010](#)) or parsing-based ([Kuznetsova et al., 2014](#)) systems. Although achieving important milestones, such systems generated the captions in a rigid way, because of their underlying rule-based text generator.

The advent of the neural encoder–decoder framework allowed researchers to tackle the problem as a sequence-to-sequence task. Inspired by the advances achieved in the MT field, a number of researchers applied the same model to the image captioning ([Mao et al., 2014](#); [Vinyals et al., 2015](#)) and the video description tasks ([Venugopalan et al., 2014](#); [Karpathy and Fei-Fei, 2015](#); [Venugopalan et al., 2015](#)). The main idea is to use a ConvNet network to extract features from the images or video frames, and feed a decoder network (typically recurrent) with such visual features.

Moreover, Xu et al. (2015) introduced the attention mechanism into the image captioning model, allowing the system to focus on certain parts of an image to generate each word. The attention mechanism was also applied to tackle video captioning problems (Yao et al., 2015), allowing the systems to attend to some specific video frames.

The research on video captioning has intensified from these seminal works (see e.g. Aafaq et al., 2018): hierarchical processing of the information (Pan et al., 2016; Song et al., 2017), inclusion of external knowledge to diversify the captions (Venugopalan et al., 2016, 2017), multi-modal attention mechanisms (Hori et al., 2017) or novel architectures, such as the Transformer (Zhou et al., 2018).

### 6.1.1 Video captioning systems

Analogously to the MT problem, we pose the video captioning problem as a sequence-to-sequence task: given an input video, represented as a sequence of  $J$  images,  $z_i^J = z_1, \dots, z_J$ , our goal is to generate a sequence of words  $\hat{y}_1^J = \hat{y}_1, \dots, \hat{y}_I$  that describes the events that occur in the video. Applying the same reasoning as in Section 1.3 leads to a similar expression as Eq. (1.5), but conditioned to the video frames, as in Eq. (6.1):

$$\hat{y}_1^J = \arg \max_{I, y_1^I} \Pr(y_1^I \mid z_1^J) \quad (6.1)$$

This expression can be tackled following the ideas from NMT (Chapter 3). Therefore, by applying Eqs. (3.1) and (3.2), we reach Eq. (6.2), the video captioning objective:

$$\hat{y}_1^J \approx \arg \max_{I, y_1^I} \sum_{i=1}^I \log p(y_i \mid y_1^{i-1}, z_1^J; \Theta) \quad (6.2)$$

As in the MT case, the parameters of the model ( $\Theta$ ) are estimated on a training dataset. In this case, each training sample is a video-caption pair  $(z, y)$ . Again, the training objective is to maximize the log-likelihood on the training dataset (Eq. (6.3)):

$$\hat{\Theta} = \arg \min_{\Theta} \sum_{s=1}^S \sum_{i=1}^{I^{(s)}} -\log(p(y_i^{(s)} \mid y_1^{i-1(s)}, z_1^J(s); \Theta)) \quad (6.3)$$

where  $z_1^{J(s)}$  and  $y^{(s)}$  denote the  $s$ -th training sample and  $I^{(s)}$  refers to the length of  $y^{(s)}$ .

Therefore, NMT and neural video captioning systems share two out of the three pattern recognition challenges (Section 1.1): the parameter estimation is performed via SGD on a training corpus, aiming to maximize the likelihood and the search problem can be tackled by a beam search method. The main difference between video captioning and NMT is found at the encoder, which is designed to process either text or video.

### 6.1.2 Neural architectures for image encoding

To obtain visual representations, we rely on techniques from the computer vision field. Nowadays, ConvNets (Section 2.3.3) are undoubtedly the most commonly employed model in computer vision. The appearance of huge image databases and the possibility of large-scale training with GPUs has allowed ConvNets to be the most prominent technique in the computer vision field (Krizhevsky et al., 2012; Simonyan and Zisserman, 2014; Szegedy et al., 2015, 2017). Since their successful reintroduction in public challenges (Krizhevsky et al., 2012), ConvNets have been demonstrated to be a powerful tool to tackle several computer vision problems, such as object detection (Russakovsky et al., 2015) or scene recognition (Zhou et al., 2014). ConvNets obtain rich image representations that can be leveraged by other applications, such as multimodal systems. Nowadays, ConvNets represent a powerful method to extract features from images.

Hence, to analyze the video frames, we employ a powerful ConvNet model that has been already trained on large datasets. Without loss of generality, we chose the so-called GoogLeNet architecture (Szegedy et al., 2015), but any other feature extractor could be used within our framework.

#### GoogLeNet

GoogLeNet was the winner of the 2014 ImageNet Large Scale Visual Recognition Challenge (ILSVRC, Szegedy et al., 2015). This network is built upon the so-called Inception modules: a block of convolutional filters with strides of  $1 \times 1$ ,  $3 \times 3$  and  $5 \times 5$  together with a maximum pooling operation applied to their inputs. In order to reduce the number of parameters of these modules, these operations are preceded by  $1 \times 1$  convolutions. The final output is obtained by concatenating the result of the convolutions. Fig. 6.1 illustrates an Inception module.

The global GoogLeNet architecture consists of 22 layers, starting with two initial convolution plus max-pooling operations, followed by a stack of Inception modules.

Max-pooling operations are applied after the second and seventh module. After this stack of Inception modules, an average pooling produces a compact representation of size  $d = 1024$ .

We use this final representation as the features extracted by the network. The original GoogLeNet architecture adds an additional layer with 1,000 units, because the network was designed for a classification problem involving 1,000 classes.

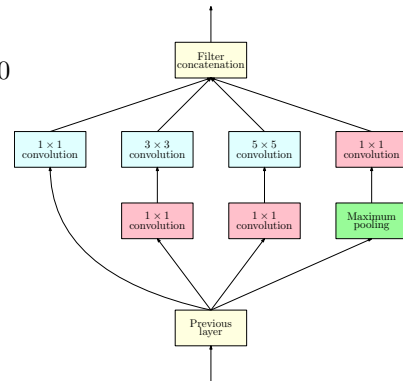
Although it varies depending on the problem and data, GoogLeNet has proven to be one of the best models for feature extraction, offering a good trade-off in terms of performance, number of parameters and computational requirements.

Regarding the features learned by the network, we propose to combine object and context-related information. We used the GoogLeNet architecture separately trained on two datasets: one for object detection (ILSVRC dataset, [Russakovsky et al., 2015](#)), and another one for scene recognition (Places 205 dataset, [Zhou et al., 2014](#)). The combination of these two sources of data can inform about the objects appearing and their surroundings, being ideal for our problem at hand.

Therefore, considering a video as a sequence of  $J$  images  $z_1, \dots, z_J$ , we apply the GoogLeNet models pre-trained on the object and scene detection datasets to each of the elements of this sequence. For each frame, we obtain two feature vectors, which are concatenated, yielding a sequence of  $J$  feature vectors,  $\mathbf{z}_1^J$ , where each element  $\mathbf{z}_j$ , with  $\mathbf{z}_j \in \mathbb{R}^d$  for  $1 \leq j \leq J$ , represents the compact representation of a video frame.

### 6.1.3 ABiViRNet: A bidirectional video captioning system

Although effective, the previous ConvNet architecture removes the temporal dependencies existing across the frames of a video. To address this issue, we apply a BRNN network ([Section 2.3.1](#)) to process the features extracted by the ConvNet. This allows the model to induce bidirectional relationships on the sequence of frames.



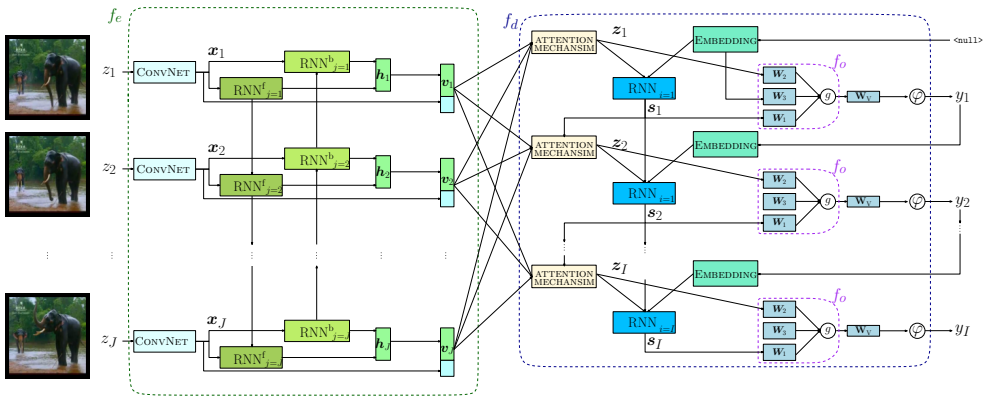
**Figure 6.1:** Inception module, as illustrated by [Szegedy et al. \(2015\)](#). It applies several convolutional filters and a maximum pooling to its inputs. The output is built by concatenating the results of the convolutions.

Hence, the sequence of feature vectors  $\mathbf{z}_1^J = \mathbf{z}_1, \dots, \mathbf{z}_J$  is processed by a BRNN ( $f_e$ ), that computes a sequence of annotations, similarly to the NMT model (Section 3.1), but from the video features, as in Eq. (6.4):

$$\mathbf{h}_1^J = f_e(\mathbf{z}_1, \dots, \mathbf{z}_J) \quad (6.4)$$

where  $\mathbf{h}_1^J$  is a sequence of  $J$  annotations of  $k$  dimensions computed by the BRNN network. As in the NMT model, each element  $\mathbf{h}_j \in \mathbb{R}^k$ , for  $1 \leq j \leq J$ , can be seen as a representation of size  $k$  of the elements surrounding the  $j$ -th video frame.

Finally, we combine the  $\mathbf{z}_1^J$  and  $\mathbf{h}_1^J$  sequences, introducing into the decoder both the original ConvNet features and the representation obtained after the BRNN. We empirically found that the system is improved from this augmented features. To that end, we concatenated the features, yielding a final sequence  $\mathbf{v}_1^J$  of  $J$  feature-augmented vectors, where each  $\mathbf{v}_j = [\mathbf{z}_j; \mathbf{h}_j] \in \mathbb{R}^{d+k}$ , for  $1 \leq j \leq J$ .



**Figure 6.2:** Architecture of the ABiViRNet model for video captioning. This model follows the attentional encoder–decoder framework described in Section 3.1. The input video is given as a sequence of frames and it is processed by a ConvNet which extracts visual features. A bidirectional RNN scans the sequence of features to compute an alternative representation, containing relationships across frames. These two representations are concatenated and fed to the decoder RNN via an attention mechanism. Following the decoder RNN, we apply a deep-output layer ( $f_o$ ), projecting the decoder state, the context vector and the embedding of the previously generated word into a common space and applying a non-linear function ( $g$ ). Finally, we project this representation to the language vocabulary space. Applying the softmax function ( $\varphi$ ) we obtain a probability distribution over the words. The caption is generated by searching the sequence of words that provides the highest probability.

The decoder is very similar to the RNN-based NMT one (Section 3.1): it consists of an RNN network, which acts as a language model, conditioned by the information provided by the encoder: in this case, the video frames. The decoder is equipped with

an attention mechanism as described in [Section 3.1](#), that is applied on the sequence computed by the encoder ( $v_1^J$ ). Considering that each of our feature vectors describes the scene at a different temporal moment, this dynamic attention mechanism acts as a trainable saliency mechanism applied through time, which is able to weight and emphasize the information from different frames.

The words are generated as in the NMT case: we project the representation computed by the decoder to a vocabulary-sized space, followed by a softmax function ([Eq. \(3.10\)](#)). To find the caption with the highest probability, we rely on the beam-search method ([Section 3.3](#)). We called this model “attention bidirectional video recurrent net” (ABiViRNet [Peris et al., 2016](#)). [Fig. 6.2](#) shows an illustration of the system.

#### 6.1.4 Experimental setup

We tested our video captioning system on the Microsoft Research Video Description (MSVD) dataset ([Chen and Dolan, 2011](#)), a collection of 1970 open domain video clips collected from YouTube and annotated using a crowd sourcing platform. Each video has a variable number of captions, written by different users. We used the splits made by [Venugopalan et al. \(2014\)](#); [Yao et al. \(2015\)](#), separating the dataset in 1200 videos for training, 100 for validation and the remaining 670 for testing. During training, the video clips and each one of their captions were treated separately, accounting for a total of more than 88,000 video-caption training samples. [Fig. 6.3](#) shows a sample from the MSVD dataset.

In order to evaluate and compare the results of the different models we used the standardized COCO-Caption evaluation package ([Chen et al., 2015](#)), which provides several metrics for text description comparison. We used three main metrics, BLEU, METEOR and CIDEr (see [Section 1.4.1](#)). Recall that, while we present BLEU and METEOR results as a percentage, the CIDEr values range from 0 to 10.

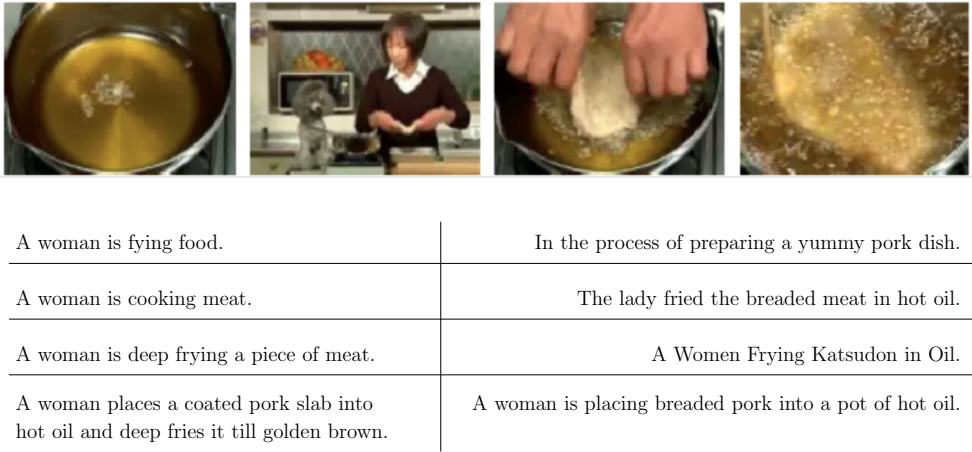
### Video captioning systems

We built our systems using the Theano framework ([Theano Development Team, 2016](#)). Our source code is publicly available<sup>1</sup>. We used LSTM ([Section 2.3.1](#)) networks as recurrent units for our system. In the case of applying a bidirectional encoder, this was also a BLSTM network. We used the concatenation operator to join the forward and backward states from the BLSTM network. Our attention mechanism had an additive attention function ([Eq. \(2.19\)](#)), with a single-layered MLP.

For extracting the visual features, we tested four main configurations:

---

<sup>1</sup><https://github.com/lvapeab/ABiViRNet>.



**Figure 6.3:** Training sample from the MSVD dataset. We only show an excerpt of 4 video frames along with 10 out of the 34 references that this video had. Note that captions can contain typos (e.g. *fying* food), since the dataset was labeled via crowdsourcing.

1. **Objects:** the features were obtained from a GoogLeNet trained for object detection. The features from the ConvNet fed directly an attentive LSTM decoder. This can be considered the baseline system, proposed by Yao et al. (2015).
2. **Objects + Scenes:** we used features from two ConvNets, trained on object detection and scene classification. Both features were concatenated.
3. **Objects + BLSTM:** represents the system described in Section 6.1.3: the visual features contained information from objects and were processed by a BLSTM before being introduced to the decoder.
4. **Objects + Scenes + BLSTM:** combination of both previous systems, visual features containing object and scene information, processed by a BLSTM.

We trained our systems using a batch size of 64 samples, using the Adadelta (Zeiler, 2012) optimizer with its default parameters. To deal with the overwhelming amount of visual information that a video contains, we applied a frame subsampling, following Yao et al. (2015). This strategy consists of picking only one image every 26 frames to reduce the computational overload. To obtain the captions, used a beam size of 10. The weight matrices were initialized following Glorot and Bengio (2010). We applied an early stopping strategy (Section 2.4) according to the BLEU of the development set. We evaluated our system every 1000 updates and the patience was set to 5.

To choose the optimal hyperparameters, we followed a random search strategy (Bergstra and Bengio, 2012). For each configuration we run 10 experiments. In each of them, we randomly set the value of the critical model hyperparameters: the size of the target word (tested on the range [300, 700]), the size of the decoder RNN (tested on the range [1000, 3000]) and the size of the BRNN encoder (tested on the range [100, 2100]). The attention mechanism size was the same of the decoder RNN. The optimal hyperparameters obtained through this exploration are shown in Table 6.1.

**Table 6.1:** Optimal hyperparameters for the captioning encoder–decoder found via random search for each configuration.

Configuration	Encoder	Decoder	Embedding
Objects	–	2231	476
Objects + Scenes	–	2256	451
Objects + BLSTM	2 × 717	484	301
Objects + Scenes + BLSTM	2 × 823	3245	628

### 6.1.5 Results and discussion

In Table 6.2 we report the results of the best models on the test set. The first row corresponds to the result obtained with our system with the object features from Yao et al. (2015). The configurations reported below the horizontal line are our proposals, where *Scenes* indicates we use scene-related features concatenated to *Objects* and *BLSTM* denotes the use of the additional BLSTM encoder.

**Table 6.2:** Text generation results for each model on the MSVD dataset. The results below the horizontal line are our proposals. † indicates the model from Yao et al. (2015) (only *Object* features) evaluated on our system.

Model	BLEU [↑]	METEOR [↑]	CIDEr [↑]
Objects†	51.5	32.5	0.66
Objects + Scenes	52.6	32.5	0.67
Objects + BLSTM	<b>53.6</b>	<b>32.6</b>	0.66
Objects + Scenes + BLSTM	52.8	31.3	<b>0.67</b>

Analyzing the results obtained, a clear improvement trend can be derived when applying the BLSTM as a temporal inference mechanism. The BLSTM addition when using *Objects* features allows to improve the result on all metrics, obtaining a benefit of more than 2 BLEU points. Adding scene-related features also slightly improves the result, although it is not as remarkable as the BLSTM improvement. The combination of *Objects+Scenes+BLSTM* offers the best CIDEr performance. Nevertheless, this



result is slightly below the *Objects*+BLSTM configuration for the rest of metrics. This behavior is probably due to the significant increase in the number of parameters to learn. It should be investigated whether the reduction of the number of parameters by reducing the size of the ConvNet features, or the use of larger datasets could lead to further improvements.

The bidirectional architecture introduced into the captioning system had the ability to infer temporal relationships from the data, in both directions. Hence, the features provided by a bidirectional encoder will incorporate information, being even more evident in the initial frames where otherwise the result would only take into account a short time-span. On the other hand, the use of a bidirectional model causes a significant increase on the number of parameters on the encoder, which increases the computational time and the amount of data needed to train the model. The use of complementary object and scene-related image features has proven to obtain richer video representations, leading to generally outperform the usage of only object-related features, for the problem at hand.

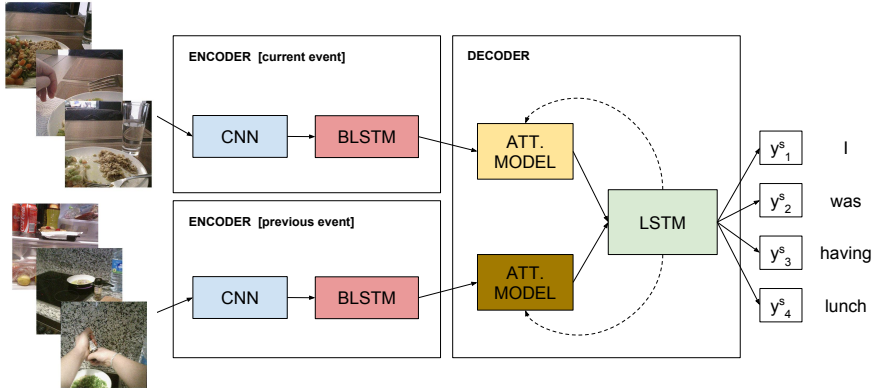
These results suggest that deep structures help to transfer the knowledge from the input sequence of frames to the output natural language caption. Hence, the next step to take must delve into the application of deeper modeling structures, such as 3D-ConvNets (Tran et al., 2015), that allow the recognition of actions and may solve some of the ambiguities existing in the tested methods, which only cope with object and scene recognition. An additional future step should study the inclusion of spatial-temporal attention models for better coping with the nature of natural videos. The usage of more powerful alternatives—deep RNNs or the Transformer model—could also bring improvements on the performance of the system. However, such deep structures involve a larger amount of parameters, whose estimation can be problematic, especially with small datasets.

## 6.2 Egocentric captioning

We now tackle the egocentric captioning problem, a particular case of a video description task. This problem is particularly challenging due to two main characteristics inherent to the egocentric images: on the one hand, most wearable cameras have a small field of view. This may condition the images, and they are frequently distorted, noisy and hence, they are complex to analyze. On the other hand, the life-logging cameras used for egocentric vision have a low temporal resolution (2-3 frames per minute). Therefore, the egocentric events are not videos, but collection of temporally ordered images.

In this section, we will focus on the problem of automatically generating captions from daily egocentric images. We hypothesize that, within a day, some of the events

that compose it follow a temporally logical relation. That is, previous actions occurring during a day influence the following ones. For instance, if a given event shows a cooking scene, it is likely that in the next event the user proceeds to eat the food.



**Figure 6.4:** High-level outline of the proposed temporally-linked multi-input attention model (TMA). To describe an event, given as a sequence of frames, the model is able to exploit information extracted from the previous event, via an independent encoder and using two attention mechanisms.

Therefore, we need a model able to capture and exploit these relationships. To that end, we modify the captioning system presented in [Section 6.1.3](#), introducing an extended context, to deal with the previous daily events. We call this approach a “temporally-linked multi-input attention model” (TMA). Our proposal is able to embed previous information coming from either image or language. In [Fig. 6.4](#), we show a high-level outline of the proposed method. We illustrate the previous example: the frames from previous events are related to food preparation and this can be employed to determine that the user is about to eat.

## Egocentric vision

As previously stated, the field of egocentric vision involves a number of different challenges and problems, like activity recognition, event classification ([Castro et al., 2015](#)), or our task of interest: event captioning. Regarding any of these subproblems, in order to provide a coherent description or labeling of the actions and events happening in egocentric images, the first step required is the segmentation of a complete day of the user into so-called events. According to [Dimiccoli et al. \(2017\)](#), an event is

*“a semantically perceptual unit that can be inferred by visual image features, without any prior knowledge of what the camera wearer is actually doing.”*

Event segmentation can be done in an automatic manner (Lu and Grauman, 2013; Poley et al., 2016; Dimiccoli et al., 2017). After the application of the day segmentation and having as output the set of events of the day, we can acquire the self-contained units of information required for the application at hand, event captioning systems, activity recognizers or event classifiers.

