UNIVERSIDAD POLITÉCNICA DE VALENCIA
DEPARTAMENTO DE SISTEMAS INFORMÁTICOS Y COMPUTACIÓN

Statistical approaches for natural language modelling and monotone statistical machine translation

Thesis
presented by Jesús Andrés Ferrer
supervised by Dr. Alfons Juan Císcar and Prof. Francisco Casacuberta Nolla

February 9, 2010

Statistical approaches for natural language modelling and monotone statistical machine translation

Jesús Andrés Ferrer

Thesis performed under the supervision of doctors
Alfons Juan Císcar and Francisco Casacuberta Nolla
and presented at the Universidad Politécnica de Valencia
in partial fulfilment of the requirements
for the degree Doctor en Informática

Valencia, February 9, 2010

Work has been partially supported by the CICYT “Centro de Investigación Científica y Tecnológica” under the Spanish project iDoc (TIC2003-08681-C02), by the Spanish research programme Consolider Ingenio 2010: MIPRCV (CSD2007-00018), by the PROMETEO program of the Generalitat Valenciana under grant Prometeo/2009/014; and by the ” *Consellería d’Empresa, Universitat i Ciència - Generalitat Valenciana”* under contract GV06/252 and under grant CTBPRA/2005/004.

Acknowledgements

Estas líneas son las más difíciles de escribir de la tesis aunque no contengan formulas ni formalismos matemáticos. Si tuviese que enumerar a todas las personas que han afectado positivamente (ó negativamente) al resultado de esta tesis, probablemente, tendría que anotar toda la tesis con comentarios al margen. Por ello, solo nombraré a las personas cuyas aportaciones han sido constantes e/o importantes sin por ello menospreciar al resto de personas que han influido.

Comenzaré por proximidad, agradeciendo a los habitantes del laboratirio 101L del departamento. No sólo a los miembros actuales, sino también a los fundadores de "Teachings of the final convergence". En particular, agradecer a Ramón Granell por nuestras discusiones; a Adrián Gimenez por ejercer de "diablo" en las reflexiones de algunos de mis problemas; y a Ricardo Sánchez por ser tan "coloquial y amigable".

Esta tésis no habría existido de no ser por el apoyo económico de la "Generalitat Valenciana" formalizado en la beca FPI con referencia CTBPRA/2005/004 recibida bajo el amparo de la "Conselleria d'Empresa, Universitat i Ciència". Así mismo estoy en deuda con Alfons Juan Císcar tanto por haber aceptado ser mi director de tesis, como por su consejo y su tesón en evitar mi dispersión en tantos temas interesantes que existen en la investigación. También debo agradecimiento a Francisco Casacuberta Nolla por su reflexiones, correcciones y guía. A parte del agradecimiento a mis directores de tesis me gustaría agradecerle a Prof. Hermann Ney por haberme acogido en Aquisgrán permitiendome así "disfrutar" tanto del clima de dicha región germana, como de sus apasionantes reflexiones que me permitieron refinar e incrementar mis conocimientos.

Mención especial requieren todos de mis compañeros de fatigas del PRHLT y del ITI en general y algunos en particular como por ejemplo Jorge Civera por soportar mis reflexiones sobre asuntos no sólo de investigación y por la plantilla de esta tesis que me ha ahorrado muchos quebraderos de cabeza; o Daniel Ortiz e Ismael García Varea por nuestras colaboraciones que son parte de esta tesis; así como a todos aquellos que con asiduidad han asistido las cenas minimalistas y no minimalistas.

Apropiandome de una cita del célebre Adrián Giménez "Cada tesis esconde un drama personal"; y este caso no podría ser una excepción. Así que estoy en deuda con todos aquellos que me han permitido disfrutar de la vida durante esta ardua e interminable tarea. Me veo en el deber de comenzar por la fuerte amistad que forgé con mis compañeros de carrera: Alex, Gabi y Jesús; amistad que todavía hoy perdura.

Desde mi retiro como exremero, no puedo sino que agradecer a todo el equipo de remo de la Universidad Politécnica de Valencia, tanto por las millas y el sudor derramado en la mar; como por las cervezas ingeridas y derramadas en la mesa de algún bar. En especial a todos aquellos remeros que me han acompañado desde el comienzo no sólo de esta tesis sino de mis estudios en informática cuando decicí practicar tan gratificante deporte.

Por último y más importante, tengo un especial y profundo agradecimiento a toda mi familia: a mis hermanos Joaquín y Juan por no ser solo hermanos sino amigos que me han acompañado durante mis estudios, mis sesiones cinemátograficas (algunas de las cuales "demasiado" antiguas), y en mis incursiones a la mar y al bar; a mi cuñada, Llum, por alimentarme con pizzas en su casa; a mi tía por estar siempre ahí como refuerzo; y a mis padres por guiarme a través de la vida, enseñarme los valores que me rigen como persona y por apoyarme en todo momento independientemente de alegría o pena. Finalmente, mi más profundo agradecimiento a mi novia Spe por su paciencia y amor incondicional que yo he compartido, comparto, y, espero, compartiré.

Jesús Andrés Ferrer
Valencia, February 9, 2010

Abstract

This thesis gathers some contributions to statistical pattern recognition and, more specifically, to several natural language processing (NLP) tasks. Several well-known statistical techniques are revisited in this thesis: parameter estimation, loss function design and probability modelling. The former techniques are applied to several NLP tasks such as text classification (TC), language modelling (LM) and statistical machine translation (SMT).

In parameter estimation, we tackle the smoothing problem by proposing a constrained domain maximum likelihood estimation (CDMLE) technique. The CDMLE avoids the need of the smoothing stage that makes the maximum likelihood estimation (MLE) to lose its good theoretical properties. This technique is applied to text classification by mean of the Naive Bayes classifier. Afterwards, the CDMLE technique is extended to leaving-one-out MLE and, then, applied to LM smoothing. The results obtained in several LM tasks reported an improvement in terms of perplexity compared with the standard smoothing techniques.

Concerning the loss function, we carefully study the design of loss functions different from the 0–1 loss. We focus our study on those loss functions that while retaining a similar decoding complexity than the 0–1 loss function, provide more flexibility. Many candidate loss functions are presented and analysed in several statistical machine translation tasks and for several translation models. We also analyse some outstanding translations rules such as the *direct translation rule*; and we give a further insight into the *log-linear models*, which are, in fact, particular cases of loss functions.

Finally, several monotone translation models are proposed based on well-known modelling techniques. Firstly, an extension to the GIATI technique is proposed to infer finite state transducers (FST). Afterwards, a phrased-based monotone translation model inspired in hidden Markov models is proposed. Lastly, a phrased-based hidden semi-Markov model is introduced. The latter model produces slightly improvements over the baseline under some circumstances.

Resumen

Esta tesis reune algunas contribuciones al reconocimiento de formas estadístico y, más específicamente, a varias tareas del procesamiento del lenguaje natural. Varias técnicas estadísticas bien conocidas se revisan en esta tesis, a saber: estimación paramétrica, diseño de la función de pérdida y modelado estadístico. Estas técnicas se aplican a varias tareas del procesamiento del lenguajes natural tales como clasificación de documentos, modelado del lenguaje natural y traducción automática estadística.

En relación con la estimación paramétrica, abordamos el problema del suavizado proponiendo una nueva técnica de estimación por máxima verosimilitud con dominio restringido (CDMLE[a]). La técnica CDMLE evita la necesidad de la etapa de suavizado que propicia la pérdida de las propiedades del estimador máximo verosímil. Esta técnica se aplica a clasificación de documentos mediante el clasificador Naive Bayes. Más tarde, la técnica CDMLE se extiende a la estimación por máxima verosimilitud por "leaving-one-out" aplicandola al suavizado de modelos de lenguaje. Los resultados obtenidos en varias tareas de modelado del lenguaje natural, muestran una mejora en términos de perplejidad.

En cuanto a la función de pérdida, se estudia cuidadosamente el diseño de funciones de pérdida diferentes a la 0–1. El estudio se centra en aquellas funciones de pérdida que reteniendo una complejidad de decodificación similar a la función 0–1, proporcionan una mayor flexibilidad. Analizamos y presentamos varias funciones de pérdida en varias tareas de traducción automática y con varios modelos de traducción. También, analizamos algunas reglas de traducción que destacan por causas prácticas como, por ejemplo, la regla de traducción directa; y, así mismo, profundizamos en la comprensión de los modelos log-lineares, que son de hecho, casos particulares de funciones de pérdida.

Finalmente, se proponen varios modelos de traducción monótonos basados en técnicas de modelado estadístico bien conocidas. En primer lugar, se propone una extensión a la técnica de GIATI, para inferir trasductores de estados finitos. Más tarde, se propone un modelo de traducción basado en secuencias de palabras e inspirado en los modelos ocultos de Markov. En último lugar, se presenta un modelo de traducción basado en secuencias de palabras y semi-modelos de Markov. Este último modelo produce mejoras sobre la referencia en ciertas circunstancias.

[a]Del inglés "Constrained Domain Maximum Likelihood Estimation"

Resum

Aquesta tesi reuneix algunes contribucions al reconeiximent de formes estadístic i més específicament a diverses tasques del processament del llenguatge natural. Diverses tècniques estadístiques ben conegudes són revisades en aquesta tesi: estimació paramètrica, disseny de la funció de pèrdua i modelatge estadístic. Les tècniques anteriors s'apliquen a diverses tasques del processament del llenguatge natural com ara classificació de documents, modelatge estadístic del llenguatge i traducció automàtica estadística.

En relació amb l'estimació paramètrica, portem a cap el problema del suavitzat proposant una tècnica d'estimació per màxima versemblança amb domini restringit (CDMLE[b]). La tècnica CDMLE evita la necessitat de l'etapa de suavitzat que afavoreix la pèrdua de les bones propietats de l'estimador per màxima versemblança. Aquesta tècnica s'aplica a classificació de documents mitjançant el classificador Naive Bayes. Més tard, la tècnica CDMLE s'exten a l'estimació per màxima versemblança amb "leaving-one-out", i aleshores, s'aplica al suavitzat de models de llenguatge. Els resultats obtinguts en diverses tasques de modelat del llenguatge mostren una millora en perplexitat.

En el diseny de la funció de pèrdua, s'estudia cuidadosament el diseny de funcions de pèrdua diferents a la 0–1. L'estudi es centra en aquelles funcions de pèrdua que retenen una complexitat de decodificació semblant a la funció 0–1 però proporcionant una major flexibilitat. Analitzem i presentem diverses funcions de pèrdua en diverses tasques de traducció automàtica amb diversos models de traducció. També analitzem algunes regles de traducció que destaquen per causes pràctiques com ara la regla de traducció directa; i així mateixa, s'aprofundeix en la comprensió dels models log-linear, que son de fet casos particulars d'aquestes funcions de pèrdua.

Finalment, es proposen diversos models de traducció monòtons basats en tècniques del modelatge estadístic ben conegudes. En primer lloc, es proposa una extensió a la tècnida de GIATI per a inferir transductors d'estats finits. Més tard, es proposa un model de traducció basat en sequències de paraules i inspirat en els models ocults de Markov. En darrer lloc, es presenta un model de traducció basat en sequències de paraules i en semi-models de Markov. Aquest darrer model produiex millores en certes circumstàncies.

[b] De l'anglés "Constrained Domain Maximum Likelihood Estimation"

Preface

Natural language processing (NLP) is a dynamic research field that aims at developing computer systems that are able to automatically generate and understand natural human language, both written and spoken. NLP is a subsoiled of artificial intelligence and linguistics, and as such it combines theories, methodologies, and experts from both worlds in order to address challenging problems.

Current technology in NLP is mainly based on inductive statistical pattern recognition approaches. These approaches define one or several statistical models that have to be estimated from a training dataset or corpus. There are several inferring criteria to perform such tasks; however, the maximum likelihood estimation (MLE) is one of the most wide-spread techniques.

The MLE verifies several desirable properties; however, a proper generalisation is conspicuous for its absence. Actually, the MLE tends to overfit the models to the training dataset, and, hence, to produce the corresponding lack of generalisation. To amend this problem, a common approach is to apply several heuristics in an additional *smoothing step*.

Once the training criterion is chosen and a proper probability model is designed for the task that is of interest, the best rule for building the system must be determined. Decision theory (DT) properly deals with this question by introducing the loss function. The loss function assesses the mistakes that a given system can produce. Typically, in many of the NLP tasks, a 0–1 loss function is assumed. Roughly speaking, this function accounts for the intuitive idea of minimising the number of errors that the system produces. This loss function yields the optimal Bayes' rule, which is the best rule that can be built in order to minimise this loss. However, this simple and intuitive loss does not take full advantage of this framework.

This statistical framework is applied to several NLP tasks. Specifically, this thesis is focused on exploring three of these NLP tasks: text classification (TC), machine translation (MT), and language modelling (LM).

Given a repository of documents, the purpose of TC is to automatically structure the documents by assigning a label to each one. The label can be automatically generated in the case of clustering or defined by an expert. In this way, the tasks of searching and browsing documents in the repository is eased. The TC technology seems to have reached a mature stage of research; nevertheless, there is still room for improvement.

The objective of MT is to make the computer automatically translate texts or utterances from one language into another language, without changing the underlying meaning. The

MT community is mainly focused on three main applications: fully-automatic MT, computer-assisted MT and understandable rough translation. The current MT technology is based on statistical methods in general and on the statistical pattern recognition theory in particular.

Fully-automatic statistical MT consists in the development of statistical models that are automatically inferred from bilingual parallel texts. In this respect, there have been different proposals for statistical translation models ranging from word-alignment translation models, such as the IBM models, the HMM word-alignment model, etc; to phrase-based and syntax-based translation models. These last models are usually based on byproducts of the word-alignment model training process. The most widespread and commonly used models are the phrased-based models; however, these models do not usually take into account the bilingual segmentation process, and are consequently heuristically trained.

The last topic of this thesis is LM itself. This problem is a very demanding and interesting problem since language models are used in a vast range of NLP problems such as speech recognition and machine translation among many others. The most widely used models, not only for their simplicity but also for their outstanding performance, are the so-called n-gram models. Similar to most of the statistical models in pattern recognition, n-gram language models suffer from overfitting. Several smoothing techniques have been proposed in the literature to deal with this problem.

