Departamento de Sistemas Informáticos y Computación
Universitat Politècnica de València

Interactive translation with neural models based on the use of mouse actions and confidence measures

MASTER THESIS

Máster en Inteligencia Artificial,
Reconocimiento de Formas e Imagen Digital

Author: Ángel Navarro Martínez

Tutor: Francisco Casacuberta Nolla

Co-tutor: Álvaro Peris Abril

Academic Course 2019-2020

Acknowledgements

I wish to express my sincere appreciation to my tutor, *Dr. Francisco Casacuberta Nolla*, who has guided me during the development of the project and has initiated me into the research world.

As well, I would like to express my gratitude to my cotutor, *Alvaro Peris Abril*, who continually has provided me with helpful information and advice.

Finally, I would like to thank the PRHLT Research Center to let me collaborate with them and provide me with the needed tools to test our results.

Abstract

The state of the art in machine translation is not good enough to be able to guarantee high quality translations at all times. In some fields, it is necessary to always obtain error-free translations, and the tendency is to use the help of a professional to correct them, making use of processes such as post-editing or interactive translation. The usual way of human-machine interaction is through the use of the keyboard and mouse. The user positions the cursor in front of an incorrect word, type the correction, and the system provides a new suffix. In this master's thesis, we are going to develop two extensions to this system: the integration of mouse actions, and the use of confidence measures. These extensions have already been developed previously for statistical models, but have not yet been developed and tested in neural models. Once we develop the extensions in an interactive neural translation system, we will compare the results of the experiments to see the improvement obtained.

In the first proposal, we introduce the use of mouse actions as the only input information to the system to correct the translations. The position of the mouse when correcting a word offers enough information to the system to be able to generate a new suffix without typing in the correction. The translation is correct from the beginning to the position where the user has moved the cursor, also indicating that the next word is incorrect. Furthermore, if the generated suffix is incorrect again, a new correction can be requested, providing the system with the extra information that the next word is incorrect again. Finally, an approach has also been developed where the user can move the cursor in the middle of the words, correcting at the character level.

In the second proposal, we reduce the effort that a translator has to make, by reducing the number of sentences and words that they have to correct. In conventional interactive translation systems, the human translator has to check every sentence and every word. In this extension, the system provides us with an estimation of how correct it thinks the translated words are, and the user only has to check those that do not exceed a certain threshold. This extension has been extended to also provide sentence estimations.

In this master's thesis, these two approaches, that attempt to reduce user effort during the translation session, have been developed. Furthermore, our results using neuronal models have been compared, with those obtained in previous works using statistical models. The results obtained show that there is a greater reduction in the number of words to write when using neural models.

Resumen

El estado del arte en traducción automática aún no es lo suficiente bueno como para ser capaz de garantizar en todo momento traducciones de alta calidad. En algunos campos es necesario obtener siempre traducciones libres de errores, y se tiende a utilizar la ayuda de un profesional que las corrija, haciendo uso de procesos como la posedición o la traducción interactiva. La forma habitual de interacción entre el traductor humano y el software de traducción es mediante el uso del teclado y el ratón. El usuario posiciona el cursor delante de la palabra incorrecta, teclea la corrección, y el sistema proporciona un nuevo sufijo. En esta tesis de máster vamos a desarrollar dos ampliaciones a este sistema: la integración de las acciones del ratón, y el uso de medidas de confianza. Estas ampliaciones ya han sido desarrolladas previamente para modelos estadísticos, pero aún no han sido desarrolladas y probadas en modelos neuronales. Una vez desarrollemos las ampliaciones en un sistema de traducción neuronal interactiva compararemos los resultados de los experimentos para ver la mejora obtenida.

En la primera propuesta planteamos la utilización de las acciones del ratón como único valor de entrada al sistema para corregir las traducciones. La posición del ratón al corregir una palabra, ofrece suficiente información al sistema como para poder generar un nuevo sufijo sin que el usuario llegue a teclear la corrección. La traducción es correcta desde el inicio, hasta la posición donde el usuario ha movido el cursor, indicando además que la siguiente palabra es incorrecta. Además, en el caso de que el sufijo generado vuelva a ser incorrecto, se puede pedir una nueva corrección, proporcionado al sistema la información extra de que la siguiente palabra vuelve a ser incorrecta. Finalmente, también se ha desarrollado una aproximación donde el usuario puede mover el cursor en medio de las palabras, realizando una corrección a nivel de carácter.

En la segunda propuesta reducimos el esfuerzo que tiene que realizar un traductor, al disminuir la cantidad de oraciones y palabras que tiene que corregir. En los sistemas convencionales de traducción interactiva el traductor humano tiene que comprobar todas las oraciones y cada una de las palabras. En esta ampliación, el sistema nos proporciona una estimación sobre como de correctas cree que son las palabras traducidas, y el usuario solamente tiene que comprobar aquellas que no superen un cierto umbral. Esta ampliación ha sido extendida para también poder proporcionar estimaciones de las oraciones.

En este trabajo de fin de máster se han desarrollado estas dos aproximaciones que intentan reducir el esfuerzo del usuario durante la sesión de traducción. Además, se han comparado nuestros resultados haciendo uso de modelos neuronales, con los obtenidos en trabajos anteriores que utilizaron modelos estadísticos. Los resultados obtenidos demuestran que hay una mayor reducción en la cantidad de palabras a escribir al utilizar modelos neuronales.

Resum

L'estat de l'art en traducció automàtica encara no és prou bo per a ser capaç de garantir en tot moment traduccions d'alta qualitat. En alguns camps és necessari obtindre sempre traduccions lliures d'errors, i es tendeix a utilitzar l'ajuda d'un professional que les corregisca, fent ús de processos com la postedició o la traducció interactiva. La forma habitual d'interacció entre el traductor humà i el programa de traducció és mitjançant l'ús del teclat i el ratolí. L'usuari posiciona el cursor davant de la paraula incorrecta, tecleja la correcció, i el sistema proporciona un nou sufix. En aquesta tesi de màster desenvoluparem dues ampliacions a aquest sistema: la integració de les accions del ratolí, i l'ús de mesures de confiança. Aquestes ampliacions ja han sigut desenvolupades prèviament per a models estadístics, però encara no han sigut desenvolupades i provades en models neuronals. Una vegada desenvolupem les ampliacions en un sistema de traducció neuronal interactiva compararem els resultats dels experiments per a veure la millora obtinguda.

En la primera proposta plantegem la utilització de les accions del ratolí com a únic valor d'entrada al sistema per a corregir les traduccions. La posició del ratolí al corregir una paraula, ofereix suficient informació al sistema com per a poder generar un nou sufix sense que l'usuari arribe a teclejar la correcció. La traducció és correcta des de l'inici, fins a la posició on l'usuari ha mogut el cursor, indicant a més que la següent paraula és incorrecta. A més, en el cas que el sufix generat torne a ser incorrecte, es pot demanar una nova correcció, proporcionant al sistema la informació extra del fet que la següent paraula torna a ser incorrecta. Finalment, també s'ha desenvolupat una aproximació on l'usuari pot moure el cursor enmig de les paraules, realitzant una correcció a nivell de caràcter.

En la segona proposta reduïm l'esforç que ha de realitzar un traductor al disminuir la quantitat d'oracions i paraules que ha de corregir. En els sistemes convencionals de traducció interactiva el traductor humà ha de comprovar totes les oracions i cadascuna de les paraules. En aquesta ampliació, el sistema ens proporciona una estimació sobre com de correctes creu que són les paraules traduïdes, i l'usuari solament ha de comprovar aquelles que no superen un cert valor. Aquesta ampliació ha sigut estesa per a també poder proporcionar estimacions de les oracions.

En aquest treball de fi de màster s'han desenvolupat aquestes dues aproximacions que intenten reduir l'esforç de l'usuari durant la sessió de traducció. A més, s'han comparat els nostres resultats fent ús de models neuronals, amb els obtinguts en treballs anteriors que van utilitzar models estadístics. Els resultats obtinguts demostren que hi ha una major reducció en la quantitat de paraules a escriure a l'utilitzar models neuronals.

Contents

List of Figures

List of Tables

Overview

This master's thesis focus on the application of different techniques about interactive machine translation applied to a neural machine translator, to decrease the effort required by the human translator to interactively correct the sentences with these systems. The thesis is structured as follows:

- *Chapter 1* begins by introducing machine translation, and move through similar disciplines that lead us to the interactive translation and the multiple approaches to improve it.
- *Chapter 2* explains the various implementations and tests done to improve the interactive translation system, trying to reduce the effort of the human translator.
- *Chapter 3* presents the results collected in our experiments and compare them with the obtained by other authors using statistical models.
- *Chapter 4* contains the conclusions of the thesis and proposes some future work to extend the content of the project.

Chapter 1

1 Introduction

Language has always been an essential aspect of our lives. It helps us to communicate with the people, allowing us to share information and ideas. Nowadays, there are more than 6000 different languages around the world, and we still have the necessity of sharing information among all of us, even with this huge mixture of languages.

Since ancient times, the translators had the arduous work to translate a huge number of documents. In additions, there was not always a translator for every pair of languages. The first modern approach to Machine Translation (MT) was proposed by Warren Weaver the year 1949, after the successes in code-breaking achieved during the Second World War.

The next years, lots of money and efforts were put in the research on MT. It was thought that in a few years, they would get fully automatic high quality translations, but that was not the case. In the year 1966, the ALPAC (Committee, 1966) published a report where concluded that MT was more expensive, less accurate, and slower than human translation and was not likely to reach the high quality wanted soon.

Despite all, some groups continued with the research in that field. In the last years, the different approaches to MT have greatly varied, from the old rule-based systems, to the new neural-based systems. But even nowadays, despite all improvements made, machine translations are not perfect. And, in most cases, a professional translator is required to obtain high quality translations.

In 1997, the Canadian government started a new project focused on Interactive Machine Translation (IMT), TransType (Langlais et al., 2000). In this field, the main objective is to decrease the effort needed of the translator to correct the sentences interactively with the computer. Since then, has been made new approaches in IMT, achieving many improvements. Section 1.7 and 1.8 explains some of these approaches.

1.1 Machine Translation

The main goal of MT is to translate text from a source language to the target language utilizing a computer. The first MT systems proposed, took a simplistic view, translating every word of the corpus, and reordering it following a set of rules of the target language. Since then, after more than 50 years of research, and new approaches to the problem, the machine translations are not perfect yet.

The different approaches to the MT problem can be classified according to different criteria (Ortiz Martínez, 2011):

- Based on the input type: text or speech.

- Based on the character of the application used by the translator. These applications are divided into four groups: applications to translate the input into a database query; applications to produce an approximated translation for its later correction in post-edition; applications to interactively generate the output in collaboration with the user; and finally, fully automated translation systems.

- Based on the translation technology. We can identify two main approaches: rule-based systems and corpus-based systems. The rule-based systems consist of a set of translation rules created by a human expert that denote how to translate from one language to another. This process has a high cost due to the knowledge needed of both languages. In recent years, a new approach appeared and gained relevance, the corpus-based systems. These, only need a parallel text to extract all the information needed by the system to generate the translations, taking away the cost of the human expert.

The project focuses on corpus-based systems. And, in Section 1.2 and 1.3, the two main approaches are explained.

1.2 Statistical Machine Translation

Statistical machine translation (SMT) is a corpus-based approach to MT, that search for patterns in large amounts of parallel texts, to be able to assign the probability of a sentence from the target language, of being the translation of another sentence from the source language.

More formally, given a sentence $x_1^J = x_1, ..., x_J$ from the source language X. To find the sentence $\hat{y}_1^{\hat{I}} = \hat{y}_1, ..., \hat{y}_{\hat{I}}$, from a target language Y, that has the highest probability of being the translation of x_1^J, we could compute it according to:

$$\hat{y}_1^{\hat{I}} = \arg\max_{I, y_1^I} \Pr(y_1^I | x_1^J) \tag{1.1}$$

In practice, the probability distribution $\Pr(y_1^I | x_1^J)$ is calculated through a log-linear combination between an array of models (Och and Ney, 2004):

$$\hat{y}_1^{\hat{I}} = \arg\max_{I, y_1^I} \left\{ \sum_{n=1}^{N} \lambda_n \cdot \log f_n(y_1^I, x_1^J) \right\} \tag{1.2}$$

where $f_n(y_1^I, x_1^J)$ can be the language model $\Pr(y_1^I)$, the translation model $\Pr(x_1^J | y_1^I)$, or any model that represents a significant feature for the translation. N is the number of different models or features used, and λ_n is the weight of each feature for the log-linear combination.

There are three main challenges on the probabilistic modelling that SMT faces:

- **Model definition**: The development of models that approximate the best possible the translation probability distribution $\Pr(y_1^I|x_1^J)$.
- **Parameter estimation**: The estimation of the parameters of the models defined through the available data, that usually is parallel texts.
- **Search problem:** The search through all the possibilities, to find the translation with the highest probability. Most systems tackle the problem via suboptimals, but fast search algorithms.