Considering the egocentric captioning problem, few works have been proposed in the literature. Only two of them tackled it as a video description problem and from an end-to-end perspective. Fan and Crandall (2016) and Fan et al. (2018) explored the problem of creating image diaries in order to apply image retrieval. As a step in the process, the authors proposed an image captioning method that processes one image at a time by applying a ConvNet to extract image features and an RNN to generate sentences. Later, they proposed to group the images and apply a sentence fusion technique to provide the final captions for each event along the day. Goel and Naik (2016) also focused on the task of image retrieval for both conventional and egocentric videos. They applied a simple method of video description composed of a ConvNet for feature extraction, a BLSTM for image sequence combination and a LSTM in the decoder to apply the final sentence generation for 5-seconds-long clips of video. The purpose of this method in the work was to provide semantic information of the available data for the posterior retrieval.

### Extending the context

For our task of describing events, we assume that such events have been previously extracted, either manually or automatically. As discussed before, it is important to take into account relevant information from previous events to generate the caption of the current one. Thus, our model treats the image sequences from different events as temporally-linked units. It jointly models and exploits intra-eventual information (flowing through the frames of a single event) together with inter-eventual information (linking temporal sequences of neighboring events).

Most work relating video captioning (e.g. Venugopalan et al. (2015); Yao et al. (2015), Section 6.1.3) assume a sample-wise independence, i.e. that a sample is meant to be unrelated to the next one. This may represent a limitation in tasks which aim to model continuous events, split according to an arbitrary criterion. This is the class of problems that we intend to address. In order to tackle it, we divide a day into events and describe each one of these events. Due to this division, our samples are conditioned between them. Therefore, the classical sample-wise independence

assumption becomes excessively severe, as critical information may be potentially lost. We propose a novel model that takes into account information from past events, being suitable to this kind of tasks.

We aim to exploit this information by incorporating into our model, at a given temporal point, the information extracted from the previous event. This information can be either textual (previous caption), visual (previous sequence of images) or both. Therefore, we develop a model that takes into account the information from both sources: the current sequence of frames together with the information coming from previous events.

Related to this, few works have been proposed to take into account mid-term temporal information in conventional videos. [Krishna et al. \(2017\)](#) extracted C3D features on variable-sized conventional videos and computed several temporal segmentation proposals in short actions. Later, the model generated textual descriptions for each action incorporating contextual information from the past and future actions belonging to the same video. In comparison to our problem of egocentric life-logging data, apart from the different perspective of the sequences, their videos are much shorter. Moreover, instead of consisting of several long-term events, theirs contain mid-term short actions belonging to the same event. Even though there are important differences to both problems, the authors proved that incorporating past and future information can help the prediction of current events or actions.

It is also worth relating the inclusion of wider contexts with the MT community. Document-level MT is a topic that is receiving increased interest. Several works proposed hierarchical structures to encode information from past events ([Wang et al., 2017](#)), following a similar strategy as in our TMA model.

### 6.2.1 Egocentric captioning of daily events

Similarly to the video description problem, in the egocentric task, we have as input a sequence of frames and we want to output a sentence that describes the input. This problem has already been tackled in the literature on conventional videos with multiple variations (see [Sections 6.1](#) and [6.1.3](#)). In [Section 6.1.3](#), we proposed to encode the frames by means of convolutional and BRNN networks. The representation obtained was fed to an RNN-based decoder, equipped with an attention mechanism, which generated the corresponding caption. We start from this architecture in order to develop our egocentric captioning system.

The main difference that characterizes the modeling for egocentric captioning is the presence of the temporally-linked events, which share a relationship. Thus, we propose a system able to take advantage from both the current sequence of egocentric images and the action happened in the previous event.

Let  $s$  be an index of the events occurred within a day. Our system considers several inputs: one belonging to the current event (the sequence of frames) and additional inputs regarding previous events. Provided that we manage two different modalities, the information relating the previous event ( $s - 1$ ) can be either the caption generated by the system for the previous event, the sequence of video frames belonging to the previous event, or both sequences. With this information, the system must generate the caption for the current event, as a sequence of words.

We process each input sequence with an independent encoder. The encoded sequences feed a recurrent decoder, through independent attention mechanisms. Similarly to NMT, the decoder defines a probability distribution over the task vocabulary. The caption is generated word by word, as in NMT. Therefore, this model is a novel extension of the classical encoder–decoder approach which introduces different encoders: one for the current sequence of images and another for the information from previous events. These encoders are described in [Section 6.2.2](#). In [Section 6.2.3](#), we detail the multi-input, attention-based RNN. Finally, in [Section 6.2.4](#), we show the full picture of the proposed model.

## 6.2.2 Encoders

Our problem at hand involves the encoding of sequences with two different modalities: images and text. Previously, we already dealt with such modalities ([Sections 3.1](#) and [6.1.2](#)): video captioning systems process the image sequences with ConvNets, to extract visual features, followed by bidirectional RNN networks, to model temporal relationships among them. Text sequences are modeled as in NMT systems, by projecting the words to a continuous space with an embedding matrix and processing the word embeddings by a bidirectional RNN.

In our model, we again use the GoogLeNet architecture (see [Section 6.1.2](#)), pre-trained on the ILSVRC dataset as image feature extractor. Note that it is possible to use any other alternative model to deal with the images representation in our TMA model. The relationships existing across the different frames are modeled by a BRNN network. Following the process described in [Section 6.1.3](#), this encoding stage generates a sequence of visual features  $\mathbf{v}_1^J = \mathbf{v}_1, \dots, \mathbf{v}_J$  from a sequence of images  $z_1^J = z_1, \dots, z_J$ .

Regarding the encoding of textual inputs, we rely on BRNN networks, as in [Section 3.1](#). Therefore, a sequence of  $K$  words  $x_1, \dots, x_K$  is encoded as a sequence of annotations  $\mathbf{h}_1^K = \mathbf{h}_1, \dots, \mathbf{h}_K$ .

### 6.2.3 Decoder: Multi-input attention RNN

Since we are in a multi-modal scenario dealing with sequences of text and images, we propose to use an RNN network that accepts multiple inputs (from multiple sources) and combines them after applying an independent attention mechanism for each input. The combination, as well as the attention mechanisms, are thus learned together with the rest of the model. We describe an LSTM version of the multi-input RNN, but this is applicable to other architectures, such as GRUs or conditional units.

Recall from [Section 2.3.2](#) that attention mechanisms can be used to bridge encoder and decoder: they compute a representation of an encoded input object, dependent on the decoding state and according to an attention function. When embedded into a recurrent decoder, they relate the sequence of representations computed by an encoder with the previous RNN hidden state  $\mathbf{s}_{i-1}$ . Our multi-input decoder is a natural extension of the classical LSTM decoders, able to process an arbitrary number of different inputs. For simplicity, in this section we assume that only two different modalities are introduced into the decoder: the sequence of image features ( $\mathbf{v}_1^J$ ), and the sequence of text annotations, also in matrix form ( $\mathbf{h}_1^K$ ). Such inputs are processed by two independent attention models, which compute, for each decoding time-step  $i$ , two dynamic context vectors, according to the previous decoder state  $\mathbf{s}_{i-1}$ :

$$\mathbf{z}_i = \varphi(a(\mathbf{s}_{i-1}, \mathbf{v}_1^J))\mathbf{v}_1^J \quad (6.5)$$

$$\mathbf{z}'_i = \varphi(a'(\mathbf{s}_{i-1}, \mathbf{h}_1^K))\mathbf{h}_1^K \quad (6.6)$$

where  $a$  and  $a'$  are the attention functions for the visual and textual data, respectively, and  $\mathbf{z}_i$  and  $\mathbf{z}'_i$  denote the visual and textual representation obtained by the attention mechanisms and  $\varphi$  is the softmax activation function ([Table 2.1](#)).

Hence, at a given time-step,  $i$ , we have two inputs from the attention models ( $\mathbf{z}_i$  and  $\mathbf{z}'_i$ ). As in regular NMT or multimedia captioning, the recurrence is applied on the output sequence, shifted one time-step to the right, in order to provide the decoder with autoregressive properties. As in regular LSTMs (see [Section 2.3.1](#)), our multi-input network maintains a hidden state and a memory state:

$$\mathbf{s}_i = \mathbf{o}_i \odot \mathbf{c}_i \quad (6.7)$$

$$\mathbf{c}_i = \mathbf{f}_i \odot \mathbf{c}_{i-1} + \mathbf{i}_i \odot \tilde{\mathbf{c}}_i \quad (6.8)$$

where  $\odot$  denotes the element-wise product,  $\mathbf{c}_{i-1}$  is the previous time-step memory state and  $\tilde{\mathbf{c}}_i$  is the updated memory state.  $\mathbf{o}_i$ ,  $\mathbf{f}_i$  and  $\mathbf{i}_i$  are the output, forget and input gates, respectively, which modulate the states.  $\tilde{\mathbf{c}}_i$  is computed taking into account the attended representations of the inputs ( $\mathbf{z}_i$  and  $\mathbf{z}'_i$ ), the last word generated by the decoder ( $y_{i-1}$ ) and the previous hidden state ( $\mathbf{s}_{i-1}$ ), as in [Eq. \(6.9\)](#):

$$\tilde{c}_i = \tanh(U_c \mathbf{E}(y_{i-1}) + \mathbf{W}_c \mathbf{s}_{i-1} + \mathbf{A}_c \mathbf{z}_i + \mathbf{B}_c \mathbf{z}'_i + \mathbf{b}_c) \quad (6.9)$$

where  $\mathbf{E}$  is the embedding matrix and  $\mathbf{E}(y_{i-1})$  denotes the word embedding of the previous output word.  $\mathbf{W}_c$ ,  $U_c$ ,  $\mathbf{A}_c$  and  $\mathbf{B}_c$  are the weight matrices for the word embedding of  $y_{i-1}$ , the previous hidden state and both context vectors from the alignment attention models, respectively, and  $\mathbf{b}_c$  is the bias term.

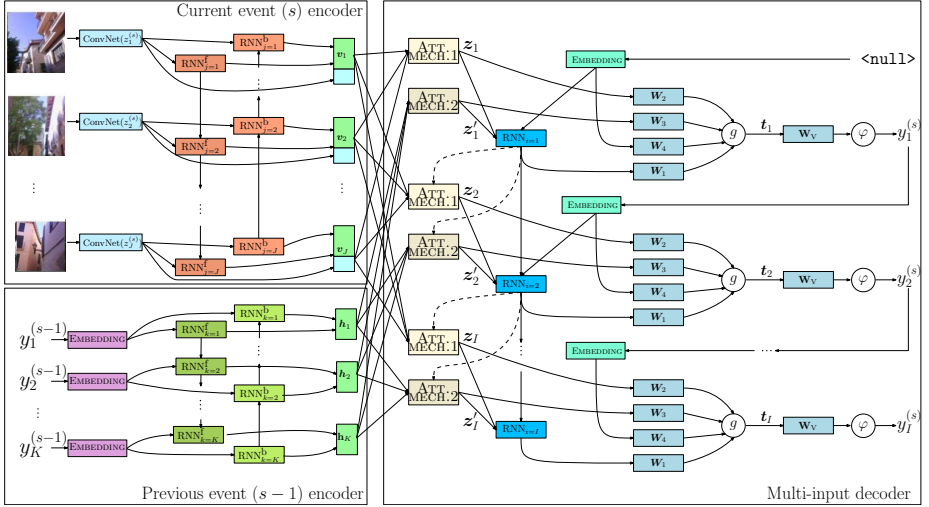
The multiple inputs are defined analogously for the three LSTM gates, as shown in Eq. (6.10):

$$\mathbf{f}_i = \sigma(U_f \mathbf{E}(y_{i-1}) + \mathbf{W}_f \mathbf{s}_{i-1} + \mathbf{A}_f \mathbf{z}_i + \mathbf{B}_f \mathbf{z}'_i + \mathbf{b}_f) \quad (6.10)$$

$$\mathbf{i}_i = \sigma(U_i \mathbf{E}(y_{i-1}) + \mathbf{W}_i \mathbf{s}_{i-1} + \mathbf{A}_i \mathbf{z}_i + \mathbf{B}_i \mathbf{z}'_i + \mathbf{b}_i) \quad (6.11)$$

$$\mathbf{o}_i = \sigma(U_o \mathbf{E}(y_{i-1}) + \mathbf{W}_o \mathbf{s}_{i-1} + \mathbf{A}_o \mathbf{z}_i + \mathbf{B}_o \mathbf{z}'_i + \mathbf{b}_o) \quad (6.12)$$

#### 6.2.4 Temporary-linked Multi-input Attention model



**Figure 6.5:** Architecture of the proposed TMA model. Unlike the traditional approaches, our architecture consists of at least two encoder stages; one for the current sequence and another one (or more) for the previous sequence; and a decoder stage that combines the information of all the previous stages using a multi-input attention RNN.

Regarding our TMA model, illustrated in Fig. 6.5, we process the sequence of images captured by the camera for the current event  $s$ ,  $z_1^J(s) = z_1(s), \dots, z_J(s)$ , to obtain the sequence of visual frames of the current event:  $\mathbf{v}_1^{(s)}, \dots, \mathbf{v}_J^{(s)}$ .

Simultaneously, we process the previous event ( $s - 1$ ). The information of the previous event can be either the sequence of images from the previous event ( $z_1^{J' (s-1)}$ ), the caption generated by the system for the previous event ( $y_1^K (s-1)$ ), or both sequences. In case of processing the previous sequence of images, the method is the same as the one applied on the current event: a ConvNet combined with a BRNN network. When the past event information is the caption  $y_1^{(s-1)}, \dots, y_K^{(s-1)}$ , we follow the textual encoding procedure to compute the sequence of annotations  $\mathbf{h}_1^{(s-1)}, \dots, \mathbf{h}_K^{(s-1)}$ . Since we are modeling the same language, the embedding matrix is shared with the decoder.

In the case we are using both modalities, the model has three different encoders: two visual encoders for the frames from the current and past events, and a textual encoder for the previous caption. Therefore, this model features three different attention mechanisms, one for each input, which are integrated into the decoder by means of the multi-input LSTM described in Section 6.2.3. For the sake of brevity, in the following we will stick to the model that takes previous frames as past event.

Analogously to the RNN-based NMT model, we apply a deep output structure after the decoder RNN, as described in Eq. (3.9): at each time-step  $i$ , the output state of the decoder ( $\mathbf{s}_i$ ) is combined with both context vectors computed by the different attention mechanisms ( $\mathbf{z}_i$  and  $\mathbf{z}'_i$ ), and the embedding of the previously generated word  $\mathbf{E}(y_{i-1})$ . We apply a non-linear activation function  $g$  (typically a tanh) to obtain an intermediate representation  $\mathbf{t}_i$ , as in Eq. (6.13):

$$\mathbf{t}_i = g(\mathbf{W}_1 \mathbf{s}_i + \mathbf{W}_2 \mathbf{z}_i + \mathbf{W}_3 \mathbf{z}'_i + \mathbf{W}_4 \mathbf{E}_i(y_{i-1}) + \mathbf{b}_i) \quad (6.13)$$

$\mathbf{W}_1, \mathbf{W}_2, \mathbf{W}_3, \mathbf{W}_4$  are the skip-connections weight matrices (orange in Fig. 6.5).

Finally, we compute a probability distribution in Eq. (6.14) by projecting  $\mathbf{t}_i$  to the target vocabulary space, by means of a fully-connected layer, followed by a softmax activation (analogously to Eq. (3.10)):

$$\mathbf{p}_i = \varphi(\mathbf{W}_V \mathbf{t}_i + \mathbf{b}_V) \quad (6.14)$$

This distribution  $\mathbf{p}_i$  represents the probability of a word  $y$  appearing in position  $i$  given the input video  $z_1^J (s)$ , the previous event  $z_1^{J' (s-1)}$ —video or caption—and the words generated so far in the current sequence  $y_1^{i-1} (s)$ , as in Eq. (6.15):

$$\mathbf{p}_i = p(y_i^{(s)} | z_1^J (s), z_1^{J' (s-1)}, y_1^{i-1} (s); \Theta) \quad (6.15)$$

which is an approximation to the true probability distribution, defined in Section 6.2.4:



$$p_i \approx \Pr(y_i^{(s)} | z_1^{J(s)}, z_1^{J'(s-1)}, y_1^{i-1(s)})$$

The model is parameterized by the set of parameters  $\Theta$ . All parameters are jointly estimated over a dataset  $\mathcal{S}$ , which consists of  $S$  image sequences and caption pairs. The training objective is to minimize the negative log-likelihood on this training set:

$$\hat{\Theta} = \arg \min_{\Theta} \sum_{s=1}^S \sum_{i=1}^{I_s} -\log p(y_i^{(s)} | z_1^{J(s)}, z_1^{J'(s-1)}, y_1^{i-1(s)}; \Theta) \quad (6.16)$$

where  $I_s$  is the length of the  $s$ -th caption. If the previous event,  $z^{(s-1)}$  is undefined, we introduce an artificial empty event.

The most likely caption is obtained with beam search as described in [Section 3.3](#).

### 6.2.5 The EDUB-SegDesc dataset

EDUB-SegDesc<sup>2</sup> (Bolaños et al., 2018a) is a dataset that can be used either for egocentric events segmentation (Dimiccoli et al., 2017) or for egocentric sequences description. It was specifically acquired and labeled for the purpose of developing a model able to describe and understand all the events appearing along the day of a person. Thus, the main application of using egocentric sequences for textual description generation is providing a memory aid for patients suffering mild cognitive impairment.

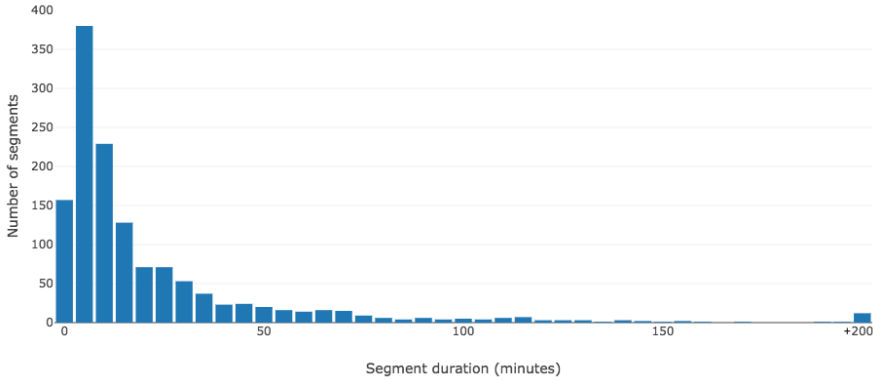
The dataset was acquired by the wearable camera Narrative<sup>3</sup>, taking a picture every 30 seconds (2 frames per minute). It consists of 55 days acquired by 9 people. Each day was manually segmented in events or sequences following the same criterion as in Dimiccoli et al. (2017).

[Fig. 6.6](#) shows a histogram with the duration (in minutes) of the resulting segmented events. We can observe a wide variability in duration. Most of the events (around 65%) have relatively short durations of 15 minutes or less, but there also exist several long events (around 5%) with a duration longer than 100 minutes.

The dataset contains a total of 48,717 images, divided in 1,339 events (or image sequences) and 3,991 captions, and has an average of 3 captions per event. It was divided in training, validation and test splits making sure that all the sequences from the same day should belong to the same data split. The division results in the figures depicted in [Table 6.3](#).

<sup>2</sup><http://www.ub.edu/cvub/edub-segdesc>

<sup>3</sup>[www.getnarrative.com](http://www.getnarrative.com)

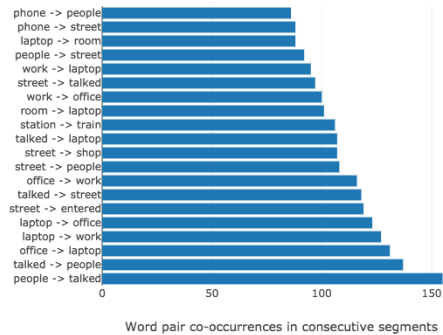


**Figure 6.6:** Histogram depicting the duration of the segmented events in the EDUB-SegDesc dataset. All segments with a duration equal or greater than 200 minutes appear grouped in the last bin (+200).

**Table 6.3:** Figures of the EDUB-SegDesc dataset, according to each partition: training, validation and test.

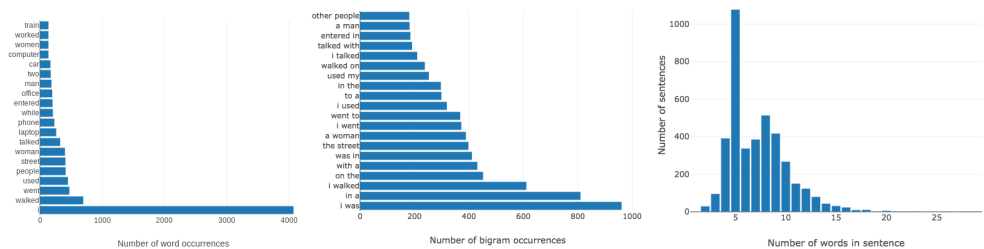
EDUB-SegDesc	Training	Validation	Test	Total
#days	39	7	9	55
#images	32,664	7,301	8,752	48,717
#events	889	204	246	1,339
#descriptions	2,652	598	741	3,991

In Fig. 6.7 we show the number of co-occurrences in consecutive events from some manually chosen keywords. This highlights the natural relationships found in consecutive events. Some notable examples of concepts appearing in consecutive events include: “people” in past events followed by “talked” in current events (social events); “laptop” followed by “work” (work-related events); “street” followed by “entered” (going from an outdoors to an indoors environment); “station” followed by “train” (transport-related events) or “phone” followed by “street” (events related to using the mobile phone on the street).



**Figure 6.7:** Number of co-occurrences in consecutive events of several keywords. Word pairs are shown in the following format: `word1 ->word2`, where `word1` appears in the past event and `word2` appears in the current event.

Fig. 6.8 shows several word-related statistics: occurrences of the most common words and bigrams and histogram sentence lengths. Note that the number of appearances of the word “I” is very high both in the single word and the bigram counts given the egocentric nature of the dataset. Other commonly occurring words and bigrams are those related to events where the user is “walking” and/or on the “street”. Curiously, there is a certain bias in the number of words contained in the annotations, where a considerable number of the sentences are composed of 5 words. Some examples of 5-word sentences are “I went to my office”, “I worked with my laptop”, or “I walked on the street”.



**Figure 6.8:** Text-related statistics of the dataset. Occurrences of the most common words (left), occurrences of the most common bigrams (center), and histogram of number of words in all the sentences (right).

Finally, we show some examples of the events and sentences contained in the dataset (Fig. 6.9). The low temporal resolution of the camera used (2 frames per minute) becomes clear in dynamic events, where the user moves and this causes us to have highly variable environments. This fact, together with the limited information present in some of the images highlights the difficulty of the problem.