The main objectives of this thesis are the followings: to present a new smoothing approach applied to TC and LM; to introduce new models in the paradigm of MT for monotone languages; and to study the loss function. The contributions of this thesis can be divided into three groups:

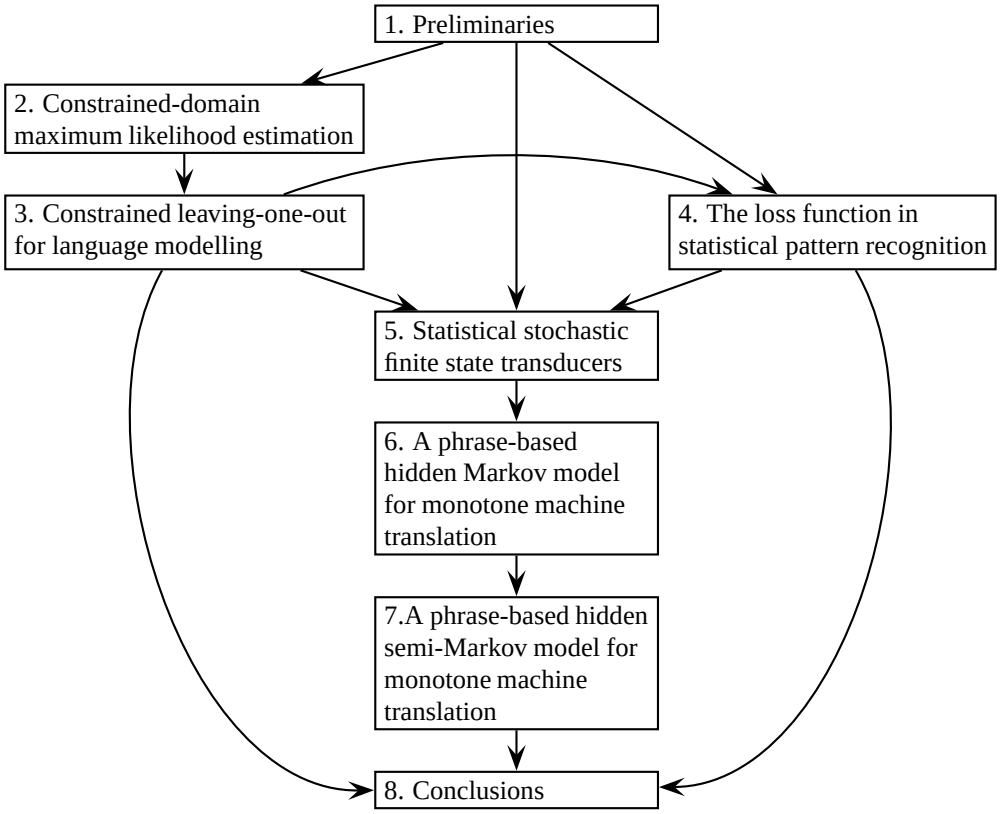
1. **Constrained-Domain Maximum likelihood estimation.** By reviewing the MLE deficiencies and the smoothing techniques used to alleviate them, we propose a modified version of the MLE that avoids the smoothing step. This new proposal is applied to two different modelling problems and tasks: text classification and language modelling. In text classification, we show how the constrained-domain maximum likelihood estimation (CDMLE), is applied to multinomial distribution avoiding the additional smoothing step. In language modelling, we use the CDMLE technique to smooth the leaving-one-out (LOO) estimation of the n-gram models since it is the milestone of the most successful n-gram smoothing techniques. This approach yields several novel smoothing techniques some of which slightly improve the standard smoothing techniques.

2. **Fundamental equation of statistical machine translation.** The optimal Bayes' rule is the basis of all statistical machine translation systems. However, this rule is obtained with the assumption of a 0–1 loss, i.e. assuming the CER as the error measure. We review classical statistical decision theory, and by changing this loss function, we boost the system performance. We apply these ideas to SMT and prove that the log-linear models are actually optimising a loss function, which resembles the actual error measure, such as the BLEU or the WER.

3. **Monotone statistical machine translation:** One of the deficiencies of the phrase-based models is that they are not "properly" modelled from a purely statistical point of view. This implies several problems in practise, since most of the systems use heuristics to estimate those models. For instance, several statistical modifications or estimation

techniques cannot be properly applied without several practical problems. We start by defining a purely statistical phrase-based finite transducer model. Once we have outlined the deficiencies of this model, we propose a *hidden Markov model (HMM)* that solves some of these deficiencies. However, forcing a HMM to take into account the bilingual segmentation process of a phrase-based model is not easy, and it yields highly demanding training algorithms. Finally, we propose an improved model that is based on *hidden semi-Markov models (HSMM)*. This latter HSMM takes into account the segmentation process smoothly, without producing highly demanding training algorithms.

The above contributions are sequentially organised in 8 chapters that cover most of the work developed in this thesis. We recommend a sequential reading of the document. However, should readers be interested in a specific research area, they can opt to read those specific chapters taking into account the following graph:



The constrained-domain MLE approach is proposed in Chapter 2. The experimental properties of this new approach are compared with classical smoothing methods in the task of TC. Later, in Chapter 3, this approach is extended to the n-gram models that are estimated and smoothed by leaving-one-out (LOO). If readers are not familiar with leaving-one-out smoothing methods for language modelling, then Section 1.2 would be helpful to them.

The optimal Bayes' decision rule obtained when a 0–1 loss function is used is one of the bases of most statistical pattern recognition systems. Specifically, when it is applied to statistical MT, it is called the fundamental equation of statistical machine translation and it is a milestone of the current MT systems. This optimal rule is revisited and studied in detail in Chapter 4.

The final part of this thesis is concerned with the definition of an efficient monotone machine translation model. Chapter 5 summarises the first incursion in this area. Inspired from Chapter 5, the following chapter tackles the monotone MT problem by defining a phrased-based hidden Markov model (PBHMM). Unfortunately, this model also had some important drawbacks. From the experience acquired with these two models, in Chapter 7, we introduce a phrased-based hidden semi-Markov model (PBHSMM) that achieves our objective: good performance, efficient training algorithms, and a properly defined statistic framework that would allow us to further improve the proposed model. This model is based on the hidden semi-Markov models (HSMMs). In Section 1.1.6 Chapter 1, we reformulate the HSMM following a new notation, that to our knowledge, has not been proposed elsewhere. Moreover, Section 1.1.6 paves the way towards the novel model proposed in Chapter 7. Should readers be unfamiliar with the HSMM, we encourage them to read Section 1.1.6 in Chapter 1.

Notation

Symbol	Meaning
$\mathrm{const}(\boldsymbol{x})$	is a constant function on $\boldsymbol{x}$, i.e., a function that *does not* depend on $\boldsymbol{x}$
$p_r(\cdots)$	is the actual unknown probability distribution
$\mathrm{p}_{\boldsymbol{\theta}}(\cdots)$	it is used to outline the fact that the probability is not the actual probability but a model that depends on a parametric vector $\boldsymbol{\theta}$
$\mathrm{p}(e)$ or p_e	the probabilities depicted in this way are already a model parameter for the event e
$A := B$	is used to stress that A is *modelled* by B, and also to stress that A is *defined* as B
$\delta(a, b)$	is the Kronecker delta function, i.e., 1 if and only if $a = b$, and 0 otherwise
$\boldsymbol{x}_1^j$	is used to denote the (sub)string $x_1 \dots x_j$
$\mathbb{N}$	is the natural number set, i.e. $\mathbb{N} = \{0, 1, 2, 3, \dots\}$
$\langle f(x) \rangle_y$	is used to denote the expectation of $f(\boldsymbol{x})$ over the probability distribution $p_r(\boldsymbol{y})$
$\langle f(x) \rangle_{\mathrm{q}(y)}$	is used to denote the expectation of $f(\boldsymbol{x})$ using the function $\mathrm{q}(\boldsymbol{y})$ as the probability distribution
$D(p\|\|q)$	is the Kullback-Leibler divergence between p and q
$\mathrm{suf}_{n-1}(\boldsymbol{x})$	stands for the $n-1$ ending elements of $\boldsymbol{x}$ or the full string $\boldsymbol{x}$ if $\|\boldsymbol{x}\| < n-1$

For denoting probability distributions throughout the thesis, we identify values and random variables whenever this entails no confusion. For instance, instead of

$$p_r(\Omega = \omega) \tag{1}$$

we use

$$p_r(\omega) \tag{2}$$

In the case of summations and products, we will omit the limits or set in which the index variable varies whenever this entails no confusion. For instance,

$$\sum_{\boldsymbol{x}} f(\boldsymbol{x}) = \sum_{\boldsymbol{x} \in \boldsymbol{X}^\star} f(\boldsymbol{x}) \quad , \tag{3}$$

or

$$\prod_{t} f(t) = \prod_{t=0}^{T} f(t) \quad . \tag{4}$$

Moreover, we use here the notation for which a summation on a void set is equal to 0,

$$\sum_{\boldsymbol{x}\in\emptyset} f(\boldsymbol{x}) = 0 \quad , \tag{5}$$

and where a void product is equal to 1,

$$\prod_{\boldsymbol{x}\in\emptyset} f(\boldsymbol{x}) = 1 \quad . \tag{6}$$

In this thesis several conditional probabilities will be used. For instance,

$$\mathrm{p}(\boldsymbol{u} \,|\, \boldsymbol{v}) \quad , \tag{7}$$

where we have used the symbol " $|$ " to divide the random variables into the given part, $\boldsymbol{v}$, and the random variable for which the probability function is defined, $\boldsymbol{u}$. However, in order to avoid cumbersome notation, the conditional probabilities will sometimes be written as follows

$$\mathrm{p}(\boldsymbol{u}/\boldsymbol{v}) \quad , \tag{8}$$

where "$/$" plays the role of " $|$ ". This ambiguity in the notation, is better understood with the following example

$$\mathrm{p}(\boldsymbol{u} \,/\, \boldsymbol{v}, |\boldsymbol{v}|, |\boldsymbol{u}|) \quad , \tag{9}$$

where if we had used " $|$ " instead of "$/$", the equation would have been awkward and unclear:

$$\mathrm{p}(\boldsymbol{u} \,|\, \boldsymbol{v}, |\boldsymbol{v}|, |\boldsymbol{u}|) \quad . \tag{10}$$

Contents

Chapter 1

Preliminaries

" *I could prove God, statistically.* " GEORGE GALLUP

Contents

1.1 Statistical Pattern Recognition

A pattern recognition problem consists in classifying each possible input or object, say $x \in \boldsymbol{X}$, in one class, say ω, from the set of all possible classes, say Ω. Examples of pattern recognition problems include text classification, speech recognition, image classification, face recognition, and machine translation, among others. A classification system is, then, characterised by the *classification function or rule*

$$\mathrm{c} : \boldsymbol{X} \longrightarrow \Omega \tag{1.1}$$

In the eighties, the most popular approaches to most of the pattern recognition problems were rule-based. Rule-based approaches define a huge set of rules based on the knowledge engineers and domain experts in order to build the classification system. The main problem of these approaches is the definition of hand-crafted rules and their maintenance. In the nineties, the rule-based approach was replaced by inductive approaches, which manly involved *statistical methods*. These approaches have numerous advantages:

- The classification function is learnt from the observation of a set of preclassified documents by an inductive process.
- The same inductive process can be applied to generate different classifiers for different domains and applications. This fact introduces an important degree of automation in the construction of ad-hoc classifiers.
- The maintenance task is significantly simplified, since it only requires to retrain the classifier with the new working conditions.
- The existence of off-the-self software to train classifiers requires less skilled man power than for constructing expert systems.
- The accuracy of classifiers based on inductive techniques competes with that of human beings and supersedes that of knowledge engineering methods in several tasks such as text classification, and speech recognition.

Several methodologies can be applied to define the classification function. Therefore, it is needed to find a measure for comparing among different classification systems. In order to quantify systems, the *classification error rate (CER)* is defined as the percentage of misclassifications performed by the system.

The classification system performance is usually measured as a function of the classification error. However, there are problems in which all the classification errors do not have the same consequences. Therefore, a function that ranks each kind of error should be provided. The *loss function*, $\mathrm{l}(\omega_p|\boldsymbol{x}, \omega_c)$, evaluates the *loss* in which the classification system incurs when classifying the object $\boldsymbol{x}$ into the class ω_p, knowing that the correct class is ω_c. An outstanding loss function is the 0–1 loss function

$$\mathrm{l}(\omega_p|\boldsymbol{x}, \omega_c) = \begin{cases} 0 & \omega_p = \omega_c \\ 1 & \text{otherwise} \end{cases} \quad . \tag{1.2}$$

If a 0–1 loss function is provided, then the optimal system minimises the classification error rate.

Taking into account the loss function definition, we define the risk when classifying an object $\boldsymbol{x}$, the so-called *conditional risk given* $\boldsymbol{x}$, as the expected value of the loss function according to the posterior class probability distribution, i.e.

$$R(\omega_p|\boldsymbol{x}) = \sum_{\omega_c \in \Omega} \mathrm{l}(\omega_p|\boldsymbol{x}, \omega_c)\, p_r(\omega_c|\boldsymbol{x}) \quad , \tag{1.3}$$

where $p_r(\omega_c|\boldsymbol{x})$ stands for the actual class posterior probability distribution. Note that by $p_r(\ldots)$ we will henceforth denote the actual probability distributions.

Usually, we want to compare system risks independently of any specific object $\boldsymbol{x}$. Using the conditional risk, we define the *the global risk* [Duda et al., 2001] as the contribution of all objects to the classifier performance, i.e.

$$R(\mathrm{c}) = \mathrm{E}_{\boldsymbol{x}}[R(\mathrm{c}(\boldsymbol{x})|\boldsymbol{x})] = \int_{\mathcal{X}} R(\mathrm{c}(\boldsymbol{x})|\boldsymbol{x})\, p_r(\boldsymbol{x}) d\boldsymbol{x} \quad , \tag{1.4}$$

where $R(\mathrm{c}(\boldsymbol{x})|\boldsymbol{x})$ is the conditional risk given $\boldsymbol{x}$, as defined in 1.4.