1.3 Neural Machine Translation

The first ideas of MT using neural networks date back to the 90s (Castaño and Casacuberta, 1997). However, the results of these works were disappointing and not further explored. Recently, the first successful Neural Machine Translation (NMT) experiments have been done (Cho et al., 2014; Sutskever et al., 2014b) , and actually is used in almost all the MT systems (Wu et al., 2016; Klein et al., 2017) . Most of them have dealt with the problem as a sequence-to-sequence transduction task (Graves, 2013), using a neural encoder-decoder model for building MT systems.

NMT follows the same idea used by SMT to solve the translation problem. Ergo, it tries to generate the sentence $\hat{y}_1^{\hat{I}}$ with the highest translation probability for the source sentence x_1^J. This is:

$$\hat{y}_1^{\hat{I}} = \arg\max_{I, y_1^I} \Pr(y_1^I|x_1^J) \tag{1.3}$$

Applying the chain rule of the probability, we can factorize this expression into:

$$\hat{y}_1^{\hat{I}} = \arg\max_{I, y_1^I} \prod_{i=1}^{I} \Pr(y_i \mid y_1^{i-1}, x_1^J) \tag{1.4}$$

We can model the conditional probability using a neural model with parameters Θ. Note that we are taking logarithms for the sake of numerical stability:

$$\hat{y}_1^{\hat{I}} \approx \arg\max_{I, y_1^I} \sum_{i=1}^{I} \log \mathrm{p}(y_i \mid y_1^{i-1}, x_1^J; \hat{\Theta}) \tag{1.5}$$

The value of these parameters Θ are obtained from trying to minimize the minus log-likelihood, on a set of a parallel corpus $S = \{x^{(s)}, y^{(s)}\}_{s=1}^S$, consisting of S sentence pairs.

$$\hat{\Theta} = \arg\min_{\Theta} \sum_{s=1}^{S} \sum_{i=1}^{I^{(s)}} -\log \mathrm{p}(y_i^{(s)} \mid y_1^{i-1(s)}, x_1^{(s)}; \Theta) \tag{1.6}$$

NMT solves the SMT problems using only one big neural network. The three main challenges that we found before are addressed as follows:

- **Model definition**: The neural networks used in NMT directly approximate the probability distribution $Pr(y_1^I|x_1^J)$.
- **Parameter estimation**: All the parameters of the neural network are trained jointly through gradient descent, trying to minimize the minus log-likelihood as seen in Equation 1.6.
- **Search problem:** Most NMT systems use the method called beam search to find the best translation.

1.3.1 Search Problem

The search problem is built on the idea of how to generate the highest probability translation sentence $\hat{y}_1^{\hat{I}}$, from the source sentence x_1^J, with the parameters Θ from the neural model. The vast majority of search methods exploit the factorization of the conditional probability, shown in the Equation 1.5.

Usually, the target sentence is generated incrementally, starting from a null-hypothesis (a hypothesis with no words) initialized with the beginning-of-sentence token (<**bos**>). Step by step, the words are being added to the hypothesis until the end-of-sentence token is added (<**eos**>). The neural model provides a score to each one of the words from the search tree. At the end of the process, we have a search tree with all the possible translations, where each hypothesis path starts from the root (marked as <**bos**>) and ends in a leaf (marked as <**eos**>). Hence, the goal is to find the path with the highest score from the root to a leaf.

Each word of the tree branches to V new paths, where V is the size of the vocabulary. The huge size of the search tree generated keeps off the possibility of making an exhaustive search. To tackle this problem we used a method called beam search (Lowerre and Reddy, 1976). This approach limits the branching factor to a maximum predefined value, called size of the beam *(b)*. At each level of the tree, the partial hypothesis set is expanded with all the vocabulary, but only the new b hypotheses with the highest score continue in this set. If a complete hypothesis is generated, it is a moved to a set of completed hypothesis, and the beam size is decreased by one. This process is repeated until the size of the beam reaches zero. Then, the method returns the most probable hypothesis from the set of completed ones.

With this method, we solve the computational high cost problem of the search, besides having the control of the cost with the parameter b. Although, we risk getting suboptimal translations by not fully extending the tree.

1.3.2 Subword NMT

The size of the vocabularies is a limiting aspect of NMT: MT is an open-vocabulary task, while the NMT models require finite vocabularies. An interesting idea is the use of subwords: instead of translating sequences of words, we can translate sequences of subwords. In this way, an unknown word can be

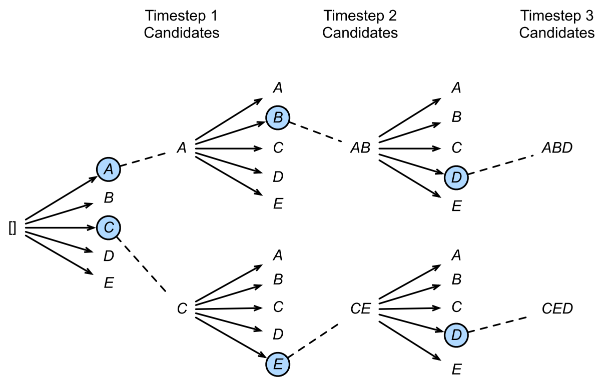


Figure 1.1: Beam Search with a Beam size of 2. At each level of the tree, despite being generated five new paths, only two of them are continued.

split into multiple known subwords for its translation. The Byte Pair Encoding (BPE) (Sennrich et al., 2016) algorithm is very adequate for this purpose.

BPE is a data compression technique that iteratively replaces the most frequent pair of bytes in a sequence with a single, unused byte. This technique is adapted for word segmentation, instead of merging frequent pairs of bytes, it merges frequent sequences of characters. At the start, the method splits each word of the vocabulary into a list of characters, plus a special end-of-word symbol. Each different character is a symbol of the new vocabulary. At each iteration, the method merges the two most frequent consecutive character sequences in a new symbol. The final symbol vocabulary size, is equal to the size of the initial vocabulary, plus the number of merge operations done. BPE obtains different granularities of the representation of words: Rare words tends to be represented as sequences of multiple subwords, due to its low frequency. While most common words could tend to be represented without segmentation.

Although BPE decreases the likelihood of finding out-of-vocabulary words, it can still happen. Even so, the general strategies for replacing unknown words can still be used (Luong et al., 2015b,a).

$$\begin{array}{c}
\text{r} \quad . \rightarrow \text{r.} \\
\text{l} \quad \text{o} \rightarrow \text{lo} \\
\text{lo} \quad \text{w} \rightarrow \text{low} \\
\text{e} \quad \text{r.} \rightarrow \text{er.} \\
\text{l} \quad \text{o} \quad \text{w} \quad \text{e} \quad \text{s} \quad \text{t} \quad \text{n} \quad \text{r} \quad \text{i} \quad \text{r.} \quad \text{lo} \quad \text{low} \quad \text{er.}
\end{array}$$

Figure 1.2: BPE merge operations learned from dictionary {'low', 'lowest', 'newer', 'wider'}, and final symbol vocabulary.

1.4 Neural Network Architectures

During the years, a large variety of neural networks architectures have been designed to tackle different problems. Our problem is of the kind sequence-to-sequence, as we are trying to get a translation from a source sentence. In this section, we review the main architectures presented in the system used for our project.

1.4.1 Recurrent Neural Network

The main feature of the Recurrent Neural Networks (RNN) is that between its connections we can find directed cycles. This allows the system to have an internal hidden-state, of size h, that will change recurrently with the time steps. Given a sequence of inputs $x_1^T = x_1, ..., x_T$, with each $x \in \mathbb{R}^x$, a standard RNN (Sutskever et al., 2014a) computes a sequence of outputs $y_1^T = y_1, ..., y_T$, with each $y \in \mathbb{R}^y$ by iterating the following equation:

$$\begin{aligned} h_t &= \text{sigm}(W^{hx} x_t + W^{hh} h_{t-1} + b^h) \\ y_t &= W^{yh} h_t + b^y \end{aligned}$$

where W^{hx}, W^{hh} and W^{yh} are the input-to-hidden, hidden-to-hidden and hidden-to-output weight matrices to estimate, b are the bias terms and 'sigm' is the sigmoid activation function. Despite being a powerful sequence modeller, it has some drawbacks that are corrected in other approaches (see Section 1.4.1 and 1.4.1).

Bidirectional recurrent neural networks

One of the downsides of RNN is that the input sequence is only used in one direction, generally from the left to the right. However, Schuster and Paliwal (1997) proposed an architecture that reads the sequence in both directions with a forward and backward RNN, the Bidirectional RNN (BRNN).

The forward $\overrightarrow{\text{RNN}}$ reads the input sequence as normally, $x_1^T = x_1, ..., x_T$, and calculates a sequence of forward hidden-states $\overrightarrow{h}_1^T = \overrightarrow{h}_1, ..., \overrightarrow{h}_T$. The backward $\overleftarrow{\text{RNN}}$ reads the sequence from right to left , $x_T^1 = x_T, ..., x_1$, and calculates a sequence of backward hidden states $\overleftarrow{h}_1^T = \overleftarrow{h}_1, ..., \overleftarrow{h}_T$.

Each element of the output sequence $y_1^T = y_1, ..., y_T$ is calculated by combining the forward hidden-states $\overrightarrow{h_t}$ with the backward hidden-states $\overleftarrow{h_t}$. The most common combination strategy is to concatenate them, although others can also be used (e.g. summation or averaging).

$$y_t = \overrightarrow{h_t} \oplus \overleftarrow{h_t}$$

Long short-term memory units

The other problem that has the general version of RNN is the called vanishing gradient (Bengio et al., 1994), which makes difficult network training when modelling long-term relationships. The Long Short-Term Memory units (LSTM) introduced by Hochreiter and Schmidhuber (1997) is one of the most popular recurrent architectures and solves this problem.

In addition to the hidden-state $h_t \in \mathbb{R}^h$ seen in the RNN, the LSTM cells also save a new internal vector $c_t \in \mathbb{R}^h$ called memory. Each cell has an input, forget and output gate whose output activation vectors are Γ_t^f, Γ_t^i, $\Gamma_t^o \in \mathbb{R}^h$. These gates module the information that flows through the cell.

In this approach, the hidden-state is calculated from the memory of the cell, and the vector from the output gate. A Hadamard product is done between both of them, also called element-wise product:

$$h_t = c_t \otimes \Gamma_t^o \tag{1.7}$$

With the memory state from the last time-step and the output vectors from the forget and input gate, we can calculate the new memory state, used in the previous equation.

$$c_t = \Gamma_t^f \otimes c_{t-1} \oplus \Gamma_t^i \otimes \bar{c_t} \tag{1.8}$$

$$\bar{c_t} = \tanh\left(W^{hh} h_{t-1} + W^{hx} x_t + b^h\right) \tag{1.9}$$

where W^{hh} and W^{hx} are the hidden-to-hidden and input-to-hidden weight matrices; b^h is the bias term and 'tanh' is the hyperbolic tangent activation function.

The forget, input and output activation gates are calculated as follows:

$$\Gamma_t^f = \mathrm{sigm}(W_f^{hx} x_t + W_f^{hh} h_{t-1} + b_f^h) \tag{1.10}$$

$$\Gamma_t^i = \mathrm{sigm}(W_i^{hx} x_t + W_i^{hh} h_{t-1} + b_i^h) \tag{1.11}$$

$$\Gamma_t^o = \mathrm{sigm}(W_o^{hx} x_t + W_o^{hh} h_{t-1} + b_o^h) \tag{1.12}$$

where 'sigm' is the sigmoid activation function; W_f^{hx}, W_i^{hx} and W_o^{hx} are the input-to-hidden weight matrics of each gate; and b_f^h, b_i^h and b_o^h are the bias terms.

1.4.2 Attention Mechanisms

Attention mechanisms (Chorowski et al., 2015) have become an essential component of neural architectures, especially in the context of NMT using sequence-to-sequence models. Generally, the sequence-to-sequence models were composed of an encoder-decoder architecture, where the encoder processed the input into a context vector of a fixed length (Sutskever et al., 2014b). Then, the decoder is initialized with this context vector and starts generating the output.

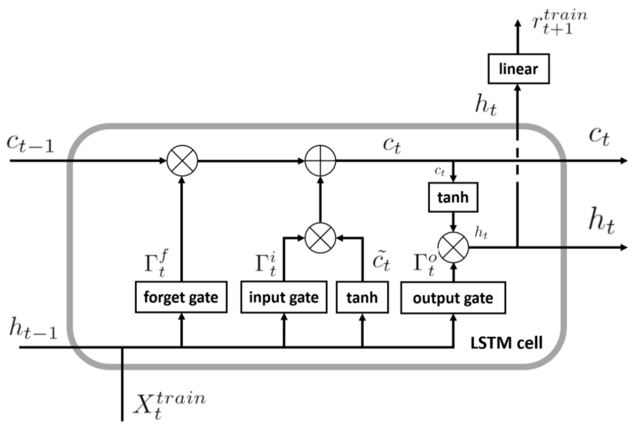


Figure 1.3: LSTM cell. The output (c_t, h_t) depends on the input and, the hidden-state and memory from the last time-step. The information is filtered through the forget, input and output gates.