### 6.2.6 Experimental setup

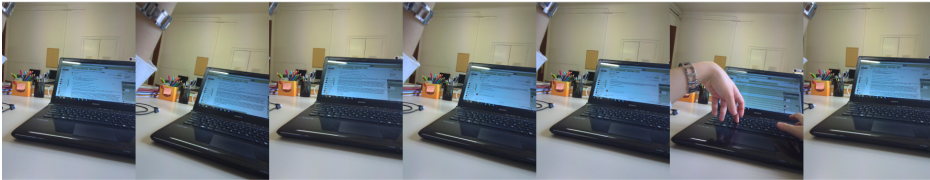
We tested our model TMA on the EDUB-SegDesc dataset. To the best of our knowledge, this was the only egocentric corpus that allowed us to exploit the correlation between events. We evaluated the proposal using standard image and video captioning metrics: we used the COCO-Caption evaluation package (Chen et al., 2015) and computed BLEU, METEOR and CIDEr.



I entered in classrooms full of people.  
I entered in classrooms and talked to different people.  
I walked between two classrooms.



I went to an office.  
I walked into my office.  
I climbed up to an office.



I worked with a laptop.  
I used a laptop.  
I used a laptop to work in an office.

**Figure 6.9:** Subset of a day from the dataset EDUB-SegDesc. We show some consecutive events and their respective references. The difficulty of the problem is highlighted by the dynamism and instability of the scene, when the user is moving. Particularly difficult examples can be seen in the second event: the references explain that the user is heading to his/her office, but this aspect is only manifested in the last image of the sequence.

## Neural models

All neural models were built using the Keras (Chollet et al., 2015) library. We release the source code of our implementation for future comparisons<sup>4</sup>. The full model was jointly trained end-to-end on the EDUB-SegDesc dataset, except for the ConvNet,

---

<sup>4</sup><https://github.com/MarcBS/TMA>.

which was already pre-trained for object detection on ImageNet (Russakovsky et al., 2015) and remained static during the training of the model.

In order to minimize the influence of randomness in our results, mostly due to the weights random initialization, each experiment was run 5 times, and we reported the average of the runs. We explored two different training strategies: training from scratch, using exclusively the data from the EDUB-SegDesc dataset; or fine-tuning pre-trained models. We studied the inclusion of pre-trained word embeddings (see Section 2.3.4), obtained using the skip-gram model from Mikolov et al. (2013a) and trained on part of Google News dataset. We also tested training the decoder as a language model on the 1 Billion words dataset (Chelba et al., 2013), but results in both cases were not better. Finally, we also tested reusing the pre-trained weights from the video captioning model from Section 6.1.4 (trained on the MSVD dataset), which improved the results under certain model configurations. As presented in the previous section, the MSVD dataset contains short clips from YouTube annotated by different users, accounting for more than 80,000 training samples. In terms of vocabulary coverage, approximately 98% of the words from EDUB-SegDesc were present in the MSVD dataset.

The main hyper-parameters of the model were selected according to the analysis reported in the video description task (Section 6.1.4). Moreover, since we conducted experiments with pre-trained models, we must keep fixed some hyper-parameters. Therefore, we stuck to the hyperparameter values shown in Table 6.1: word embeddings of size 301, BLSTM encoder of 717 units on each layer, and a LSTM decoder of 484 units. In the case of using pre-trained word embeddings, they had a dimension of 300. The initial state of the decoder LSTM was initialized with the hyperbolic tangent of the mean of the video features obtained by the encoder (Xu et al., 2015). We took at most 26 frames evenly distributed from each complete sequence of the dataset.

We trained the model by mini-batch SGD. We tested the Adam and Adadelta update rules. The best performance was obtained using Adadelta with the default parameters; or Adam with an initial learning rate of 0.001 and a decay of 0.995 at the end of every epoch. We set the batch size to 64 and used early stopping on the validation set based on BLEU, setting the patience to 20 and checking the performance every 50 updates. The size of the beam during the search was 10. Thus, we report the best results in each case. During training, the norm of the gradients was clipped to 10 (Pascanu et al., 2013).

In order to prevent over-fitting, we applied layer normalization (Section 2.4). In contrast to other works, we observed that the use of dropout (Section 2.4) combined with layer normalization produced better performance (using  $p = 0.5$ ). We also

applied weight decay ( $10^{-4}$ ) and Gaussian noise ( $\sigma = 10^{-2}$ ) to the non-recurrent weights.

## 6.2.7 Results and discussion

First, we show the performance of a regular video captioning system and study the influence of several pre-training methods. Next, we compare other state of the art video captioning systems to the TMA model. Finally, we describe several architectural choices that degraded the performance.

### What to fine-tune?

Table 6.4 shows the results of tackling the task as a regular video captioning problem, without incorporating information from previous events. The system model is the same as in Section 6.1.3: the ABiViRNet model. We tested three different configuration: training the system from scratch, using only the data from the EDUB-SegDesc dataset; starting with pre-trained word embeddings and fine-tuning a system trained on the MSVD dataset.

**Table 6.4:** EDUB-SegDesc results on different pre-trained decoders. ABiViRNet refers to the video captioning system from Section 6.1.3. BLEU and METEOR metrics are given in percentage. We compare the basic model trained from scratch to the same model with certain pre-trained components. #params denotes the number of parameters to estimate, given in millions.

	BLEU [↑]	METEOR [↑]	CIDEr [↑]	#params
ABiViRNet	29.6	20.3	0.79	27.3M
ABiViRNet + word2vec	26.0	20.1	0.90	27.3M
ABiViRNet + MSVD	28.5	21.2	0.89	35.1M

As shown Table 6.4, the inclusion of pre-trained embeddings worsens the performance of the system in terms of BLEU and METEOR. We hypothesize that this is due to the different domains on which the word embeddings are trained. The word2vec vectors were trained on a more general domain and their capabilities cannot be exploited to the full in our problem. If we start from the parameters learned from the MSVD data and fine-tune the model with the egocentric dataset, the BLEU score is also lowered, although to a lower extent than with word2vec. Nevertheless, the performance of the MSVD model in terms of METEOR and CIDEr is increased.

## Extending the context

The effect of including information from previous events is shown in [Table 6.5](#). For comparison, we also include the only similar approach in the egocentric video captioning literature: DeepSeek ([Goel and Naik, 2016](#)). This model consists of a non-attentional BLSTM encoder with two layers. The encoder feeds a similar decoder to the one of [Yao et al. \(2015\)](#). We performed additional tests with other state-of-the-art models for video captioning: Enc-Dec Global ([Yao et al., 2015](#)), hLSTMat ([Song et al., 2017](#)) and the already tested ABiViRNet ([Section 6.1.3](#)). Note that none of them considers long-term temporal information from different events.

**Table 6.5:** EDUB-SegDesc results for variations of the TMA model compared to the state of the art, which do not consider information from previous events. We use either the previous caption, the previous sequence of images (video) or both to make the current prediction. BLEU and METEOR metrics are given in percentage. #params denotes the number of parameters to estimate, given in millions. The best results for each metric are shown in boldface. Results with the symbol \* were obtained with the Adam optimizer instead of Adadelta.

	BLEU [↑]	METEOR [↑]	CIDEr [↑]	#params
Enc-Dec Global ( <a href="#">Yao et al., 2015</a> )	28.1	20.8	0.88	44.4M
hLSTMat ( <a href="#">Song et al., 2017</a> )	25.6	20.8	0.88	—
ABiViRNet ( <a href="#">Section 6.1.3</a> )	29.6	20.3	0.79	27.3M
DeepSeek ( <a href="#">Goel and Naik, 2016</a> )	27.9	21.4	0.99	27.2M
TMA previous-caption	30.6	20.4	0.90	39.1M
TMA previous-caption + MSVD	28.3	21.3	0.94	46.9M
TMA previous-video*	31.0	21.4	1.01	43.3M
TMA previous-video + MSVD*	<b>31.9</b>	<b>22.1</b>	<b>1.07</b>	51.0M
TMA previous-video-caption	30.4	21.8	1.00	55.1M
TMA previous-video-caption + MSVD*	29.7	<b>22.1</b>	0.93	62.8M

As detailed in [Section 6.2.3](#), we tested our TMA model introducing the previous caption, the previous sequence of images or both previous caption and images. As before, we distinguish between training from scratch with the EDUB-SegDesc dataset or start from the weights learned with MSVD pre-trained model. Given the results observed in [Table 6.4](#), we dropped the use of word2vec word embeddings.

According to [Table 6.5](#), if we compare the results obtained by the state of the art methods (rows 1-4), which only consider the current sequence of images, to the different configurations of our method (rows 5-10), we can see that our method outperforms the rest. This behavior can be explained by the inclusion of information from previous events in the TMA model. Since it considers a broader context, it is

able to better understand the given event and generally increases the performance of the system. Furthermore, considering the characteristics of egocentric life-logging photo streams, the data provided by a single event often lacks enough information to easily understand what is happening. Thus, providing context from previous events usually improves the captioning results.

In contrast, some state of the art methods outperform certain configurations of our model (see row 3, BLEU; and row 4, METEOR and CIDEr). We understand that this phenomenon can occur, because although providing context from previous events can often be useful, in some cases noise could be introduced for two reasons: 1) the error of the predictions of previous events can be propagated, and 2) some consecutive events could lack any semantic relationship or have a very low number of samples in the training set.

Considering the differences between different configurations of our TMA model, the inclusion of the previous video event as input yields the best generalization results during test evaluation. The previous caption also enhances the system, but to a lower extent. We observed that the inclusion of past videos and captions produced good results on the development set, but the performance of this configuration was dropped on the test set. This phenomenon could imply that, although the model has greater potential, it is also more complex considering the number of parameters, which produces a quicker over-fitting on the training data and the consequent impact produced by the model selection of the validation set (Reunanen, 2003).

The results obtained with the TMA model show that taking into account the previous video event is more effective than considering the previous caption. This effect is partially produced because, at test time, we use as input an output previously generated by the model. Obviously, this output may be erroneous. Therefore, it may lead to the introduction of noise to the system and to error propagation.

From Tables 6.4 and 6.5, it can be concluded that, when fine-tuning from the MSVD model, the METEOR scores are always better than when starting from scratch, but BLEU scores are lowered. This is probably due to vocabulary differences from both tasks. The pre-trained model tends to produce captions with the structure that it learned from the MSVD dataset, while the model trained from scratch on the EDUB-SegDesc generates ad-hoc captions for the task at hand. Since BLEU is based on  $n$ -gram counts, the latter model generally obtains higher scores than the first one, because its captions are more literal. Since the METEOR metric stems and employs synonyms, it is less rigid than BLEU. Therefore, pre-trained models are able to score better than those trained from scratch.



## Failed experiments

In [Table 6.6](#) we report an additional set of comparisons that show several configurations that did not provide good results, compared to those shown in [Table 6.5](#). First, we compared several models either using dropout or not. It appears that combining the use of dropout, layer normalization and Gaussian noise in the same model clearly helps obtaining better results in all the cases. The removal of layer normalization or Gaussian noise produced even worse results.

**Table 6.6:** EDUB-SegDesc results for several additional configurations tested that did not provide good enough results. For each bad result we report its counter example configuration with better results. BLEU and METEOR metrics are given in percentage. #params denotes the number of parameters to estimate, given in millions. The best results for each metric are shown in boldface. Results with the symbol \* were obtained with the Adam optimizer instead of Adadelta. NoDrop relates to systems without dropout and NonInfo to systems trained excluding non-informative images.

	BLEU [↑]	METEOR [↑]	CIDEr [↑]	#params
ABiViRNet - NoDrop	24.8	18.6	0.80	27.3M
ABiViRNet	29.6	20.3	0.79	27.3M
ABiViRNet + MSVD - NoDrop	27.9	19.4	0.89	35.1M
ABiViRNet + MSVD	28.5	21.2	0.89	35.1M
TMA previous-caption - NoDrop	28.1	20.1	0.94	39.1M
TMA previous-caption	30.6	20.4	0.90	39.1M
TMA previous-video + MSVD - NonInfo*	30.3	21.5	0.99	51.0M
TMA previous-video + MSVD*	31.9	22.1	1.07	51.0M

In the second part of the table we compare either using all the images available in the dataset or removing all non-informative images, following the strategy from [Lidon et al. \(2017\)](#). An image is considered to be non-informative if it is dark, blurry or pointing to the sky or ground without showing any object. We observed that the removal of such images has a negative impact on the system, as it worsens performance.

### 6.2.8 Qualitative analysis

We now review some illustrative examples, in order to understand the weaknesses and strengths of the TMA model, as well as the major difficulties that appear in the task at hand.

Fig. 6.10 shows some examples of the predictions produced by the TMA model on consecutive events in the test set. We can observe the influence that the previous event had on the captioning of the second sample. In sample #2, the *TMA previous-video* model is aware that the user was previously in a restaurant. This conditions the model, which generates the caption “*I went to the bathroom*”, which is a likely action when a person is in a restaurant. The *ABiViRNet* model is unable to capture this information.

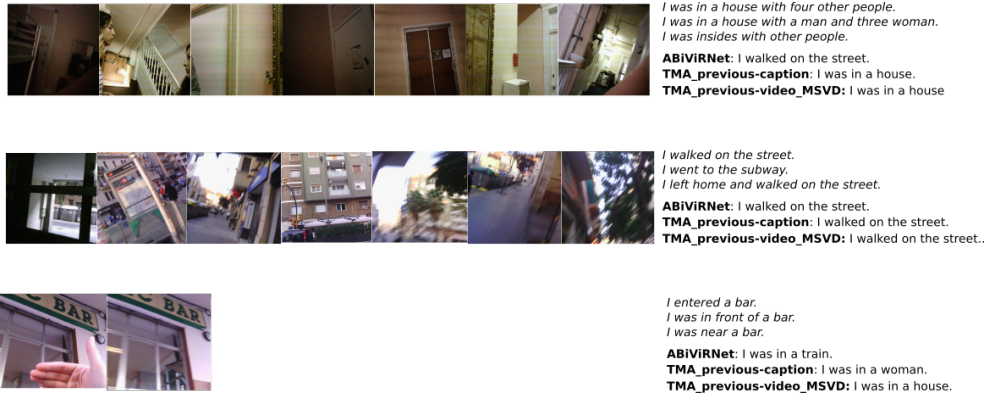


**Figure 6.10:** Comparison of the results obtained by the baseline method ABiViRNet and two of the proposals TMA models, including the previous caption and the previous video + MSVD. Samples relate consecutive events from the test split and the reference captions are in *italic*.

The influence of previous actions can also be observed in events #5 and #6. At event #5, the user is shopping and in the next event, he/she continues shopping. The

*TMA previous-video* model can distinguish that the shopping action occurred in event #6 due to incorporation of the visual information from the previous one. On the other hand, the *ABiViRNet* model is not able to make this inference and deduces that the user is in the street. Similarly, the *TMA previous-caption* model is more prone to error propagation: when it fails to understand a certain event, it is more likely to be also wrong in the following one. For example, in sample #4 the model generates “*I was in a supermarket*”, which leads the model to also fail on the consecutive event #5.

Fig. 6.11 depicts some examples of initial events of different days. We can see that in the case of TMA models, the fact that information from previous events can not be used in this particular cases does not influence the model and obtains better or comparable results than non-TMA models.



**Figure 6.11:** Comparison of the results obtained by the baseline method ABiViRNet and two TMA proposals: previous-caption and TMA previous-video + MSVD. Events corresponding to the start of the day are shown.

Fig. 6.12 shows additional examples of successes and failures of the system. In the success case #1, we can see that the model is able to understand that the user went to the bathroom, although it is not specifically described in the references and it is hard to distinguish just looking at the images (note the third image where the user is drying his hands). In addition, this exemplifies one of the major challenges occurring within the event captioning problem: during an event, many situations may occur. The system may focus only on part of the event, resulting in captions that, although correct, are not found in the references.

In the success case #2, we can see that both *ABiViRNet* and *TMA previous-video* model correctly describe the first event. But as we move on to the next one, the first model is unable to detect that the user is having lunch, while the TMA

## Success cases

#1



*I was on the computer.  
I worked on the computer.  
I worked with a laptop.*

**ABiViRNet:** I used my laptop.  
**TMA\_previous-video\_MSVD:** I worked with a laptop.

#2



*I walked in the hallway.  
I walked in the university.  
I walked around the university.*

**ABiViRNet:** I used my phone.  
**TMA\_previous-video\_MSVD:** I went to the bathroom.

#2



*I walked on the street with friends.  
I walked in a city with a group of friends.  
I walked on the street with a group of friends.*

**ABiViRNet:** I was in the street.  
**TMA\_previous-video\_MSVD:** I walked on the street.

#2




*I was in a restaurant and ate lunch with my friends.  
I was in a restaurant with a group of friends.  
I ate lunch with a group of people.*

**ABiViRNet:** I went to a woman.  
**TMA\_previous-video\_MSVD:** I had lunch with a woman.

## Failure cases

#1



*I went to the underground.  
I waited for the tube in the metro station.  
I was in the underground.*

**ABiViRNet:** I was in the train.  
**TMA\_previous-video\_MSVD:** I waited for the tube.

#1



*I was in the tube.  
I was using the tube.  
I was inside the metro.*

**ABiViRNet:** I was in the train.  
**TMA\_previous-video\_MSVD:** I was in a supermarket.


#2



*I talked to people, walked on the street and used a mobile.  
I walked on the street and used my phone.  
I walked on the street with people and used my mobile phone.*

**ABiViRNet:** I walked on the street.  
**TMA\_previous-video\_MSVD:** I walked on the street.

#2



*I entered in a pub.  
I was with other people in a pub.  
I went to a pub.*

**ABiViRNet:** I was on the street.  
**TMA\_previous-video\_MSVD:** I was in a parade.

**Figure 6.12:** Two success cases (left) and two failure cases (right) of the TMA model *TMA previous-video + MSVD*. All the examples belong to the test set. For each case, the top sequence is the previous event and the bottom sequence is the event that we are predicting. In success case #1, we can see that the TMA model correctly recognizes that the user went to the bathroom even though it is not specified in the reference sentences and it is not straightforward to see in the images (note the third image using a hand drier). The two samples at the right exemplify cases in which the TMA model fails due to certain images appearing in the sequences. For instance, in the failure case #1, the TMA model probably considers that the user is in a supermarket due to the first image, where a vending machine appears. In failure case #2, the TMA model may infer that the user is seeing a parade due to the multiple people and colored lights appearing in the images.

model is capable to infer that the event has changed. Therefore, it incorporates this information and correctly guesses that the user is having lunch. Below these examples, we show some failures. We observe certain conditions on the input images that may explain these failures. In the failure case #1, the model probably infers that the user is in a supermarket due to the vending machine in the first image. The second failure case relates images with multiple persons and colored lights. Hence, the model interprets the images as a parade.

### 6.3 Interactive-predictive captioning

We explore now the application of the interactive-predictive framework described in [Chapter 4](#) to multimodal systems. We will tackle the video captioning task described in the previous sections, together with the image captioning task.

The interactive-predictive framework, as described in [Section 4.3](#) is directly applicable to these multimodal systems. The main difference is the way of encoding the input object. Before being introduced into the encoder–decoder system, we need to compute an adequate representation of it. Depending on the modality of the input object, we thus apply a different feature extractor:

**Images:** as explained in [Section 2.3.3](#) ConvNets are powerful feature extractors. We process the image with a ConvNet and use as features the final representation computed by the ConvNet that preserves positional information. A complete image is thus seen as a sequence of image crops. Then, we can directly apply the sequence-to-sequence framework, as done by [Xu et al. \(2015\)](#).

**Videos:** We follow the encoding process described in [Section 6.1.2](#). Following [Section 6.1.4](#), we computed global features for each video image and subsampled the frames introduced to the system.

We evaluated our interactive-predictive framework in these two different tasks, on two datasets per task:

**Image captioning:** we tackled two common datasets: Flickr8k ([Hodosh et al., 2013](#)) and Flickr30k ([Plummer et al., 2015](#)). The goal is to generate descriptions of pictures crawled from Flickr users. The first dataset comprises 30,000 training samples while the second one contains 145,000. Both datasets have test sets of 1,000 images.

**Video captioning:** we tested our systems on the datasets used in the previous sections: MSVD ([Section 6.1.4](#)) and the EDUB-SegDesc ([Section 6.2.5](#)).

As discussed in [Section 1.4.1](#), when dealing with interactive-predictive systems, we need to assess the quality of the initial predictions of the system and the amount of human effort required to correct these predictions. As in [Section 6.1.5](#), we measure the first case in terms of BLEU and METEOR. The human effort required by static systems will be approximated by TER ([Zaidan and Callison-Burch, 2010](#)) while that required by interactive-predictive systems will be evaluated using KSMR.

Again, we rely on simulated users, as described in [Section 1.4.1](#). The users will follow the character-based protocol, introducing the corrections at character level (see [Sections 4.3.1](#) and [4.3.3](#)). We set the ground-truth samples from the different datasets to be the desired outputs by our simulated users.

Our neural sequence-to-sequence systems were developed with NMT-Keras (see [Appendix A](#), [Peris and Casacuberta, 2018a](#)). We tested the RNN and Transformer architectures. The systems were similar to those described in [Section 3.5.1](#). In the case of image and video captioning, we reduced all model sizes to 256, since we are dealing with smaller datasets. We applied an early-stopping strategy, observing the BLEU score on the development set. At decoding time, we used a beam size of 6 in all cases.

The image captioning systems were trained using Adam ([Kingma and Ba, 2014](#)), with a learning rate of 0.0002. The image features were extracted using a NASNet architecture ([Zoph et al., 2018](#)), trained on the ImageNet dataset ([Deng et al., 2009](#)).

In the case of video captioning, we obtained better performance using Adadelta ([Zeiler, 2012](#)), for both datasets. Note that the RNN-based video captioning model was slightly different to the one from [Section 6.1.3](#). We did not experiment with the TMA model from [Section 6.2.4](#), but the application of the interactive-predictive framework the temporarily-linked model is trivial.

### 6.3.1 Results and discussion

We now show and discuss the results obtained by our systems. First, we assess the systems quantitatively, in terms of prediction quality and effort required during the correction stage. Next, in order to gain some insights into the behavior of the system, we analyze an image captioning example.

## Quantitative evaluation

We start by evaluating the systems in a traditional way, assessing their prediction quality. Table 6.7 shows the BLEU and METEOR results of the different systems for all tasks. These results are similar to those reported obtained in previous sections (Sections 6.1.5 and 6.2.6) and the literature (Xu et al., 2015; Yao et al., 2015; Peris et al., 2016; Bolaños et al., 2018a).

**Table 6.7:** Prediction quality for the different tasks, datasets and models. The RNN column denotes RNN-based system and the “Trans.” column indicates a Transformer model.

Task	Dataset	BLEU [↑]		METEOR [↑]		TER [↓]	
		RNN	Trans.	RNN	Trans.	RNN	Trans.
Image Captioning	Flickr8k	22.1	19.6	20.8	19.8	70.6	72.3
	Flickr30k	22.2	19.3	20.0	18.5	73.2	74.6
Video Captioning	MSVD	49.6	45.7	33.4	30.7	50.7	54.1
	EDUB-SegDesc	30.4	25.8	21.9	20.3	63.1	65.9

It is worth noting that RNN-based systems slightly outperformed Transformer. This latter model is more data-eager than RNN systems. Many of the recent advances yielded with this architecture leverage huge data collections (e.g. Radford et al. (2019)). We also contrasted this fact in our experimentation, as happened with small MT datasets (see Section 3.5.3). These captioning datasets are also relatively small. Hence, the Transformer model only was fully exploited in the MT case.