In practise, the global risk is approximated by the law of great numbers for a given test set $T = (\boldsymbol{x}_n, \omega_n)_{n=1}^N$ i.i.d. according to $p_r(\omega, \boldsymbol{x})$,

$$\bar{R}_T(\mathrm{c}) = \frac{1}{N} \sum_{n=1}^{N} \mathrm{l}(\mathrm{c}(\boldsymbol{x}_n)|\boldsymbol{x}_n, \omega_n) \quad . \tag{1.5}$$

The approximation of the global risk using a test set T is called *empirical risk* on the test set T. If we use the 0–1 loss function, then the empirical risk simplifies to the formerly defined classification error rate

$$\bar{R}_T(\mathrm{c}) = \frac{1}{N} \sum_{n=1}^{N} \delta(\mathrm{c}(\boldsymbol{x}_n), \omega_n) \quad , \tag{1.6}$$

where δ stands for the Kronecker delta function.

Our aspiration is to design the classification function that minimises the global risk. Since minimising the conditional risk for each object $\boldsymbol{x}$ is a sufficient condition to minimise the global risk, without any loss of generality, the optimal classification rule, namely *minimum Bayes' risk*, is the one that minimises the conditional risk for each object

$$\hat{\mathrm{c}}(\boldsymbol{x}) = \arg\min_{\omega \in \Omega} R(\omega \,|\, \boldsymbol{x}) \quad . \tag{1.7}$$

If 0–1 loss function is assumed, then the conditional risk is simplified to

$$R(\omega_p|\boldsymbol{x}) = 1 - p_r(\omega_p|\boldsymbol{x}) \quad , \tag{1.8}$$

and then the optimal classification rule is given by

$$\hat{\mathrm{c}}(\boldsymbol{x}) = \arg\max_{\omega \in \Omega} p_r(\omega \,|\, \boldsymbol{x}) \quad . \tag{1.9}$$

This equation is well-known and often assumed to be optimal for all pattern recognition problems, although, the assumption of a 0-1 loss function is always taken, either consciously or unconsciously.

1.1.1 Statistical modelling

In Eq. (1.9) the *class-posterior probability* is used in order to find the optimal class, although this probability is unknown. If we knew such probability, then we could define the best classifier for this framework, the so-called *Bayes classifier*, and its CER would be the minimum possible CER, the so-called *Bayes classification error rate*.

Since the posterior probability in Eq. (1.9) has to be approximated with a model, a common preliminary approach is to use the Bayes' theorem in Eq. (1.9) yielding

$$\mathrm{c}(x) = \arg\max_{\omega} \left\{ \frac{\mathrm{p}_r(x \,|\, \omega) p_r(\omega)}{p_r(x)} \right\} = \arg\max_{\omega} \left\{ \mathrm{p}_r(\omega) p_r(x \,|\, \omega) \right\} \quad , \tag{1.10}$$

where the posterior probability is substituted by two probabilities: the class prior $\mathrm{p}_r(\omega)$, and the object posterior $p_r(x \mid \omega)$. If actual probabilities are known, then both Eqs. (1.10) and (1.9) are equivalent. However, the latter Eq. (1.10) typically yield better approximations on real systems provided that actual probabilities are unknown, and it is needed to model them.

It is particularly worthy of note that from Eq. (1.10) and knowing that $p_r(A, B) = p_r(A)p_r(B \mid A)$ the following equation is obtained

$$\mathrm{c}(x) = \arg\max_{\omega} \{p_r(x, \omega)\} \quad . \tag{1.11}$$

Provided that we are focused on approximations to actual probabilities, most of the modelling techniques are based on statistics. Typically, classical or frequentist statistics are applied, producing a classification of the models in two categories

- *Parametric models:* where the actual probabilities are modelled according to any statistical distribution, such as, the normal distribution, or the beta distribution.
- *Non-parametric models:* where the actual probability is decomposed using statistical equivalences and afterwards modelled directly.

Another emerging modelling technique that has successfully been applied to several tasks such as text classification [Sutton and McCallum, 2006], or speech recognition [Heigold et al., 2007] is the *discriminative models* [Berger et al., 1996] or the *log-linear models*. A log-linear model is defined as an approximation to a probability distribution parametrised in the following way

$$p_{\boldsymbol{\theta}}(\omega|\boldsymbol{x}) = \frac{1}{Z_{\boldsymbol{\theta}}(\boldsymbol{x})} \exp(\sum_{k=1}^{K} \theta_k f_k(\boldsymbol{x}, \omega)) \quad , \tag{1.12}$$

with the set of parameters $\boldsymbol{\theta}$, and where $\boldsymbol{f}(\boldsymbol{x}, \omega)$ is a vector of *features*, defined a priory as a part of the modelling process. Finally, $Z_{\boldsymbol{\theta}}(\boldsymbol{x})$ is the normalisation constant that ensures that the posterior-class probability sums up to one,

$$Z_{\boldsymbol{\theta}}(\boldsymbol{x}) = \sum_{\omega \in \Omega} \exp(\sum_{k=1}^{K} \theta_k f_k(\boldsymbol{x}, \omega)) \quad . \tag{1.13}$$

The feature vector $\boldsymbol{f}(\boldsymbol{x}, \omega)$ is whatever vectorial function that obtains K real values from the object $\boldsymbol{x}$ and its class ω. Anyway, the features are often count events, such as whether a certain word appears or not; or such as the number of occurrences of a given word.

1.1.2 Training criterion

In order to train the model parameters, the *optimal* set of parameters, say $\hat{\boldsymbol{\theta}}$ must be found. The problem of the training criterion rises up because of the word "optimal". Appropriateness depends upon a criterion, which is summarised by the *criterion function ($\mathcal{C}$)*. Given a criterion function, the optimal set of parameters, $\hat{\boldsymbol{\theta}}$, is determined by

$$\hat{\boldsymbol{\theta}} = \arg\max_{\boldsymbol{\theta} \in \boldsymbol{\Theta}} \{\mathcal{C}(\boldsymbol{\theta})\} \quad . \tag{1.14}$$

Often the criterion $\mathcal{C}(\boldsymbol{\theta})$ cannot be mathematically calculated, and then, it is necessary a sample to approximate it, $D = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\}$,

$$\hat{\boldsymbol{\theta}} = \arg\max_{\boldsymbol{\theta} \in \boldsymbol{\Theta}} \{\mathcal{C}(\boldsymbol{\theta}; D)\} \quad . \tag{1.15}$$

Nevertheless the expression in Eq. (1.14) is used indistinctly of whether a sample is needed or not.

It is important to remark the difference between the loss function defined in Section 1.1 and the training criterion defined in the Eq. (1.14). The former defines the best way to build a system for given probability functions, whereas the latter determines the best way to obtain the optimal parameters, which would be used to approximate those probabilities.

There are several well-known and studied criteria such as maximum likelihood estimation (MLE), maximum a posteriori probability (MAP) or minimum mean energy (MME). We focus on the former, the wide-spread MLE.

1.1.3 Maximum likelihood estimation (MLE)

The maximum likelihood estimation (MLE) criterion is one of the most wide-spread criteria which has a well-founded motivation. It can be argued that since we are interested in the actual probability distribution, we should minimise the "distance" (in terms of the Kullback-Leibler divergence) between the model and the actual distribution, that is

$$\hat{\boldsymbol{\theta}} = \underset{\boldsymbol{\theta}\in\boldsymbol{\Theta}}{\arg\min} \left\{ \mathrm{KL}(p_r || p_{\boldsymbol{\theta}}) \right\} \quad , \tag{1.16}$$

where $\mathrm{KL}(p_r || \, \mathrm{p}_{\boldsymbol{\theta}})$ is the Kullback-Leibler distance between the model and the actual probability, defined as follows

$$\mathrm{KL}(p_r || p_{\boldsymbol{\theta}}) = \int_{\boldsymbol{X}} p_r(\boldsymbol{x}) \log p_r(\boldsymbol{x}) d\boldsymbol{x} - \int_{\boldsymbol{X}} p_r(\boldsymbol{x}) \log p_{\boldsymbol{\theta}}(\boldsymbol{x}) d\boldsymbol{x} \quad . \tag{1.17}$$

Plugging previous Eq. (1.17) into Eq. (1.16) yields

$$\hat{\boldsymbol{\theta}} = \underset{\boldsymbol{\theta}\in\boldsymbol{\Theta}}{\arg\max} \left\{ \int_{\boldsymbol{X}} p_r(\boldsymbol{x}) \log p_{\boldsymbol{\theta}}(\boldsymbol{x}) d\boldsymbol{x} \right\} \quad . \tag{1.18}$$

Since Eq. (1.18) is typically unfeasible to solve, it can be approximated by the law of great numbers. For a given sample $D = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\}$ i.i.d. according to $p_r(\boldsymbol{x})$, Eq. (1.18) is approximated by

$$\hat{\boldsymbol{\theta}} = \underset{\boldsymbol{\theta}\in\boldsymbol{\Theta}}{\arg\max} \left\{ \sum_n \log p_{\boldsymbol{\theta}}(\boldsymbol{x}_n) \right\} \tag{1.19}$$

If we define the *log-likelihood function (*LL*)* as follows

$$\mathrm{LL}(\boldsymbol{\theta}) = \sum_n \log p_{\boldsymbol{\theta}}(\boldsymbol{x}_n) \quad , \tag{1.20}$$

then Eq. (1.19) is expressed as

$$\hat{\boldsymbol{\theta}} = \underset{\boldsymbol{\theta}\in\boldsymbol{\Theta}}{\arg\max} \left\{ \mathrm{LL}(\boldsymbol{\theta}) \right\} \quad .$$

Therefore, minimising the divergence between the actual probability distribution and the model yields the log-likelihood function as the criterion function, i.e. $\mathcal{C}(\boldsymbol{\theta}) = \mathrm{LL}(\boldsymbol{\theta})$. This criterion is named after the log-likelihood function and is so-called *maximum likelihood (ML)* criterion. Maximum likelihood criterion typically leads to the intuitive solution of the relative frequencies. Note that since the logarithmic is an increasing function, maximising the log-likelihood function depicted in Eq. (1.20) is the same that maximising the likelihood function defined as follows,

$$\mathrm{L}(\boldsymbol{\theta}) = \prod_n p_{\boldsymbol{\theta}}(\boldsymbol{x}_n) \quad . \tag{1.21}$$

In summary, given an independent and interchangeable sample $D = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\}$, the MLE consists in solving the following maximisation

$$\hat{\boldsymbol{\theta}} = \arg\max_{\boldsymbol{\theta}\in\boldsymbol{\Theta}}\{\sum_n \log p_{\boldsymbol{\theta}}(\boldsymbol{x}_n)\} \quad . \tag{1.22}$$

The maximum likelihood estimation has been a core technique in pattern recognition. However, there is a little confusion in the bibliography around the MLE term. The term ML criterion is understood in statistics as the statistical criterion that we have presented here which is used to estimate the optimal set of parameters for any given probability distribution. In the pattern recognition literature, MLE refers to the estimation of the probability $p_r(\boldsymbol{x}, \omega)$ using "statistical MLE", i.e. maximising the following expression

$$\hat{\boldsymbol{\theta}} = \arg\max_{\boldsymbol{\theta}\in\boldsymbol{\Theta}}\{\sum_n \log p_{\boldsymbol{\theta}}(\boldsymbol{x}_n \mid \omega_n) + \log p_{\boldsymbol{\theta}}(\omega_n)\} \quad . \tag{1.23}$$

The MLE has several desirable properties:

- The MLE is asymptotically unbiased
- The MLE is asymptotically efficient, i.e., asymptotically, no unbiased estimator has lower mean squared error than the MLE
- The MLE is asymptotically normal. As the number of samples increases, the distribution of the MLE tends to the Gaussian distribution with the actual value as a mean and covariance matrix equal to the inverse of the Fisher information matrix
- The maximum likelihood estimator is consistent

There are some regularity conditions which must be satisfied to ensure this behaviour:

- The first and second order derivatives of the log-likelihood function must be defined
- The Fisher information matrix must be continuous and not zero-valued

Although, the MLE is asymptotically unbiased, the MLE is biased in practice for "small" datasets. The term small depends on the ratio of the dataset size to the number of parameters. In pattern recognition, this problem is very common and it is known as the overfitting problem. The overfitting problem is understood in pattern recognition as the fact that the learnt set of parameters is very specialised for the training data, and hence, a small amount of probability remains to be distributed among the unseen data.

A typical approach to alleviate this problem is to resort to a smoothing technique. A smoothing technique distorts the optimal set of parameters, $\hat{\boldsymbol{\theta}}$, in order to obtain a "smoothed" version of them, $\tilde{\boldsymbol{\theta}}$. Several of the smoothing techniques are heuristically inspired and make the optimal solution to lose all its theoretical properties.

1.1.4 Maximum likelihood estimation for hidden variable models

Maximum likelihood estimation usually leads to simple convex optimisation problems. However, if some variables were unobserved, finding the optimal parameter set is not a simple problem any more. Many useful models are *hidden variable models*, i.e. part of the random variables are not observed in practice. Fortunately, the *Expectation-Maximisation (EM)* algorithm [Dempster et al., 1977a, Neal and Hinton, 1998, Wu, 1983] finds the maximum likelihood parameters estimates in such problems. In this section, we briefly review the EM algorithm according to [Neal and Hinton, 1998].

A model is said to be a hidden variable model if part of the model variable is not seen in our training data. Therefore, it is diffuse whether hidden refers to the model or to the sample. In such a case, we

split the observation $\boldsymbol{x}$ into two random variables, $\boldsymbol{x} = (\boldsymbol{y}, \boldsymbol{z})$: the hidden part $\boldsymbol{y}$ and the visible part $\boldsymbol{z}$. Therefore, the model probability is given by

$$p_r(\boldsymbol{x}) := \mathrm{p}_{\boldsymbol{\theta}}(\boldsymbol{y}, \boldsymbol{z}) \quad . \tag{1.24}$$

The model in Eq.(1.24) is referred to as *the complete model*.