The main problem of using a fixed-length context vector is that it has to remember long sequences. Once it has processed the entire sequence, is very common that the earlier parts of it have been forgotten and do not show in the last hidden-state. The attention mechanisms were created to solve this problem, using a dynamic context vector.

Generalizing this concept, given an input sequence $x_1^n = x_1, ..., x_n$, the attention mechanism generates a representation sequence $z_1^n = z_1, ..., z_n$. This representation is contextualized to focus on those elements that are more important for the current time-step.

For this purpose, we use a state vector, called *query*, for weighing the input sequence x_1^n. Both objects are related by means of an attention function, that calculates the compatibility scores for each one of the elements from the input sequence. Finally, these scores are used to compute a weighted average of the elements of the sequence, z_1^n.

1.5 Computer Assisted Translation

Nowadays, the MT systems are still not able to generate translations with high quality, despite all the advances achieved and the big data-sets of parallel texts procured. There are domains where is strictly necessary that all the translations are error-free, in these cases is needed an expert to assist the translation process.

In the beginning, the users received a set of machine-translated sentences and had to correct them by themselves. Over time and technological improvements, the interaction human-machine increased cause of the development of an array of new tools that aimed to assist the human in the translation process.

Computer-Aided Translation (CAT) (Gambier and Doorslaer, 2010) aims to use computer software to assist a human expert to generate translations. In this case, the translation process is carried principally by a human, not like in MT, where the human only is involved in the pre- or post-editing.

Since 1960, after realizing the difficulty of getting high quality translations from a fully automatic MT, researchers have been working on CAT. With the rapid evolution of technology, the tools became more accessible, affordable, popular and even necessary. Some of the most used tools are:

- **Translation Memories**: Translation Memories (TM) software is the most well-known CAT tool. TM divides the text to be translated into segments, and lookup for similar segments already translated in its database. Finally, when it finds a similar segment it suggests to reuse it.
- **Language Search-Engines**: Language search-engines work like traditional search engines, but without using the Internet to search the result. It seeks the results of the search in a large database of translation memory, finding similar fragments of previously translated texts.
- **Terminology Managers**: Terminology management software let the user manage their terminology bank, with the ability to automatically search for these terms and check whether have been translated correctly.
- **Aligners**: Aligners build a parallel corpus from the source and destination documents. The tool divides the text into segments, and determine which segments match with each other.
- **Post-Editors**: Post-Edition software provides the user with a first machine-translated output from the text to translate. Domain specific models are used to get better results.
- **Interactive Machine Translation**: Interactive Machine Translation, is a paradigm where the human translator interacts with the predictions done by the computer to generate the final translations. There are different approaches to IMT like Interactive-predictive machine translation and Confidence Measures.

1.6 Interactive Machine Translation

Interactive Machine Translation (IMT) is a sub-field of CAT where translator and machine software work interactively. The MT system generates a hypothesis with the available information. Whenever it is wrong, the user provides feedback to the system, and a new hypothesis is generated.

One of the most recent projects in this field is *Casmacat* (Alabau et al., 2013, 2014), funded by *the European Commission*. It aimed to be a working environment for translators with an array of innovative features that were not available in other tools. It included and combined an innovative set of IMT features: Intelligent autocompletion (Barrachina et al., 2009b), Confidence measures (González-Rubio et al., 2010), Prediction length control (Alabau et al., 2012), Search and Replace, Word alignment information (Brown et al., 1993) and Prediction rejection (Sanchis-Trilles et al., 2008b).

This sub-field of CAT encompasses many different techniques. With the improvement in the MT models, the effort that the translator needs to do will decrease because of the higher quality translations. Besides all those improvements, human interaction plays a very important role, so techniques or features that speed up the work also are very useful.

In this project, we focus on reducing the word strokes that the user has to do in order of correcting the translation. On the one hand, we generate new translations before the user types the correct word, eliminating the writing cost in the case of getting it right (see Section 2.1). On the other hand, we use confidence measure to select which words need checking and which are presumably correct (see Section 2.2).

1.7 Interactive-Predictive Machine Translation

Interactive-Predictive Machine Translation (IPMT) was firstly introduced by (Barrachina et al., 2009a) following the TransType project ideas and formalizing the interactive-predictive translation framework under a statistical point of view.

As previously mentioned, translations done with the current models, and therefore the systems, are still far from perfect. One alternative to the post-editing could be this approach, the IPMT paradigm. Under this paradigm, translations are considered as an iterative process between translator and computer. This way, the model takes into account the input sentence and the corrections done by the user. Generally, this interaction is a left to right process, although other kinds of interactions are possible.

SOURCE (x):		Para encender la impresora:
REFERENCE (y):		To power on the printer:
ITER-0	($\mathbf{p}$)	( )
	($\hat{s_h}$)	*To switch on:*
ITER-1	($\mathbf{p}$)	To
	(s_t)	*switch on:*
	(k)	power
	($\hat{s_h}$)	*on the printer:*
ITER-2	($\mathbf{p}$)	To power on the printer:
	(s_t)	( )
	(k)	(#)
	($\hat{s_h}$)	( )
FINAL	($p \equiv y$)	To power on the printer:

Figure 1.4: IPMT session to translate a Spanish sentence into English. Non-validated hypotheses are displayed in italics, whereas accepted prefixes are printed in normal font.

Figure 1.4 illustrates a typical IPMT session. Initially, the user is provided with a source sentence x to be translated. The reference y, is the correct translation of the sentence that the user will try to obtain. At iteration 0, the IPMT system provides the first hypothesis $\hat{s_h}$, and the prefix $\mathbf{p}$ remains empty cause the user still does not supply any corrections. At the next iteration, the user validates the prefix $\mathbf{p}$ as correct, positioning the cursor in a certain position of s, and corrects the next word by typing k. With this correction, the IPMT system provides a new suffix $\hat{s_h}$, that the user will have to validate in the next iterations. The process continues until the whole sentence is correct and is validated introducing the special token '#'.

1.7.1 Neural Framework

Following the works that introduced the Interactive Neural Machine Translation Knowles and Koehn (2016); Peris et al. (2017). The paths on the search tree mentioned in Section 1.3.1, are generated from left to right. In the same way, this kind of feedback interaction, validates the sentences from left to right, doing natural the inclusion of IPMT into NMT systems. Given a translation hypothesis $\hat{y}_1^{\hat{I}} = \hat{y}_1, ..., \hat{y}_{\hat{I}}$, the validated part with the correction made by the user has the form $f = \hat{y}_1^i$. Given a prefix $\hat{y}_1^i$, only a single path from the search tree can include it. Introducing the user feedback $f = \hat{y}_1^i$, Equation 1.5 can be reformulated as next to get the suffix $\hat{y}_{i+1}^{\hat{I}}$:

$$\hat{y}_{i+1}^{\hat{I}} \approx \underset{I, y_{i+1}^I}{\arg\max} \sum_{i'=i+1}^{I} \log \mathrm{p}(y_{i'} \mid y_{i+1}^{i'-1}, x_1^J, f = \hat{y}_1^i; \Theta) \tag{1.13}$$

As well, if we want a more general Equation to generate another hypothesis from the root of the search tree, we can tweak it, adding the Kronecker delta function.

$$\hat{y}_1^{\hat{I}} \approx \underset{I, y_1^I}{\arg\max} \sum_{i'=1}^{I} \log \mathrm{p}(y_{i'} \mid y_1^{i'-1}, x_1^J, f = \hat{y}_1^i; \Theta) \tag{1.14}$$

the probability distribution can be expressed as follows:

$$\mathrm{p}(y_{i'} \mid y_1^{i'-1}, x_1^J, f = \hat{y}_1^i; \Theta) = \begin{cases} \delta(y_{i'}, \hat{y_{i'}}), & \textit{if } i' \leq i \\ \mathrm{p}(y_{i'} \mid y_1^{i'-1}, x_1^J, f = \hat{y}_1^i; \Theta), & \text{otherwise} \end{cases} \tag{1.15}$$

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

$$\delta(y_{i'}, \hat{y_{i'}}) = \begin{cases} 1, & y_{i'} \equiv \hat{y_{i'}} \\ 0, & \text{otherwise} \end{cases} \tag{1.16}$$

This process assures that the validated prefix is present in the search tree, and can be seen as generating the most probable suffix.

1.8 Confidence Measures

The other IMT approach for MT that we use in the project is the Confidence Measures (CM). Normally, the user has to check each of the words, from the hypothesis generated by the machine, to correct them. This approach reduces the effort that the user needs to do by decreasing the number of words that have to check per sentence.

Confidence estimation has been extensively studied for speech recognition, and recently introduced to MT (Blatz et al., 2004; Ueffing et al., 2003). Since then, research has studied which models calculate the most reliable confidence estimations (Ueffing and Ney, 2005b; Sanchis et al., 2007). CM can be used, among other applications, within IPTM systems (González-Rubio et al., 2010; González-Rubio et al., 2010). The main feature of these systems is the human-machine interactivity, for this reason, the model used to obtain the confidence scores should get them the fastest possible without interrupting the interaction. In this project, we used the IBM Model 1(see Section 2.2).

From our point of view, we can see CM as a conventional pattern classification problem, where each word of a given translation is classified whether correct or incorrect. Picked a word classification threshold, all the words with a confidence estimation below it are classified as incorrect. Although individual words are more likely to be correct than are whole sentences, this process can be extrapolated to a sentence level (see Section 2.2.2).

SOURCE (x):		Para encender la impresora:
REFERENCE (y):		To power on the printer:
ITER-0	(**p**)	( )
	($\hat{s_h}$)	*To switch on* $\underline{a}$ *printer:*
ITER-1	(**p**)	To switch on
	(s_t)	*a printer*
	(k)	the
	($\hat{s_h}$)	*printer:*
ITER-2	(**p**)	To switch on the printer:
	(s_t)	( )
	(k)	(#)
	($\hat{s_h}$)	( )
FINAL	(**p**)	To switch on the printer:

Figure 1.5: IPMT session to translate a Spanish sentence into English. Non-validated hypotheses are displayed in italics, whereas accepted prefixes are printed in normal font. Words classified as incorrect are displayed underlined.

Figure 1.5 illustrates a typical IPMT session with CM. Initially, the user is provided with a source sentence x to be translated. The reference y, is the correct translation of the sentence that the user will try to obtain. At iteration 0, the IPMT system provides the first hypothesis $\hat{s}_h$, and the prefix p remains empty cause the user still does not supply any correction. At the next iteration, the system requests a correction by the user of the next word marked as incorrect, *'a'*, and the user corrects it by typing the word k. With this correction, the IPMT system provides a new suffix $\hat{s}_h$. The process continues until the system thinks that does not remain more incorrect words, and the user validated it.

As seen in Figure 1.5, although the obtained sentence is correct, it is different from the reference. One main problem of CM applied to the IPMT is that we have no longer assure perfect translations. Instead, we have to find the CM model and threshold value that best suits our case, reducing the human effort without giving up the quality of the translations.

1.9 Conclusion

In this chapter, we have seen how started the machine translation field, and how once researchers figured that it was very difficult to obtain high-quality translations, started to investigate on how to improve the human-machine interaction in the translation process. Until now, there is a good range of features that could be implemented and combined in an IMT tool to reduce the effort done by the user, but we focus on IPMT and CM systems.

In the next chapter, we are going to see how are implemented in an INMT model two different approaches. The first of them being the introduction of mouse actions into IPMT systems, and the second, the use of IBM Model 1 in a CM system.

Chapter 2

2 Introducing mouse actions and confidence measures in INMT

As said previously in Section 1.5, despite all the improvements done in MT, the current state of the art is far away from getting high-quality translations. Because of this, the research on IMT has increased, and new approaches have appeared.

In recent years, new approaches to this paradigm have been researched, and the vast majority of them were developed using SMT models. Due to how approximately recent are neural models, most of these approaches have not been tried on them.

In this project we are going to implement two approaches on a Interactive Neural Machine Translation (INMT) system. On one hand, we implemented the most basic kind of human interaction, the mouse action (see Section 2.1). On the other, we added a CM, calculated with the IBM Model , to a IPMT based on NMT (see Section 2.2).

2.1 Mouse Actions in IPMT

Generally, the feedback that the IPTM systems receive is the correction of one word of the sentence provided (see Section 1.7). To give this feedback, the professional human must have moved the mouse to the position with the error, clicked and wrote the correct word.