Next, we evaluate the performance of the interactive-predictive systems. These results are shown in Table 6.8. Due to the novelty of this scenario, we lack from references in the literature, regarding the other tasks. The results indicate that a user approximately needed to press 13 and 9 keys per sentence (on average) to correct the outputs of the image and video captioning systems, respectively.

As in the case of INMT, a crucial aspect of the usability of interactive-predictive systems is their response time. The average response time of our systems was below 0.15 seconds. According to Nielsen (1993), this provides the user of a feeling of almost instant reactivity.

**Table 6.8:** Effort required to correct the outputs of interactive-predictive systems, using RNN and Transformer (Trans.) models, in terms of KSMR.

Task	Dataset	KSMR [↓]	
		RNN	Trans.
Image Captioning	Flickr8k	36.6	36.9
	Flickr30k	36.0	40.0
Video Captioning	MSVD	36.4	40.5
	EDUB-SegDesc	40.0	38.0

### Qualitative analysis

We now show and analyze an image captioning example. Other examples for the video captioning systems are alike. The example is taken from our multimodal systems showcase<sup>5</sup> (see [Appendix A.3](#) and shown in [Fig. 6.13](#)).

We can see that the caption generated by the system (at iteration 0) has an error. The user wants to indicate that the people are sitting on a bench. Hence, the feedback introduced is the character “b”. The system is able to properly complete the word “bench”, with this single interaction. The same happens when the user wants to introduce the clause “under a”. With only typing the character “u”, the system generates this clause. Finally, it is interesting to observe the behavior of the last interaction. The user introduced the character “n” to the word “a”. Hence, the next word must start with a vowel. The system is able to properly account for this concordance and generates the word “umbrella”. We observe that the systems also handle correctly other concordances, such as singular/plural clauses.

## 6.4 Summary

In this chapter we addressed the problem of egocentric captioning of daily events. We tackled this challenge as an automatic video description task which in turn can be tackled as a sequence-to-sequence problem. Hence, we firstly investigated on regular video captioning scenario, showing that the inclusion of bidirectional RNNs to encode frames can bring improvements to a captioning system. Next, we addressed our problem at hand, considering a natural characteristic of this problem: since the egocentric sequences were captured consecutively along a day, there exists a relationship between a given situation and the previous one. We aimed to include such dependency in an automatic captioning system. In order to do this, we developed a natural extension of RNN networks, able to deal with multiple inputs, from different modalities.

<sup>5</sup><http://casmacat.prhlt.upv.es/interactive-seq2seq>





<b>Iter 0</b>	<b>System</b>	A group of people sit on a ramp.
<b>Iter 1</b>	<b>User</b> <b>System</b>	<i>A group of people sit on a <span style="border: 1px solid black; padding: 0 2px;">r</span>amp.</i> <i>A group of people sit on a bench.</i>
<b>Iter 2</b>	<b>User</b> <b>System</b>	<i>A group of people sit on a bench <span style="border: 1px solid black; padding: 0 2px;">u</span>.</i> <i>A group of people sit on a bench under a building.</i>
<b>Iter 3</b>	<b>User</b> <b>System</b>	<i>A group of people sit on a bench under a <span style="border: 1px solid black; padding: 0 2px;">n</span>building.</i> <i>A group of people sit on a bench under an umbrella.</i>
<b>Iter 4</b>	<b>User</b>	<i>A group of people sit on a bench under an umbrella.</i>

**Figure 6.13:** Interactive-predictive session example, to correct the caption generated for the image. At each iteration, the user introduces a character correction (boxed). The system modifies its hypothesis, taking into account this feedback: keeping the correct prefix (green) and generating a compatible suffix. Post-editing this sample in a static way, would have required the deletion of 4 characters and the addition of 23 characters.

We assessed our proposal on a dataset for egocentric captioning. Image sequences were obtained using a wearable camera and were manually segmented and annotated. Both source code of our system and dataset are publicly available. We carried out an automatic evaluation of the system, with clear results: the inclusion of previous information effectively enhances the performance of the system.

Moreover, if we had available the sequence of events of a full day, we could introduce not only previous events to the system, but also the following ones. Since the TMA model defined in this work supports an arbitrary number of inputs, the inclusion of the context coming from following events becomes natural. Therefore, the captioning results could be refined, not only by the previous, but also by incorporating

the following information. Furthermore, we could aim to look further than the immediately previous event, incorporating longer-term memory. The long-term learning challenge is taking off recently: [Kaiser et al. \(2017\)](#) proposed a memory-augmented network, able to learn very long-term relationships. Moreover, Transformer-like architectures ([Section 3.2](#)) usually manage better large contexts ([Agrawal et al., 2018](#)) than RNNs. A Transformer model can be used directly on our framework, as encoder or decoder. We leave the experimentation with such alternative architectures as future work.

Additionally, we applied the interactive-predictive framework to these multimodal, neural sequence-to-sequence systems. We tackled the image and video captioning tasks, using two state-of-the-art models and, in all cases, the interactive-predictive systems were able to decrease the human effort required to correct the outputs of the system. We obtained savings of approximately 50%. We also analyzed these systems through an online demo website.

## Chapter 7

# Conclusions

We conclude this dissertation summarizing the achievements accomplished. We also provide a list of the scientific publications derived from this thesis. Finally, we trace several lines of future work, that we consider interesting steps to take.

### 7.1 Scientific contributions

We explored three main research lines in this thesis, regarding neural sequence-to-sequence learning. We summarize the contributions achieved for each one of them.

#### 7.1.1 Interactive-predictive pattern recognition

We pioneered the application of the interactive-predictive framework to NMT and to other sequence-to-sequence tasks. This was achieved independently and in parallel to other works (Knowles and Koehn, 2016; Wuebker et al., 2016). We also augmented the classical left-to-right interactions, providing the user with a more flexible interactive-predictive protocol, namely, segment-based interaction. We also devised an effective strategy to introduce user feedback at character level, while maintaining the inference at the word (or subword) level. Finally, we applied this interactive-predictive protocol beyond MT, yielding interactive-predictive image and video captioning systems.

We conducted a wide set of experiments, showing that INMT systems behave much better than classical PB-SMT models: they have a better recovery from unexpected feedback and provide more fluent hypotheses than PB-SMT systems. Moreover, we empirically demonstrated the capabilities of character-level interactions, reducing the effort required by the user up to 50%.

In the multimodal framework, the interactive-predictive systems also performed well: the user feedback was well integrated by the system, which provided fluent and coherent hypotheses, usually accounting for lexical and grammatical concordances.

We deployed all these systems in an online demonstration, accessible through a public website<sup>1</sup>.

### 7.1.2 Adaptive NMT

The second research direction was devoted to the creation of adaptive NMT systems. To that end, we relied on the data generated during the exploitation of the system in a computer-assisted translation environment. We studied two main paradigms: online and active learning.

#### Online learning

In the online learning setup, the system is incrementally updated with corrected samples, acquired via translation post-editing or IMT. In the case of NMT, this can be done using regular training methods, based on gradient descent. We also proposed two alternative update strategies, that consider how wrong was the system prediction to perform the update.

We studied the behavior of the system in three different scenarios, accounting for the use-cases that an in-production system may face: having available a large pool of in-domain data, lacking in-domain data and having available both in-domain and out-of-domain data. We also evaluated the regular post-editing and the INMT processes. We explored different optimizers and number of updates per sample.

The results showed that the optimal parameters are task-dependent, but more structured and repetitive tasks benefit from more aggressive updates. The online adaptation of NMT brought significant improvements in terms of translation quality and reducing the human effort in all three scenarios and for all NMT architectures (RNN-based and Transformer). It is also worth mentioning that our alternative update rules did not perform better than classical methods.

---

<sup>1</sup><http://casmacat.prhlt.upv.es/interactive-seq2seq>

We also tested the application of adaptive NMT systems with professional translators, who usually make use of translation post-editing. We observed decreases in the time used during the post-editing process and an improvement in the translation quality of adaptive systems. Moreover, the users were very pleased with the adaptive system, preferring this to a static one.

### **Active learning**

The active learning scenario regards the selection, from a large pool of instances, of a subset of samples to correct. These selected samples will be used to adapt the system. This is useful for the periodical translation of large amounts of data, potentially available in the form of stream. We applied this paradigm to NMT, developing ad-hoc strategies to obtain better selections to supervise.

The evaluation of these framework was positive. Under an active learning setup, we were able to enhance the performance of the NMT system in terms of translation quality. In the case of using an interactive-predictive system, we reduced the effort required to reach a given translation quality by approximately 25%.

### **7.1.3 Multimodal captioning**

We also studied the application of the sequence-to-sequence framework to multimodal problems. More precisely, we focused on the task of video captioning. We tackled this problem following the neural encoder–decoder architecture, using BRNNs for modeling temporal relationships among the video frames.

Derived from this, we tackled the egocentric captioning: generating descriptions of sequences of egocentric images. A major issue in this task is the temporal relationship existing among the different daily events: an event is conditioned on the previous ones. Aiming to exploit this, we proposed a model that accounted for the previous event. This previous event was characterized either by its sequence of frames, its generated caption or both. We observed that considering only the sequence of frames provided the most reliable results.

## 7.2 Dissemination of the work

Part of the work presented in this thesis was accepted in several international conferences and journals. We provide a list of these publications.

### 7.2.1 Interactive-predictive and adaptive NMT

The development of interactive-predictive adaptive systems described in [Chapters 4](#) and [5](#) was presented in two international journal articles:

- Á. Peris, M. Domingo, and F. Casacuberta. Interactive neural machine translation. *Computer Speech & Language*, 45:201–220, 2017. JCR Q2.
- Á. Peris and F. Casacuberta. Online learning for effort reduction in interactive neural machine translation. *Computer Speech & Language*, 58:98–126, 2019. JCR Q2.

The application of the active learning framework to NMT described in [Chapter 5](#) was presented in an international conference:

- Á. Peris and F. Casacuberta. Active learning for interactive neural machine translation of data streams. In *Proceedings of the Conference on Computational Natural Language Learning*, pages 151–160, 2018. CORE A.

The human evaluation of adaptive systems ([Section 5.4](#)) was also presented in

- Miguel Domingo, Mercedes García-Martínez, Álvaro Peris, Alexandre Helle, Amando Estela, Laurent Bié, Francisco Casacuberta and Manuel Herranz. Incremental Adaptation of NMT for Professional Post-editors: A User Study. In *Proceedings of Machine Translation Summit XVII Volume 2: Translator, Project and User Tracks*, pages 219–227, 2019. CORE B.

Moreover, we also developed the segment-based interactive-predictive protocol for PB-SMT systems, although this was not reported in this dissertation. This derived in two articles in an international journal and an international conference:

- M. Domingo, Á. Peris and F. Casacuberta. Interactive-predictive translation based on multiple word-segments. In *Proceedings of the 19th Annual Conference of the European Association for Machine Translation*, pages 282–291, 2016. CORE B. **Best paper award.**
- M. Domingo, Á. Peris, and F. Casacuberta. Segment-based interactive-predictive machine translation. *Machine Translation*, 31(4):163–185, 2017.

### 7.2.2 Multimodal captioning

Our solution to the captioning of videos, described in [Chapter 6](#) was presented in an international conference:

- Á. Peris, M. Bolaños, P. Radeva, and F. Casacuberta. Video description using bidirectional recurrent neural networks. In *Proceedings of the International Conference on Artificial Neural Networks*, pages 3–11, 2016. CORE B.

The captioning of egocentric, temporally-dependent videos ([Section 6.2](#)) was presented in an international journal:

- M. Bolaños, Á. Peris, F. Casacuberta, S. Soler, and P. Radeva. Egocentric video description based on temporally-linked sequences. *Journal of Visual Communication and Image Representation*, 50:205216, 2018. JCR Q2.

The application of the interactive-predictive framework to multimodal systems was presented in an international conference:

- Á. Peris and F. Casacuberta. Interactive-predictive neural multimodal systems. In *Iberian Conference on Pattern Recognition and Image Analysis, Lecture Notes in Computer Science*, in press. 2019.

In addition, although not reported in this thesis, we also tackled the visual question answering problem, yielding a participation in an international contest and a publication in an international conference:

- M. Bolaños, Á. Peris, F. Casacuberta, and P. Radeva, P. VIBIKNet: Visual bidirectional kernelized network for the VQA Challenge. In *VQA Challenge Workshop in IEEE Conference on Computer Vision and Pattern Recognition*. 2016. WORKSHOP IN CORE A\* CONFERENCE.
- M. Bolaños, Á. Peris, F. Casacuberta, and P. Radeva. VIBIKNet: Visual bidirectional kernelized network for visual question answering. In *Iberian Conference on Pattern Recognition and Image Analysis*, volume 10255 of *Lecture Notes in Computer Science*, pages 372–380. 2017.

### 7.2.3 Additional work

In parallel to the research described in this dissertation, we also studied the domain adaptation field for MT. This research derived in the following publications:

- Á. Peris, M. Chinea-Rios and F. Casacuberta. Neural networks classifier for data selection in statistical machine translation. *The Prague Bulletin of Mathemati-*

cal Linguistics, 108(1):283–294. 2017. Presented in *the 20th Annual Conference of the European Association for Machine Translation*. CORE B.

- M. China-Rios, Á. Peris, F. Casacuberta. Adapting neural machine translation with parallel synthetic data. In: *Proceedings of the Second Conference on Machine Translation*, pages 138–147, 2017.
- M. China-Rios, Á. Peris, F. Casacuberta. Are automatic metrics robust and reliable in specific machine translation tasks? In: *Proceedings of the 21st Annual Conference of the European Association for Machine Translation*, pages 89–98. 2018. CORE B.

Moreover, we described the NMT technology for a non-expert audience (in Spanish):

- F. Casacuberta and Á. Peris. Traducció automàtica neuronal. *Revista Tradumàtica: tecnologies de la traducció*, 15(1):66–74. 2017.

#### 7.2.4 Open-source software

We released all the source code developed during this thesis as open-source repositories, with MIT license:

- <https://github.com/lvapeab/nmt-keras> implements all NMT systems (Chapter 3) used along this thesis. The online and active learning paradigms (Chapter 5) were also developed using this toolkit.
- [https://github.com/lvapeab/multimodal\\_keras\\_wrapper](https://github.com/lvapeab/multimodal_keras_wrapper) contains the search modifications that the interactive-predictive framework requires (Chapter 4). This repository also encapsulates the methods used to work with multimodal data.
- <https://github.com/lvapeab/ABiViRNet> implements the ABiViRNet model (Section 6.1.3).
- <https://github.com/MarcBS/TMA> implements the TMA model (Section 6.2.4).

In addition to these repositories and their corresponding online documentation, the description of the main pieces of software (see Appendix A) were yielded an international journal publication and two presentations in an international conference:

- Á. Peris and F. Casacuberta. NMT-Keras: a very flexible toolkit with a focus on interactive NMT and online learning. *The Prague Bulletin of Mathematical Linguistics*, 111:113–124, 2018.



- Á. Peris and F. Casacuberta. A neural, interactive-predictive system for multimodal sequence to sequence tasks. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 81–86, 2019. CORE A\*.
- Miguel Domingo, Mercedes García-Martínez, Amando Estela Pastor, Laurent Bié, Alexander Helle, Álvaro Peris, Francisco Casacuberta and Manuel Heranz Pérez. Demonstration of a neural machine translation system with online learning for translators. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 70–74, 2019. CORE A\*.

Furthermore, it is worth mentioning that the NMT-Keras toolkit has served since 2017 in the practical lessons of the *Machine Translation* course, belonging to the *Master’s Degree in Artificial Intelligence, Pattern Recognition and Digital Imaging*, from *Universitat Politècnica de València*.

## 7.3 Future work

The work presented in this thesis opens several future research directions, that we aim to explore in a near future.

### 7.3.1 Interactive-predictive pattern recognition

The next step to take is to conduct a human experimentation in order to obtain accurate measures of the actual effort spent by the users. Recent works showed positive user perceptions on interactive-predictive neural systems, although with mixed results in terms of productivity (Daems and Macken, 2019; Knowles et al., 2019). Performing these evaluations with our systems with real users is at the top of our agenda.

The inclusion of active interaction (González-Rubio et al., 2010a) into NMT also seems an interesting future direction. This protocol consists of only correcting those parts of the hypothesis that the system considers more prone to be erroneous. Hence, the user must only review some parts from the hypothesis. The work by Lam et al. (2018) is close to this scenario and it should be further extended.

Moreover, the encouraging results obtained with multimodal interactive-predictive systems open several avenues for future research. These systems can be used to tackle a number of structured prediction tasks, e.g. data to text (Puduppully et al., 2018). More precisely, we are interested in applying this framework to the automatic report of medical images or to the generation of automatic life-loggers, as those described in [Section 6.2](#). In addition to an end-user application, these tools can also be exploited

by human annotators, to create datasets in a more efficient way, that allow us to push forward the research on multiple areas.

In this thesis, we experimented only with multimodal inputs. In future, we want to explore the inclusion of multimodal feedback signals. This was already done for statistical models, integrating speech (Alabau et al., 2011) and electronic pens (Alabau et al., 2014) into the interactive-predictive pipeline. We think that neural models are especially well-suited to fully exploit these multimodal feedback signals.

Finally, we used a different system for each task. In future, we would like to explore the construction of a single interactive-predictive, multitask, multimodal system. The recent advances achieved in multitask learning (Radford et al., 2019) heavily support this research direction.

### 7.3.2 Adaptive NMT

One of the main drawbacks relating to NMT adaptation is the independence, up to some point, of the objective function (cross-entropy) with respect to the evaluation metric (e.g. BLEU or TER). The alternative update methods described in this thesis (Section 5.2.2) manipulated the loss function, but they were less effective than performing regular updates. To boost the effectiveness of online adaptation for NMT, the reinforcement learning field brings exciting avenues for future work.

Reinforcement learning has been used to directly optimize BLEU, as a complementary method to the traditional maximum likelihood training (Wu et al., 2016), during decoding (Gu et al., 2017) or to leverage weak user feedback (Sokolov et al., 2016b; Kreutzer et al., 2017). The exploration of these approaches seems a promising research line.

Moreover, the relationship between the interactive-predictive framework and the reinforcement learning field seems clear: the feedback provided by the user can be seen as system rewards. The conjunction of these two worlds is at its early stages and few works have gone into in any depth (Sokolov et al., 2016a; Lam et al., 2018). Hence, taking immediate advantage from user feedback via reinforcement learning also seems a plausible future objective.

In order to develop useful adaptive systems, we also need to address the concerns noted by the post-editors (Section 5.4.6), namely the degradation of domain-specific terms and the incorrect generation of words due to subwords. To that end, we should study and analyze the hypotheses produced by the adaptive system and the post-edits performed by the users, similarly to Koponen et al. (2019) and develop effective methods to prevent these degradations.

In addition, we want to integrate our adaptive systems together with other translation tools, such as translation memories or terminological dictionaries, with the aim of improving the productivity of the post-editing process. With this feature-rich system, we would like to conduct additional evaluations, involving more diverse languages and domains and larger numbers of professional post-editors.

Regarding the active learning framework that we applied to NMT (Section 5.5), we also want explore new lines of work in future. First, we intend to apply our method to other datasets, involving linguistically diverse language pairs and low-resource scenarios, in order to observe whether the results obtained in this work hold. We also aim to devise sampling strategies that take into account the cognitive effort or time required to interactively translate a sentence. Moreover, the proposed sampling strategies (Section 5.5.2) can be used as a data selection technique, to perform domain adaptation. It would be interesting to assess their performance on this task.

### 7.3.3 Multimodal captioning

The results obtained in the video captioning task suggest that deep structures help to transfer the knowledge from an input sequence of frames to the output caption. The application of three-dimensional ConvNets (Tran et al., 2015) allows the recognition of actions and may solve some of the ambiguities existing in the tested methods, which only cope with object and scenes recognition. Preliminary results, although not reported in this thesis, indicate the capabilities of these architectures. Moreover, the attention mechanism implemented in our model exploited the temporal relationships of the video frames. An additional future step should study the inclusion of spatial-temporal attention models to better model videos.

Regarding to egocentric vision and our temporarily-linked model, we should study methods of increasing even more the context. If we have available the sequence of events of a full day, we could introduce not only previous events to the system, but also the following ones. Furthermore, we should aim to look further than the immediately previous event, incorporating longer-term memory. Transformer-like architectures are very powerful modelers of long-range dependencies (Agrawal et al., 2018; Devlin et al., 2018; Radford et al., 2019). Moreover, application of life-long learning techniques (e.g., Kaiser et al., 2017) can also bring improvements to the modeling of these long-term relationships.

Finally, and as stated above, we intend to apply these multimodal techniques, combined with the interactive-predictive framework, to the development of medical applications, such as x-ray imaging analysis (Bustos et al., 2019), or the efficient exploitation of medical databases such as the DICOM standard (Mildenberger et al., 2002), following the structured data-to-text framework.



## Appendix A

# The NMT-Keras toolkit

To easily develop new deep learning models is a key feature in this fast-moving field. Within the scope of this thesis, we developed NMT-Keras<sup>1</sup>, a flexible toolkit for neural machine translation, based on the Keras library for deep learning (Chollet et al., 2015). Keras is an API written in Python which provides high-level interfaces to numerical computation libraries, such as Theano (Theano Development Team, 2016) or TensorFlow (Abadi et al., 2016). Keras is well-structured and documented, with designing principles that make it modular and extensible, being easy to construct complex models.

Following the spirit of the Keras library, we developed NMT-Keras, released under MIT license, that aims to provide a highly-modular and extensible framework to NMT. NMT-Keras supports advanced features, including support of INMT (Chapter 4), continuous adaptation (Section 5.2) and active learning (Section 5.5.1) strategies. An additional goal, is to ease the usage of the library, but allowing the user to configure most of the options involving the NMT process.

Several toolkits currently offer fully-fledged NMT systems. Among them, we can highlight OpenNMT (Klein et al., 2017), Tensor2Tensor (Vaswani et al., 2018) or Nematus (Sennrich et al., 2017). NMT-Keras differentiates from them by offering interactive-predictive and long-term learning functionalities, with the final goal of fostering a more productive usage of the NMT system.

---

<sup>1</sup><https://github.com/lvapeab/nmt-keras>

This appendix describes the main design and features of NMT-Keras. First, we review the deep learning framework which is the basis of the toolkit. Next, we summarize the principal features and functionality of the library.

## A.1 Design

NMT-Keras relies on two main libraries: a modified version of Keras<sup>2</sup>, which provides the framework used to train the neural models; and a wrapper around it, named Multimodal Keras Wrapper<sup>3</sup>, designed to ease the usage of Keras and the management of data. These tools represent a general deep learning framework, able to tackle different problems, involving several data modalities. The problem of NMT is an instantiation of the sequence-to-sequence task, applied to textual inputs and outputs. NMT-Keras is designed to tackle this particular task. The reason to rely on a fork of Keras is because this allows us to independently design functions for our problems at hand, which may be confusing for the general audience of Keras. However, in a near future we hope to integrate our contributions into the main Keras repository.