Suppose that the joint probability $\boldsymbol{y}$ and $\boldsymbol{z}$ is parametrised using $\boldsymbol{\theta}$, then the marginal probability of $\boldsymbol{z}$, the so-called *incomplete model*, is given by

$$\mathrm{p}_{\boldsymbol{\theta}}(\boldsymbol{z}) = \sum_{\boldsymbol{y}} \mathrm{p}_{\boldsymbol{\theta}}(\boldsymbol{y}, \boldsymbol{z}) \quad , \tag{1.25}$$

where, for simplicity, we have assumed that $\boldsymbol{y}$ has a discrete domain, as is often the case; anyway the results can be generalised.

Given the observed data $\boldsymbol{z}$, we wish to find the maximum likelihood estimate for the model parameters, that is to say the value of $\boldsymbol{\theta}$ that maximises the incomplete log-likelihood function given by

$$\mathrm{LL}(\boldsymbol{\theta}) = \log \mathrm{p}_{\boldsymbol{\theta}}(\boldsymbol{z}) = \log \sum_{\boldsymbol{y}} \mathrm{p}_{\boldsymbol{\theta}}(\boldsymbol{z}, \boldsymbol{y}) \quad . \tag{1.26}$$

The EM algorithm starts with some initial point $\boldsymbol{\theta}^{(0)}$, and then it proceeds to iteratively generate successive estimates, $\boldsymbol{\theta}^{(1)}, \boldsymbol{\theta}^{(2)}, \ldots$ by repeatedly applying two steps: the Expectation (E) step and the Maximisation (M) step. On the one hand, the E step consists in finding the (best) distribution for the unobserved variables, given the observed variables and the current estimate of the parameters. On the other hand, the M step re-estimates the parameters to be those with maximum likelihood, under the assumption that the distribution found in the E step is the actual distribution for the latent or unobserved variables.

It can be shown that each EM iteration improves the log-likelihood or leaves it unchanged. Note that this implies that the EM algorithm is able to find a local maximum but not a global one. This is both the most important property and also the most important drawback of this technique. Broadly speaking, the main drawback of the EM is that it delegates to the initialisation the responsibility of finding a global maximum, and, hence, a bad initialisation of the parameters can ruin the system performance.

The basis of the EM algorithm relay on defining an alternative objective function $\mathcal{L}(\cdots)$ to the log-likelihood function in Eq. (1.26), and then maximise this alternative criterion. This alternative objective function is a variation and hence, one of its parameters is a probability function. Therefore, given a parameter set $\boldsymbol{\theta}$ and a probability function $\mathrm{q}(\boldsymbol{y})$, the variation $\mathcal{L}(\mathrm{q}, \boldsymbol{\theta})$ is defined as follows

$$\mathcal{L}(\mathrm{q}, \boldsymbol{\theta}) = \mathrm{LL}(\boldsymbol{\theta}) - D(\mathrm{q} \,||\, \mathrm{p}_{\boldsymbol{\theta}}) \quad , \tag{1.27}$$

where by $\mathrm{p}_{\boldsymbol{\theta}}$ we denote $\mathrm{p}_{\boldsymbol{\theta}}(\boldsymbol{y} \mid \boldsymbol{z})$ and $D(\cdot||\cdot)$ is the Kullback-Leibler divergence. It can be proved [Neal and Hinton, 1998] that if a local (or global) maximum of $\mathcal{L}$ occurs at $\hat{\boldsymbol{\theta}}$ and $\hat{\mathrm{q}}$, then $\mathrm{LL}(\boldsymbol{\theta})$ has a local (global) maximum at $\hat{\boldsymbol{\theta}}$.

An iteration of the standard EM algorithm can be expressed in terms of the function $\mathcal{L}$, since each steps corresponds to the maximisation of one of its parameters while retaining the other fixed, i.e.,

E step Set $\mathrm{q}^{(k)}(\cdots)$ to the $\mathrm{q}(\boldsymbol{y})$ that *maximises* $\mathcal{L}(\mathrm{q}, \boldsymbol{\theta}^{(k-1)})$.

M Step Set $\boldsymbol{\theta}^{(k)}$ to the $\boldsymbol{\theta}$ that *maximises* $\mathcal{L}(\mathrm{q}^{(k)}, \boldsymbol{\theta})$.

It has been proved [Neal and Hinton, 1998] that the E step is maximised when $\mathrm{q}^{(k)} = \mathrm{p}_{\boldsymbol{\theta}}(\boldsymbol{y} \mid \boldsymbol{z})$. Then the EM algorithm can be expressed in its conventional way [Dempster et al., 1977a]

E step Compute the distribution $\mathrm{q}^{(k)}$ over the domain of $\boldsymbol{y}$[a] such that $\mathrm{q}(\boldsymbol{y}) = \mathrm{p}_{\boldsymbol{\theta}^{(k-1)}}(\boldsymbol{y} \mid \boldsymbol{z})$

[a] Actually over the domain of the random variable $\mathcal{Y}$, that accordingly to our notation is identified with its value $\boldsymbol{y}$.

M Step Set $\boldsymbol{\theta}^{(k)}$ to the $\boldsymbol{\theta}$ that maximises expected value of $\log \mathrm{p}_{\boldsymbol{\theta}}(\boldsymbol{y}, \boldsymbol{z})$ with respect to $\mathrm{q}^{(k)}$, that is to say, $\langle \log \mathrm{p}_{\boldsymbol{\theta}}(\boldsymbol{y}, \boldsymbol{z}) \rangle_{\mathrm{q}^{(k)}}$.

Moreover, the M step of the EM algorithm may be only partially implemented; and then the new estimates for the parameters $\boldsymbol{\theta}^{(k)}$ improve the function $\mathcal{L}(\mathrm{q}^{(k)}, \boldsymbol{\theta})$ given the distribution found in the E step $\mathrm{q}^{(k)}$, but do not maximise it. This partial M step always produces an improvement of the true likelihood. This variant is known as the *Generalised Expectation-Maximisation (GEM)* [Dempster et al., 1977a]. On the other hand, the E step may also be only partially implemented, with the new estimate for the hidden probability distribution, $\mathrm{q}^{(k)}$, only improving the function $\mathcal{L}(\mathrm{q}, \boldsymbol{\theta}^{(k-1)})$ given the optimal parameter set in previous iteration, instead of maximising it. In summary, the EM algorithm can be expressed as follows

E step Find a q that *improves* $\mathcal{L}(\mathrm{q}, \boldsymbol{\theta}^{(k-1)})$ with respect to $\mathcal{L}(\mathrm{q}^{(k-1)}, \boldsymbol{\theta}^{(k-1)})$, and set $\mathrm{q}^{(k)}$ to it.

M Step Find a $\boldsymbol{\theta}$ that *improves* $\mathcal{L}(\mathrm{q}^{(k)}, \boldsymbol{\theta})$ with respect to $\mathcal{L}(\mathrm{q}^{(k)}, \boldsymbol{\theta}^{(k-1)})$, and set $\boldsymbol{\theta}^{(k)}$ to it.

These two steps are repeated until convergence. The convergence is typically achieved when the relative increment of log-likelihood from iteration k to $k+1$ goes below a given threshold or when a maximum number of iterations is achieved.

The so-called *Viterbi* EM, is an outstanding version of the EM algorithm. In this version, the hidden probability distribution q obtained in the E step is assumed to be a Dirac's delta function at the maximum point $\hat{\boldsymbol{y}}$ according to $\mathrm{p}_{\boldsymbol{\theta}^{(k-1)}}(\boldsymbol{y} \mid \boldsymbol{z})$. This assumption restates the E step as follows

$$\boldsymbol{y}^{(k)} = \arg\max_{\boldsymbol{y} \in \boldsymbol{Y}} \mathrm{p}_{\boldsymbol{\theta}^{(k-1)}}(\boldsymbol{y} \mid \boldsymbol{z}) \quad , \tag{1.28}$$

and yields the hidden probability distribution q

$$\mathrm{q}^{(k)}(\boldsymbol{y}) = \begin{cases} 1 & \boldsymbol{y} = \boldsymbol{y}^{(k)} \\ 0 & \text{otherwise} \end{cases} \quad . \tag{1.29}$$

The maximisation step is, then, reduced to a usual MLE without latent variables but with the sample completed with the $\boldsymbol{y}^{(k)}$ estimates.

Since the Viterbi algorithm constrains the family of functions to optimise in the E-step, i.e. $\mathrm{q}^{(k)}(\boldsymbol{y}) = \delta(\boldsymbol{y}, \boldsymbol{y}^{(k)})$; the parameter set that is obtained by the Viterbi approximation is typically worse than that of the actual EM without any constraint in the E-step.

In the remaining subsections, we will analyse two outstanding hidden variable models: the hidden Markov models (HMMS) and the hidden semi-Markov models (HSMMs).

1.1.5 Hidden Markov models (HMMs)

Although initially introduced and studied in the late 1960s and early 1970s, statistical methods of Markov source or *Hidden Markov modelling (HMM)* have become an increasingly field of interest in the last few decades. There are two strong reason because these HMMs have become so popular. First, the models are very rich in mathematical structure. Second, the models obtain very good results in practise when applied properly. The applications of HMM range from several pattern recognition problems such as speech recognition to the field of bio-informatics [Rabiner, 1989]. In this section, we briefly introduce the model in a generic way.

Given a vector $\boldsymbol{x}_1^J$, we want to model its probability distribution, $p_r(\boldsymbol{x})$. For simplicity reasons, we assume through this section that all the inputs have a known and fixed length J, that is to say, when we write $p_r(\boldsymbol{x})$, we are actually referring to $p_r(\boldsymbol{x} \mid J)$. In order to model the previous probability, we assume that each element x_j of the vector $\boldsymbol{x}$ has been produced or *emitted* in a different state $q_j \in \boldsymbol{Q}$ and, hence, the same element x_j can have different probability distribution depending on the state in

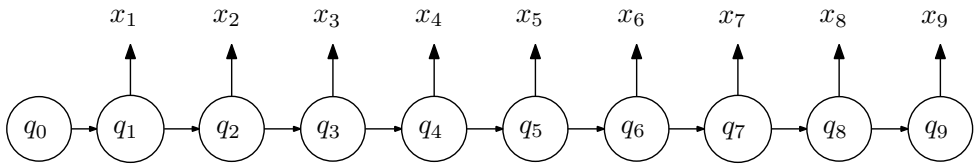


Figure 1.1: A graphical representation of the emission of a sequence of outputs $\boldsymbol{x}_1^9$ by a HMM. Note that this is not a graphical representation of a HMM topology.

which it was emitted, i.e., $\mathrm{p}(x_j \,|\, q_j)$. Since we do not observe the vector of states $\boldsymbol{q}$ in the training data, we need to model the emission probability with a hidden variable model. Mathematically,

$$p_r(\boldsymbol{x}) = \sum_{\boldsymbol{q}} p_r(\boldsymbol{x} \,|\, \boldsymbol{q}) p_r(\boldsymbol{q}) \quad , \tag{1.30}$$

where, as discussed above,

$$p_r(\boldsymbol{x} \,|\, \boldsymbol{q}) := \prod_j \mathrm{p}_{\boldsymbol{\theta}}(x_j \,|\, q_j) \quad . \tag{1.31}$$

A left-to-right decomposition is taken to model the state probability $p_r(\boldsymbol{q})$ under a first order Markovian assumption, i.e.,

$$p_r(\boldsymbol{q}) = \prod_j p_r(q_j \,|\, \boldsymbol{q}_0^{j-1}) := \mathrm{p}(q_0) \prod_j \mathrm{p}(q_j \,|\, q_{j-1}) \quad . \tag{1.32}$$

So far, no special model for the emission probabilities $\mathrm{p}_{\boldsymbol{\theta}}(x_j \,|\, q)$ is assumed.

Each state $q_j \in \boldsymbol{Q}$ can represent either an index $\boldsymbol{Q} = \{0, 1, \ldots, Q\}$, or something relevant. It is also valuable to highlight that there is one special state q_0, the so-called *initial state*, which does not emit any dimension of $\boldsymbol{x}$. This special state models the chances that any of the remaining states has to be the state to emit the first output element. Note that although we have add here our notation to the initial state notation, the initial state event is also modelled by using an initial state distribution [Rabiner, 1989].

Given a set of parameters γ, which comprises the transition probabilities $\mathrm{p}(q \,|\, q')$ and the emission parameters $\boldsymbol{\theta}$; the HMM is given by

$$\mathrm{p}_{\boldsymbol{\gamma}}(\boldsymbol{x}) = \sum_{\boldsymbol{q}} \mathrm{p}(q_0) \prod_j \mathrm{p}(q_j \,|\, q_{j-1}) \, \mathrm{p}_{\boldsymbol{\theta}}(x_j \,|\, q) \quad , \tag{1.33}$$

where $\mathrm{p}(q_0) = 1$ since this "phantom" state is is always present (it is a sure event); where $\mathrm{p}(q \,|\, q')$ are the model parameters for the transition probabilities and where $\mathrm{p}_{\boldsymbol{\theta}}(x_j \,|\, q)$ is the emission probability modelled with the parameter set $\boldsymbol{\theta}$.

There are many interesting questions to solve when dealing with HMM, however we focus here on 3 of them:

1. How to compute the probability $\mathrm{p}_{\boldsymbol{\gamma}}(\boldsymbol{x})$ for a given set of parameters $\boldsymbol{\gamma}$ and a given object $\boldsymbol{x}$.
2. How to obtain the (most probable) sequence of states $\hat{\boldsymbol{q}}$ that has emitted a specific object $\boldsymbol{x}$.
3. How to estimate the optimal set of parameters $\boldsymbol{\theta}$ for a given training set $D = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\}$.