Sanchis-Trilles et al. (2008b) proposed to enrich the user-machine interaction by introducing Mouse Actions (MA) as an additional information source for the system. We will consider two types of MAs, called non-explicit (or positioning) MAs (see Section 2.1.1) and interaction-explicit MAs (see Section 2.1.2). The main feature of both is that they just use actions that the expert can do with the mouse, by positioning and clicking, with the assumption that these interactions have a lower cost than writing the correct word.

2.1.1 Non-Explicit Mouse Actions

In IPTM systems, before typing a new word to correct a hypothesis, the user needs to position the cursor in place to change it. By doing so, the user already is providing very useful information to the system: he is validating a prefix, since the start until the mouse position, and at the same time he is marking the next word as incorrect.

As said, usually, the user would have to type the correct word, but just with this information provided with the positioning of the mouse, we can generate a new hypothesis. This new hypothesis has the same corrected prefix, and the next word is changed to another different. Note that this does not mean the next hypothesis will be correct, but in the worst case, the user only would have to type the correct word as usually, as the mouse is already in position.

This kind of MA is called non-explicit because it does not require any additional action from the user: to correct a word he has to position the mouse in the place he wants, and we are taking advantage of this action suggesting a new suffix hypothesis.

SOURCE (x):		Para encender la impresora:
REFERENCE (y):		To power on the printer:
ITER-0	($\mathbf{p}$)	( )
	($\hat{s_h}$)	*To switch on:*
ITER-1	($\mathbf{p}$)	To
	(s_t)	‖ *switch on:*
	($\hat{s_h}$)	*power on the printer:*
ITER-2	($\mathbf{p}$)	To power on the printer:
	(s_t)	( )
	(k)	(#)
	($\hat{s_h}$)	( )
FINAL	($\mathbf{p} \equiv \mathbf{y}$)	To power on the printer:

Figure 2.1: IPMT session to translate a Spanish sentence into English using *non-explicit* Mouse Actions. Non-validated hypotheses are displayed in italics, whereas accepted prefixes are in normal font. The MAs are indicated by the symbol '‖'.

Figure 2.1 illustrates a typical IPMT session where a non-explicit MA is done. Initially, the user is provided with a source sentence x to be translated. The reference y, is the correct translation of the sentence that the user will try to obtain. At iteration 0, the IPMT system provides the first hypothesis $\hat{s}_h$, and the prefix p remains empty cause the user still does not supply any correction. At the next iteration, the user position the cursor before word *'switch'*, with the purpose of typing in *'power'*. By doing so, the user is validating the prefix *'To'*, and signalling that the word *'switch'* is incorrect. Before typing in anything, the system provides a new suffix $\hat{s}_h$. Finally, the user only has to accept the last translation as the system has corrected it correctly.

This scenario is very similar to the explained in Section 1.7.1, where the user inserts the word k into the system to correct the hypothesis. Both cases use a prefix to generate the new hypothesis, but now, instead of finding a suffix that starts with a given word k, we are looking for a suffix that must not start with a given word s_l. Equation 1.14 can be reformulated as next to get the new hypothesis:

$$\hat{y}_1^{\hat{I}} \approx \underset{I, y_1^I}{\arg\max} \sum_{i'=1}^{I} \log \mathrm{p}(y_{i'} \mid y_1^{i'-1}, x_1^J, f = \hat{y}_1^i,\ s_l; \Theta) \tag{2.1}$$

where $f = \hat{y}_1^i$ is the validated prefix by the user. Note, that in this case we do not have the corrected word to add it. And s_l is the first word of the incorrect suffix.

Now, we have a new case in the probability distribution for the first word of the suffix:

$$\mathrm{p}(y_{i'}|y_1^{i'-1}, x_1^J, f = \hat{y}_1^i, s_l; \Theta) = \begin{cases} \delta(y_{i'}, \hat{y_{i'}}), & if\ i' \leq i \\ [\, y_{i'} \neq s_l \,]\, \mathrm{p}(y_{i'} \mid y_1^{i'-1}, x_1^J, f = \hat{y}_1^i,\ s_l; \Theta), & if\ i' = i+1 \\ \mathrm{p}(y_{i'} \mid y_1^{i'-1}, x_1^J, f = \hat{y}_1^i,\ s_l; \Theta), & \text{otherwise} \end{cases} \tag{2.2}$$

where $\delta(\cdot, \cdot)$ is the Kronecker delta and $[\, P \,]$ is an Iverson bracket:

$$\delta_1(y_{i'}, \hat{y_{i'}}) = \begin{cases} 1, & y_{i'} \equiv \hat{y_{i'}} \\ 0, & \text{otherwise} \end{cases} \tag{2.3}$$

$$[\, y_{i'} \neq s_l] = \begin{cases} 1, & y_{i'} \neq s_l \\ 0, & \text{otherwise} \end{cases} \tag{2.4}$$

This process assures that the validated prefix is present in the search tree, and can be seen as generating the most probable suffix assuring that the first word is different from the wrong one.

2.1.2 Interaction-explicit Mouse Actions

Until now, all the actions done by the user did not suppose an extra cost, because to correct a word, the user still must move the cursor to the correct position. If the system provides suggestions which are good enough, even if the first suggestion of the system was wrong, the user may ask for a new suffix, performing a MA without introducing a whole new word.

To perform this kind of MA, the user needs to perform a click before typing any word. In this case, the user needs to indicate explicitly that he wants a new suggestion, in contrast to the non-explicit positioning. For that, this kind of action is called interaction-explicit MA. The user can make as many MAs as needed, having in mind that are less costly than introducing a whole new word.

Figure 2.2 illustrates a typical IPMT session where an interaction-explicit MA is done. Initially, the user is provided with a source sentence x to be translated. The reference y, is the correct translation of the sentence that the user will try to obtain. At iteration 0, the IPMT system provides the first hypothesis $\hat{s}_h$, and the prefix p remains empty cause the user still does not supply any correction. At the next iteration, the user position the cursor before the word '*installation*', with the purpose of typing in '*type*'. By doing so, the user is validating the prefix '*Select the*', and signalling that the word '*installation*' is incorrect. Before typing in anything, the system provides a new suffix. This new suffix still wrong, and the user gives another opportunity to the system, before typing the correct word, by performing an interaction-explicit MA. Now, the system provides a new suffix knowing that the word 'install' is also wrong.

SOURCE (x): Seleccione el tipo de instalación.
REFERENCE (y): Select the type of installation.

ITER-0	$(\mathbf{p})$	( )
	$(\hat{s_h})$	*Select the installation wizard.*
ITER-1	$(\mathbf{p})$	Select the
	(s_t)	‖ *installation wizard*
	$(\hat{s_h})$	*install script.*
ITER-2	$(\mathbf{p})$	Select the
	(s_t)	‖ *install script*
	$(\hat{s_h})$	*type of installation.*
ITER-3	$(\mathbf{p})$	Select the type of installation.
	(s_t)	( )
	(k)	(#)
	$(\hat{s_h})$	( )
FINAL	$(\mathbf{p} \equiv \mathbf{y})$	Select the type of installation.

Figure 2.2: IPMT session to translate a Spanish sentence into English using *non-explicit* and *interaction-explicit* Mouse Actions. Non-validated hypotheses are displayed in italics, whereas accepted prefixes are in normal font. The MAs are indicated by the symbol '‖'.

This time, the suffix given by the system is correct, and the user just has to validate it as correct.

To generate a new hypothesis we can adjust Equation 2.1 as follows to instead of just banning one word s_l, forbid a sequence of words $s_l^n = s_l^1, ..., s_l^n$, where n is the number of MAs performed by the user at the same position:

$$\hat{y}_1^{\hat{I}} \approx \arg\max_{I, y_1^I} \sum_{i'=1}^{I} \log \mathrm{p}(y_{i'} \mid y_1^{i'-1}, x_1^J, f = \hat{y}_1^i,\ s_l^n; \Theta) \tag{2.5}$$

$$\mathrm{p}(y_{i'} | y_1^{i'-1}, x_1^J, f = \hat{y}_1^i, s_l^n; \Theta) = \begin{cases} \delta(y_{i'}, \hat{y_{i'}}), & \mathit{if}\ i' \leq i \\ [\, y_{i'} \notin s_l^n \,]\, \mathrm{p}(y_{i'} \mid y_1^{i'-1}, x_1^J, f = \hat{y}_1^i,\ s_l^n; \Theta), & \mathit{if}\ i' = i+1 \\ \mathrm{p}(y_{i'} \mid y_1^{i'-1}, x_1^J, f = \hat{y}_1^i,\ s_l^n; \Theta), & \text{otherwise} \end{cases} \tag{2.6}$$

where $\delta(\cdot, \cdot)$ is the Kronecker delta and $[\, P \,]$ is an Iverson bracket:

$$\delta_1(y_{i'}, \hat{y_{i'}}) = \begin{cases} 1, & y_{i'} \equiv \hat{y_{i'}} \\ 0, & \text{otherwise} \end{cases} \tag{2.7}$$

$$[\, y_{i'} \notin s_l^n] = \begin{cases} 1, & y_{i'} \notin s_l^n \\ 0, & \text{otherwise} \end{cases} \tag{2.8}$$

As in the previous case, this process guarantees that the validated prefix is present in the search tree. Also, assures that the first word of the suffix is not present in the sequence of incorrect words.

2.1.3 MAs at Charater Level

Until now, we have supposed that the user can only position the cursor before a word. There is another approach, which enables the user to position the cursor before any character of the sentence.

In this approach, the user is validating all the words up to the position of the cursor, and if the cursor is in the middle of a word the left part of it. More formally, assuming that the user has clicked in the *u-th* position of the *i'-th* word of the hypothesis, the validated prefix could be represented as follows:

$$f = (\hat{y}_1^{i'-1}, \hat{y}_{i'}{}_1^{u})$$

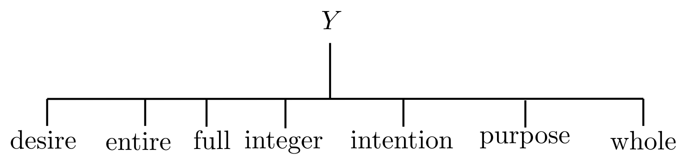
where $\hat{y}_1^{i'-1}$ is the sequence of validated words until the position $i'-1$, and $\hat{y}_{i'},_1^{u}$ is the correct part of the word $\hat{y}_{i'}$.

Figure 2.3 illustrates an example of a typical IPMT session where the user performs a *non-explicit* MA at character level. The user is provided with a source sentence x, that will translate to the reference sentence y by means of the system. At iteration 0, the system suggests the first hypothesis $\hat{s}_h$, how it is incorrect the user do not validate it. At iteration 1, the expert realizes that the sentence is correct until the middle of the *4-th* word, and instead of moving the cursor at the start of the word, performs a MA at a character level positioning it at the last correct character, validating the section $f = (\hat{y}_1^3, \hat{y}_4{}_1^4)$. Then, the system generates a new suffix $\hat{s}_h$ with this new information. This process is repeated until all the sentence is validated by the user.

SOURCE (x):		Va de la intención a la acción
REFERENCE (y):		It's going from intention to action
ITER-0	(**p**)	( )
	$(\hat{s_h})$	*It's going from interest to action*
ITER-1	(**p**)	It's going from inte
	(s_t)	‖ *rest to action*
	$(\hat{s_h})$	*ntion to action*
ITER-2	(**p**)	It's going from intention to action
	(s_t)	( )
	(k)	(#)
	$(\hat{s_h})$	( )
FINAL	$(\mathbf{p} \equiv \mathbf{y})$	It's going from intention to action

Figure 2.3: IPMT session to translate a Spanish sentence into English using *non-explicit* Mouse Actions at character level. Non-validated hypotheses are displayed in italics, whereas accepted prefixes are in normal font. The MAs are indicated by the symbol '‖'.

At word level, for each MA performed at the same position, only one new word was added to the sequence of forbidden words. This happens because, the only information that we had about the new word was that the current one was incorrect, this approach gives us more information about it. The fact that the user can move the cursor to the middle of a word tells us that the previous part of it is correct and that the next letter does not. Just the words of the vocabulary with that beginning, excluding those which have the incorrect character, are possible candidates. This list of possible next words is expressed as $m_1^n = m_1, ..., m_n$.



$M = \{\text{integer, intention}\}$

Figure 2.4: Constrainment of the vocabulary for character level interaction. For this example, we assume a vocabulary of 7 words. The user provides a validated prefix $f = (\hat{y}_1^{i'-1}, \hat{y}_{i'},{}_1^u)$ where the validated part of the last word is '*inte*' and the next incorrect character is '*r*'. From the vocabulary, the only words that begin with that prefix and do not continue with the character '*r*' are integer and intention. Those words compose the sequence of possible candidates.