### A.1.1 Keras

As mentioned before, Keras is a high-level deep learning API, which provides a neat interface to numerical computation libraries. Keras allows to easily implement complex deep learning models by defining the layers as building blocks. This simplicity, together with the quality of its code, has made Keras to be one of the most popular deep learning frameworks. It is also well-documented, and supported by a large community, which fosters its usage.

In Keras, a model is defined as a directed graph of layers or operations, containing one or more inputs and one or more outputs. Once a model has been defined, it is compiled for training, aiming to minimize a loss function. The optimization process is carried out, via gradient descent, by means of an optimizer.

Once the model is compiled, we feed it with data, training it as desired. Once a model is trained, it is ready to be applied on new input data.

---

<sup>2</sup><https://github.com/MarcBS/keras>

<sup>3</sup>[https://github.com/lvapeab/multimodal\\_keras\\_wrapper](https://github.com/lvapeab/multimodal_keras_wrapper)

### A.1.2 Multimodal Keras Wrapper

The Multimodal Keras Wrapper allows to handle the training and application of complex Keras models, data management (including multimodal data) and application of additional callbacks during training. The wrapper defines two main objects and includes a number of extra features:

**Dataset:** A Dataset object is a database adapted for Keras, which acts as data provider. It manages the data splits (training, validation, test). It accepts several data types, including text, images, videos and categorical labels. In the case of text, the Dataset object builds the vocabularies, loads and encodes text into a numerical representation and also decodes the outputs of the network into natural language. In the case of images or videos, it also normalizes and equalizes the images; and can apply data augmentation.

**Model wrapper:** This is the core of the wrapper around Keras. It connects the Keras library with the Dataset object and manages the functions used to train and apply the Keras models. When dealing with sequence-to-sequence models, it implements a beam search procedure. It also contains a training visualization module and ready-to-use convolutional neural networks architectures.

**Extra:** Additional functionalities include extra callbacks, I/O management and evaluation of the system outputs. To compute the translation quality metrics of the models, we use the coco-caption evaluation tool (Chen et al., 2015), which provides common evaluation metrics: BLEU, METEOR, CIDEr, and ROUGE-L. Moreover, we modified it<sup>4</sup> to also include TER.

### A.1.3 NMT-Keras

The NMT-Keras library makes use of the aforementioned libraries, to build a complete NMT toolkit. The library is compatible with Python 2 and 3. The training of NMT models is done with the `main.py` file. The hyperparameters are set via a configuration file (`config.py`), and can also be set from the command line interface. To train an NMT system with NMT-Keras is straightforward:

1. Set the desired configuration in `config.py`.
2. Launch `main.py`.

The training process will then prepare the data, constructing the Dataset object and the Keras model. The default models implemented in NMT-Keras are an attentional RNN encoder-decoder (Bahdanau et al., 2015; Sennrich et al., 2017), and the

---

<sup>4</sup><https://github.com/lvapeab/coco-caption>

Transformer model (Vaswani et al., 2017). Once the model is compiled, the training process starts, following the specified configuration. To translate new text with a trained model, we use beam search.

## A.2 Features

As we keep our Keras fork constantly up-to-date with the original library, NMT-Keras has access to the full Keras functionality, including (but not limited to):

**Layers:** All the architectures described in Section 2.3: Fully-connected layers, ConvNets, RNNs (including LSTM, GRU and their bidirectional and conditional variants) with attention mechanisms, multi-head attention layers, or embedding layers. Moreover, all the techniques described in Section 2.4 to improve the generalization capabilities of the model are available: early-stopping, weight decay, noise injection, dropout, batch and layer normalization.

**Initializers:** The weights of a model can be initialized to a constant value, to values drawn from statistical distributions or according to strategies described in Section 2.2.4.

**Optimizers:** A number of SGD update rules variants are implemented (Section 2.2.3): plain SGD, RMSProp, Adagrad, Adadelta, Adam, Adamax or hypergradient descent-based methods (Baydin et al., 2017). The learning rate can be scheduled according to several strategies (linear, exponential, “noam” (Vaswani et al., 2017)).

**Regularizers and constraints:** Keras allows to set penalties and constraints to the parameters and to the layer outputs of the model.

Our version of Keras implements additional layers, useful for sequence-to-sequence problems:

**Improved RNNs:** All RNN architectures can be used in an autoregressive way, i.e. taking into account the previously generated token. They also integrate attention mechanisms, supporting the *add* (Bahdanau et al., 2015) and *dot* (Luong et al., 2015a) models.

**Conditional RNNs:** Conditional LSTM/GRU layers (Section 2.3.2, consisting of cascaded applications of LSTM/GRU cells, with attention models in between.

**Multi-input RNNs:** LSTM/GRU networks with two and three different inputs and independent attention mechanisms (Bolaños et al., 2018a).



**Transformer layers:** Multi-head attention layers, positional encodings and position-wise feed-forward networks (Vaswani et al., 2017).

**Convolutional layers:** Class activation maps (Zhou et al., 2016).

Finally, NMT-Keras supports a number of additional options. Here we list the most important ones, but we refer the reader to the library documentation<sup>5</sup> for an exhaustive listing of all available options:

**Deep models:** Deep residual RNN layers, deep output layers (Pascanu et al., 2014) and deep fully-connected layers to initialize the state of the RNN decoder.

**Embeddings:** Incorporation of pre-trained embeddings in the NMT model and embedding scaling options.

**Regularization strategies:** Label smoothing (Szegedy et al., 2015), early-stop, weight decay, doubly stochastic attention regularizer (Xu et al., 2015) and a fine-grained application of dropout.

**Search options:** Normalization of the beam search process by length and coverage penalty. The search can be also constrained according to the length of the input/output sequences.

**Unknown word replacement:** Replace unknown words according to the attention model (Jean et al., 2015a). The replacement may rely on a statistical dictionary.

**Tokenizing options:** Including full BPE support (Sennrich et al., 2016).

**Integration with other tools:** Support for Spearmin (Gelbart et al., 2014), for Bayesian optimization of the hyperparameters and Tensorboard, the visualization tool of TensorFlow.

Apart from these model options, NMT-Keras also contains scripts for ensemble decoding and generation of  $N$ -best lists; sentence scoring, model averaging and construction of statistical dictionaries (for unknown words replacement). It also contains a client-server architecture, which allows us to access to NMT-Keras via web. Finally, in order to introduce newcomers to NMT-Keras, a set of tutorials are available, explaining step-by-step how to construct an NMT system.

---

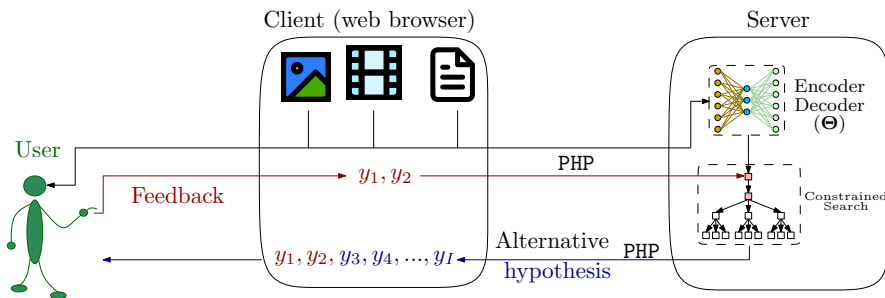
<sup>5</sup><https://nmt-keras.readthedocs.io>

### A.3 Interactive-predictive and adaptive pattern recognition

All systems and protocols described in [Chapter 4](#) and [Section 6.3](#) have been implemented and integrated into NMT-Keras. We built a demo website<sup>6</sup> of these interactive, adaptive systems using the client-server features of the toolkit. The system is deployed as a Python HTTP server that handles the requests from the client. The client is an HTML website, which manages the interactions with the user using JavaScript. Client and server are communicated via the HTTP protocol, using the PHP curl tool. All code is open-source and publicly available<sup>7,8</sup>.

#### A.3.1 Usage of the interactive system

Our interactive-predictive system works as follows: initially, an input object is presented to the user in the client website. The user requests an automatic prediction of it. Next, the client communicates the server via PHP. The server queries the neural system, which produces an initial hypothesis applying [Eq. \(1.6\)](#). The hypothesis is then sent back to the client website.



**Figure A.1:** System architecture. The client, a website, presents the user several input objects (images, videos or texts) and a prediction. The user then introduces a feedback signal, to correct this prediction. After being introduced, the feedback signal is sent to the server—together with the input object—for generating an alternative hypothesis, which takes into account the user corrections.

Next, the interactive-predictive process starts: the user searches in this hypothesis the first error, and introduces a correction with the keyboard (writing one or more characters). When the user stops typing the correction, the system reacts, sending to the server a request containing the input object and the user feedback (the sequence of characters that conform the correct prefix). Then, the neural model implements

<sup>6</sup><http://casmacat.prhlt.upv.es/interactive-seq2seq>.

<sup>7</sup>Server source code: <https://github.com/lvapeab/interactive-keras-captioning>.

<sup>8</sup>Client source code: [https://github.com/lvapeab/inmt\\_demo\\_web/tree/general\\_ipr](https://github.com/lvapeab/inmt_demo_web/tree/general_ipr).

Eq. (4.1) and produces an alternative hypothesis, such that it completes the correct prefix. This is implemented as a constrained beam search, as described in Chapter 4. This iteration of the process is illustrated in Fig. A.1.

This protocol is repeated until the user finds satisfactory the hypothesis given by the system. Then, it is validated. As soon as the sentence is validated, the system can be incrementally updated with this sample, following the online learning setup Chapter 5. Hence, in future interactions, the system will be progressively updated, tailoring to a given domain or to the user preferences.

### A.3.2 System showcase

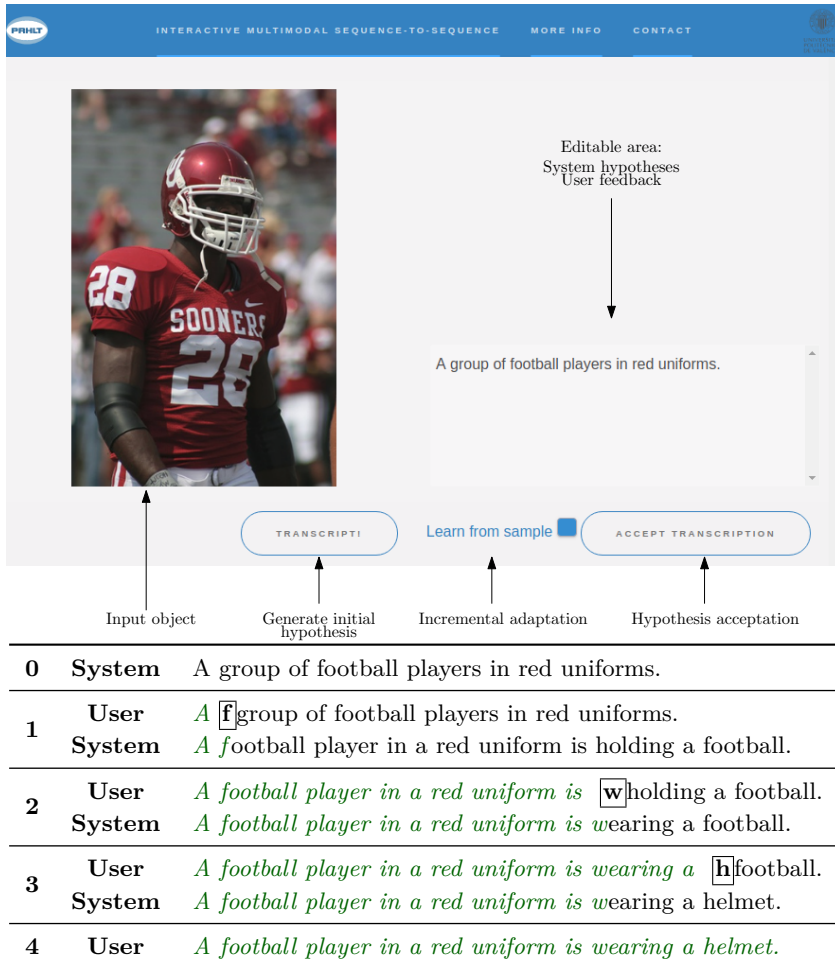
To show the interactive-predictive protocol described in the previous sections, we developed a website which hosts a demonstration of the system. Our demonstration system handles three different problems, regarding three different data modalities: text-to-text (NMT), image-to-text (image captioning) and video-to-text (video captioning). To tackle these tasks, we use the RNN-based model described in Section 3.1. Our framework has also support for Transformer-like architectures Section 3.2.

The NMT task regards the UFAL task Section 1.4.3. The image and video captioning systems are the Flickr8k and MSVD, as described in Section 6.3. Finally, the systems can be retrained after the validation of each sample. In our demonstration, the systems are updated via gradient descent, but using a learning rate of 0, which prevents a degradation of the model due to accidental misuse.

#### Example: image captioning

We show and analyze an image captioning example. The NMT and video captioning tasks are alike. Fig. A.2 shows the demo website, for the image captioning task together with the interactive-predictive captioning session, to obtain a correct sample.. In the left part of the screen, the input object is shown, in this case, an image. As the user clicks in the “Transcript!” button, the system generates a caption of the image, displaying it in an editable area on the right part of the screen. The user can then introduce the desired corrections to this hypothesis. As a correction is introduced, the system reacts, providing an alternative caption, but always considering the feedback given by the user. As can be seen in Fig. A.2, the caption generated by the system has some errors. With three interactions, the system was able to obtain a correct caption for the image.

It is particularly interesting to observe that the system correctly accounts for the singular/plural concordance of the clause *in red uniform(s)*, depending on the subject (*A football player/A group of football players*).



**Figure A.2:** Frontend of the client website. As the button “Transcript!” is clicked, an initial hypothesis for the input object—in this case, an image—appears in the right area. The user then introduces corrections of this text. The system reacts to each translation, producing alternative hypotheses, always compliant with the user feedback. At each iteration, the user introduces a character correction (boxed). The system modifies its hypothesis, taking into account this feedback: keeping the correct prefix (green) and generating a compatible suffix.

## A.4 Additional applications

The modular design of Keras and Multimodal Keras Wrapper allows us to use them to tackle other problems. Following the spirit of the toolkit, it has been modified and applied in a variety of tasks, including sentence classification (Peris et al., 2017b), food recognition and localization (Bolaños and Radeva, 2016; Bolaños et al., 2017), visual question answering (Bolaños et al., 2017), MT quality estimation (Ive et al., 2018), language transliteration (Le et al., 2019) or semantic failover for network managing (Hsueh et al., 2018).

## A.5 Summary

NMT-Keras is a toolkit built on top of Keras that aims to ease the deployment of complex NMT systems by having a modular and extensible design. NMT-Keras has a strong focus on building adaptive and interactive NMT systems; which leverage the effort reduction of a user willing to obtain high-quality translations. Finally, its flexibility allows the NMT-Keras framework to be applied directly to other problems, such as multimedia captioning or sentence classification.

We would like to improve the frontend of our website. Inspecting the attributes of black-box neural models is a relevant research topic, and it is under active development (e.g. Zeiler and Fergus, 2014; Ancona et al., 2017). Visualizing these relevant attributes would help to understand the model predictions and behavior. Moreover, a more sophisticated frontend would allow us to implement interesting features, such as mapping the attention weights through the input sequence or the implementation of more complex interaction protocols, such as touch-based interaction (Marie and Max, 2015) or segment-based interaction (Peris et al., 2017c).

We also intend to continue the active development of the tool, including new functionalities and improving the quality of the source code. Moreover, we hope to integrate our tool into the Keras ecosystem in a near future.



## Appendix B

# Human post-editors survey

We show the results of the survey conducted in [Section 5.4.6](#). Recall that this research was done in collaboration with Miguel Domingo and the team from Pangeanic.

### B.1 Initial questionnaire

Prior to starting the post-editing task, we collected some demographics of post-editors:

	User 1	User 2	User 3
Gender	Female	Male	Female
Age	27	24	30
Years of training	6	3	6
Years of experience	5	1.5	5
1 <sup>st</sup> language	Spanish	Spanish/Catalan	Spanish
2 <sup>nd</sup> language		English	
Frequency of use of MT		Every 2-3 weeks	
Time using MT	Several months	1.5 years	2 years
Satisfaction w.r.t. MT		Somewhat satisfied	
Will to apply MT in the future		Not sure	Yes
Feasibility of MT in industry	Somewhat likely	Highly likely	

Moreover, they freely answered to the question **What experience, if any, do you have of post-editing machine translation?:**

**User 1:** *It has been quite positive when post-editing texts of certain fields. But it has been limited.*

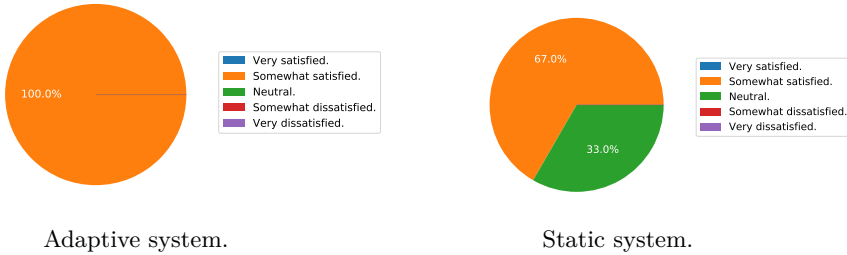
**User 2:** *I've post-edited texts and reviewed post-edition of documents with different scope and extension for about a year. Depending on the request and type of text, it can save up time and enhance productivity. In my opinion, clear criteria to the extent and amount of post-edition work should be stated for each task (or generally).*

**User 3:** *It is slow and not accurate with some kinds of texts.*

## B.2 Final questionnaire

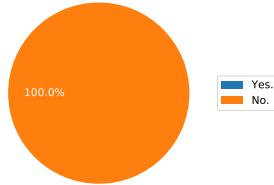
After performing each post-editing task, users completed the following questionnaire. We show the question in boldface font and next, we plot the answers given by the users, grouping them according to the uses of an adaptive or a static system.

**How satisfied are you with the translation you have produced?**

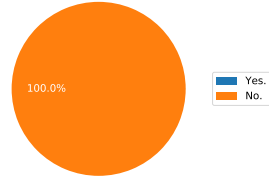




Would you have preferred to work on your translation from scratch instead of post-editing machine translation?

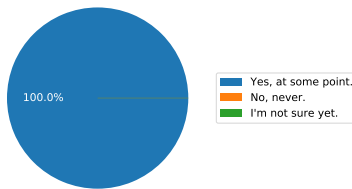


Adaptive system.

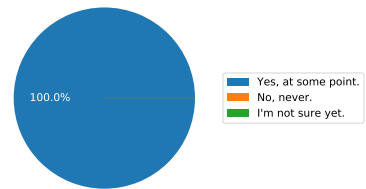


Static system.

Do you think that you will want to apply machine translation in your future translation tasks?

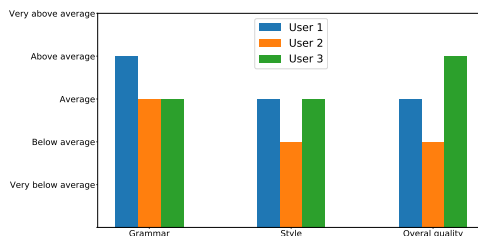


Adaptive system.

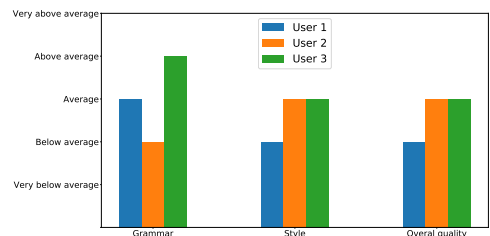


Static system.

Based on the post-editing task you have performed, how much do you rate machine translation outputs on the following attributes?

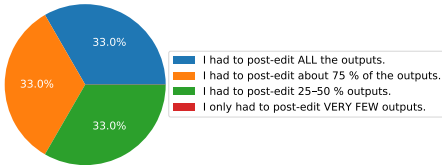


Adaptive system.

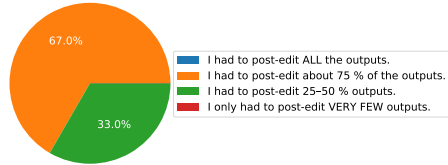


Static system.

Based on the post-editing task you have performed, which of these statements will you go for?

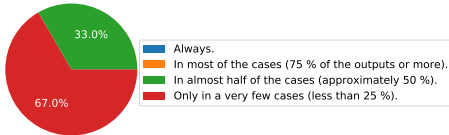


Adaptive system.

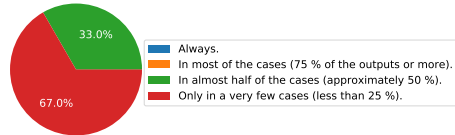


Static system.

Based on the post-editing task you have performed, how often would you have preferred to translate from scratch rather than post-editing machine translation?



Adaptive system.



Static system.

Once the users finished their translation jobs, we asked them to guess which system was adaptive and which was static. All they guessed correctly.