The first question is typically solved by defining the so-called *forward* recursion. This recursion is obtained by reordering the sums of the probability in Eq. (1.33), i.e.,

$$\mathrm{p}_{\boldsymbol{\gamma}}(\boldsymbol{x}_1^J) = \sum_{q_0} \sum_{q_1} \cdots \sum_{q_J} \prod_j \mathrm{p}_{\boldsymbol{\theta}}(x_j \,|\, q_j) \, \mathrm{p}(q_j \,|\, q_{j-1}) \quad , \tag{1.34}$$

is equal to

$$\mathrm{p}_{\boldsymbol{\gamma}}(\boldsymbol{x}_1^J) = \sum_{q_0} \mathrm{p}(q_0) \cdots \sum_{q_j} \mathrm{p}_{\boldsymbol{\theta}}(x_j \,|\, q_j)\, \mathrm{p}(q_j \,|\, q_{j-1}) \cdots \sum_{q_J} \mathrm{p}_{\boldsymbol{\theta}}(x_J \,|\, q_J)\, \mathrm{p}(q_J \,|\, q_{J-1}) \quad . \tag{1.35}$$

In this way, the forward recursion $\alpha_j(q)$ is defined as the joint probability of emitting the prefix $\boldsymbol{x}_1^j$ and emitting the last element x_j in the state q, i.e.

$$\alpha_j(q) := \mathrm{p}_{\boldsymbol{\theta}}(\boldsymbol{x}_1^j, \mathcal{Q}_j = q) = \mathrm{p}_{\boldsymbol{\theta}}(x_j \,|\, q) \sum_{q'} \mathrm{p}(q \,|\, q') \alpha_{j-1}(q') \quad , \tag{1.36}$$

with the base case $\alpha_0(q_0) = 1$. The forward recurrence requires a time complexity of $O(Q^2 J)$ to fill in a table of $O(QJ)$ elements.

Finally, the probability of a given sequence $\boldsymbol{x}_1^J$ is computed as follows

$$\mathrm{p}_{\boldsymbol{\theta}}(\boldsymbol{x}_1^J) = \sum_{q} \alpha_J(q) \quad . \tag{1.37}$$

A similar recursion can be defined using a post-fix arrangement of the sums instead of a prefix one. The so-called *backward recursion* is defined as follows

$$\beta_j(q) = \mathrm{p}_{\boldsymbol{\theta}}(\boldsymbol{x}_{j+1}^J \,|\, \mathcal{Q}_j = q) = \sum_{q'} \mathrm{p}_{\boldsymbol{\theta}}(x_{j+1} \,|\, q')\, \mathrm{p}(q' \,|\, q) \beta_{j+1}(q') \quad , \tag{1.38}$$

with the base case $\beta_J(q) = 1$. The computational requirements for computing the backward recurrence are the same of that of the forward recursion.

The second question is answered by defining the *Viterbi recurrence*. Given an emitted object $\boldsymbol{x}$ we want to obtain the state vector $\boldsymbol{q}$ that maximises the probability of emitting such an object, i.e.

$$\hat{\boldsymbol{q}} = \arg\max_{q_0} \left\{ \arg\max_{q_1} \left\{ \cdots \arg\max_{q_J} \left\{ \prod_j \mathrm{p}_{\boldsymbol{\theta}}(x_j \,|\, q_j)\, \mathrm{p}(q_j \,|\, q_{j-1}) \right\} \cdots \right\} \right\} \tag{1.39}$$

By reordering the maximisations and the products, the Viterbi recurrence is defined as follows

$$\delta_j(q) = \arg\max_{q'} \left\{ \delta_{j-1}(q')\, \mathrm{p}(q \,|\, q') \right\} \mathrm{p}_{\boldsymbol{\theta}}(x_j \,|\, q) \quad , \tag{1.40}$$

with the base case $\delta_0 = \boldsymbol{q}_0$. Note that, by backtracking from $\max_q \delta_J(q)$, the optimal $\hat{\boldsymbol{q}}$ is obtained in a time complexity of $O(Q^2 J)$ with the aid of a recursion table of $O(QJ)$ elements.

Since the HMM is a hidden or latent model, some approximate inference algorithm is needed, and hence the third question has as many answers as approximate algorithms. The classical algorithms are obtained as the result of applying the EM. There are two main algorithms: Viterbi based training and Baum-Welch training. The former is the *Viterbi* EM training (see Section 1.1.4) applied to HMM, in which the Viterbi recursion is used in the E step.

The latter is the instantiation of the conventional EM training to the HMM case. The re-estimation equation in the M step for the transition probabilities in that case are given by

$$\mathrm{p}(q \,|\, q') = \frac{\sum_n \sum_j \xi_{nj}(q, q')}{\sum_n \sum_j \gamma_{nj}(q')} \tag{1.41}$$

where $\gamma_{nj}(q)$ is the probability of using the state q in the j-th emission independently of which is the previous state

$$\gamma_{nj}(q) = \sum_{q'} \xi_{nj}(q, q') \tag{1.42}$$

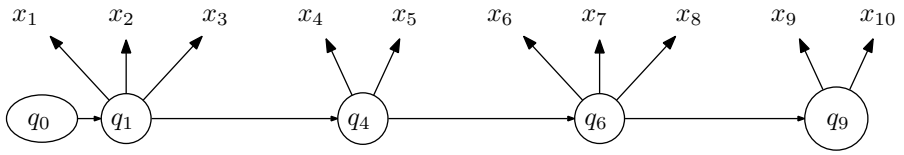


Figure 1.2: An instance of the generative segmentation process carried out by a HSMM for an output sequence $\boldsymbol{x}$ of 10 elements.

and where $\xi_{nj}(q, q')$ stands for the probability of using the state q for emitting the j-element having used the state q' for the previous emission, i.e.

$$\xi_{nj}(q, q') = \mathrm{p}_{\boldsymbol{\theta}}(\mathcal{Q}_j = q, \mathcal{Q}_{j-1} = q' \,|\, \boldsymbol{x}_n) = \frac{\alpha_j(q')\,\mathrm{p}_{\boldsymbol{\theta}}(x_j \,|\, q)\,\mathrm{p}(q \,|\, q')\beta_{j+1}(q)}{\mathrm{p}_{\boldsymbol{\theta}}(\boldsymbol{x})} \tag{1.43}$$

We omit the estimation equations for the emission probability $\mathrm{p}_{\boldsymbol{\theta}}(x_j \,|\, q)$ since no assumption is made in its modelling. It should be noted that this is the most important part of the model, and if it is modelled incorrectly, then it can ruin the system performance. However, for our interest, it is not needed to further assume any specific emission distribution.

The main advantage of the Viterbi training with respect to the Baum-Welch is its speed. The Baum-Welch training is slower than the Viterbi training. However, it is also expected that Baum-Welch training obtains better practical results than the Viterbi training since the Viterbi training only takes into account the most probable path for each sample instead of all the possible paths. Recall that this topic has already been addressed in Section 1.1.4.

Throughout all the section we have assumed that all the output objects $\boldsymbol{x}$ had the very same length J. In order to make the same model able to manage different lengths, a usual approximation is to add a non-emitting final state q_F or output symbol \$, and hence, the transition probability $\mathrm{p}(q_F \,|\, q)$ or the emission probability $\mathrm{p}(\$ \,|\, q_{J+1})$ is used for modelling the length. For instance in the case of the final state the parametrised model is given by:

$$\mathrm{p}_{\boldsymbol{\gamma}}(\boldsymbol{x}) = \sum_{\boldsymbol{q}} \mathrm{p}(q_0) \prod_j \mathrm{p}(q_j \,|\, q_{j-1})\,\mathrm{p}_{\boldsymbol{\theta}}(x_j \,|\, q)\,\mathrm{p}(q_F \,|\, q_J) \quad . \tag{1.44}$$

Note that a further subtle assumption is hidden in this model since each transition probability and emission probability does not depend on the length J, and, hence, the emission and transition probabilities have been merged for all the lengths [Rabiner, 1989].

1.1.6 Hidden semi-Markov models (HSMMs)

Given a sequence of observations $\boldsymbol{x}_1^J$, a hidden semi-Markov model (HSMM) [Ostendorf et al., 1996] is a modification of HMM. A HSMM emits a segment $\boldsymbol{x}_j^{j+l-1}$ at each state instead of constraining the emission to one element x_j as the (conventional) HMM. This way, the probability of emitting a sequence of observations $\boldsymbol{x}_j^{j+l-1}$ in any state depends on the segment length l and the state itself. The figure 1.2 depicts an instance of the emission process carried out by a HSMM.

In a hidden Markov model (HMM), the probability of emitting a segment of length l while remaining in the same state q, can only be simulated by transitions to the same state q. This approximation yields a exponential decaying length probability model

$$\mathrm{p}(l \,|\, q) = [\mathrm{p}(q \,|\, q)]^{l-1} \quad , \tag{1.45}$$

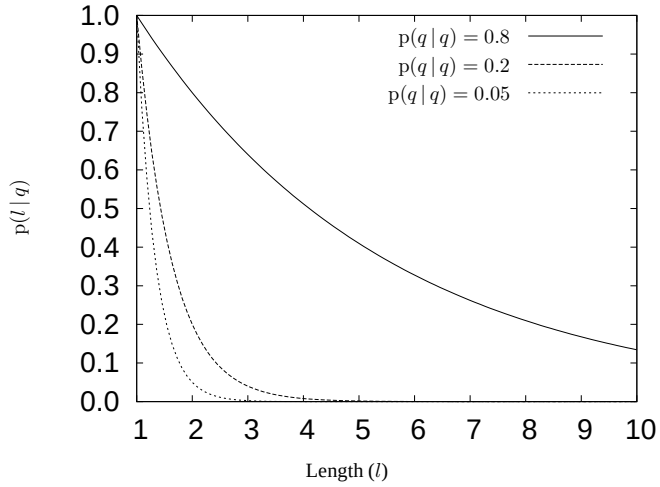


Figure 1.3: The "simulated" length probability in a HMM for several values of loop probabilities $\mathrm{p}(q \,|\, q)$. Note that we have used a continuous plot instead of a histogram plot for clarity's sake.

which is depicted in Fig. 1.3. This exponential decaying length probability model is not a good approximation for many cases.

Even if the decaying length probability is not an important issue for a given problem, the simulation of a HSMM using a HMM also constrains the segment emission probability to be a naive Bayes decomposition, i.e.

$$\mathrm{p}_{\boldsymbol{\theta}}(\boldsymbol{x}_j^{j+l_j-1} \,|\, q, l) = \prod_{k=j}^{j+l_j-1} \mathrm{p}_{\boldsymbol{\theta}}(x_k \,|\, q) \quad , \tag{1.46}$$

which again is not the best choice for many cases.

These two differences between an HMM and a HSMM, i.e., the exponential length distribution and the naive Bayes posterior emitting probability; introduce the HSMM as a very interesting, appealing and powerful extension to the conventional HMM.

There are several ways to formalise the HSMM extension. Here, we advocate for a similar formalisation of that found in Murhpy [2007]. The HSMMs are based on a hidden state sequence $\boldsymbol{q}$, which is a property inherited from HMM. Additionally, HSMMs also need to define a length random variable, $\boldsymbol{l}$, which stores the length of the segments emitted at each state. Oppositely to traditional HMM, the state sequence and the length vector can show different dimension to that of the emitted object $\boldsymbol{x}_1^J$ (see Fig. 1.2 for an example). However, in order to clearly specify the semi-Markov modelling technique, a special representation of the state sequence and length vector is needed. Under this representation, both state and length random variables share the length with the output object, i.e., J.

We define the length vector $\boldsymbol{l} = (l_1, l_2, \cdots, l_J)$ as a random variable that stores the length of each segment at the position at which the segment begins. All the remaining positions, which are not segment beginnings, are set to 0. For instance, in the example given in Fig. 1.2 the length vector $\boldsymbol{l}$ is $\boldsymbol{l}_1^{10} = (3, 0, 0, 2, 0, 3, 0, 0, 2, 0)$. The vector $\boldsymbol{l}$ can be extended with an additional element $l_0 = 0$ whenever it is necessary, yielding $\boldsymbol{l}_0^J = (l_0, l_1, l_2, \cdots, l_J)$.

Given a length emission vector $\boldsymbol{l}$, we define its prefix counterpart $\bar{\boldsymbol{l}}$ as

$$\bar{l}_j = \sum_{k=1}^{j} l_k, \quad j = 0, 1, \cdots, J \tag{1.47}$$

as well as its previous segment prefix length

$$\pi_j = 1 + \sum_{k=1}^{i(j)} l_k, \quad j = 0, 1, \cdots, J \tag{1.48}$$

with $i(j)$ equal to the starting of the previous segment. For instance, in Fig. 1.2, the prefix segment length is $\bar{\boldsymbol{l}} = \bar{\boldsymbol{l}}_0^{10} = (0, 3, 3, 3, 5, 5, 8, 8, 8, 10, 10)$ and the previous prefix segment length is given by $\boldsymbol{\pi} = \boldsymbol{\pi}_0^{10} = (1, 1, 1, 1, 4, 4, 6, 6, 6, 9, 9)$. Note that if we know a prefix of $\boldsymbol{l}$, say $\boldsymbol{l}_1^j$, then the corresponding prefixes of $\bar{l}$ and $\boldsymbol{\pi}$, say $\bar{\boldsymbol{l}}_0^j$ and $\boldsymbol{\pi}_0^j$, are known as well.

A similar definition to that of the length vector $\boldsymbol{l}$ would be handful for dealing with the state sequence vector. Let $\boldsymbol{Q}$ be the set of all possible states in which the model can emit an output segment, and let $\odot \notin \boldsymbol{Q}$ be an extra null state. The state sequence vector $\boldsymbol{q}_0^J$ stores the state in which each segment has been emitted at the starting position of the segment. Hence, the remaining positions not corresponding to the starting of any segment are set to the null state, $\odot$. For instance in the example in Fig. 1.2 the state vector $\boldsymbol{q}$ is defined as $\boldsymbol{q}_0^{10} = (q_0, q_1, \odot, \odot, q_4, \odot, q_6, \odot, \odot, q_9, \odot)$. Given a pair made of a state vector and a length vector $(\boldsymbol{q}, \boldsymbol{l})$, they must verify that the null states $\odot$ and the 0 lengths co-occur at the very same positions with the exception of q_0 which can never be $\odot$, in other words, $q_j = \odot$ if and only if $l_j = 0$, for $j = 1, 2, \ldots, J$.

Note that we assume no specific parametrisation for the segment emission probability distribution $\mathrm{p}_{\boldsymbol{\theta}}(\boldsymbol{x}_j^{j+l_j-1} \mid q_j, l_j)$ through the remaining of the current section.