If, after the user performs a non-explicit MA, the suffix is still incorrect, the user can perform interaction-explicit MAs. For each one, the filtering of the words from the vocabulary is more strict, as we have a new incorrect character.

We can formulate the set of Equations 2.5-2.8, to instead of excluding the words from a sequence, only accept the ones that are included there:

$$\hat{y}_1^{\hat{I}} \approx \arg\max_{I, y_1^I} \sum_{i'=1}^{I} \log \mathrm{p}(y_{i'} \mid y_1^{i'-1}, x_1^J, f = \hat{y}_1^{i-1},\ m_1^n; \Theta) \tag{2.9}$$

$$\mathrm{p}(y_{i'}|y_1^{i'-1}, x_1^J, f = \hat{y}_1^{i-1}, m_1^n; \Theta) = \begin{cases} \delta(y_{i'}, \hat{y_{i'}}), & \textit{if } i' < i \\ [\, y_{i'} \in m_1^n \,]\, \mathrm{p}(y_{i'} \mid y_1^{i'-1}, x_1^J, f = \hat{y}_1^{i-1},\ m_1^n; \Theta), & \textit{if } i' = i \\ \mathrm{p}(y_{i'} \mid y_1^{i'-1}, x_1^J, f = \hat{y}_1^{i},\ m_1^n; \Theta), & \text{otherwise} \end{cases} \tag{2.10}$$

where $\delta(\cdot, \cdot)$ is the Kronecker delta and $[P]$ is an Iverson bracket:

$$\delta_1(y_{i'}, \hat{y_{i'}}) = \begin{cases} 1, & y_{i'} \equiv \hat{y_{i'}} \\ 0, & \text{otherwise} \end{cases} \tag{2.11}$$

$$[\, y_{i'} \in m_1^n \,] = \begin{cases} 1, & y_{i'} \in m_1^n \\ 0, & \text{otherwise} \end{cases} \tag{2.12}$$

Dealing with subwords

To achieve better results, we apply the method BPE (see Section 1.3.2), which segments the words of the vocabulary in a new set of subwords. Although this helps to prevent finding unknown words, also, complicates the filtering of the vocabulary to get the sequence of candidate words.

For example, if the word '*intention*' was a set of the subwords '*in*' and '*tention*', none of them begins full with the prefix'*inte*', as needed in Figure 2.4. This, makes us have to explore all the possible combinations to generate the given prefix.

The algorithm implemented to make this search efficiently has been optimized to make it as quick as possible, we do not want that the user waits more than 1 second, as it could interrupt her flow of thought (Miller and Kelley, 1991).

The algorithm searches for combinations of subwords that contains the prefix, and a minimum of one character more, that can not be one of the marked as incorrect. The search algorithm of the system will pick the paths of the tree with the combinations, and the sequence of next subwords that obtains the highest score. So we do not have to find all the possible ways to finish the word, the search algorithm automatically will pick the best.

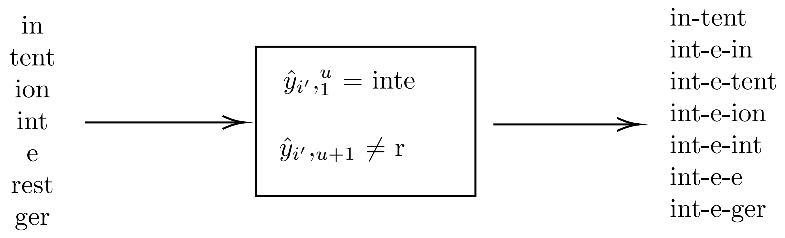


Figure 2.5: Subword combination process. Given a set of 7 subwords, a validated prefix $\hat{y}_{i'},_1^u$ and an incorrect character $\hat{y}_{i',u+1}$, the algorithm provides a list with all the suitable combinations. This process assures that the prefix is present in the new word, and at the same time is reducing the number of possible combinations of subwords.

2.2 IBM Model 1 as CM

In the previous approach, the user had to check all the words of the hypothesis to validate it. As seen in Section 1.8, in the systems with a CM technique integrated the user only have to check those words that scored lower than the threshold set.

Most of the confidence measures which have been presented in the literature calculate the scores either based on simple translation models such as IBM-1 or make use of information provided by an SMT system such as N-best lists or word graphs (Blatz et al., 2004; Gandrabur and Foster, 2003; Ueffing et al., 2003).

In this project, we used the IBM model 1 (see Section 2.2.1) to calculate the scores. It is used in two different modes: at word level (see Section 2.2.1), and at sentence level (see Section 2.2.2).

2.2.1 CM model

In this project, we used the IBM Model 1 as our CM model, because it lets us calculate the confidence scores of each word very fast, without losing performance in identifying correct words compared to others word CMs, as seen in the results from Blatz et al. (2004). We are in an interactive framework, the speed is a crucial feature of the system.

Given a source sentence x_1^J and a translation hypothesis y_1^I, the word confidence score c_w is computed as:

$$c_w(x_1^J, y_i) = \arg\max_{0 \leq j \leq J} \mathrm{p}(y_i|x_j) \tag{2.13}$$

where $\mathrm{p}(y_i|x_j)$ is the translation probability between the words y_i and x_j, which we can obtain as easy as query in a dictionary. x_0 is reserved for the empty source word. We substitute, the usually used, average, for a maximal lexicon probability because work by Ueffing and Ney (2005a) shows that the average is dominated by this maximum.

After computing the confidence scores of each word, they are classified as either correct or incorrect, depending on whether its confidence is lower or higher the classification threshold.

2.2.2 Sentence Level

We also consider another approach where instead of getting a confidence estimation of each word of the hypothesis, we just get one confidence estimation about all the sentence. From the word confidence score Equation (2.13), we compute two different CMs which differs in the way the word confidence scores are combined:

MEAN CM ($c_M(x_1^J, y_i^I)$) is computed as the geometric mean of the confidence scores of the words in the sentence.

$$c_M(x_1^J, y_i^I) = \sqrt[I]{\prod_{i=1}^{I} c_w(x_1^J, y_i)} \tag{2.14}$$

RATIO CM ($c_R(x_1^J, y_i^I)$) is computed as the percentage of words classified as correct in the sentence. A word is classified as correct if its confidence exceeds a word classification threshold h_w.

$$c_R(x_1^J, y_i^I) = \frac{|\{y_i / c_w(x_1^J, y_1) > h_w\}|}{I} \tag{2.15}$$

We set a sentence classification threshold value h_s, the sentences that get a confidence estimation lower than it are corrected using the conventional IPMT procedure, and the ones that get a higher or equal value are classified as correct and do not need correction. If we set the threshold $h_s = 0.0$ all the sentences will be classified as correct, and the system would work as a fully automatic

NMT, and if we set it as $h_s = 1.0$ all the sentences will be classified as incorrect and the system would work as a IPMT.

2.3 Conclusion

First of all, we have seen how are integrated the MAs as a new type of feedback in an IPMT system, the differences between the non-explicit and interaction-explicit MAs, and the use of MA at a character level. Also, we have seen how are implemented in the NMT system all this set of MAs.

Secondly, we have explained how works the CMs at word and sentence level, the equations for the two types of sentence level confidence estimations, and how are they implemented in the NMT system.

In the next chapter, we are going to see the results obtained with our models, and compare them with the obtained in previous works that integrate the same approaches in a system based on an SMT model.

Chapter 3

3 Experimental Setup

In this chapter, both methodologies are tested with different parallel corpus where we can compare the results with the obtained in SMT systems.

First of all, we introduce the software and corpora used in the experimentation; the metrics used to compare the results; how is simulated the user in the new cases raised by the CMs; and how the experiments are organized. Finally, the obtained results are shown, discussed and compared.

3.1 Software

In this section is described the main software used in the development of the project.

NMT-KERAS

NMT-Keras (Álvaro Peris and Casacuberta, 2018) is a flexible toolkit for training neural models, which puts special effort in the development of advanced applications, such as IPMT and Online Learning.

MOSES

Moses (Koehn et al., 2007) is an open-source toolkit which implements state of the art SMT techniques. Moreover, also provides a huge set of methods to perform the preprocessing of the corpora, and tokenize it correctly.

SUBWORD-NMT

Subword-nmt (Sennrich et al., 2015) is an open-source set of scripts which segments text into subword units. It learns a BPE just with the training text and can apply its rules to any file.

GIZA++

GIZA++ (Och and Ney, 2003) is a word-alignment toolkit, which between others can estimate the IBM Models 1 through 5.

3.2 Evaluation Metrics

In this section are explained all the metrics used to assess the results:

- **BLEU**: *BiLingual Evaluation Understudy* (BLEU) (Papineni et al., 2002) is a method for the automatic evaluation of machine translation. It computes a geometric mean of the precision of n-grams p_n, multiplied by a factor PB to penalise short sentences.

$$\text{BLEU} = \text{BP} \exp\left(\sum_{n=1}^{N} \frac{\log p_n}{N}\right) \tag{3.1}$$

- **TER**: *Translation Edit Rate* (TER) (Snover et al., 2006) is defined as the minimum number of edits needed to change a hypothesis to match the reference, normalized by the average length of the reference.

$$\text{TER} = \frac{\text{minimum n}^{\text{o}} \text{ of edits}}{\text{average n}^{\text{o}} \text{ reference words}} \tag{3.2}$$

  Possible edits include the insertion, deletion, and substitution of single words as well as shifts of word sequences.

- **CER**: *Classification Error Rate* (CER) is computed as the number of classification errors divided by the total number of classified words.

$$\text{CER} = \frac{\text{n}^{\text{o}} \text{ misclassified words}}{\text{n}^{\text{o}} \text{ classified words}} \tag{3.3}$$

- **WSR**: *Word Stroke Ratio* (WSR) (Tomás and Casacuberta, 2006) is computed as the number of words that the user would need to perform to generate the reference translation, normalized by the total number of words in the sentence.

$$\text{WSR} = \frac{\text{n}^{\text{o}} \text{ word strokes performed}}{\text{n}^{\text{o}} \text{ reference words}} \tag{3.4}$$

- **MAR**: *Mouse Action Ratio* (MAR) (Sanchis-Trilles et al., 2008b) is computed as the number of mouse actions that the user would need to perform in order to generate the reference translation, normalized by the total number of words in the sentence.

$$\text{MAR} = \frac{\text{n}^{\text{o}} \text{ mouse actions performed}}{\text{n}^{\text{o}} \text{ reference words}} \tag{3.5}$$

- **cMAR**: *Character level MAR* (cMAR) is a variant of the MAR metric where instead of normalizing the value between the number of words of the reference, it is normalized by the total number of characters of it.

$$\text{cMAR} = \frac{\text{n}^{\text{o}} \text{ mouse actions performed}}{\text{n}^{\text{o}} \text{ reference characters}} \tag{3.6}$$

- **uMAR**: *Useful MAR* (uMAR) (Sanchis-Trilles et al., 2008b) indicates the amount of MAs which were *useful*, i.e. the MAs that actually generates a new suffix with the first word correct.

$$\text{uMAR} = \frac{\text{MAC} - n\text{WSC}}{\text{MAC}} \tag{3.7}$$

where MAC stands for "Mouse Action Count", WSC for "Word Stroke Count" and n is the maximum number of MAs allowed for the same position.

As different experiments are done, not all the metrics appear in all of them. We use the WSR, MAR, cMAR and uMAR metrics, in the mouse actions experiments, to see the reduction in the word strokes needed against the increase in the number of mouse actions performed. The BLEU, TER, CER and WSR metrics are used in the confidence measure experiments to compare the WSR reduction obtained with the quality of the sentences generated.

3.3 Corpora

In this section, we describe all the corpora that were used through the development of the project. Three different parallel texts were used: Xerox, EU and Europarl.

All the corpora were preprocessed of the same way. First of all, with the scripts included in Moses, we cleaned, lower-cased and tokenized all the corpora. Once we have it tokenized, we applied the subword subdivision with a maximum of 32000 merges. As we wanted to compare our results with experiments already done in SMT systems, all the corpora were already divided into training, development and test from previous works.

In each corpus, we show the number of sentences, the average length of them, the number of total words, and the size of the vocabulary from the training set. The symbol "K" means that the number given is in thousands.

Xerox

The Xerox corpus (Esteban et al., 2004) is a compendium of user manuals for Xerox printers and photocopiers. The source language of the corpora was English, and the language services of Xerox provided the references for German, Spanish and French.

		De-En		Es-En		Fr-En	
		German	English	Spanish	English	French	English
Training	Sentences (K)	49		55		52	
	Avg. Length	10	12	13	11	13	11
	Run. Words (K)	538	593	750	665	677	615
	Vocabulary (K)	25	13	16	14	16	14
Development	Sentences	1000		1000		1000	
	Avg. Length	11	11	15	14	11	11
	Run. Words (K)	11	11	16	14	12	11
Test	Sentences	1000		1000		1000	
	Avg. Length	11	12	8	7	12	11
	Run. Words (K)	12	12	10	8	12	11

Table 3.1: Characteristics of the Xerox corpus.