Finally, we asked them to give their opinions regarding the experiment:

### Opinions on adaptive systems

**User 1:** *Machine translation was very good in some of the outputs but there were few sentences which were incomplete or useless. Also -if this has to resemble a translation project in real life- we should have some reference, other TMs, instructions about terminology, etc.*

**User 2:** *Despite recurrent mistakes also found in Test 1 (repetition, mistranslation, nonsense words...), the online learning system was noticeable, for product names, grammatical structures and lexical-wise as well.*

**User 3:** *The corrections applied in one segment were in general reflected in the following segments, which was very helpful, as only style has to be slightly corrected in those segments. However, it seemed that the learning was not performed in the instance of the corrected term appearing immediately after, but it was some instances below when the term started to be translated appropriately after learning. As I progressed in the translation, some times it seemed that the term was "forgotten", and again was wrongly translated. It is important to mention that in some cases, totally made-up words appeared as a result of the MT (for ex. "absolvido", "los padtillas"), which I had not introduced myself.*

### **Opinions on static systems**

**User 1:** *In this case the translations I confirmed were not influencing the following MT outputs, so the task of post-editing has been less fluent.*

**User 2:** *In many instances, the machine translation output was helpful. There were other cases where the segment was not fully translated (only a few words present, I saved some screenshots) or there were unnecessary repetition of terms, which needed a greater deal of edition. I didn't get to appreciate the online training system, or at least to a full extent. I had to correct the name of the product in every segment, for example.*

**User 3:** *In some cases the output was just a couple of words instead of the full sentence or paragraph. Moreover, the fuzzy matches were not applied and it seemed that learning was not performed as I kept having the same wrong terms after having previously changed them and confirmed segments.*



# List of Figures

1.1	MT technology in WMT . . . . .	9
2.1	MLP with a single layer . . . . .	21
2.2	Bidirectional recurrent neural network . . . . .	30
2.3	LSTM cell . . . . .	31
2.4	Conditional RNN unit . . . . .	35
2.5	Multi-head attention mechanism . . . . .	36
3.1	RNN encoder–decoder . . . . .	46
3.2	The Transformer model . . . . .	52
4.1	Single iteration of prefix-based IMT . . . . .	68
4.2	Segment-based IMT iteration . . . . .	70
4.3	Constrained search for segment-based interaction . . . . .	73
4.4	Character-level interaction . . . . .	75
4.5	Constrained vocabulary for character-level interaction . . . . .	76

4.6	INMT at subword level . . . . .	77
4.7	Word-level versus character-level interaction . . . . .	84
4.8	PB-SMT prefix-based IMT session . . . . .	86
4.9	INMT prefix-based session . . . . .	87
4.10	INMT segment-based session. . . . .	88
4.11	INMT prefix-based session at character level . . . . .	89
5.1	Static and adaptive IMT . . . . .	92
5.2	Translation quality depending on the optimizer . . . . .	105
5.3	Translation quality depending on the number of updates per sample . . . . .	106
5.4	Translation quality for the alternative optimizers . . . . .	108
5.5	Samples from the XRCE, UFAL and TED tasks. . . . .	115
5.6	Cumulative TER and KSMR of static and adaptive NMT systems . . . . .	121
5.7	INMT session from scenario #1 . . . . .	122
5.8	INMT session from scenario #1 with online learning . . . . .	123
5.9	SDL Trados Studio user interface . . . . .	125
5.10	hTER and hBLEU per sentence . . . . .	128
5.11	Evolution of BLEU with respect to the amount of training data . . . . .	137
5.12	Translation quality as a function of human effort . . . . .	138
6.1	Inception module . . . . .	146
6.2	Architecture of the ABiViRNet model for video captioning . . . . .	147
6.3	MSVD sample . . . . .	149
6.4	Outline of the TMA model . . . . .	152
6.5	Architecture of the TMA model . . . . .	157
6.6	Histogram of duration of events from EDUB-SegDesc . . . . .	160

6.7	Keyword co-occurrence frequency . . . . .	160
6.8	Text-related statistics from the EDUB-SegDesc . . . . .	161
6.9	Subset from a day from the dataset EDUB-SegDesc . . . . .	162
6.10	Egocentric captioning results . . . . .	168
6.11	Egocentric captioning results samples from initial events . . . . .	169
6.12	Success and failure cases of the TMA model . . . . .	170
6.13	Multimodal interactive-predictive session . . . . .	175
A.1	Interactive-predictive system architecture . . . . .	192
A.2	Frontend of the demonstration website . . . . .	194





# List of Tables

1.1	Main figures of the XRCE, TED, UFAL and Europarl corpora . . . . .	15
2.1	Activation functions . . . . .	22
2.2	Parameter update rules . . . . .	26
3.1	Translation quality for the XRCE task . . . . .	57
3.2	Translation quality for the TED task . . . . .	58
3.3	Translation quality for the UFAL task . . . . .	59
3.4	Translation quality for the Europarl task . . . . .	60
4.1	Prefix-based interaction results . . . . .	79
4.2	Segment-based interaction results . . . . .	81
4.3	KSMR of interactive-predictive systems at character level . . . . .	82
4.4	Average system response time per interaction . . . . .	85
5.1	RRR and UNF for all corpora . . . . .	103
5.2	Best online optimizer for each task. . . . .	107

5.3	Static versus adaptive RNN-based NMT systems . . . . .	110
5.4	Static versus adaptive Transformer-based NMT systems . . . . .	110
5.5	Static versus adaptive INMT systems . . . . .	111
5.6	Vocabulary coverage, RRR and UNF of in-domain sets with respect to the out-of-domain . . . . .	112
5.7	Out-of-domain static versus adaptive RNN-based NMT systems . . . . .	113
5.8	Static versus adaptive Transformer-based NMT systems . . . . .	114
5.9	KSMR of adaptive INMT systems compared to static INMT . . . . .	116
5.10	Static versus adaptive fine-tuned RNN-based NMT systems . . . . .	117
5.11	Static versus adaptive Transformer-based NMT systems . . . . .	118
5.12	KSMR of fine-tuned adaptive INMT compared to static INMT . . . . .	119
5.13	Online learning times . . . . .	120
5.14	Results of the simulated experiments. TER and BLEU were computed considering the reference sentences. * indicates statistically significant differences between the static and the adaptive systems. . . . .	126
5.15	Distribution of users, test sets and scenarios for post-editing . . . . .	126
5.16	Results on human post-editing . . . . .	127
6.1	Optimal hyperparameters for each captioning model . . . . .	150
6.2	Text generation results for each model on the MSVD dataset . . . . .	150
6.3	Figures of the EDUB-SegDesc dataset . . . . .	160
6.4	Results of different pre-trained decoders on EDUB-SegDesc . . . . .	164
6.5	TMA results on EDUB-SegDesc compared to the state of the art . . . . .	165
6.6	Unsatisfactory models for EDUB-SegDesc . . . . .	167
6.7	Prediction quality for image and video captioning systems . . . . .	173
6.8	Effort required by interactive-predictive multimodal systems . . . . .	174

# Bibliography

- N. Aafaq, S. Z. Gilani, W. Liu, and A. Mian. Video description: A survey of methods, datasets and evaluation metrics. *arXiv:1806.00186*, 2018.
- M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, et al. Tensorflow: A system for large-scale machine learning. In *Proceedings of the 12th USENIX Symposium on Operating Systems Design and Implementation*, volume 16, pages 265–283, 2016.
- M. Aghaei, M. Dimiccoli, C. C. Ferrer, and P. Radeva. Towards social pattern characterization in egocentric photo-streams. *Computer Vision and Image Understanding*, 171:104–117, 2018.
- R. R. Agrawal, M. Turchi, and M. Negri. Contextual handling in neural machine translation: Look behind, ahead and on both sides. In *Proceedings of the 21st Annual Conference of the European Association for Machine Translation*, pages 11–20, 2018.
- V. Alabau, L. Rodríguez-Ruiz, A. Sanchis, P. Martínez-Gómez, and F. Casacuberta. On multimodal interactive machine translation using speech recognition. In *Proceedings of the International Conference on Multimodal Interaction*, pages 129–136, 2011.
- V. Alabau, R. Bonk, C. Buck, M. Carl, F. Casacuberta, M. García-Martínez, J. González-Rubio, P. Koehn, L. A. Leiva, B. Mesa-Lao, D. Ortiz-Martínez, H. Saint-Amand, G. Sanchis-Trilles, and C. Tsoukala. CASMACAT: An open source workbench for advanced computer aided translation. *The Prague Bulletin of Mathematical Linguistics*, 100:101–112, 2013.

- V. Alabau, A. Sanchis, and F. Casacuberta. Improving on-line handwritten recognition in interactive machine translation. *Pattern Recognition*, 47(3):1217–1228, 2014.
- V. Alabau, M. Carl, F. Casacuberta, M. García-Martínez, J. González-Rubio, B. Mesa-Lao, D. Ortiz-Martínez, M. Schaeffer, and G. Sanchis-Trilles. *New Directions in Empirical Translation Process Research*, chapter Learning Advanced Post-editing, pages 95–110. New Frontiers in Translation Studies. 2016.
- R. B. Allen. Several studies on natural language and back-propagation. In *Proceedings of the IEEE First International Conference on Neural Networks*, pages 335–341, 1987.
- L. B. Almeida, T. Langlois, and J. D. Amaral. Parameter adaptation in stochastic optimization. *On-line Learning in Neural Networks*, pages 111–134, 1998.
- M. Ancona, E. Ceolini, C. Öztireli, and M. Gross. Towards better understanding of gradient-based attribution methods for deep neural networks. *arXiv:1711.06104*, 2017.
- A. G. Arenas. Productivity and quality in the post-editing of outputs from translation memories and machine translation. *Localisation Focus*, 7(1):11–21, 2008.
- A. G. Arenas. *Productivity and quality in MT post-editing*. 2009.
- D. Ataman and M. Federico. Compositional representation of morphologically-rich input for neural machine translation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*, pages 305–311, 2018.
- F. Azadi and S. Khadivi. Improved search strategy for interactive machine translation in computer-assisted translation. In *Proceedings of the XV Machine Translation Summit*, pages 319–332, 2015.
- J. L. Ba, J. R. Kiros, and G. Hinton. Layer normalization. *arXiv:1607.06450*, 2016.
- P. Bahar, C. Brix, and H. Ney. Towards two-dimensional sequence to sequence model in neural machine translation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3009–3015, 2018.
- D. Bahdanau, K. Cho, and Y. Bengio. Neural machine translation by jointly learning to align and translate. *arXiv:1409.0473*, 2015.
- A. V. M. Barone, J. Helcl, R. Sennrich, B. Haddow, and A. Birch. Deep architectures for neural machine translation. In *Proceedings of the Second Conference on Machine Translation*, pages 99–107, 2017.

- 
- S. Barrachina, O. Bender, F. Casacuberta, J. Civera, E. Cubel, S. Khadivi, A. Lagarda, H. Ney, J. Tomás, E. Vidal, and J.-M. Vilar. Statistical approaches to computer-assisted translation. *Computational Linguistics*, 35(1):3–28, 2009.
- A. G. Baydin, R. Cornish, D. M. Rubio, M. Schmidt, and F. Wood. Online learning rate adaptation with hypergradient descent. *arXiv:1703.04782*, 2017.
- A. G. Baydin, B. A. Pearlmutter, A. A. Radul, and J. M. Siskind. Automatic differentiation in machine learning: a survey. *Journal of Machine Learning Research*, 18(153):1–43, 2018.
- O. Bender, S. Hasan, D. Vilar, R. Zens, and H. Ney. Comparison of generation strategies for interactive machine translation. In *Proceedings of the Annual Conference of the European Association for Machine Translation*, pages 33–40, 2005.
- Y. Bengio, P. Simard, and P. Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, 5(2):157–166, 1994.
- Y. Bengio, R. Ducharme, P. Vincent, and C. Janvin. A neural probabilistic language model. *Machine Learning Research*, pages 1137–1155, 2003.
- Y. Bengio, J. Louradour, R. Collobert, and J. Weston. Curriculum learning. In *Proceedings of the 26th annual International Conference on Machine Learning*, pages 41–48, 2009.
- L. Bentivogli, N. Bertoldi, M. Cettolo, M. Federico, M. Negri, and M. Turchi. On the evaluation of adaptive machine translation for human post-editing. *IEEE/ACM Transactions on Audio, Speech and Language Processing*, 24(2):388–399, 2016.
- J. Bergstra and Y. Bengio. Random search for hyper-parameter optimization. *Journal of Machine Learning Research*, 13(1):281–305, 2012.
- J. Bergstra, O. Breuleux, F. Bastien, P. Lamblin, R. Pascanu, G. Desjardins, J. Turian, D. Warde-Farley, and Y. Bengio. Theano: A cpu and gpu math compiler in python. In *Proceedings of the 9th Python in Science Conference*, volume 1, pages 3–10, 2010.
- N. Bertoldi and M. Federico. Domain adaptation for statistical machine translation with monolingual resources. In *Proceedings of the Workshop on Statistical Machine Translation*, pages 182–189, 2009.
- N. Bertoldi, M. Cettolo, and M. Federico. Cache-based online adaptation for machine translation enhanced computer assisted translation. *Proceedings of the XIV Machine Translation Summit*, pages 35–42, 2013.
- A. Betancourt, P. Morerio, C. S. Regazzoni, and M. Rauterberg. The evolution of first person vision methods: A survey. *IEEE Transactions on Circuits and Systems for Video Technology*, 25(5):744–760, 2015.

- E. Biçici and D. Yuret. Optimizing instance selection for statistical machine translation with feature decay algorithms. *IEEE/ACM Transactions on Audio, Speech and Language Processing*, 23(2):339–350, 2015.
- C. M. Bishop. Training with noise is equivalent to tikhonov regularization. *Neural Computation*, 7(1):108–116, 1995.
- C. M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer, 2006. ISBN 0387310738.
- J. Blatz, E. Fitzgerald, G. Foster, S. Gandrabur, C. Goutte, A. Kulesza, A. Sanchis, and N. Ueffing. Confidence estimation for machine translation. In *Proceedings of the International Conference on Computational Linguistics*, pages 315–321, 2004.
- M. Bloodgood and C. Callison-Burch. Bucking the trend: Large-scale cost-focused active learning for statistical machine translation. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 854–864, 2010.
- O. Bojar, C. Buck, C. Callison-Burch, B. Haddow, P. Koehn, C. Monz, M. Post, H. Saint-Amand, R. Soricut, and L. Specia, editors. *Proceedings of the Eighth Workshop on Statistical Machine Translation*. 2013.
- O. Bojar, C. Buck, R. Chatterjee, C. Federmann, Y. Graham, B. Haddow, M. Huck, A. J. Yepes, P. Koehn, and J. Kreutzer. *Proceedings of the Second Conference on Machine Translation*. 2017a.
- O. Bojar, B. Haddow, D. M. , R. Sudarikov, A. Tamchyna, and D. Vari. Report on building translation systems for public health domain (deliverable D1.1). Technical Report H2020-ICT-2014-1-644402, Technical report, Health in my Language (HimL), 2017b.
- O. Bojar, R. Chatterjee, C. Federmann, M. Fishel, Y. Graham, B. Haddow, M. Huck, A. J. Yepes, P. Koehn, C. Monz, M. Negri, A. N ev ol, M. Neves, M. Post, L. Specia, M. Turchi, and K. Verspoor, editors. *Proceedings of the Third Conference on Machine Translation*. 2018.
- M. Bola nos, M. Dimiccoli, and P. Radeva. Toward storytelling from visual lifelogging: An overview. *IEEE Transactions on Human-Machine Systems*, 47(1):77–90, 2017.
- M. Bola nos and P. Radeva. Simultaneous food localization and recognition. In *Proceedings of the 23rd International Conference on Pattern Recognition*, pages 3140–3145, 2016.
- M. Bola nos, A. Ferr a, and P. Radeva. Food ingredients recognition through multi-label learning. In *Proceedings of the International Conference on Image Analysis and Processing*, pages 394–402, 2017.

- 
- M. Bolaños, Á. Peris, F. Casacuberta, and P. Radeva. Vibiknet: Visual bidirectional kernelized network for visual question answering. In *Iberian Conference on Pattern Recognition and Image Analysis*, volume 10255 of *Lecture Notes in Computer Science*, pages 372–380, 2017.
- M. Bolaños, Á. Peris, F. Casacuberta, S. Soler, and P. Radeva. Egocentric video description based on temporally-linked sequences. *Journal of Visual Communication and Image Representation*, 50:205–216, 2018a.
- M. Bolaños, M. Valdivia, and P. Radeva. Where and what am i eating? image-based food menu recognition. In *Proceedings of the European Conference on Computer Vision*, pages 590–605, 2018b.
- L. Bottou. *Une approche theorique de l'apprentissage connexionniste et applications a la reconnaissance de la parole*. PhD thesis, 1991.
- S. Boyd, L. Xiao, and A. Mutapcic. Subgradient methods, 2003. Lecture notes of EE392o, Stanford University, Autumn Quarter.
- D. Britz, A. Goldie, M.-T. Luong, and Q. Le. Massive exploration of neural machine translation architectures. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1442–1451, 2017.
- P. F. Brown, J. Cocke, S. A. D. Pietra, V. J. D. Pietra, F. Jelinek, J. D. Lafferty, R. L. Mercer, and P. S. Roossin. A statistical approach to machine translation. *Computational Linguistics*, 16:79–85, 1990.
- P. F. Brown, V. J. D. Pietra, S. A. D. Pietra, and R. L. Mercer. The mathematics of statistical machine translation: Parameter estimation. *Computational Linguistics*, 19(2):263–311, 1993.
- A. Bustos, A. Pertusa, J.-M. Salinas, and M. de la Iglesia-Vayá. Padchest: A large chest x-ray image dataset with multi-label annotated reports. 2019.
- D. Cai, H. Zhang, and N. Ye. Improvements in statistical phrase-based interactive machine translation. In *Proceedings of the International Conference on Asian Language Processing*, pages 91–94, 2013.
- C. Callison-Burch, C. Fordyce, P. Koehn, C. Monz, and J. Schroeder. (Meta-) evaluation of machine translation. In *Proceedings of the Workshop on Statistical Machine Translation*, pages 136–158, 2007.
- A. Cartas, M. Dimiccoli, and P. Radeva. Batch-based activity recognition from egocentric photo-streams. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2347–2354, 2017.

- G. A. Casañ and M. A. Castaño. Distributed representation of vocabularies in the RECONTRA neural translator. In *Proceedings of the Sixth European Conference on Speech Communication and Technology*, pages 2423–2426, 1999.
- M.-A. Castaño and F. Casacuberta. A connectionist approach to machine translation. In *Proceedings of the International Conference on Theoretical and Methodological Issues in Machine Translation*, pages 160–167, 1997.
- S. Castilho, J. Moorkens, F. Gaspari, I. Calixto, J. Tinsley, and A. Way. Is neural machine translation the new state of the art? *The Prague Bulletin of Mathematical Linguistics*, 108(1):109–120, 2017.
- D. Castro, S. Hickson, V. Bettadapura, E. Thomaz, G. Abowd, H. Christensen, and I. Essa. Predicting daily activities from egocentric images using deep learning. In *Proceedings of the 2015 ACM International symposium on Wearable Computers*, pages 75–82, 2015.
- W. Chan, N. Jaitly, Q. Le, and O. Vinyals. Listen, attend and spell: A neural network for large vocabulary conversational speech recognition. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 4960–4964, 2016.
- O. Chapelle, B. Schölkopf, and A. Zien. *Semi-Supervised Learning*. MIT press, 2006.
- C. Chelba, T. Mikolov, M. Schuster, Q. Ge, T. Brants, P. Koehn, and T. Robinson. One billion word benchmark for measuring progress in statistical language modeling. *arXiv:1312.3005*, 2013.
- B. Chen and C. Cherry. A systematic comparison of smoothing techniques for sentence-level bleu. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*, pages 362–367, 2014.
- D. L. Chen and W. B. Dolan. Collecting highly parallel data for paraphrase evaluation. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics*, pages 190–200, 2011.
- X. Chen, H. Fang, T.-Y. Lin, R. Vedantam, S. Gupta, P. Dollár, and C. L. Zitnick. Microsoft COCO captions: Data collection and evaluation server. *arXiv:1504.00325*, 2015.
- S. Cheng, S. Huang, H. Chen, X.-Y. Dai, and J. Chen. Print: A pick-revise framework for interactive machine translation. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1240–1249, 2016.
- D. Chiang. Hope and fear for discriminative training of statistical translation models. *Journal of Machine Learning Research*, 13:1159–1187, 2012.



- 
- M. Chinea-Rios, Á. Peris, and F. Casacuberta. Adapting neural machine translation with parallel synthetic data. In *Proceedings of the Second Conference on Machine Translation*, pages 138–147, 2017.
- T. Ching, D. S. Himmelstein, B. K. Beaulieu-Jones, A. A. Kalinin, B. T. Do, G. P. Way, E. Ferrero, P.-M. Agapow, M. Zietz, M. M. Hoffman, W. Xie, G. L. Rosen, B. J. Lengerich, J. Israeli, J. Lanchantin, S. Woloszynek, A. E. Carpenter, A. Shrikumar, J. Xu, E. M. Cofer, C. A. Lavender, S. C. Turaga, A. M. Alexandari, Z. Lu, D. J. Harris, D. DeCaprio, Y. Qi, A. Kundaje, Y. Peng, L. K. Wiley, M. H. S. Segler, S. M. Boca, S. J. Swamidass, A. Huang, A. Gitter, and C. S. Greene. Opportunities and obstacles for deep learning in biology and medicine. *Journal of The Royal Society Interface*, 15(141), 2018.
- K. Cho, B. van Merriënboer, D. Bahdanau, and Y. Bengio. On the properties of neural machine translation: Encoder-decoder approaches. In *Proceedings of the Workshop on Syntax, Semantic and Structure in Statistical Translation*, pages 103–111, 2014.
- F. Chollet et al. Keras. <https://github.com/keras-team/keras>, 2015. GitHub repository.
- L. Chrisman. Learning recursive distributed representations for holistic computation. *Connection Science*, 3(4):345–366, 1991.
- D. Cohn, L. Atlas, and R. Ladner. Improving generalization with active learning. *Machine Learning*, 15(2):201–221, 1994.
- R. Collobert, K. Kavukcuoglu, and C. Farabet. Torch7: A matlab-like environment for machine learning. In *BigLearn Workshop in Advances in Neural Information Processing Systems*, 2011.
- K. Crammer and Y. Singer. Ultraconservative online algorithms for multiclass problems. In *Proceedings of the Annual Conference on Computational Learning Theory*, pages 99–115, 2001.
- K. Crammer, O. Dekel, J. Keshet, S. Shalev-Shwartz, and Y. Singer. Online passive-aggressive algorithms. *Journal of Machine Learning Research*, 7:551–585, 2006.
- J. M. Crego, J. Kim, G. Klein, A. Rebollo, K. Yang, J. Senellart, E. Akhanov, P. Brunelle, A. Coquard, Y. Deng, S. Enoue, C. Geiss, J. Johanson, A. Khalsa, R. Khiari, B. Ko, C. Kobus, J. Lorieux, L. Martins, D. Nguyen, A. Priori, T. Riccardi, N. Segal, C. Servan, C. Tiquet, B. Wang, J. Yang, D. Zhang, J. Zhou, and P. Zoldan. SYSTRAN’s pure neural machine translation systems. *arXiv:1610.05540*, 2016.
- G. Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals and Systems*, 2(4):303–314, 1989.

- J. Daems and L. Macken. Interactive adaptive smt versus interactive adaptive nmt: a user experience evaluation. *Machine Translation*, pages 1–18, 2019.
- I. Dagan and S. P. Engelson. Committee-based sampling for training probabilistic classifiers. In *Proceedings of the Twelfth International Conference on Machine Learning*, pages 150–157. 1995.
- A. A. Dara, J. van Genabith, Q. Liu, J. Judge, and A. Toral. Active learning for post-editing based incrementally retrained MT. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics, volume 2: Short Papers*, pages 185–189, 2014.
- J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009.
- M. Denkowski, C. Dyer, and A. Lavie. Learning from post-editing: Online model adaptation for statistical machine translation. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pages 395–404, 2014a.
- M. Denkowski, A. Lavie, I. Lacruz, and C. Dyer. Real time adaptive machine translation for post-editing with cdec and transcenter. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pages 72–77, 2014b.
- J. Devlin. Sharp models on dull hardware: Fast and accurate neural machine translation decoding on the CPU. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2820–2825, 2017.
- J. Devlin, R. Zbib, Z. Huang, T. Lamar, R. Schwartz, and J. Makhoul. Fast and robust neural network joint models for statistical machine translation. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, pages 1370–1380, 2014.
- J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. *arXiv:1810.04805*, 2018.
- M. Dimiccoli, M. Bolaños, E. Talavera, M. Aghaei, S. G. Nikolov, and P. Radeva. SR-clustering: Semantic regularized clustering for egocentric photo streams segmentation. *Computer Vision and Image Understanding*, 155:55–69, 2017.
- A. R. Doherty, S. E. Hodges, A. C. King, A. F. Smeaton, E. Berry, C. J. Moulin, S. Lindley, P. Kelly, and C. Foster. Wearable cameras in health. *American Journal of Preventive Medicine*, 44(3):320–323, 2013.