Finally, in order to model the output emission probability for a given sequence of observations $\boldsymbol{x}$ we unhide the state vector $\boldsymbol{q}$ and the length vector $\boldsymbol{l}$

$$p_r(\boldsymbol{x}_1^J) = \sum_{\boldsymbol{l}} \sum_{\boldsymbol{q}} p_r(\boldsymbol{q}, \boldsymbol{l}) p_r(\boldsymbol{x} \mid \boldsymbol{q}, \boldsymbol{l}) \quad , \tag{1.49}$$

where we have introduced two probabilities: the state and length probability $p_r(\boldsymbol{q}, \boldsymbol{l})$ and the emission probability $p_r(\boldsymbol{x} \mid \boldsymbol{q}, \boldsymbol{l})$.

The first probability in Eq. (1.49) is decomposed from left to right as follows

$$p_r(\boldsymbol{q}, \boldsymbol{l}) = \prod_{j=1}^{J} p_r(q_j, l_j \mid \boldsymbol{q}_0^{j-1}, \boldsymbol{l}_1^{j-1}) \quad . \tag{1.50}$$

The state transition probability in Eq. (1.50), is modelled only in the segment boundaries. Let say that j is one of these boundaries, then $\bar{l}_{j-1} + 1 = j$ and thereby, the current state cannot be null, $q_j \neq \odot$. Note also that if there is a segment boundary at j, then the previous segment length cannot be 0 and the previous state cannot be null; that is to say $l_{\pi_{j-1}} > 0$ and $\boldsymbol{q}_{\pi_{j-1}} \neq \odot$. Finally, note that all the segment lengths $\boldsymbol{l}$ (and states $\boldsymbol{q}$), must be 0 (and null), between π_{j-1} and $j-1$; that is to say $\boldsymbol{l}_{\pi_{j-1}+1}^{j-1} = \boldsymbol{0}$ (and $\boldsymbol{q}_{\pi_{j-1}}^{j-1} = \odot$). Although many of those conditions are redundant, we have specified them for clarity's sake. In order to simplify notation, we use $C(j)$ to denote the predicate that corresponds to all the previous conditions, i.e.,

$$C(j) \equiv \bar{l}_{j-1} + 1 = j,\, \boldsymbol{l}_{\pi_{j-1}+1}^{j-1} = \boldsymbol{0},\, l_{\pi_{j-1}} > 0, q_j \neq \odot,\, \boldsymbol{q}_{\pi_{j-1}}^{j-1} = \odot,\, q_{\pi_{j-1}} \neq \odot \quad . \tag{1.51}$$

Using the previous definition, the state transition probability $p_r(q_j, l_j \mid \boldsymbol{q}_0^{j-1}, \boldsymbol{l}_1^{j-1})$ is, finally, modelled as follows

$$p_r(q_j, l_j \mid \boldsymbol{q}_0^{j-1}, \boldsymbol{l}_1^{j-1}) := \begin{cases} \mathrm{p}(q_j, l_j \mid q_{\pi_{j-1}}) & C(j) \\ 1 & l_{\bar{l}_{j-1}+1} \neq j,\ l_j = 0,\ q_j = \odot \\ 0 & \text{otherwise} \end{cases} \tag{1.52}$$

Note that the latter case $p_r(q_j, l_j \mid \boldsymbol{q}_0^{j-1}, \boldsymbol{l}_1^{j-1}) = 0$, is only possible if a segment length vector or state vector is out of the domain.

In this way, the state path probability in Eq. (1.52) is simplified to

$$p_r(\boldsymbol{q}, \boldsymbol{l}) := \prod_{j \in \mathcal{Z}(\boldsymbol{q})} 1 \prod_{j \notin \mathcal{Z}(\boldsymbol{q})} \mathrm{p}(q_j, l_j \mid q_{\pi_{j-1}}) \quad , \tag{1.53}$$

where $\mathcal{Z}(\boldsymbol{q})$ or simply $\mathcal{Z}$ stands for the set of positions j for which q_j is the null state $\odot$, or alternatively the positions for which $\boldsymbol{l}$ is 0, i.e. $l_j = 0$. For instance, in Fig. 1.2 we have $\mathcal{Z} = \{2, 3, 5, 7, 8, 10\}$.

Since one of the two products in Eq. (1.52) simplifies to 1, the state path probability in Eq. (1.53) is equal to

$$p_r(\boldsymbol{q}, \boldsymbol{l}) := \prod_{j \notin \mathcal{Z}} \mathrm{p}(q_j, l_j \mid q_{\pi_{j-1}}) \quad , \tag{1.54}$$

or, in order to simplify notation, to

$$p_r(\boldsymbol{q}, \boldsymbol{l}) := \prod_{t} \mathrm{p}(q_t, l_t \mid q_{\pi_{t-1}}) \quad , \tag{1.55}$$

where we have explicitly omitted that $t \in \mathcal{Z}$, but we use the index t instead of j to keep in mind the whole simplification process without the need of specifying any part of it.

Note that although we have modelled the transition and length probabilities with the same parameter $\mathrm{p}(q_j, l_j \mid q_i)$, these parameters are typically modelled with the following two parameters

$$\mathrm{p}(q_j, l_j \mid q_i) := \mathrm{p}(q_j \mid q_i)\, \mathrm{p}(l_j \mid q_j) \quad . \tag{1.56}$$

Since this does not affect to the algorithms significantly, we keep the more general notation $\mathrm{p}(q_j, l_j \mid q_i)$.

The emission probability in Eq. (1.49) can be decomposed in a similar way to the state transition probability as follows

$$\mathrm{p}_{\boldsymbol{\theta}}(\boldsymbol{x} \mid \boldsymbol{q}, \boldsymbol{l}) := \prod_{t} \mathrm{p}_{\boldsymbol{\theta}}(\boldsymbol{x}(t) \mid q_t, l_t) \quad , \tag{1.57}$$

where $\boldsymbol{x}(t)$ stands for $\boldsymbol{x}_t^{t+l_t-1}$.

Plugging Eqs. (1.55) and (1.57) into Eq. (1.49) the emission probability for HSMM is defined as follows

$$\mathrm{p}_{\boldsymbol{\theta}}(\boldsymbol{x}_1^J) := \sum_{\boldsymbol{l}} \sum_{\boldsymbol{q}} \prod_{t} \mathrm{p}(q_t, l_t \mid q_{\pi_{t-1}})\, \mathrm{p}_{\boldsymbol{\theta}}(\boldsymbol{x}(t) \mid q_t, l_t) \quad . \tag{1.58}$$

Similarly to the HMMs discussed in Section 1.1.5, we should answer to the same questions such as how to compute the probability for a given output $\boldsymbol{x}$. The answer to these questions is very similar to the HMM case, but taking into account an extra sum on the state emission lengths. For instance, the *forward recursion* for this model is defined by

$$\alpha_t(q) = \mathrm{p}_{\boldsymbol{\theta}}(\boldsymbol{x}_1^t, \mathcal{Q}_t = q) = \sum_{l} \sum_{q'} \mathrm{p}_{\boldsymbol{\theta}}(\boldsymbol{x}_{t-l+1}^t \mid q, l)\, \mathrm{p}(q, l \mid q')\alpha_{t-l}(q') \quad . \tag{1.59}$$

The time complexity for computing such a recurrence is $O(Q^2J^2)$, which is J times slower than the HMM counterpart. However, the segment length is often constrained to a maximum length M yielding a time complexity of $O(Q^2JM)$; which is just M times slower and *does not scale with the sentence length* J.

The analogous *Viterbi* and backward recursions to the ones defined for HMM are easily defined in a similar way. On the one hand, the backward recursion exploits the following equation

$$\beta_t(q) = \mathrm{p}_{\boldsymbol{\theta}}(\boldsymbol{x}_{t+1}^J \mid \mathcal{Q}_t = q) = \sum_l \sum_{q'} \mathrm{p}_{\boldsymbol{\theta}}(\boldsymbol{x}_{t+1}^{t+l} \mid q', l)\, \mathrm{p}(q', l \mid q)\beta_{t+l}(q') \quad . \tag{1.60}$$

On the other hand, the Viterbi recursion needs an additional maximisation for the emitted segment length in order to exploit the recursion, i.e.,

$$\delta_t(q) = \max_l \left\{ \max_{q'} \left\{ \delta_{t-l}(q')\, \mathrm{p}(q, l \mid q') \right\} \mathrm{p}_{\boldsymbol{\theta}}(\boldsymbol{x}_{t-l+1}^t \mid q, l) \right\} \quad . \tag{1.61}$$

Finally, the estimation of the model parameters is performed in a similar way to the HMM but using the re-defined recurrences.

1.2 Language Modelling

Language modelling is a core task in several natural language processing problems such as statistical machine translation (see Section 1.3) or speech recognition [Nadas, 1984] among others. The language modelling (LM) task is stated as the problem of designing appropriate models that approximate the probability of a given text, $p_r(\boldsymbol{w})$. Therefore, given a sentence or text $\boldsymbol{w}$ made up of T words chosen from a lexicon $\mathcal{W}$ with replacement, our aim is to model the probability $p_r(\boldsymbol{w})$ with an "appropriate" model $\mathrm{p}_{\boldsymbol{\theta}}(\boldsymbol{w})$.

There are several models for approximating the actual language probability distribution. For instance, hierarchical models use context-free grammars to capture long term dependencies [Benedí and Sánchez, 2005]. However, one of the most widespread models is the n-gram model [Goodman, 2001], which obtains surprisingly good performance although it only captures short term dependencies. The main advantage of this model is the simplicity and good performance compared with other more complex models.

The n-gram model decomposes the language probability from left to right as follows

$$p_r(\boldsymbol{w}) = \prod_{t=1}^{T} p_r(w_t | \boldsymbol{w}_1^{t-1}) \quad . \tag{1.62}$$

In theory we could turn each product term in Eq. (1.62) into a parameter. In practice, however, this would lead to a huge set of parameters that would be unfeasible to train. Therefore, a n-Markovian assumption is made in order to keep a manageable amount of parameters. That is the same to say that, the probability of the t-th word is assumed to depend only on the $n-1$ previous words, yielding the n-gram model

$$p_r(\boldsymbol{w}) := \prod_{t=1}^{T} \mathrm{p}(w_t | h_n(\boldsymbol{w}_1^{t-1})) \quad , \tag{1.63}$$

where $h_n(\boldsymbol{w}_1^{t-1})$ or simply h, stands for the $(n-1)$ words previous to the current position t, i.e.

$$h_n(\boldsymbol{w}_1^{t-1}) := \boldsymbol{w}_{\max\{t-n+1,1\}}^{t-1} \quad . \tag{1.64}$$

We will henceforth use $\bar{h}$ to denote the $n-2$ previous words

$$\bar{h} = h_{n-1}(\boldsymbol{w}_1^{t-1}) \quad , \tag{1.65}$$

$\bar{\bar{h}}$ to denote the $n-3$ previous words, and so on.

Note that in the Eq. (1.63), we have abused of notation since, for instance, the first term in the product is $\mathrm{p}(w_1)$, which is not a unigram but the probability of w_1 to be in the *first position of the sentence*. Hence, if there are not $n-1$ previous positions in the history h, then the probability $\mathrm{p}(w_t \,|\, \boldsymbol{w}_1^{t-1})$ is not a t-gram probability, but the probability for w_t to be the t-th word knowing that $\boldsymbol{w}_1^{t-1}$ are at the $t-1$ first positions. This fact is usually made explicit in practice by concatenating at the beginning of the sentence a special symbol, say "<s>".

1.2.1 Evaluation

In order to compare between different language models the *(conditional) perplexity* [Bahl et al., 1983] on a test set, $S = \{\boldsymbol{s}_1, \ldots, \boldsymbol{s}_M\}$ is defined as

$$\mathrm{PP}(S) = 2^{\frac{1}{N}\sum_{m=1}^{M} \log_2 p_{LM}(\boldsymbol{s}_m)} \quad , \tag{1.66}$$

for a given *language model (LM)* $p_{LM}(\cdots)$; and where N stands for the total number of words in the test set. If the LM is a n-gram model, then

$$\mathrm{PP}(S) = 2^{\frac{1}{N}\sum_{m=1}^{M}\sum_{l=1}^{T_m} \log_2 p(s_{ml} \,|\, h_n(\boldsymbol{s_m}_1^{l-1}))} \quad , \tag{1.67}$$

where T_m stands for the length of the m-th outcome, $\boldsymbol{s}_m$.

Note that the *(conditional) perplexity* is a geometric average of the log-likelihood,

$$\mathrm{PP}(S) = 2^{\frac{1}{N}\,\mathrm{LL}_{LM}(S)} \quad . \tag{1.68}$$

The perplexity can be understood as the average number of possible words that can come after a given prefix. The perplexity depends on two factors: the model efficiency and the task complexity. Under the same circumstances, the less the perplexity is the better. Therefore, if we compare two different smoothing techniques or models, the one with the smallest perplexity is the best one. However, the criticism to this measure lies on the fact that an improvement in perplexity is not always related to an improvement on the system performance.

1.2.2 Maximum Likelihood Estimation

Once the n-gram model is simplified by the $n-1$ previous words assumption, its parameter set is still large enough to annihilate our chances for obtaining a reliable estimation from the training data. Specifically, the n-gram parameter set is[b]

$$\{\mathrm{p}(w \,|\, h)\} \quad \forall w \in \mathcal{W}, \forall h \in \mathcal{W}^{n-1} \quad , \tag{1.69}$$

with the following normalisation constraints

$$\sum_{w} \mathrm{p}(w \,|\, h) = 1 \qquad \forall h \in \mathcal{W}^{n-1} \quad , \tag{1.70}$$

where $\mathcal{W}$ denotes the vocabulary.

[b] We have intentionally omitted the parameters that initialise the history for simplicity sake, for instance $p(w_1)$.

Due to the large number of free parameters and the always scarce data, it is a common approach to resort to smoothing techniques. For instance, for a trigram language model, the events that occur only once or not at all typically represent a huge percentage of the total occurrences. Therefore, the probabilities of these events are difficult to estimate with conventional methods.