EU

The EU corpus (Khadivi and Goutte, 2003) is formed from the Bulletin of the European Union, which exists in all official languages of the European Union, and is publicly available on the internet. We used the German↔English, Spanish↔English and French↔English pairs of languages.

		De-En		Es-En		Fr-En	
		German	English	Spanish	English	French	English
Training	Sentences (K)	222		214		215	
	Avg. Length	24	25	27	24	27	24
	Run. Words (K)	5348	5698	6000	5350	5806	5283
	Vocabulary (K)	152	86	84	69	91	83
Development	Sentences	400		400		400	
	Avg. Length	24	25	29	25	28	25
	Run. Words (K)	10	10	12	10	11	10
Test	Sentences	800		800		800	
	Avg. Length	23	25	28	25	28	24
	Run. Words (K)	18	19	23	20	22	20

Table 3.2: Characteristics of the EU corpus.

Europarl

The Europarl corpus (Koehn, 2005) is built from the Proceedings of the European Parliament, which exists in all official languages of the European Union, and is publicly available on the internet. Just like before, we used the German↔English, Spanish↔English and French↔English pairs of languages.

		De-En		Es-En		Fr-En	
		German	English	Spanish	English	French	English
Training	Sentences (K)	751		730		688	
	Avg. Length	20	21	21	20	22	20
	Run. Words (K)	15257	16101	15724	15268	15599	13849
	Vocabulary (K)	195	65	102	64	80	61
Development	Sentences	2000		2000		2000	
	Avg. Length	27	29	30	29	33	29
	Run. Words (K)	55	59	60	59	67	59
Test	Sentences	2000		2000		2000	
	Avg. Length	27	29	30	29	33	29
	Run. Words (K)	54	58	67	58	66	58

Table 3.3: Characteristics of the Europarl corpus.

3.4 User Simulation

3.4.1 MA

In the MA scenario, we assume that the user trusts the system enough to perform up to the maximum mouse actions allowed, a value set by us, to correct the error. If the user reaches the maximum mouse actions allowed the correct word is typed and a new suffix is generated.

Figure 3.1 illustrates an example where the translator has reached the limit of MAs allowed for the session. Although he asked the system two times to generate a new suffix, none of them corrected the word correctly, and the user had to type manually the correct word.

By correcting manually those cases in which the system does not correct the error, all the sentences produced are perfect, so in the next experiments, we do not need the metrics that provide us with a quality score like BLEU or Ter.

SOURCE (x):		Seleccione el tipo de instalación.
REFERENCE (y):		Select the type of installation.
ITER-0	$(\mathbf{p})$	( )
	$(\hat{s_h})$	*Select the installation wizard.*
ITER-1	$(\mathbf{p})$	Select the
	(s_t)	‖ *installation wizard*
	$(\hat{s_h})$	*install script.*
ITER-2	$(\mathbf{p})$	Select the
	(s_t)	‖ *install script*
	$(\hat{s_h})$	*kind installation.*
ITER-3	$(\mathbf{p})$	Select the
	(s_t)	*kind installation.*
	(k)	type
	$(\hat{s_h})$	*of installation*
ITER-4	$(\mathbf{p})$	Select the type of installation.
	(s_t)	( )
	(k)	(#)
	$(\hat{s_h})$	( )
FINAL	$(\mathbf{p} \equiv \mathbf{y})$	Select the type of installation.

Figure 3.1: IPMT session to translate a Spanish sentence into English using a maximum of 2 MAs. Non-validated hypotheses are displayed in italics, whereas accepted prefixes are in normal font. The MAs are indicated by the symbol '‖'.

3.4.2 CM

In the CM scenario, as we can validate imperfect sentences that differ from the reference, we have to make two assumptions to solve the new cases that we can find where the validated part is wrong. First, we assume that the CM makes no mistakes in classifying the words. Second, the user is always able to correct the word without taking into account its context.

The first assumption implies that the translator only checks the words that the system has classified as incorrect, skipping the other words. The confidence estimation of the system is not perfect, some words could be misclassified, provoking that the sentence generated by the system is not guaranteed to be equal to the reference.

The second assumption is a consequence of the first one. If we skip words that could be incorrect, the correctness of the validated prefix is no longer guaranteed. Even so, the translator should be capable of correcting each word, even if the previous part is wrong. We use the reference sentence to correct the words classified as incorrect.

Although these assumptions could seem unrealistic, they are made to simplify the IMT scenario and focus on the impact of adding CM to these systems.

Figure 3.2 illustrates an example where we can see how works both assumptions. The first assumption is very easy to show, at the first iteration, the first word tagged as incorrect is 'wizard', so the system misclassified the previous word 'installation'. To correct the word, the simulated user types the word that the reference has in the same position, without looking at its context, as explained in the second assumption. Finally, the system does not classify another word as incorrect and the translation finishes. In a perfect world, the final sentence generated would be equal to the reference.

SOURCE (x):		Seleccione el tipo de instalación
REFERENCE (y):		Select the type of installation.
ITER-0	(**p**)	( )
	($\hat{s_h}$)	*Select the installation wizard.*
ITER-1	(**p**)	Select the installation
	(s_t)	*wizard.*
	(k)	of
	($\hat{s_h}$)	*installation.*
ITER-2	(**p**)	Select the installation of installation.
	(s_t)	( )
	(k)	(#)
	($\hat{s_h}$)	( )
FINAL	(**p**)	Select the installation of installation.

Figure 3.2: IPMT session with CM to translate a Spanish sentence into English. Non-validated hypotheses are displayed in italics, whereas accepted prefixes are printed in normal font. Words classified as incorrect are displayed underlined.

3.5 Experimental Results

As a first step, we built an NMT system for each pair of languages and corpus cited in the previous section. This was done by means of the NMT-Keras toolkit, which is a complete system for building NMT models. The system adjusted the weights optimizing the BLEU score obtained on the development partition.

We used the architecture '*AttentionRNNEncoderDecoder*' from NMT-Keras, which uses a bidirectional RNN as an encoder, and as a decoder, a RNN with an attention mechanism, followed by a deep output function, and a fully-connected output layer with a softmax activation function to obtain the probabilities of the target words.

Once we had all the models built, we proceed to generate all the results and compare them with the obtained in SMT systems. In Section 3.5.1 we compare the results of introducing the mouse actions into NMT systems and in Section 3.5.2 the results of using CM with NMT systems.

3.5.1 Mouse Actions Results

Xerox and EU

We started by replicating the experiments done by Sanchis-Trilles et al. (2008a), with our NMT models. In those experiments they tested the improvements in WSR by using non-explicit and interaction-explicit MAs in the corpora Xerox and EU. Their results can be seen in Tables 3.4 and 3.6, and ours in Tables 3.5 and 3.7.

	baseline		non-explicit			explicit		
	cMAR	WSR	cMAR	WSR	WSR red.	cMAR	WSR	WSR red.
De-En	13.5	58.5	13.5	56.2	3.9	58.4	51.9	11.3
En-De	12.6	65.1	12.6	63.2	2.9	65.0	59.2	9.1
Es-En	13.5	31.7	13.5	27.0	14.8	31.3	23.8	24.9
En-Es	10.0	27.4	10.0	24.3	11.3	27.1	21.6	21.2
Fr-En	15.7	55.0	15.7	51.5	6.4	54.6	47.1	14.4
En-Fr	13.6	55.4	13.6	52.1	6.0	54.9	48.3	12.8

Table 3.4: Experimental results with the Xerox corpus with a SMT system. All results are in percentage. Results from Sanchis-Trilles et al. (2008a)

	baseline		non-explicit			explicit		
	cMAR	WSR	cMAR	WSR	WSR red.	cMAR	WSR	WSR red.
De-En	9.0	41.1	9.4	32.5	20.9	31.2	24.5	40.5
En-De	8.7	49.9	9.2	41.1	17.7	31.7	31.4	37.1
Es-En	6.8	34.2	7.3	27.9	18.4	20.4	22.5	34.2
En-Es	5.7	29.0	6.0	23.3	19.6	17.0	18.7	35.5
Fr-En	9.5	45.2	10.1	35.6	21.3	32.9	26.6	41.3
En-Fr	8.3	41.2	8.9	32.1	22.1	28.6	23.5	42.9

Table 3.5: Experimental results with the Xerox corpus with a NMT system. All results are in percentage.

	baseline		non-explicit			explicit		
	cMAR	WSR	cMAR	WSR	WSR red.	cMAR	WSR	WSR red.
De-En	14.4	60.5	14.4	58.5	3.3	60.5	53.4	11.7
En-De	13.5	62.2	13.5	60.6	2.6	62.3	56.4	9.3
Es-En	13.8	48.5	13.8	45.5	6.2	48.2	40.8	15.9
En-Es	15.9	52.1	15.9	49.0	6.0	52.3	44.9	13.8
Fr-En	14.4	44.0	14.4	40.3	8.4	43.8	36.3	17.5
En-Fr	15.6	49.6	15.6	47.0	5.2	49.8	43.1	13.1

Table 3.6: Experimental results with the EU corpus with a SMT system. All results are in percentage. Results from Sanchis-Trilles et al. (2008a)

	baseline		non-explicit			explicit		
	cMAR	WSR	cMAR	WSR	WSR red.	cMAR	WSR	WSR red.
De-En	7.2	39.9	7.4	30.3	23.9	24.6	20.0	49.8
En-De	6.5	42.8	6.7	33.1	22.7	22.8	23.1	46.0
Es-En	5.7	30.9	5.9	21.7	29.9	17.7	13.2	57.3
En-Es	5.7	31.2	5.9	21.6	30.8	17.6	13.1	57.9
Fr-En	5.9	32.2	6.1	23.1	28.1	19.0	15.1	53.0
En-Fr	5.8	31.4	6.0	22.4	28.9	18.3	14.1	55.2

Table 3.7: Experimental results with the EU corpus with a NMT system. All results are in percentage.

By definition, the MAR obtained in the baseline should be equal to the obtained with the non-explicit MAs. Due to that the non-explicit MAs are those which are already done in the baseline in order to position the cursor before correcting a word. In the previous work, this happens as supposed, but in ours, we can see a little difference in the values.

In our case, this happens because when the user corrects a word, we are setting the score of that word in that position to 1, and all the others to 0. But when the user does a non-explicit MA, only changes the score of the incorrect word in that position to 0, giving the possibility of obtaining different suffixes even when both options corrected the word.

It can be seen the big difference between applying this input information in both models. Throughout all the languages pairs and for both corpora, in the previous work, the maximum improvement using non-explicit MAs in the WSR was of 14.8%. In our work, with an NMT system, the minimum improvement was of 17.7%. This difference is even higher if we take a look at the results with interaction-explicit MAs. They get a maximum improvement of 24.9%, and we get a minimum of 34.2%.

One of the main reasons why our models obtained a better WSR improvement is because our models generate higher-quality translations, this idea is supported in the fact that in the vast majority of the cases we get a lower WSR in the baseline. Also, the better a model is, it is more likely that the model generates the correct suffix with less MAs.

We can go further and compare the models that we obtained from each corpus. Although the WSR baseline is almost the same for both cases, the improvement obtained with the EU corpus tends to be greater. We can see the difference between the models if we take a look at the percentage of uMAR with the increasing of explicit MAs performed (see Figure 3.3).

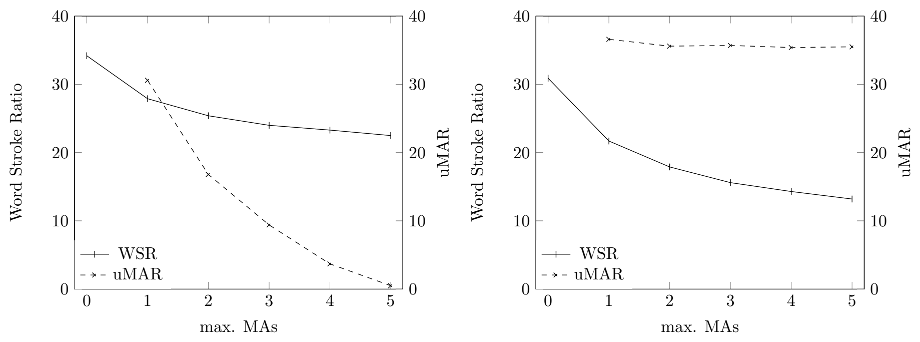


Figure 3.3: uMar evolution through the increasing of the maximum number of explicit MAs. Xerox Es-En (left), EU Es-En (right).