- M. Domingo, Á. Peris, and F. Casacuberta. Interactive-predictive translation based on multiple word-segments. In *Proceedings of the 19th Annual Conference of the European Association for Machine Translation*, pages 282–291, 2016.
- M. Domingo, Á. Peris, and F. Casacuberta. Segment-based interactive-predictive machine translation. *Machine Translation*, 31:1–23, 2018.
- K. Dranch, R. Beninato, and T. Johnson. The size and state of the language services industry, including ranking of top 100 LSPs by revenue. Technical report, Nimdzi, 2018.
- J. Duchi, E. Hazan, and Y. Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12:2121–2159, 2011.
- C. Dyer, V. Chahuneau, and N. A. Smith. A simple, fast, and effective reparameterization of IBM Model 2. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 644–648, 2013.
- S. Edunov, M. Ott, M. Auli, D. Grangier, and M. Ranzato. Classical structured prediction losses for sequence to sequence learning. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 355–364, 2018.
- M. Elbayad, L. Besacier, and J. Verbeek. Pervasive attention: 2d convolutional neural networks for sequence-to-sequence prediction. In *Proceedings of the 22nd Conference on Computational Natural Language Learning*, pages 97–107, 2018.
- J. L. Elman. Finding structure in time. *Cognitive Science*, 14(2):179–211, 1990.
- J. L. Elman. Distributed representations, simple recurrent networks, and grammatical structure. *Machine Learning*, 7(2-3):195–225, 1991.
- C. Fan and D. J. Crandall. Deepdiary: Automatically captioning lifelogging image streams. In *Proceedings of European Conference on Computer Vision*, pages 459–473, 2016.
- C. Fan, Z. Zhang, and D. J. Crandall. Deepdiary: Lifelogging image captioning and summarization. *Journal of Visual Communication and Image Representation*, 55:40–55, 2018.
- A. Farhadi, M. Hejrati, M. A. Sadeghi, P. Young, C. Rashtchian, J. Hockenmaier, and D. Forsyth. Every picture tells a story: Generating sentences from images. In *Proceedings of the European Conference on Computer Vision*, pages 15–29, 2010.

- A. Fathi, A. Farhadi, and J. M. Rehg. Understanding egocentric activities. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 407–414, 2011.
- M. Federico, N. Bertoldi, M. Cettolo, M. Negri, M. Turchi, M. Trombetti, A. Cattelan, A. Farina, D. Lupinetti, A. Martines, A. Massidda, H. Schwenk, L. Barrault, F. Blain, P. Koehn, C. Buck, and U. Germann. The matecat tool. In *Proceedings of the 25th International Conference on Computational Linguistics: System Demonstrations*, pages 129–132, 2014.
- M. L. Forcada and R. P. Neco. Recursive hetero-associative memories for translation. In *Proceedings of the International Work-Conference on Artificial Neural Networks*, pages 453–462, 1997.
- G. Foster, P. Isabelle, and P. Plamondon. Target-text mediated interactive machine translation. *Machine Translation*, 12:175–194, 1997.
- J. French. The time travellers CAPM. *Investment Analysts Journal*, 46(2):81–96, 2017.
- K. Fukushima. Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological Cybernetics*, 36(4):193–202, 1980.
- P. Gage. A new algorithm for data compression. *The C Users Journal*, 12(2):23–38, 1994.
- S. Gandrabur and G. Foster. Confidence estimation for text prediction. In *Proceedings of the Conference on Computational Natural Language Learning*, pages 315–321, 2003.
- J. Gehring, M. Auli, D. Grangier, D. Yarats, and Y. N. Dauphin. Convolutional sequence to sequence learning. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 1243–1252, 2017.
- M. A. Gelbart, J. Snoek, and R. P. Adams. Bayesian optimization with unknown constraints. In *Proceedings of the Thirtieth Conference on Uncertainty in Artificial Intelligence*, pages 250–259, 2014.
- F. A. Gers, J. Schmidhuber, and F. Cummins. Learning to forget: Continual prediction with LSTM. *Neural Computation*, 12(10):2451–2471, 2000.
- X. Glorot and Y. Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256, 2010.

- K. Goel and J. Naik. Deepseek: A video captioning tool for making videos searchable. cs224d.stanford.edu, 2016.
- J. González-Rubio and F. Casacuberta. Cost-sensitive active learning for computer-assisted translation. *Pattern Recognition Letters*, 37:124–134, 2014.
- J. González-Rubio, D. Ortiz-Martínez, and F. Casacuberta. Balancing user effort and translation error in interactive machine translation via confidence measures. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, pages 173–177, 2010a.
- J. González-Rubio, D. Ortiz-Martínez, and F. Casacuberta. On the use of confidence measures within an interactive-predictive machine translation system. In *Proceedings of the Annual Conference of the European Association for Machine Translation*, page [no page numbers], 2010b.
- J. González-Rubio, D. Ortiz-Martínez, and F. Casacuberta. An active learning scenario for interactive machine translation. In *Proceedings of the 13th international conference on multimodal interfaces*, pages 197–200, 2011.
- J. González-Rubio, D. Ortiz-Martínez, and F. Casacuberta. Active learning for interactive machine translation. In *Proceedings of the Conference of the European Chapter of the Association for Computational Linguistics*, pages 245–254, 2012.
- J. González-Rubio, D. Ortiz-Martínez, J.-M. Benedí, and F. Casacuberta. Interactive machine translation using hierarchical translation models. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 244–254, 2013.
- J. González-Rubio, J.-M. Benedí, D. Ortiz-Martínez, and F. Casacuberta. Beyond prefix-based interactive translation prediction. In *Proceedings of the Conference on Computational Natural Language Learning*, pages 198–207, 2016.
- I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT Press, 2016.
- E. Granell, V. Romero, and C. D. Martínez-Hinarejos. An interactive approach with off-line and on-line handwritten text recognition combination for transcribing historical documents. In *Proceedings of the International Association for Pattern Recognition Workshop on Document Analysis Systems*, pages 269–274, 2016.
- A. Graves. Generating sequences with recurrent neural networks. *arXiv:1308.0850*, 2013.
- A. Graves, M. Liwicki, S. Fernández, R. Bertolami, H. Bunke, and J. Schmidhuber. A novel connectionist system for unconstrained handwriting recognition. *IEEE transactions on Pattern Analysis and Machine Intelligence*, 31(5):855–868, 2009.

- A. Graves, A.-r. Mohamed, and G. Hinton. Speech recognition with deep recurrent neural networks. In *Proceedings of the Institute of Electrical and Electronics Engineers International Conference on Acoustics, Speech and Signal Processing*, pages 6645–6649, 2013.
- S. Green, J. Heer, and C. D. Manning. The efficacy of human post-editing for language translation. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 439–448, 2013a.
- S. Green, S. Wang, D. Cer, and C. D. Manning. Fast and adaptive online training of feature-rich translation models. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, volume 1, pages 311–321, 2013b.
- S. Green, J. Chuang, J. Heer, and C. D. Manning. Predictive translation memory: A mixed-initiative system for human language translation. In *Proceedings of the Annual Association for Computing Machinery Symposium on User Interface Software and Technology*, pages 177–187, 2014.
- K. Greff, R. K. Srivastava, J. Koutník, B. R. Steunebrink, and J. Schmidhuber. LSTM: A search space odyssey. *IEEE Transactions on Neural Networks and Learning Systems*, 28(10):2222–2232, 2017.
- J. Gu, K. Cho, and V. O. Li. Trainable greedy decoding for neural machine translation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1958–1968, 2017.
- G. Haffari, M. Roy, and A. Sarkar. Active learning for statistical phrase-based machine translation. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 415–423, 2009.
- E. Hasler, B. Haddow, and P. Koehn. Margin infused relaxed algorithm for mooses. *The Prague Bulletin of Mathematical Linguistics*, 96:69–78, 2011.
- E. Hasler, A. Gispert, G. Iglesias, and B. Byrne. Neural machine translation decoding with terminology constraints. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, volume 2, pages 506–512, 2018.
- G. E. Hinton. Learning distributed representations of concepts. In *Proceedings of the Annual Meeting of the Cognitive Science Society*, pages 12–24, 1986.
- S. Hochreiter. Untersuchungen zu dynamischen neuronalen netzen. *Diploma, Technische Universität München*, 91:1, 1991.
- S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.

- M. Hodosh, P. Young, and J. Hockenmaier. Framing image description as a ranking task: Data, models and evaluation metrics. *Journal of Artificial Intelligence Research*, 47:853–899, 2013.
- C. Hokamp and Q. Liu. Lexically constrained decoding for sequence generation using grid beam search. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, volume 1, pages 1535–1546, 2017.
- C. Hori, T. Hori, T.-Y. Lee, Z. Zhang, B. Harsham, J. R. Hershey, T. K. Marks, and K. Sumi. Attention-based multimodal fusion for video description. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4203–4212, 2017.
- S.-W. Hsueh, T.-Y. Lin, W.-I. Lei, C.-L. P. Ngai, Y.-H. Sheng, and Y.-S. Wu. Semantic failover in software-defined networking. In *Proceedings of the IEEE 23rd Pacific Rim International Symposium on Dependable Computing*, pages 299–308, 2018.
- K. Hu and P. Cadwell. A comparative study of post-editing guidelines. In *Proceedings of the 19th Annual Conference of the European Association for Machine Translation*, pages 34206–353, 2016.
- J. Hutchins. Two precursors of machine translation: Artsrouni and trojanskij. *International Journal of Translation*, 16(1):11–31, 2004.
- J. M. Inesta and C. Pérez-Sancho. Interactive multimodal music transcription. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 211–215, 2013.
- S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of the International Conference on Machine Learning*, pages 448–456, 2015.
- J. Ive, F. Blain, and L. Specia. deepQuest: A framework for neural-based quality estimation. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 3146–3157, 2018.
- Y. Iwashita, A. Takamine, R. Kurazume, and M. S. Ryoo. First-person animal activity recognition from egocentric videos. In *Proceedings of the 22nd International Conference on Pattern Recognition*, pages 4310–4315, 2014.
- S. Jean, K. Cho, R. Memisevic, and Y. Bengio. On using very large target vocabulary for neural machine translation. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics and the International Joint Conference on Natural Language Processing*, pages 1–10, 2015a.
- S. Jean, O. Firat, K. Cho, R. Memisevic, and Y. Bengio. Montreal neural machine translation systems for WMT15. In *Proceedings of the Tenth Workshop on Statistical Machine Translation*, pages 134–140, 2015b.

- Y. Jia, M. Carl, and X. Wang. Post-editing neural machine translation versus phrase-based machine translation for english–chinese. *Machine Translation*, pages 1–21, 2019.
- M. I. Jordan. Artificial neural networks. chapter Attractor Dynamics and Parallelism in a Connectionist Sequential Machine, pages 112–127. 1990.
- Ł. Kaiser, O. Nachum, A. Roy, and S. Bengio. Learning to remember rare events. *arXiv:1703.03129*, 2017.
- N. Kalchbrenner and P. Blunsom. Recurrent continuous translation models. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1700–1709, 2013.
- S. Kang and K. Cho. Conditional molecular design with deep generative models. *arXiv:1805.00108*, 2018.
- S. Karimova, P. Simianer, and S. Riezler. A user-study on online adaptation of neural machine translation to human post-edits. *Machine Translation*, 32(4):309–324, 2018.
- A. Karpathy and L. Fei-Fei. Deep visual-semantic alignments for generating image descriptions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3128–3137, 2015.
- M. Kay. The proper place of men and machines in language translation. Technical report, 1980. Xerox Palo Alto Research Center.
- H. Khayrallah, B. Thompson, K. Duh, and P. Koehn. Regularized training objective for continued training for domain adaptation in neural machine translation. In *Proceedings of the 2nd Workshop on Neural Machine Translation and Generation*, pages 36–44, 2018.
- Y. Kim. Convolutional neural networks for sentence classification. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1746–1751, 2014.
- D. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv:1412.6980*, 2014.
- G. Klein, Y. Kim, Y. Deng, J. Senellart, and A. Rush. Opennmt: Open-source toolkit for neural machine translation. *Proceedings of the Annual Meeting of the Association for Computational Linguistics, System Demonstrations*, pages 67–72, 2017.
- R. Kneser and H. Ney. Improved backing-off for m-gram language modeling. In *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing*, volume 1, pages 181–184, 1995.



- R. Knowles and P. Koehn. Neural interactive translation prediction. In *Proceedings of the Association for Machine Translation in the Americas*, pages 107–120, 2016.
- R. Knowles, M. Sanchez-Torron, and P. Koehn. A user study of neural interactive translation prediction. *Machine Translation*, In Press, 2019. doi: 10.1007/s10590-019-09235-8.
- P. Koehn. Statistical significance tests for machine translation evaluation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 388–395, 2004.
- P. Koehn. Europarl: A parallel corpus for statistical machine translation. In *Proceedings of the Machine Translation Summit*, pages 79–86, 2005.
- P. Koehn. A process study of computer-aided translation. *Machine Translation*, 23(4):241–263, 2010a.
- P. Koehn. *Statistical machine translation*. Cambridge University Press, 2010b.
- P. Koehn and R. Knowles. Six challenges for neural machine translation. In *Proceedings of the First Workshop on Neural Machine Translation*, pages 28–39, 2017.
- P. Koehn and C. Monz, editors. *Proceedings on the Workshop on Statistical Machine Translation*. 2006.
- P. Koehn, H. Hoang, A. Birch, C. Callison-Burch, M. Federico, N. Bertoldi, B. Cowan, W. Shen, C. Moran, R. Zens, C. Dyer, O. Bojar, A. Constantin, and E. Herbst. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, pages 177–180, 2007.
- P. Koehn, C. Tsoukala, and H. Saint-Amand. Refinements to interactive translation prediction based on search graphs. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, pages 574–578, 2014.
- M. Koponen, L. Salmi, and M. Nikulin. A product and process analysis of post-editor corrections on neural, statistical and rule-based machine translation output. *Machine Translation*, pages 1–30, 2019.
- S. S. R. Kothur, R. Knowles, and P. Koehn. Document-level adaptation for neural machine translation. In *Proceedings of the 2nd Workshop on Neural Machine Translation and Generation*, pages 64–73, 2018.
- J. Kreutzer, A. Sokolov, and S. Riezler. Bandit structured prediction for neural sequence-to-sequence learning. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1503–1513, 2017.

- R. Krishna, K. Hata, F. Ren, L. Fei-Fei, and J. Carlos Niebles. Dense-captioning events in videos. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 706–715, 2017.
- N. Krishnamoorthy, G. Malkarnenkar, R. J. Mooney, K. Saenko, and S. Guadarrama. Generating natural-language video descriptions using text-mined knowledge. In *Proceedings of the Twenty-Seventh Association for the Advancement of Artificial Intelligence Conference on Artificial Intelligence*, volume 1, pages 541–547, 2013.
- A. Krizhevsky, I. Sutskever, and G. E. Hinton. ImageNet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, pages 1097–1105, 2012.
- A. Krogh and J. A. Hertz. A simple weight decay can improve generalization. In *Advances in Neural Information Processing Systems*, pages 950–957, 1992.
- P. Kuznetsova, V. Ordonez, T. Berg, and Y. Choi. Treetalk: Composition and compression of trees for image descriptions. *Transactions of the Association of Computational Linguistics*, 2(1):351–362, 2014.
- A. L. Lagarda, D. Ortiz-Martínez, V. Alabau, and F. Casacuberta. Translating without in-domain corpus: Machine translation post-editing with online learning techniques. *Computer Speech & Language*, 32(1):109–134, 2015.
- S. M. Lakew, M. Cettolo, and M. Federico. A comparison of transformer and recurrent neural networks on multilingual neural machine translation. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 641–652, 2018.
- T. K. Lam, J. Kreutzer, and S. Riezler. A reinforcement learning approach to interactive-predictive neural machine translation. In *Proceedings of the European Association for Machine Translation conference*, pages 169–178, 2018.
- P. Langlais, G. Lapalme, and M. Lorange. TransType: Development-evaluation cycles to boost translator’s productivity. *Machine Translation*, 17(2):77–98, 2002.
- A. Lavie and M. J. Denkowski. The METEOR metric for automatic evaluation of machine translation. *Machine Translation*, 23(2-3):105–115, 2009.
- N. T. Le, F. Sadat, L. Menard, and D. Dinh. Low-resource machine transliteration using recurrent neural networks. *ACM Transactions on Asian and Low-Resource Language Information Processing*, 18(2):13, 2019.
- Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Backpropagation applied to handwritten zip code recognition. *Neural Computation*, 1(4):541–551, 1989a.

- Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.
- Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *Nature*, 521(7553):436, 2015.
- Y. LeCun et al. Generalization and network design strategies. *Connectionism in perspective*, pages 143–155, 1989b.
- J. Lee, K. Cho, and T. Hofmann. Fully character-level neural machine translation without explicit segmentation. *Transactions of the Association for Computational Linguistics*, 5:365–378, 2017.
- A. Levenberg, C. Callison-Burch, and M. Osborne. Stream-based translation models for statistical machine translation. In *Proceedings of the Human Language Technologies: the 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 394–402, 2010.
- D. D. Lewis and J. Catlett. Heterogeneous uncertainty sampling for supervised learning. In *Machine Learning Proceedings*, pages 148–156, 1994.
- J. Libovický, A. Tamchyna, and P. Pecina. Adaptation to hungarian, swedish, and spanish (deliverable D1.4). Technical Report H2020-ICT-2014-1-644753, Technical report, KConnect, 2016.
- A. Lidon, M. Bolaños, M. Dimiccoli, P. Radeva, M. Garolera, and X. Giro-i Nieto. Semantic summarization of egocentric photo stream events. In *Proceedings of the 2nd Workshop on Lifelogging Tools and Applications*, pages 3–11, 2017.
- Z. Lin, M. Feng, C. N. d. Santos, M. Yu, B. Xiang, B. Zhou, and Y. Bengio. A structured self-attentive sentence embedding. *arXiv:1703.03130*, 2017.
- B. T. Lowerre. The HARPY speech recognition system. Technical report, Carnegie-Mellon University, 1976.
- Z. Lu and K. Grauman. Story-driven summarization for egocentric video. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2714–2721, 2013.
- M.-T. Luong and C. D. Manning. Achieving open vocabulary neural machine translation with hybrid word-character models. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, pages 1054–1063, 2016.
- T. Luong, H. Pham, and C. D. Manning. Effective approaches to attention-based neural machine translation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1412–1421, 2015a.

- T. Luong, I. Sutskever, Q. Le, O. Vinyals, and W. Zaremba. Addressing the rare word problem in neural machine translation. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics and the International Joint Conference on Natural Language Processing*, pages 11–19, 2015b.
- E. Macklovitch. TransType2: The last word. In *Proceedings of the International Conference on Language Resources and Evaluation*, pages 167–172, 2006.
- J. Mao, W. Xu, Y. Yang, J. Wang, and A. L. Yuille. Explain images with multimodal recurrent neural networks. *arXiv:1410.1090*, 2014.
- B. Marie and A. Max. Touch-based pre-post-editing of machine translation output. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1040–1045, 2015.
- D. Martín-Albo, V. Romero, and E. Vidal. Interactive off-line handwritten text transcription using on-line handwritten text as feedback. In *Proceedings of the 12th International Conference on Document Analysis and Recognition*, pages 1280–1284, 2013.
- P. Martínez-Gómez, G. Sanchis-Trilles, and F. Casacuberta. Online adaptation strategies for statistical machine translation in post-editing scenarios. *Pattern Recognition*, 45(9):3193–3203, 2012.
- P. Mathur, M. Cettolo, M. Federico, and F.-F. B. Kessler. Online learning approaches in computer assisted translation. In *Proceedings of the Eighth Workshop on Statistical Machine Translation*, pages 301–308, 2013.
- C. Mauro, G. Christian, and F. Marcello. Wit3: Web inventory of transcribed and translated talks. In *Proceedings of the Conference of European Association for Machine Translation*, pages 261–268, 2012.
- W. S. McCulloch and W. Pitts. A logical calculus of the ideas immanent in nervous activity. *The Bulletin of Mathematical Biophysics*, 5(4):115–133, 1943.
- P. Michel and G. Neubig. Extreme adaptation for personalized neural machine translation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*, pages 312–318, 2018.
- T. Mikolov, M. Karafiát, L. Burget, J. Cernocký, and S. Khudanpur. Recurrent neural network based language model. In *Proceedings of the Annual Conference of the International Speech Communication Association*, pages 1045–1048, 2010.
- T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient estimation of word representations in vector space. *arXiv:1301.3781*, 2013a.

- 
- T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems*, pages 3111–3119, 2013b.
- P. Mildenerger, M. Eichelberg, and E. Martin. Introduction to the DICOM standard. *European radiology*, 12(4):920–927, 2002.
- N. Murata. A statistical study of on-line learning. *Online Learning and Neural Networks*. Cambridge University Press, Cambridge, UK, pages 63–92, 1998.
- L. Nepveu, G. Lapalme, P. Langlais, and G. Foster. Adaptive language and translation models for interactive machine translation. In *Proceedings of the Conference on Empirical Method in Natural Language Processing*, pages 190–197, 2004.
- Y. E. Nesterov. A method for solving the convex programming problem with convergence rate  $O(1/k^2)$ . In *Doklady Akademii Nauk SSSR*, volume 269, pages 543–547, 1983.
- H. Ney. On the probabilistic interpretation of neural network classifiers and discriminative training criteria. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(2):107–119, 1995.
- J. Nielsen. *Usability Engineering*. Morgan Kaufmann Publishers Inc., 1993.
- E. W. Noreen. *Computer-intensive methods for testing hypotheses*. Wiley New York, 1989.
- F. J. Och. Minimum error rate training in statistical machine translation. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, pages 160–167, 2003.
- F. J. Och and H. Ney. Discriminative training and maximum entropy models for statistical machine translation. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, pages 295–302, 2002.
- F. Olsson. A literature survey of active machine learning in the context of natural language processing. Technical report, Swedish Institute of Computer Science, 2009.
- D. Ortiz-Martínez. *Advances in Fully-Automatic and Interactive Phrase-Based Statistical Machine Translation*. PhD thesis, Universidad Politécnica de Valencia, 2011.
- D. Ortiz-Martínez. Online learning for statistical machine translation. *Computational Linguistics*, 42(1):121–161, 2016.
- D. Ortiz-Martínez, I. García-Varea, and F. Casacuberta. Online learning for interactive statistical machine translation. In *Proceedings of the North American Chapter of the Association for Computational Linguistics - Human Language Technologies*, pages 546–554, 2010.