For better understand the overfitting problem in the n-gram LM context, we must bare in mind the (conventional) MLE estimation. Given a training corpus, $w_1 \dots w_n \dots w_N$; we know in each text position, n, the observed word w_n and its conditional history h_n. In this case, the log-likelihood function is given by:

$$\begin{aligned} \mathrm{LL}(\{\mathrm{p}(w|h)\}) = & \quad \sum_{n=1}^{N} \log \mathrm{p}(w_n \,|\, h_n) & (1.71) \\ = & \quad \sum_{wh} N(w,h) \log \mathrm{p}(w \,|\, h) \quad , & (1.72) \end{aligned}$$

where in the last line we have changed the summation index by grouping the occurrences for the same word w and history h. We will henceforth denote by $N(w,h)$ to all the occurrences in the training set of a given n-gram, hw.

In order to obtain the MLE for the n-gram model, we must maximise $\mathrm{LL}(\{\mathrm{p}(w|h)\})$ constrained by Eq. (1.70). Applying some convex optimisation techniques [Ney et al., 1997], the MLE is computed as follows

$$\hat{\mathrm{p}}(w|h) = \frac{N(w,h)}{N(h)} \quad , \tag{1.73}$$

where $N(h)$ stands for the occurrences of the history h in the training corpus, that is to say

$$N(h) = \sum_{w} N(w,h) \quad . \tag{1.74}$$

It is seen that the n-gram parameter set is very sparse, leading to poorly estimated MLE probabilities. For instance, in Eq. (1.73), it is observed that for the unseen n-grams the probability is estimated as 0 even if all the words of the n-gram occur in the training data. This overfitting problem derived from the scarce training data, is one of the most important drawbacks of the n-gram model.

1.2.3 Leaving-one-out smoothing techniques

The smoothing techniques for n-gram models [Goodman, 2001] range from adding a pseudo-count to each occurrence count, to discounting a probability mass B_h from the seen n-grams for each history h and redistribute it according to a smoothing distribution, $\beta(\cdot \,|\, \bar{h})$. For instance, the linear interpolation [Chen and Goodman, 1998] distributes the gained probability mass B_h among all words according to the smoothing distribution. On the other hand, the backing-off redistributes the probability only among unseen events.

The most successful smoothing techniques are based on the *Turing-Good (TG) counts* [Good, 1953, Nadas, 1984] which are obtained by *leaving-one-out (LOO)*. Specifically, the modified Kneser-Ney, which obviously is a modified version of the Kneser-Ney smoothing [Kneser and Ney, 1995], obtains the best results [Goodman, 2001].

The main idea of the LOO-based smoothing models is to discount a probability mass from all the seen n-grams by means of a discounting parameter λ_r for counts $r < R$ being R the maximum count. Then, the gained probability mass, B_h, is redistributed among the unseen events according to a smoothing probability distribution $\beta(w \,|\, \bar{h})$. No probability is discounted from the most frequent n-gram, R. Therefore, in order to smooth the n-gram language model, we smooth the probability

estimates $\mathrm{p}(w|h)$ with the following smoothing model

$$\tilde{p}_{\boldsymbol{\lambda}}(w|h) := \begin{cases} \frac{R}{N(h)} & N(w,h) = R \\ (1-\lambda_r)\frac{r}{N(h)} & 0 < N(w,h) = r < R \\ B_h \beta(w|\bar{h}) & N(w,h) = 0 \end{cases} \quad , \tag{1.75}$$

where B_h is the discounted probability mass defined as follows

$$B_h = \sum_{r=1}^{R-1} \lambda_r n_r(h) \frac{r}{N(h)} \quad , \tag{1.76}$$

so that the probability defined in Eq. (1.75) sums up to 1, and where $n_r(h)$ are the counts-of-counts conditional to the previous history h, i.e.

$$n_r(h) = \sum_w \delta(r, N(w,h)) \quad . \tag{1.77}$$

The discounting probability mass, $\beta(w|\bar{h})$, is a lower order smoothing probability distribution defined over the unseen words, i.e.

$$\sum_{w:N(w,h)=0} \beta(w|\bar{h}) = 1 \quad . \tag{1.78}$$

The *leaving-one-out (LOO)* criterion [Katz, 1987] is based on the MLE criterion, where each sample plays the role of both training and testing. We summarise the basis of the formalisation found in [Ney et al., 1997, Sec. 4]. Firstly, equivalence classes are formed by gathering all n-grams hw which share the very same count $r = N(w,h)$ and history h, into the same equivalence class. Note that these equivalence classes simulate the result of the conventional MLE in Eq. (1.73), where all the n-grams with the same count share the same probability $\hat{\mathrm{p}}(w \,|\, h)$. Secondly, we count the number of different n-grams in each class labelled with the count r, $r = 0, 1, ..., R$; and denote them by $n_r(h)$ (see Eq. (1.77)). Finally, by leaving-one-out an n-gram observation in the class with count r for testing, it is moved into the class with count $r-1$. Thus, the associated probability is replaced with the probability of the class $r-1$, obtaining in this way the LOO probabilities $\tilde{p}_{loo}(w \,|\, h)$

$$\tilde{p}_{loo}(w|h) := \begin{cases} (1-\lambda_{r-1})\frac{r-1}{N(h)} & 1 < N(w,h) = r \leq R \\ B_h \beta(w|\bar{h}) & N(w,h) = 1 \end{cases} \quad . \tag{1.79}$$

If this process is repeated for all occurrences and for all equivalence classes $r = 1, ..., R$; then, the LOO log-likelihood criterion is obtained

$$\mathrm{F}(\{\boldsymbol{\lambda}_1^{R-1}\}) = \sum_{wh} N(w,h) \log \tilde{p}_{loo}(w \,|\, h) \tag{1.80}$$

$$= \sum_{r=2}^{R} r n_r \log(1-\lambda_{r-1}) + \sum_h n_1(h) \log \left(\sum_{r=1}^{R-1} \lambda_r n_r(h) r \right) + \mathrm{const}(\boldsymbol{\lambda}_1^{R-1}) \quad , \tag{1.81}$$

where n_r stands for the the counts-of-counts unconditional to any previous history, i.e.,

$$n_r = \sum_h n_r(h) = \sum_{wh} \delta(r, N(w,h)) \quad . \tag{1.82}$$

However, Eq. (1.81) is very difficult to deal with; furthermore, no close solution for λ_r is known. Therefore, we make the following assumption [Ney et al., 1997, pag. 186]

$$\sum_{r=1}^{R-1} \lambda_r n_r(h) r = \phi_h \sum_{r=1}^{R-1} \lambda_r n_r r \quad , \tag{1.83}$$

where ϕ_h is a constant value depending on the previous history h but not in the counts r.

After taking assumption in Eq. (1.83) into Eq. (1.81), the function to maximise is given by

$$\mathrm{F}(\boldsymbol{\lambda}_1^{R-1}) = \sum_{r=2}^{R} r n_r \log(1-\lambda_{r-1}) + n_1 \log\left(\sum_{r=1}^{R-1} \lambda_r n_r r\right) + \mathrm{const}(\boldsymbol{\lambda}_1^{R-1}) \quad , \tag{1.84}$$

for which the solution is given by [Ney et al., 1997, pag. 186]

$$\hat{\lambda}_r = 1 - \frac{r^\star}{r}(1 - n_R R/N) \quad , \tag{1.85}$$

where $r^\star$ stands for the Good-Turing count [Good, 1953, Nadas, 1984, Ney et al., 1997]

$$r^\star = \frac{n_{r+1}(r+1)}{n_r} \quad , r = 1, 2, \ldots, R-1 \quad , \tag{1.86}$$

and abusing of notation, $R^\star = R$. If we further assume that $n_R R/N \ll 1$, then Eq. (1.85) simplifies to

$$\hat{\lambda}_r = 1 - \frac{r^\star}{r} \quad . \tag{1.87}$$

Finally, the solution to the smoothed model is given by plugging Eq. (1.87) into Eq. (1.75), i.e.,

$$\tilde{p}(w|h) := \begin{cases} \frac{r^\star}{N(h)} & 0 < N(w,h) = r \leq R \\ B_h \beta(w|\bar{h}) & N(w,h) = 0 \end{cases} \quad , \tag{1.88}$$

where the smoothing distribution $\beta(w \,|\, \bar{h})$ is also estimated by LOO [Ney et al., 1997].

It is very illustrative to define the discounted probability mass B_h in terms of the TG counts. Using the smoothing model solution in Eq. (1.88), and the definition of the discounting probability mass in Eq. (1.76); the former B_h is computed as follows

$$\begin{aligned} B_h &= 1 - \sum_{r=1}^{R} n_r(h) \frac{r^\star}{N(h)} \\ &= \frac{1}{N(h)} \left(N(h) - \sum_{r=1}^{R} n_r(h) r^\star \right) \quad , \end{aligned} \tag{1.89}$$

where taking into account the following property

$$N(h) = \sum_{r=1}^{R} r n_r(h) \quad , \tag{1.90}$$

B_h is expressed as

$$B_h = \frac{1}{N(h)} \left(\sum_{r=1}^{R} r n_r(h) - \sum_{r=1}^{R} r^\star n_r(h) \right) \quad , \tag{1.91}$$

and grouping common terms

$$B_h = \frac{1}{N(h)} \sum_{r=1}^{R} (r - r^\star) n_r(h) \quad . \tag{1.92}$$

Note that the question of whether the normalisation constraints in Eq. (1.70) are satisfied depends on B_h and, hence, on the TG counts $r^\star$. This is due to the assumption in Eq. (1.83), since the definition of our model ensured these normalisation constraint to be verified.

It may be said that the conditional dependence of the counts on h is dropped when the assumption in Eq. (1.83) is taken; resembling, in this way, the solution of a joint smoothing model [Ney et al., 1997]. Consequently, we should analyse a joint smoothing model to see whether this statement is true or not. The joint smoothing model approximates the joint probabilities $\tilde{p}_{\boldsymbol{\lambda}}(w, h)$ instead of the conditional ones, $\tilde{p}_{\boldsymbol{\lambda}}(w|h)$, as follows

$$\tilde{p}_{\boldsymbol{\lambda}}(w, h) := \begin{cases} \frac{R}{N} & N(w, h) = R \\ (1 - \lambda_r)\frac{r}{N} & 0 < N(w, h) = r < R \\ B\beta(w, \bar{h}) & N(w, h) = 0 \end{cases} \tag{1.93}$$

with the gained probability B independent from the previous history as follows

$$B = \frac{1}{N} \sum_{r=1}^{R-1} \lambda_r n_r r \quad . \tag{1.94}$$

Note that both Eqs. (1.93) and (1.94) are analogous to Eqs. (1.75) and (1.76), except for the normalisation constant that is $N(h)$ in the latter and has been replaced for N in the former.

If we apply LOO to the joint smoothing model in Eq. (1.93), then the solution in Eq. (1.85) is obtained without any assumption. Therefore, we conclude that taking assumption in Eq. (1.83) with the conditional model in Eq. (1.75) is equivalent to taking the assumption of optimising the parameters $\boldsymbol{\lambda}_1^{R-1}$ for maximising the joint LOO log-likelihood function in Eq. (1.81) and, then, use them as if they were the optimal parameters for the conditional model in Eq. (1.75). In theory, this assumption can degenerate the probabilities as commented above, not ensuring the normalisation constraints in Eq. (1.70). Moreover, in practice, we are interested in maximising the conditional model, since it would produce smaller perplexities and eventually, this should improve the system performance. The magnitude of this assumption mainly depends on the finally behaviour of these smoothing models.

It is worth noting that we have broadly reviewed the standard formulation given in [Ney et al., 1997] for introducing the Turing-Good counts. Furthermore, the Kneser-Ney (KN) smoothing [Kneser and Ney, 1995] is a special case of the model defined in Eq. (1.75) that ties all the parameters $\boldsymbol{\lambda}_1^{R-1}$ with one discounting parameter b, i.e.,

$$\lambda_r(b) = \frac{b}{r} \quad , \tag{1.95}$$

leaving just one free parameter to estimate: the former discounting parameter, b. Furthermore, an upper and lower bound to this parameter is obtained by LOO [Ney et al., 1997]

$$\frac{n_1}{n_1 + 2n_2 + \sum_{r \geq 3} n_r} < b < \frac{n_1}{n_1 + 2n_2} \quad . \tag{1.96}$$

In this case, similarly to other smoothing techniques, both joint and conditional smoothing models lead to the same solution, without degrading the probabilities by not verifying the normalisation constraints in Eq. (1.70).

1.2.4 Language modelling for text classification

The unigram language model is a special case of the n-gram language model, for $n = 1$. In such case, the probability for a given sentence or text $\boldsymbol{w}_1^T$ is given by

$$p_r(\boldsymbol{w}) := \prod_{t=1}^{T} \mathrm{p}(w_t) \quad , \tag{1.97}$$

where we have assumed that the occurrence probability of each word is independent of its position and other words, the so-called *Naive Bayes assumption*. Note that if the probability of a text is given by Eq.(1.97), then the count vector of words, $\boldsymbol{x}$, such that $x_d = N_{\boldsymbol{w}}(v_d)$ is the number of occurrences of the word v_d in the text $\boldsymbol{w}$, follows a multinomial distribution, i.e.,

$$p_{\boldsymbol{\theta}}(\boldsymbol{x}|L) = \left(\begin{array}{c} L \\ \boldsymbol{x} \end{array} \right) \prod_{d=1}^{D} \theta_d^{x_d} \quad , \tag{1.98}$$

where θ_d stands for the probability of the word v_d to occur and, hence, they must sum up to one

$$\sum_{d=1}^{D} \theta_d = 1 \quad , \tag{1.99}$$

with the definition

$$\left(\begin{array}{c} L \\ \boldsymbol{x} \end{array} \right) = \frac{L!}{\prod_{d=1}^{D} x_d!} \quad . \tag{1.100}$$

The *naive Bayes* language model has long been a core technique in information retrieval and, more recently, it has attracted significant interest in pattern recognition and machine learning [Lewis, 1998]. This technique is specially outstanding in text classification [Juan and Ney, 2002, Vilar et al., 2004]. In Chapter 2, the naive Bayes language model is further analysed.