In the models obtained from the Xerox corpus, the uMAR tends to decrease when we increase the maximum number of MAs. In the pair of languages Es-En, the percentage of useful MAs decrease from 30.6% to 0.5% with a maximum of 5 MAs. While in the EU corpus, the uMAR remains at 33%. Ergo, although both models seemed equally good only looking at the baseline, just one of them is taking considerable benefits of the new MAs performed by increasing the maximum number.

Europarl

The experiments performed by Sanchis-Trilles et al. (2008b) used the Europarl corpus, and not only take into account the interaction-explicit MAs, but they also take a look into the improvement of the system considering one to five maximum MAs. They compare the metrics WSR, MAR and uMAR.

Once we have built our models, for each pair of languages of the corpus, we replicate the experiments, and the results were very similar to the obtained in the previous subsection. The results and comparison of them can be seen in Table 3.8. The reduction of WSR that we obtained from only performing non-explicit MAs, on average was of 25.0%, a value very impressive compared to the 3.2% obtained in the previous work.

	Sanchis-Trilles et al. (2008b)			our work		
	baseline	non-explicit	WSR red.	baseline	non-explicit	WSR red.
De-En	71.6	69.0	3.6	44.6	33.4	25.2
En-De	75.9	73.5	3.2	48.2	36.9	23.6
Es-En	63.0	59.2	6.0	42.0	31.1	26.0
En-Es	63.8	60.5	5.2	44.2	33.2	24.9
Fr-En	62.9	59.2	5.9	43.4	32.5	25.1
En-Fr	63.4	60.0	5.4	40.4	30.2	25.3

Table 3.8: WSR improvement when consideraing non-explicit MAs. All results are given in percentages.

Once we have compared the behaviour of both systems just taking account the non-explicit MAs, we can analyse the effect of performing to a maximum of 5 MAs per incorrect word. More specifically, taking a look at the metrics in each step, to see how the WSR improves while, at the same time, the MAR degrades.

In our case, performing to a maximum of 5 MAs per incorrect word produces an average improvement in WSR of about 51%, and the uMAR maintains around 30%. The number of word strokes that the user would have to perform is halved, and at each MA, the user has a probability of 0.3^n of getting the correct word, where n is the number MAs performed already.

In Figure 3.4, we have a comparison more in detail of both models for the pairs of languages '*Spanish→English*'. It can be seen how although both models start with a similar value of MAR, our model ends with a significantly lower value, due to the high percentage of uMAR, while in the other case, the lower value of it provokes that more MA were needed to be performed to correct the sentences.

Although in Figure 3.4 is only the comparison of one pair of languages, the experiments were done for all the pairs, and the results obtained were very similar to the shown there. Because of this, and to not saturate with multiple graphs with very similar values, we decided to omit them.

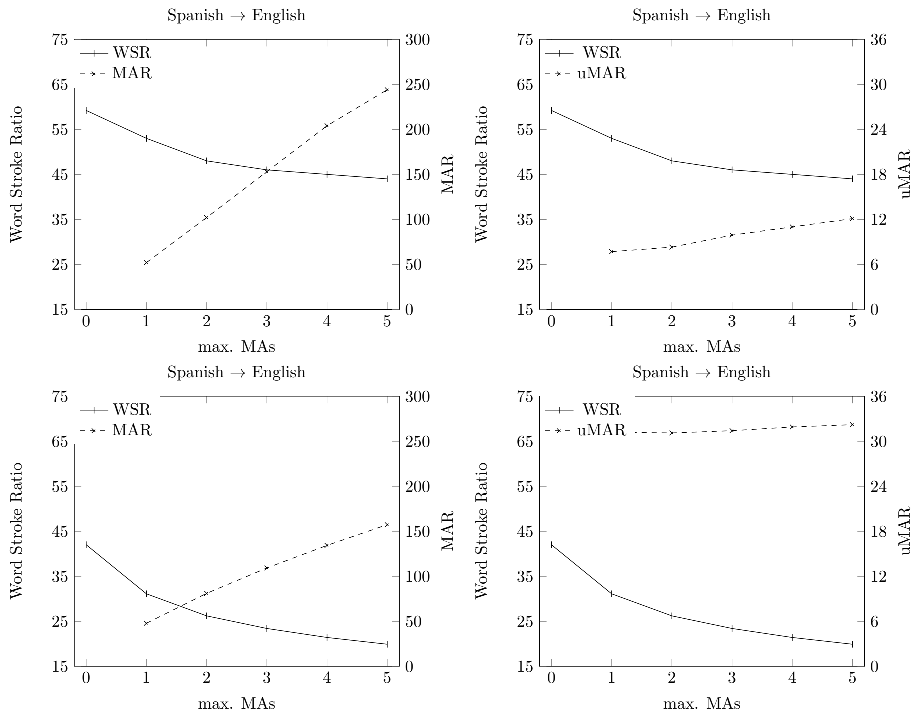


Figure 3.4: WSR improvement when considering one to five maximum MAs in Europarl corpus. All results are given in percentages. Sanchis-Trilles et al. (2008b) results (up), our results (down).

3.5.2 Confidence Measures Results

We built the IBM Model 1 needed to generate the confidence scores by means of the GIZA++ toolkit. We configure it to do 10 iterations of the EM algorithm before saving the probability matrix.

All the experiments were performed using the pair of languages Spanish-English of the EU corpora. As it was used in the experiments that we are going to replicate and compare next.

Word Level

First of all, we are going to replicate the experiments done by González-Rubio et al. (2010), where they treat the confidence scores of each word from the hypothesis individually. As in the previous section, they use SMT systems to generate the new hypothesis, and next, we are going to see the differences from using a NMT system instead.

Figure 3.5 shows CER for different values of the classification threshold in both projects. The two extreme values of the graph, 0.0 and 1.0, does not support any useful information to the system as they are tagging all the words equally. Specifically, when we set the threshold to 0.0 all the words are classified as correct, while when we use a value of 1.0, the words are classified as incorrect.

In our results, it can be seen easily in which zone where are achieved the best values of CER. The best value was obtained for a threshold of 0.05, where only 8% of the words were classified as incorrect. This value is so near to 0.0 that in this case would not be necessary use this CM, as it will be similar to not provide any information to the user.

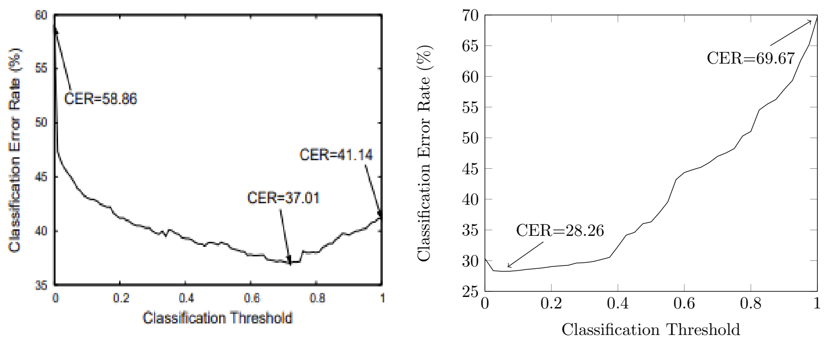


Figure 3.5: CER for different classification threshold values when translating from Spanish into English. All results are given in percentages. González-Rubio et al. (2010) (left) our work (right).

In the previous work, they reach the opposite conclusion. As shown in the graph, their evolution of CER for different classification thresholds is very different than ours. Their best value is obtained with a classification threshold of 0.75 and gets 37.0 CER.

In our case, the system provides so good hypothesis that even classifying all the words as correct, the CER obtained is better than the achieved in the previous work. These confidence scores are calculated with the IBM Model 1 which although is very quick, it does not have in consideration the context of the word. We could have reached the point where we need better models for our confidence scores to make them reliable again.

Next, we performed a series of experiments in which we simulated an IPMT session where the user only is asked to correct those words that are classified as incorrect according to the confidence classification threshold. In these experiments, we compare the effort done by the user, WSR, with the quality of the translations, TER and BLEU.

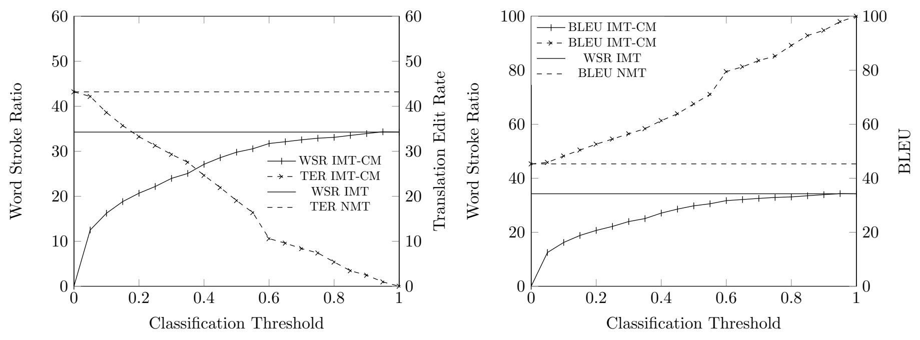


Figure 3.6: TER (left) and BLEU (right) translation scores against WSR for different values of the confidence classification threshold when translation from Spanish into English.

Figure 3.6 shows the results of these experiments, where '*WSR IMT-CM*', '*TER IMT-CM*' and '*BLEU IMT-CM*' correspond to the values obtained by our simulations; '*TER SMT*' and '*BLEU MST*' are the scores obtained by a fully-automatic MT; and '*WSR-IMT*' is the score from a conventional IPMT system.

Figure 3.6 shows a smooth transition between the fully automatic NMT system and the conventional IPMT system. As we increase the score, more words are classified as incorrect, and therefore, more words are corrected by the user. We can see how affects the system, that our best CER score in Figure 3.5 is obtained with a very low classification threshold (0.05): the WSR rise very quick in comparison with the changes in TER and BLEU.

Sentence Level

Finally, we are going to replicate, with the same models than before, the experiments performed by González-Rubio et al. (2010), which tried to reduce the translator effort depending on the confidence score of the whole sentence. The confidence scores are calculated following Equations 2.14 (MEAN CM) and 2.15 (RATIO CM).

Figure 3.7 shows the results of both projects obtained from using the MEAN CM to calculate the confidence score of the sentence. Note, how in our results we have obviated values greater than 0.3 because for all of them, the system behaved like a conventional IPMT system. In our case, the transition between the two behaviours is done between 0.0 and 0.2, the vast majority of the sentence confidence scores obtained were very low. While in the previous work, with the SMT model, this transition occurs more smoothly between the classification thresholds 0.0 and 0.6. Even so, from 0.6 and onwards, the system behaves like a conventional IPMT system. In both cases, this is an undesired effect, since for a large range of values there is no change in the behaviour of the system.

The brief transition obtained in our results could be related to the quality of the translations. As seen in the previous subsection, our model gets the best CER score at a very low confidence classification threshold (0.05), while with the SMT models the best CER was obtained at 0.75.

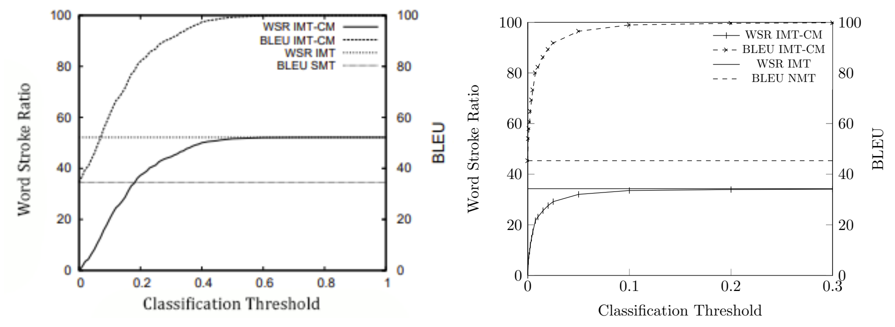


Figure 3.7: BLEU translation scores against WSR for different values of the sentence classification threshold using the MEAN CM. All results are given in percentages. González-Rubio et al. (2010) (left) our work (right)

The RATIO CM confidence values depend on a word classification threshold h_w. We have performed various experiments ranging h_w between 0.0 and 1.0 and see that we can expand the transition between the behaviours with this value. The larger the value selected, the smoother the transition between the fully automatic MT system and the conventional IPMT system. Figure 3.8 shows the results obtained from using $h_w = 0.4$ and $h_w = 0.6$. According to Figure 3.8, using $h_w = 0.6$ and a sentence classification threshold value of 0.5 we obtain a WSR reduction of 21% relative and an almost perfect translation quality of 89 BLEU points.