- P. Pan, Z. Xu, Y. Yang, F. Wu, and Y. Zhuang. Hierarchical recurrent neural encoder for video representation with application to captioning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1029–1038, 2016.
- K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu. BLEU: a method for automatic evaluation of machine translation. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, pages 311–318, 2002.
- A. Parikh, O. Täckström, D. Das, and J. Uszkoreit. A decomposable attention model for natural language inference. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2249–2255, 2016.
- R. Pascanu, T. Mikolov, and Y. Bengio. On the difficulty of training recurrent neural networks. In *Proceedings of the International Conference on Machine Learning*, pages 1310–1318, 2013.
- R. Pascanu, C. Gulcehre, K. Cho, and Y. Bengio. How to construct deep recurrent neural networks. *arXiv:1312.6026*, 2014.
- P. Passban, Q. Liu, and A. Way. Improving character-based decoding using target-side morphological information for neural machine translation. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, volume 1, pages 58–68, 2018.
- A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer. Automatic differentiation in pytorch. In *NIPS 2017 Autodiff Workshop: The Future of Gradient-based Machine Learning Software and Techniques*, 2017.
- J. Pennington, R. Socher, and C. Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pages 1532–1543, 2014.
- Á. Peris and F. Casacuberta. NMT-Keras: a very flexible toolkit with a focus on interactive NMT and online learning. *The Prague Bulletin of Mathematical Linguistics*, 111:113–124, 2018a.
- Á. Peris and F. Casacuberta. Active learning for interactive neural machine translation of data streams. In *Proceedings of the Conference on Computational Natural Language Learning*, pages 151–160, 2018b.
- Á. Peris and F. Casacuberta. Online learning for effort reduction in interactive neural machine translation. *Computer Speech & Language*, 58:98–126, 2019a.
- Á. Peris and F. Casacuberta. Interactive-predictive neural multimodal systems. In *Iberian Conference on Pattern Recognition and Image Analysis*, Lecture Notes in Computer Science. In press., 2019b.

- 
- Á. Peris, M. Bolaños, P. Radeva, and F. Casacuberta. Video description using bidirectional recurrent neural networks. In *Proceedings of the International Conference on Artificial Neural Networks*, pages 3–11, 2016.
- Á. Peris, L. Cebrián, and F. Casacuberta. Online learning for neural machine translation post-editing. *arXiv:1706.03196*, 2017a.
- Á. Peris, M. Chinea-Ríos, and F. Casacuberta. Neural networks classifier for data selection in statistical machine translation. *The Prague Bulletin of Mathematical Linguistics*, 108(1):283–294, 2017b.
- Á. Peris, M. Domingo, and F. Casacuberta. Interactive neural machine translation. *Computer Speech & Language*, 45:201–220, 2017c.
- M. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, volume 1, pages 2227–2237, 2018.
- J. R. Pierce and J. B. Carroll. *Language and machines: Computers in translation and linguistics*. National Academy of Sciences/National Research Council, 1966.
- B. A. Plummer, L. Wang, C. M. Cervantes, J. C. Caicedo, J. Hockenmaier, and S. Lazebnik. Flickr30k entities: Collecting region-to-phrase correspondences for richer image-to-sentence models. In *Proceedings of the International Conference of Computer Vision*, pages 2641–2649, 2015.
- Y. Poley, A. Ephrat, S. Peleg, and C. Arora. Compact cnn for indexing egocentric videos. In *Proceedings of the IEEE Winter Conference on Applications of Computer Vision*, pages 1–9, 2016.
- B. T. Polyak. Some methods of speeding up the convergence of iteration methods. *USSR Computational Mathematics and Mathematical Physics*, 4(5):1–17, 1964.
- M. Popel and O. Bojar. Training tips for the transformer model. *The Prague Bulletin of Mathematical Linguistics*, 110(1):43–70, 2018.
- R. Possas, S. Pinto Caceres, and F. Ramos. Egocentric activity recognition on a budget. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5967–5976, 2018.
- M. Post and D. Vilar. Fast lexically constrained decoding with dynamic beam allocation for neural machine translation. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1314–1324, 2018.

- R. Puduppully, L. Dong, and M. Lapata. Data-to-text generation with content selection and planning. *arXiv:1809.00582*, 2018.
- J. A. Pérez-Ortiz, D. Torregrosa, and M. Forcada. Black-box integration of heterogeneous bilingual resources into an interactive translation system. In *Proceedings of the European Chapter of the Association for Computational Linguistics Workshop on Humans and Computer-assisted Translation*, pages 57–65, 2014.
- L. Quirós, C.-D. Martínez-Hinarejos, A. H. Toselli, and E. Vidal. Interactive layout detection. In *Proceedings of the Iberian Conference on Pattern Recognition and Image Analysis*, volume 10255 of *Lecture Notes in Computer Science*, pages 161–168, 2017.
- A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever. Language models are unsupervised multitask learners. Technical report, Open-AI, 2019.
- M. Ranzato, S. Chopra, M. Auli, and W. Zaremba. Sequence level training with recurrent neural networks. *arXiv:1511.06732*, 2015.
- J. Reunanen. Overfitting in making comparisons between variable selection methods. *Journal of Machine Learning Research*, 3(Mar):1371–1382, 2003.
- S. Riezler and J. T. Maxwell. On some pitfalls in automatic evaluation and significance testing for mt. In *Proceedings of the Association for Computational Linguistics Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*, pages 57–64, 2005.
- H. Robbins and S. Monro. A stochastic approximation method. *The Annals of Mathematical Statistics*, pages 400–407, 1951.
- L. Rodríguez, F. Casacuberta, and E. Vidal. Computer assisted transcription of speech. In *Proceedings of the Iberian Conference on Pattern Recognition and Image Analysis*, pages 241–248, 2007.
- M. Rohrbach, W. Qiu, I. Titov, S. Thater, M. Pinkal, and B. Schiele. Translating video content to natural language descriptions. In *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition*, pages 433–440, 2013.
- F. Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological Review*, 65(6):386, 1958.
- D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning representations by back-propagating errors. *Nature*, 323(6088):533, 1986.
- O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, et al. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015.



- 
- G. Sanchis-Trilles and F. Casacuberta. Improving translation quality stability using bayesian predictive adaptation. *Computer Speech & Language*, 34(1):1–17, 2015.
- G. Sanchis-Trilles, D. Ortiz-Martínez, J. Civera, F. Casacuberta, E. Vidal, and H. Hoang. Improving interactive machine translation via mouse actions. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 485–494, 2008.
- A. M. Saxe, J. L. McClelland, and S. Ganguli. Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. *arXiv:1312.6120*, 2013.
- SchlumbergerSema S.A., Intituto Tecnológico de Informática, Rheinisch Westfälische Technische Hochschule Aachen Lehrstuhl für Informatik VI, Recherche Appliquée en Linguistique Informatique Laboratory University of Montreal, Celer Soluciones, Société Gamma, and Xerox Research Centre Europe. TT2. TransType2-computer-assisted translation. Project technical annex. Technical Report IST-2001-32091, Information Society Technologies Programme, 2001.
- M. Schuster and K. K. Paliwal. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11):2673–2681, 1997.
- H. Schwenk. Continuous space language models. *Computer Speech & Language*, 21(3):492–518, 2007.
- F. M. Segarra, L. A. Leiva, and R. Paredes. A relevant image search engine with late fusion: mixing the roles of textual and visual descriptors. In *Proceedings of the 16th international conference on Intelligent user interfaces*, pages 455–456, 2011.
- A. J. Sellen, A. Fogg, M. Aitken, S. Hodges, C. Rother, and K. Wood. Do life-logging technologies support memory for the past?: an experimental study using sensecam. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 81–90, 2007.
- R. Sennrich, B. Haddow, and A. Birch. Neural machine translation of rare words with subword units. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, pages 1715–1725, 2016.
- R. Sennrich, O. Firat, K. Cho, A. Birch, B. Haddow, J. Hitschler, M. Junczys-Dowmunt, S. Läubli, A. V. Miceli Barone, J. Mokry, and M. Nadejde. Nematus: a toolkit for neural machine translation. In *Proceedings of the Software Demonstrations at the 15th Conference of the European Chapter of the Association for Computational Linguistics*, pages 65–68, 2017.
- B. Settles. Active learning literature survey. Computer Sciences Technical Report 1648, University of Wisconsin–Madison, 2009.

- H. S. Seung, M. Opper, and H. Sompolinsky. Query by committee. In *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*, pages 287–294, 1992.
- C. E. Shannon. A mathematical theory of communication. *Bell System Technical Journal*, 27:479–523, 1948.
- S. Shen, Y. Cheng, Z. He, W. He, H. Wu, M. Sun, and Y. Liu. Minimum risk training for neural machine translation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, pages 1683–1692, 2016.
- N. Z. Shor, N. G. Zhurbenko, A. P. Likhovid, and P. I. Stetsyuk. Algorithms of non-differentiable optimization: Development and application. *Cybernetics and Systems Analysis*, 39(4):537–548, 2003.
- K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv:1409.1556*, 2014.
- K. K. Singh, K. Fatahalian, and A. A. Efros. Krishnacam: Using a longitudinal, single-person, egocentric dataset for scene understanding tasks. In *Proceedings of the IEEE Winter Conference on Applications of Computer Vision*, pages 1–9, 2016.
- J. Slocum. A survey of machine translation: its history, current status, and future prospects. *Computational Linguistics*, 11(1):1–17, 1985.
- A. Smith, C. Hardmeier, and J. Tiedemann. Climbing mount BLEU: The strange world of reachable high-BLEU translations. *Baltic Journal of Modern Computing*, 4(2):269, 2016.
- M. Snover, B. Dorr, R. Schwartz, L. Micciulla, and J. Makhoul. A study of translation edit rate with targeted human annotation. In *Proceedings of the Association for Machine Translation in the Americas*, pages 223–231, 2006.
- A. Sokolov, J. Kreutzer, C. Lo, and S. Riezler. Learning structured predictors from bandit feedback for interactive nlp. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, volume 1, pages 1610–1620, 2016a.
- A. Sokolov, J. Kreutzer, S. Riezler, and C. Lo. Stochastic structured prediction under bandit feedback. In *Advances in Neural Information Processing Systems*, pages 1489–1497, 2016b.
- J. Song, L. Gao, Z. Guo, W. Liu, D. Zhang, and H. T. Shen. Hierarchical LSTM with adjusted temporal attention for video captioning. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, pages 2737–2743, 2017.
- A. Spector, L. Thorgrimsen, B. Woods, L. Royan, S. Davies, M. Butterworth, and M. Orrell. Efficacy of an evidence-based cognitive stimulation therapy programme for people with dementia. *The British Journal of Psychiatry*, 183(3):248–254, 2003.

- N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958, 2014.
- E. Strubell, A. Ganesh, and A. McCallum. Energy and policy considerations for deep learning in NLP. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3645–3650, 2019.
- M. Sundermeyer, T. Alkhouli, J. Wuebker, and H. Ney. Translation modeling with bidirectional recurrent neural networks. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 14–25, 2014.
- I. Sutskever, O. Vinyals, and Q. V. Le. Sequence to sequence learning with neural networks. In *Proceedings of the Advances in Neural Information Processing Systems*, volume 27, pages 3104–3112, 2014.
- R. S. Sutton and A. G. Barto. *Introduction to reinforcement learning*, volume 135. MIT press, 1998.
- P. Swietojanski, J. Li, and S. Renals. Learning hidden unit contributions for unsupervised acoustic model adaptation. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 24(8):1450–1463, 2016.
- C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–9, 2015.
- C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2818–2826, 2016.
- C. Szegedy, S. Ioffe, V. Vanhoucke, and A. A. Alemi. Inception-v4, inception-resnet and the impact of residual connections on learning. In *Proceedings of the Association for the Advancement of Artificial Intelligence Conference on Artificial Intelligence*, volume 4, pages 4278–4284, 2017.
- M. Tatsumi. Correlation between automatic evaluation metric scores, post-editing speed, and some other factors. In *Proceedings of the Machine Translation Summit*, pages 332–339, 2009.
- Theano Development Team. Theano: A Python framework for fast computation of mathematical expressions. *arXiv:1605.02688*, 2016.
- B. Thompson, J. Gwinnup, H. Khayrallah, K. Duh, and P. Koehn. Overcoming catastrophic forgetting during domain adaptation of neural machine translation. In

- Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2062–2068, 2019.
- J. Tomás and F. Casacuberta. Statistical phrase-based models for interactive computer-assisted translation. In *Proceedings of the International Conference on Computational Linguistics/Association for Computational Linguistics*, pages 835–841, 2006.
- A. Toral and A. Way. What level of quality can neural machine translation attain on literary text? In J. Moorkens, S. Castilho, F. Gaspari, and S. Doherty, editors, *Translation Quality Assessment: From Principles to Practice*, pages 263–287. 2018.
- A. Toral, S. Castilho, K. Hu, and A. Way. Attaining the unattainable? reassessing claims of human parity in neural machine translation. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, pages 113–123, 2018.
- D. Torregrosa, M. L. Forcada, and J. A. Pérez-Ortiz. An open-source web-based tool for resource-agnostic interactive translation prediction. *The Prague Bulletin of Mathematical Linguistics*, 102:69–80, 2014.
- D. Torregrosa, J. A. Pérez-Ortiz, and M. L. Forcada. Comparative human and automatic evaluation of glass-box and black-box approaches to interactive translation prediction. *The Prague Bulletin of Mathematical Linguistics*, 108(1):97–108, 2017.
- A. Toselli, V. Romero, L. Rodríguez, and E. Vidal. Computer assisted transcription of handwritten text images. In *Proceedings of the Ninth International Conference on Document Analysis and Recognition*, volume 2, pages 944–948, 2007.
- A. H. Toselli, E. Vidal, and F. Casacuberta. *Multimodal interactive pattern recognition and applications*. Springer Science & Business Media, 2011.
- D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri. Learning spatiotemporal features with 3D convolutional networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4489–4497, 2015.
- H. Tseng, P. Chang, G. Andrew, D. Jurafsky, and C. Manning. A conditional random field word segmenter for sighthan bakeoff 2005. In *Proceedings of the fourth SIGHAN workshop on Chinese language Processing*, 2005.
- Z. Tu, Z. Lu, Y. Liu, X. Liu, and H. Li. Modeling coverage for neural machine translation. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, volume 1, pages 76–85, 2016.
- Z. Tu, Y. Liu, L. Shang, X. Liu, and H. Li. Neural machine translation with reconstruction. In *Proceedings of the Thirty-First Association for the Advancement of Artificial Intelligence Conference on Artificial Intelligence*, pages 3097–3103, 2017.

- 
- M. Turchi, M. Negri, M. A. Farajian, and M. Federico. Continuous learning from human post-edits for neural machine translation. *The Prague Bulletin of Mathematical Linguistics*, 108(1):233–244, 2017.
- J. P. Turian, L. Shea, and I. D. Melamed. Evaluation of machine translation and its evaluation. In *Proceedings of the Machine Translation Summit*, pages 386–393, 2003.
- R. Udupa and H. K. Maji. Computational complexity of statistical machine translation. In *Proceedings of the 11th Conference of the European Chapter of the Association for Computational Linguistics*, pages 25–32, 2006.
- N. Ueffing and H. Ney. Application of word-level confidence measures in interactive statistical machine translation. In *Proceedings of the European Association for Machine Translation conference*, pages 262–270, 2005.
- N. Ueffing and H. Ney. Word-level confidence estimation for machine translation. *Computational Linguistics*, 33:9–40, 2007.
- Z. Vakil and S. Khadivi. A new search approach for interactive-predictive computer-assisted translation. *Proceedings of COLING 2012*, pages 1261–1270, 2012.
- V. Vapnik and A. Y. Chervonenkis. On the uniform convergence of relative frequencies of events to their probabilities. *Theory of Probability & Its Applications*, 16(2):264–280, 1971.
- M. Vasconcellos and M. León. SPANAM and ENGSPAN: machine translation at the pan american health organization. *Computational Linguistics*, 11(2-3), 1985.
- A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008, 2017.
- A. Vaswani, S. Bengio, E. Brevdo, F. Chollet, A. N. Gomez, S. Gouws, L. Jones, Ł. Kaiser, N. Kalchbrenner, N. Parmar, et al. Tensor2tensor for neural machine translation. *arXiv:1803.07416*, 2018.
- R. Vedantam, C. Lawrence Zitnick, and D. Parikh. CIDEr: Consensus-based image description evaluation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4566–4575, 2015.
- S. Venugopalan, H. Xu, J. Donahue, M. Rohrbach, R. Mooney, and K. Saenko. Translating videos to natural language using deep recurrent neural networks. *arXiv:1412.4729*, 2014.
- S. Venugopalan, M. Rohrbach, J. Donahue, R. Mooney, T. Darrell, and K. Saenko. Sequence to sequence-video to text. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4534–4542, 2015.

- S. Venugopalan, L. A. Hendricks, R. Mooney, and K. Saenko. Improving LSTM-based video description with linguistic knowledge mined from text. *arXiv:1604.01729*, 2016.
- S. Venugopalan, L. Anne Hendricks, M. Rohrbach, R. Mooney, T. Darrell, and K. Saenko. Captioning images with diverse objects. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5753–5761, 2017.
- D. Vilar. Learning hidden unit contribution for adapting neural machine translation models. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, volume 2, pages 500–505, 2018.
- O. Vinyals, A. Toshev, S. Bengio, and D. Erhan. Show and tell: A neural image caption generator. In *Proceedings of the Conference on Computer Vision and Pattern Recognition*, pages 3156–3164, 2015.
- J. von Neumann. Zur theorie der gesellschaftsspiele. *Mathematische Annalen*, 100(1): 295–320, 1928.
- A. Waibel, T. Hanazawa, G. Hinton, K. Shikano, and K. J. Lang. Phoneme recognition using time-delay neural networks. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 37(3):328–339, 1989.
- L. Wang, Z. Tu, A. Way, and Q. Liu. Exploiting cross-sentence context for neural machine translation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2826–2831, 2017.
- T. Watanabe, J. Suzuki, H. Tsukada, and H. Isozaki. Online large-margin training for statistical machine translation. In *Proceedings of the Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, 2007.
- W. Weaver. Translation. In *Machine Translation of Languages*, pages 15–23. Wiley Periodicals Inc., 1949.
- C. Wendt. Better translations with user collaboration integrated mt at microsoft. In *Proceedings of Ninth Biennial Conference of the Association for Machine Translation in the Americas*, 2010.
- P. J. Werbos. Backpropagation through time: What it does and how to do it. *Proceedings of the Institute of Electrical and Electronics Engineers*, 78(10):1550–1560, 1990.
- P. J. Whitelock, M. M. Wood, B. J. Chandler, N. Holden, and H. J. Horsfall. Strategies for interactive machine translation: the experience and implications of the UMIST

- 
- japanese project. In *Proceedings of the 11th conference on Computational Linguistics*, pages 329–334, 1986.
- B. Widrow. Adaptive switching circuits. In *Western Electric Show and Convention Record*, volume 4, page 96, 1960.
- R. J. Williams and J. Peng. An efficient gradient-based algorithm for on-line training of recurrent network trajectories. *Neural Computation*, 2(4):490–501, 1990.
- R. J. Williams and D. Zipser. A learning algorithm for continually running fully recurrent neural networks. *Neural Computation*, 1(2):270–280, 1989.
- Y. Wu, M. Schuster, Z. Chen, Q. V. Le, M. Norouzi, W. Macherey, M. Krikun, Y. Cao, Q. Gao, K. Macherey, J. Klingner, A. Shah, M. Johnson, X. Liu, Ł. Kaiser, S. Gouws, Y. Kato, T. Kudo, H. Kazawa, K. Stevens, G. Kurian, N. Patil, W. Wang, C. Young, J. Smith, J. Riesa, A. Rudnick, O. Vinyals, G. Corrado, M. Hughes, and J. Dean. Google’s neural machine translation system: Bridging the gap between human and machine translation. *arXiv:1609.08144*, 2016.
- J. Wuebker, S. Green, J. DeNero, S. Hasan, and M.-T. Luong. Models and inference for prefix-constrained machine translation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, volume 1, pages 66–75, 2016.
- J. Wuebker, P. Simianer, and J. DeNero. Compact personalized models for neural machine translation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 881–886, 2018.
- K. Xu, J. Ba, R. Kiros, K. Cho, A. Courville, R. Salakhutdinov, R. Zemel, and Y. Bengio. Show, attend and tell: Neural image caption generation with visual attention. In *Proceedings of the International Conference on Machine Learning*, pages 2048–2057, 2015.
- B. Z. Yao, X. Yang, L. Lin, M. W. Lee, and S.-C. Zhu. I2T: Image parsing to text description. *Proceedings of the IEEE*, 98(8):1485–1508, 2010.
- L. Yao, A. Torabi, K. Cho, N. Ballas, C. Pal, H. Larochelle, and A. Courville. Describing videos by exploiting temporal structure. In *Proceedings of the International Conference on Computer Vision*, pages 4507–4515, 2015.
- Y. Yao, M. Xu, C. Choi, D. J. Crandall, E. M. Atkins, and B. Dariush. Egocentric vision-based future vehicle localization for intelligent driving assistance systems. *arXiv:1809.07408*, 2018.
- O. F. Zaidan and C. Callison-Burch. Predicting human-targeted translation edit rate via untrained human annotators. In *Proceedings of the Annual Conference of the*

- North American Chapter of the Association for Computational Linguistics*, pages 369–372, 2010.
- N. Zeghidour, N. Usunier, G. Synnaeve, R. Collobert, and E. Dupoux. End-to-end speech recognition from the raw waveform. In *Proceedings of the 19th Annual Conference of the International Speech Communication Association*, pages 142–146, 2018.
- M. D. Zeiler. ADADELTA: An adaptive learning rate method. *arXiv:1212.5701*, 2012.
- M. D. Zeiler and R. Fergus. Visualizing and understanding convolutional networks. In *Proceedings of the European Conference on Computer Vision*, pages 818–833, 2014.
- R. Zens, F. J. Och, and H. Ney. Phrase-based statistical machine translation. In *Proceedings of the Annual German Conference on Advances in Artificial Intelligence*, volume 2479, pages 18–32, 2002.
- B. Zhou, A. Lapedriza, J. Xiao, A. Torralba, and A. Oliva. Learning deep features for scene recognition using places database. In *Advances in Neural Information Processing Systems*, pages 487–495, 2014.
- B. Zhou, A. Khosla, A. Lapedriza, A. Oliva, and A. Torralba. Learning deep features for discriminative localization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2921–2929, 2016.
- L. Zhou, Y. Zhou, J. J. Corso, R. Socher, and C. Xiong. End-to-end dense video captioning with masked transformer. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8739–8748, 2018.
- D. Zissis, E. K. Xidias, and D. Lekkas. A cloud based architecture capable of perceiving and predicting multiple vessel behaviour. *Applied Soft Computing*, 35:652–661, 2015.
- B. Zoph, V. Vasudevan, J. Shlens, and Q. V. Le. Learning transferable architectures for scalable image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8697–8710, 2018.