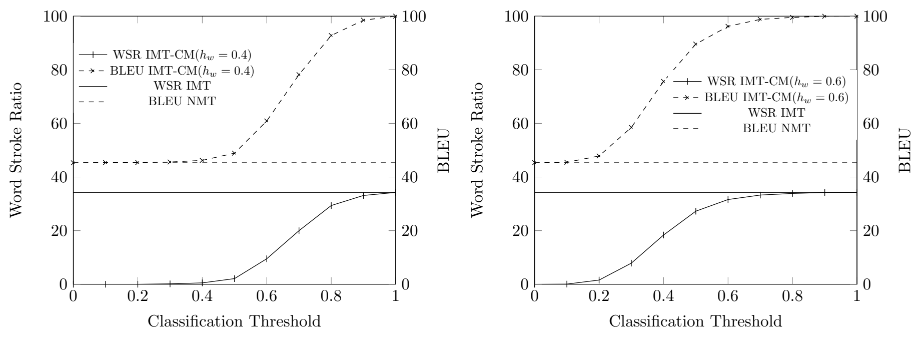


Figure 3.8: BLEU translation scores against WSR for different values of the sentence classification threshold using the RATIO CM with $h_w = 0.4$ and $h_w = 0.6$. All results are given in percentage. Results obtained in our work.

As in our experiments, in the previous work obtained that the greater was the word classification threshold, the smoother was the transition between the MT system and the IPMT system. Figure 3.9 shows the results obtained in the previous work with the RATIO CM with a $h_w = 0.4$, and it can be seen that is very similar to our graph with a word classification threshold of 0.6. With a sentence classification threshold value of 0.6, they obtain a WSR reduction of 20% relative and translation quality of 87 BLEU points.

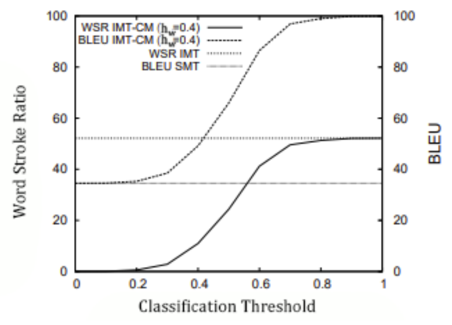


Figure 3.9: BLEU translation scores against WSR for different values of the sentence classification threshold using the RATIO CM with $h_w = 0.4$. All results are given in percentage. Results obtained from González-Rubio et al. (2010).

The results obtained in the previous subsections were not encouraging of the integration of CM into IPMT systems with NMT models, cause the best results were obtained with a classification threshold too near to 0.0. These last results, with the RATIO CM, indicates the opposite. With the correct values of the word classification threshold, we can smooth the transition and select a classification threshold that obtains an almost perfect translation quality with less WSR than using SMT models.

Chapter 4

4 Conclusions and Future Work

4.1 Conclusions

In this thesis, we have applied different techniques about interactive machine translation to NMT systems, with the premise of reducing the effort done by the translator in comparison with the needed when using SMT systems.

The techniques were tested using different corpora, and compared with previous works that used these same techniques but with different systems. These results show how the use of these techniques into NMT systems reduce greatly the effort that the user needs to do.

The results obtained from introducing the mouse actions as additional input information, were very good. In all the experiments the baseline was improved, proving that this kind of input information could be very useful and can reduce drastically the effort needed to do to correct a translation.

On the other hand, the results obtained from integrating confidence measures to the IPTM process were doubtful. At a sentence level, using the RATIO CM, the results were good and proved that in this scenario the confidence scores are useful: we can reduce the effort done by risking a bit the quality of the translations. In the other scenarios, the classification threshold values that obtained the best scores were so near to 0.0 that the system works as a fully automatic SMT.

Overall, the results proved that there is still work to do in this field. With the improvement of the models, also improve the effect that these interactive translation techniques have on them.

4.2 Future Work

In all the experiments that we have performed the user have been simulated following some basic rules. Not all the users behave in the same way, some of them could do not understand how the system works, or even need time to learn which is the best way to use it. As future work, we need to test the techniques implemented with real translators, to study their behaviours using the system and corroborate our results obtained with the user simulation.

Another line of work is to test the CM system using additional models to obtain the word confidence scores. As we have seen in Section 2, the IBM Model 1 does not have obtained the results that we thought, other models could work better with NMT systems, even without slowing down the human-machine interaction.

In this project, we have tested two different IMT approaches to reduce the human effort in INTM systems. We have to implement and test new approaches to INMT systems to compare the results and study which approach obtain the higher WSR reduction in this kind of systems.

References

Vicent Alabau, Luis A Leiva, Daniel Ortiz-Martínez, and Francisco Casacuberta. User evaluation of interactive machine translation systems. In *Proceedings European Association for Machine Translation*, pages 20–23, 2012.

Vicent Alabau, Ragnar Bonk, Christian Buck, Michael Carl, Francisco Casacuberta, Mercedes García-Martínez, Jesús González, Philipp Koehn, Luis Leiva, Bartolomé Mesa-Lao, et al. Casmacat: An open source workbench for advanced computer aided translation. *The Prague Bulletin of Mathematical Linguistics*, 100(1):101–112, 2013.

Vicent Alabau, Christian Buck, Michael Carl, Francisco Casacuberta, Mercedes García-Martínez, Ulrich Germann, Jesús González-Rubio, Robin Hill, Philipp Koehn, Luis A Leiva, et al. Casmacat: A computer-assisted translation workbench. In *Proceedings of the Demonstrations at the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pages 25–28, 2014.

Álvaro Peris and Francisco 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.

Sergio Barrachina, Oliver Bender, Francisco Casacuberta, Jorge Civera, Elsa Cubel, Shahram Khadivi, Antonio Lagarda, Hermann Ney, Jesús Tomás, Enrique Vidal, and Juan-Miguel Vilar. Statistical approaches to computer-assisted translation. *Computational Linguistics*, 35(1):3–28, 2009a.

Sergio Barrachina, Oliver Bender, Francisco Casacuberta, Jorge Civera, Elsa Cubel, Shahram Khadivi, Antonio Lagarda, Hermann Ney, Jesús Tomás, Enrique Vidal, and Juan-Miguel Vilar. Statistical approaches to computer-assisted translation. *Computational Linguistics*, 35(1):3–28, 2009b.

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.

John Blatz, Erin Fitzgerald, George Foster, Simona Gandrabur, Cyril Goutte, Alex Kulesza, Alberto Sanchis, and Nicola Ueffing. Confidence estimation for machine translation. In *Internation Conference on Computational Linguistics 2004: Proceedings of the 20th International Conference on Computational Linguistics*, pages 315–321. aug 23–aug 27 2004.

Peter F Brown, Stephen A Della Pietra, Vincent J Della Pietra, and Robert L Mercer. The mathematics of statistical machine translation: Parameter estimation. *Computational linguistics*, 19(2):263–311, 1993.

M. Asunción Castaño and Francisco Casacuberta. A connectionist approach to machine translation. In *Proceeding of the EUROSPEECH'97*, pages 91 – 94, 1997.

Kyunghyun Cho, Bart Van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259*, 2014.

Jan K Chorowski, Dzmitry Bahdanau, Dmitriy Serdyuk, Kyunghyun Cho, and Yoshua Bengio. Attention-based models for speech recognition. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 577–585. Curran Associates, Inc., 2015.

Automatic Language Processing Advisory Committee. Language and machines computers in translation and linguistics, 1966.

José Esteban, José Lorenzo, Antonio S Valderrábanos, and Guy Lapalme. Transtype2-an innovative computer-assisted translation system. In *Proceedings of the ACL Interactive Poster and Demonstration Sessions*, pages 94–97, 2004.

Gambier and Doorslaer, editors. *Handbook of Translation Studies.* John Benjamins Publishing Company, 2010.

Simona Gandrabur and George Foster. Confidence estimation for translation prediction. In *Proceedings of the Seventh Conference on Natural Language Learning at North American Chapter of the Association for Computational Linguistics 2003*, pages 95–102, 2003.

Jesús González-Rubio, Daniel Ortiz-Martínez, and Francisco Casacuberta. Balancing user effort and translation error in interactive machine translation via confidence measures. In *Proceedings of the Association for Computational Linguistics 2010 Conference Short Papers*, pages 173–177, 2010.

Jesús González-Rubio, Daniel Ortiz-Martínez, and Francisco Casacuberta. On the use of confidence measures within an interactive-predictive machine translation system. In *Proceedings of 14th Annual Conference of the European Association for Machine Translation*, 2010.

Alex Graves. Generating sequences with recurrent neural networks. *CoRR*, abs/1308.0850, 2013.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

S Khadivi and C Goutte. Tools for corpus alignment and evaluation of the alignments (deliverable d4. 9). Technical report, Technical report, TransType2 (IST-2001-32091), 2003.

Guillaume Klein, Yoon Kim, Yuntian Deng, Jean Senellart, and Alexander M Rush. Opennmt: Open-source toolkit for neural machine translation. *arXiv preprint arXiv:1701.02810*, 2017.

Rebecca Knowles and Philipp Koehn. Neural interactive translation prediction. In *Proceedings of the Association for Machine Translation in the Americas*, pages 107–120, 2016.

Philipp Koehn. Europarl: A parallel corpus for statistical machine translation. In *MT summit*, volume 5, pages 79–86. 2005.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, et al. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th annual meeting of the ACL on interactive poster and demonstration sessions*, pages 177–180. 2007.

Philippe Langlais, George Foster, and Guy Lapalme. Transtype: a computer-aided translation typing system. In *Associaton for Neuro Linguistic Programming-North American Chapter of the Association for Computational Linguistics 2000 Workshop: Embedded Machine Translation Systems*, 2000.

B. Lowerre and R. Reddy. The harpy speech recognition system: performance with large vocabularies. *The Journal of the Acoustical Society of America*, 60 (S1):S10–S11, 1976.

Minh-Thang Luong, Quoc V Le, Ilya Sutskever, Oriol Vinyals, and Lukasz Kaiser. Multi-task sequence to sequence learning. *arXiv preprint arXiv:1511.06114*, 2015a.

Thang Luong, Hieu Pham, and Christopher D. Manning. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1412–1421. September 2015b.

D. L. Miller and M. L. Kelley. Interventions for improving homework performance: A critical review. *School Psychology Quarterly*, 6(3):174–185, 1991.

Franz Och and Hermann Ney. The alignment template approach to statistical machine translation. *Computational Linguistics*, 30:417–449, 12 2004.

Franz Josef Och and Hermann Ney. A systematic comparison of various statistical alignment models. *Computational Linguistics*, 29(1):19–51, 2003.

Daniel Ortiz Martínez. *Advances in fully-automatic and interactive phrase-based statistical machine translation.* PhD thesis, Universidad Politécnica de Valencia, 2011.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, pages 311–318, 2002.

Álvaro Peris, Miguel Domingo, and Francisco Casacuberta. Interactive neural machine translation. *Computer Speech & Language*, 45:201–220, 2017.

Alberto Sanchis, Alfons Juan, and Enrique Vidal. Estimation of confidence measures for machine translation. *Proceedings of the MT Summit XI*, pages 407–412, 2007.

Germán Sanchis-Trilles, Maria-Teresa González, Francisco Casacuberta, Enrique Vidal, and Jorge Civera. Introducing additional input information into interactive machine translation systems. In *International Workshop on Machine Learning for Multimodal Interaction*, pages 284–295. 2008a.

Germán Sanchis-Trilles, Daniel Ortiz-Martínez, Jorge Civera, Francisco Casacuberta, Enrique Vidal, and Hieu Hoang. Improving interactive machine translation via mouse actions. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 485–494. October 2008b.

Mike Schuster and Kuldip K Paliwal. Bidirectional recurrent neural networks. *IEEE transactions on Signal Processing*, 45(11):2673–2681, 1997.

Rico Sennrich, Barry Haddow, and Alexandra Birch. Neural machine translation of rare words with subword units. *arXiv preprint arXiv:1508.07909*, 2015.

Rico Sennrich, Barry Haddow, and Alexandra Birch. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725. August 2016.

Matthew Snover, Bonnie Dorr, Richard Schwartz, Linnea Micciulla, and John Makhoul. A study of translation edit rate with targeted human annotation. In *Proceedings of association for machine translation in the Americas*, volume 200. 2006.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. Sequence to sequence learning with neural networks. *CoRR*, abs/1409.3215, 2014a.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112, 2014b.

Jesús Tomás and Francisco Casacuberta. Statistical phrase-based models for interactive computer-assisted translation. In *Proceedings of the COLING/ACL 2006 Main Conference Poster Sessions*, pages 835–841, 2006.

Nicola Ueffing and H. Ney. Application of word-level confidence measures in interactive statistical machine translation. In *Proceedings of the 10th Annual Conf. of the European Association for Machine Translation*, pages 262–270, 2005a.

Nicola Ueffing and Hermann Ney. Word-level confidence estimation for machine translation using phrase-based translation models. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pages 763–770, 2005b.

Nicola Ueffing, Klaus Macherey, and Hermann Ney. Confidence measures for statistical machine translation. In *In Proceedings MT Summit IX*, pages 394–401. 2003.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*, 2016.