1.3 Statistical Machine Translation

In this Section we review state-of-the-art applications and approaches in the field of *machine translation (MT)*, focusing on the statistical approach. The goal of MT is the automatic translation of a source sentence $\boldsymbol{x}$ into a target sentence $\boldsymbol{y}$,

$$\begin{aligned} \boldsymbol{x} &= x_1 \dots x_j \dots x_J, \quad x_j \in \boldsymbol{X}, \quad j = 1, \dots, J \\ \boldsymbol{y} &= y_1 \dots y_i \dots y_I, \quad y_i \in \boldsymbol{Y}, \quad i = 1, \dots, I \end{aligned}$$

where x_j and y_i denote source and target words; and $\boldsymbol{X}$ and $\boldsymbol{Y}$, the source and target vocabularies respectively.

On the one hand, current MT technology is focused on three main applications:

- Fully-automatic MT in limited domains like weather forecast [Langlais et al., 2005], hotel reception desk [Amengual et al., 2000b], appointment scheduling, etc.
- Post-editing for CAT, i.e., post-editing the human amendment of automatic translations produced by an MT system.
- Understandable rough translation in which the aim is to allow a human to decide whether the translated text includes relevant information. For instance, this is used for document finding purposes or user assistance in software troubleshooting.

- Interactive machine translation where a synergy between the user and the system is achieved by restating the user interaction as an iterative process in which the user corrects the translations given by the system which proposes a new hypothesis for the unvalidated part of the translation in its turn.

On the other hand, state-of-the-art MT approaches can be classified according to the level of analysis of the source sentence before translating:

- The interlingua approach [Arnold et al., 1993, Nirenburg et al., 1992, Nyberg and Mitamura, 1992].
- The transfer approach decomposes the translation process into three steps: analysis, transfer and generation. A review of transfer-based systems is presented in [Hutchins and Somers, 1992].
- The direct approach refers to the word-by-word translation from the source sentence into the target sentence. Under this approach we find example-based MT and statistical MT.

In *statistical machine translation (SMT)*, this translation process is usually presented as a statistical pattern recognition problem in which for a given source sentence $\boldsymbol{x}$, the optimal target sentence $\hat{\boldsymbol{y}}$ is searched according to

$$\hat{\boldsymbol{y}} = \arg\max_{y} \; p_r(\boldsymbol{y} \mid \boldsymbol{x}) \quad , \tag{1.101}$$

where $p_r(\boldsymbol{y} \mid \boldsymbol{x})$ is the probability for $\boldsymbol{y}$ to be the actual translation of $\boldsymbol{x}$. Note that Eq. (1.101) is simply the adoption of the Bayes' optimal classification rule in Eq. (1.9) into the machine translation scope.

The so-called *search problem* is to compute a target sentence $\hat{\boldsymbol{y}}$ for which this probability is maximum. Applying Bayes' theorem we can reformulate Eq. (1.101) as follows

$$\hat{\boldsymbol{y}} = \arg\max_{y} \; p_r(\boldsymbol{x} \mid \boldsymbol{y}) p_r(\boldsymbol{y}) \quad , \tag{1.102}$$

where the term $p(\boldsymbol{y} \mid \boldsymbol{x})$ has been decomposed into a *translation model* $p_r(\boldsymbol{x} \mid \boldsymbol{y})$ and a *language model* $p_r(\boldsymbol{y})$. Intuitively, the translation model is responsible for modelling the correlation between source and target sentence, but it can also be understood as a mapping function from target to source words. Whereas the language model $p_r(\boldsymbol{y})$ represents the well-formedness of the candidate translation $\boldsymbol{y}$ [Stolcke, 2002].

The application of Eq. (1.101), minimises the CER which in MT scope corresponds to *the sentence error rate (SER)*. However, the SER measure provides a rough and superficial evaluation of the system translation quality and it is rarely used in favour of other more popular evaluation measures described in Section 1.3.3.

The search problem presented in Eq. (1.102) was proved to be an NP-complete problem [Knight, 1999, Udupa and Maji, 2006]. However various research groups have developed efficient search algorithms by using suitable simplifications and applying optimisation methods. Starting from the IBM work based on a stack-decoding algorithm [Berger et al., 1996], greedy [Berger et al., 1994, Germann et al., 2001, Wang and Waibel, 1998] and integer-programming [Germann et al., 2001] approaches to dynamic-programming search [García-Varea and Casacuberta, 2001, Tillmann and Ney, 2003].

Nevertheless, most of the current statistical MT systems present an alternative modelling of the translation process different from that presented in Eq. (1.101). The posterior probability is modelled as a log-linear combination of feature functions [Och and Ney, 2004] under the framework of maximum entropy [Berger et al., 1996]

$$\hat{\boldsymbol{y}} = \arg\max_{y} \sum_{m=1}^{M} \lambda_m h_m(\boldsymbol{x}, \boldsymbol{y}) \quad , \tag{1.103}$$

where λ_m is the interpolation weight and $h_m(\boldsymbol{x}, \boldsymbol{y})$ is a function that assigns a score to the sentence pair $(\boldsymbol{x}, \boldsymbol{y})$. Examples of features range from $h_m(\boldsymbol{x}, \boldsymbol{y}) = \log p_r(\boldsymbol{x} \mid \boldsymbol{y})$ or $h_m(\boldsymbol{x}, \boldsymbol{y}) = \log p_r(\boldsymbol{y})$, to

$h_m(\boldsymbol{x}, \boldsymbol{y}) = \exp(1)$. Note that under this framework, Eq. (1.102) is a particular case where

$$h_1(\boldsymbol{x}, \boldsymbol{y}) = \log p_r(\boldsymbol{x} \mid \boldsymbol{y}) \tag{1.104}$$

$$h_2(\boldsymbol{x}, \boldsymbol{y}) = \log p_r(\boldsymbol{y}) \quad , \tag{1.105}$$

and $\lambda_1 = \lambda_2 = 1$.

It is particularly worthy of note that Eq. (1.103) is quite similar to a *log-linear* model depicted in Eq. (1.12) without the normalisation coefficient $Z_{\boldsymbol{\theta}}(\boldsymbol{x})$. For this reason, these models are commonly refereed to as log-linear models [Och and Ney, 2004]. However, note that the ellipsis of the normalisation constant plays an interesting role in these translations models since its omission is conserved through the training process as well, in contrast to the standard log-linear models. We will further analyse these differences in Chapter 4.

1.3.1 Statistical word-based translation systems

A great variety of statistical translation models have been proposed since the word alignment models were proposed [Brown et al., 1993a, 1990]. Most of state-of-the-art statistical MT systems are based on bilingual phrases [Callison-Burch et al., 2007]. These bilingual phrases are sequences of words in the two languages and not necessarily phrases in the linguistic sense. The phrase-based approach to MT is further explored in Section 1.3.2.

Another approach which has become popular in recent years is grounded on the integration of syntactic knowledge into statistical MT systems [Ding and Palmer, 2005, Graehl and Knight, 2004, Lin, 2004, Wu, 1996, Yamada and Knight, 2001]. This approach parses the sentence in one or both of the involved languages, defining then, the translation operations on parts of the parse tree. In [Chiang, 2007], Chiang constructs hierarchical transducers for translation. The model is a syntax-free grammar which is learnt from a bilingual corpus without any syntactic information. It consists of phrases which can contain sub-phrases, so that a hierarchical structure is induced.

The third main approach, which is currently investigated in statistical MT, is the modelling of the translation process as a finite-state transducer [Alshawi et al., 2000, Bangalore and Riccardi, 1995, Casacuberta and Vidal, 2004, Kanthak and Ney, 2004, Mariño et al., 2006]. This approach solves the translation problem by estimating a language model on sentences of extended symbols derived from the association of source and target words coming from the same bilingual pair. The translation transducer is basically an acceptor for this language of extended symbols.

In this section we briefly review the word based models presented in Brown et al. [1993a]. In this work, the models were presented in its inverse way, i.e., $p_r(\boldsymbol{x} \mid \boldsymbol{y})$. However, since in Chapter 4 we make use of direct translation models, we present here the IBM models in its corresponding direct way, i.e., $p_r(\boldsymbol{y} \mid \boldsymbol{x})$. In the direct version of IBM models, the translation of a source sentence $\boldsymbol{x}$ into a target sentence $\boldsymbol{y}$, is carried out using *alignments* between words, i.e. a target word y_i is aligned to the set of source word positions $\boldsymbol{a}_i = \{j_1, \ldots, j_l\}$, if the target word is directly generated as translation of the source word group $x_{j_1}, \ldots, x_{j_l}$. This model requires the use of a hidden variable model since the alignments are typically never seen in training

$$p_r(\boldsymbol{y} \mid \boldsymbol{x}) = p_r(I \mid \boldsymbol{x}) \sum_{\boldsymbol{a}_1} \cdots \sum_{\boldsymbol{a}_I} p_r(\boldsymbol{y}, \boldsymbol{a}_1^I \mid \boldsymbol{x}, I) \quad , \tag{1.106}$$

where $\boldsymbol{a}_i$ is the alignment vector that indicates which source words are aligned with the i-th target word y_i, i.e.

$$\boldsymbol{a}_i \subseteq \{1, \ldots, J\} \quad , \tag{1.107}$$

and where $p_r(I \mid \boldsymbol{x})$ is a length distribution which is usually uniformly modelled, and consequently ignored.

Some constraints are usually added to the alignment sets $\boldsymbol{a}_1^I$, due to practical restrictions. For instance, the *coverage constraint* requires all the source words to be in at least one alignment set.

The complete probability model in Eq. (1.106), $p_r(\boldsymbol{y}, \boldsymbol{a}_1^I \,|\, \boldsymbol{x})$, can be decomposed left to right as follows

$$p_r(\boldsymbol{y}, \boldsymbol{a}_1^I \,|\, \boldsymbol{x}, I) = \prod_i p_r(\boldsymbol{a}_i \,|\, \boldsymbol{x}, \boldsymbol{a}_1^{i-1}, \boldsymbol{y}_1^{i-1}, I) p_r(y_i \,|\, \boldsymbol{x}, \boldsymbol{a}_1^i, \boldsymbol{y}_1^{i-1}, I) \quad , \tag{1.108}$$

where two probabilities are used:

- The alignment probability $p_r(\boldsymbol{a}_i \,|\, \boldsymbol{x}, \boldsymbol{a}_1^{i-1}, \boldsymbol{y}_1^{i-1}, I)$
- The translation dictionary probability $p_r(y_i \,|\, \boldsymbol{x}, \boldsymbol{a}_1^i, \boldsymbol{y}_1^{i-1}, I)$

Different alignment models were proposed in [Brown et al., 1993b] (in its inverse form) based on this idea, although only 2 models where directly modelled constraining the probabilities in Eq. (1.108) directly. These two models constrain the alignment sets cardinality to 1 or 0, that is to say each target word can be aligned to either one word or no word at all. In order to simplify notation, we redefine the alignment variables since each alignment is composed of one word. Therefore, we say that $a_i = j$ if the target word y_i is "aligned" to the source word x_j, where j can be any source position ($\{1, \ldots, J\}$) or 0 indicating that y_i is not aligned to any word. In order to represent the "non-alignment" event, a NULL word is introduced at the beginning of $\boldsymbol{x}$, i.e. $\boldsymbol{x} = x_0 x_1 \cdots x_J$. If a target word y_i is aligned to x_0 ($a_i = 0$), the so-called NULL word, then it is equivalent to say that this target word y_i is not aligned to any source word.

IBM model 1

The first of the IBM models, the so-called IBM model 1, is essentially defined as a statistical bilingual dictionary.

The IBM model 1 [Brown et al., 1993b] makes the following assumptions

- The alignment probability is uniform, i.e.

$$p_r(\boldsymbol{a}_i \,|\, \boldsymbol{x}, \boldsymbol{a}_1^{i-1}, \boldsymbol{y}_1^{i-1}, I) := \frac{1}{J+1} \quad . \tag{1.109}$$

- The dictionary probability depends only on the aligned word, i.e.

$$p_r(y_i \,|\, \boldsymbol{x}, \boldsymbol{a}_1^i, \boldsymbol{y}_1^{i-1}, I) := \mathrm{p}(y_i \,|\, x_{a_i}) \quad , \tag{1.110}$$

  where note that we have introduced the notation p to refer to parameters, and where the following normalisation constraint must be verified

$$\sum_b \mathrm{p}(b \,|\, a) = 1 \qquad \forall a \in \boldsymbol{X} \quad . \tag{1.111}$$

Taking into account the assumptions in Eqs. (1.109), and (1.110), the model probability is given by

$$p_r(\boldsymbol{y} \,|\, \boldsymbol{x}) := \left(\frac{1}{J+1}\right)^I \prod_i \sum_{j=0}^{J} \mathrm{p}(y_i \,|\, x_j) \quad . \tag{1.112}$$

Since the model is a hidden variable model, the EM algorithm [Dempster et al., 1977b] is used to estimate the parameter set: $\boldsymbol{\Theta} = \{\mathrm{p}(b \,|\, a) \,|\, b \in \boldsymbol{Y}, a \in \boldsymbol{X}\}$.

The aim of the IBM model 1 typically is to initialise the training of superior IBM models. Another interesting property of the IBM model 1 is the concavity of its log-likelihood function, and therefore the

uniqueness of a maximum value under non-degenerated[c] initialisation. However, the IBM model 1 has been widely applied to different tasks of statistical MT, cross-lingual information retrieval and bilingual TC due to its simplicity and applicability of its parameter values.

In statistical MT, the IBM model 1 has traditionally been an important ingredient in applications such as the alignment of bilingual sentences [Moore, 2002], the alignment of syntactic tree fragments [Ding et al., 2003], the segmentation of bilingual long sentences for improved word alignment [Nevado et al., 2003], the extraction of parallel sentences from comparable corpora [Munteanu et al., 2004], the estimation of word-level confidence measures [Ueffing and Ney, 2007] and serves as inspiration for lexicalised phrase scoring in phrase-based systems [Koehn, 2005, Koehn et al., 2003]. Furthermore, it has also received attention to improve non-structural problems [Moore, 2004].

IBM model 2

The IBM model 2 is an extension of the IBM model 1 where the alignment probability is not uniformly modelled. Specifically, the IBM model 2 parametrises the alignment probability as follows

$$p_r(\boldsymbol{a}_i \,|\, \boldsymbol{x}, \boldsymbol{a}_1^{i-1}, \boldsymbol{y}_1^{i-1}, I) := \mathrm{p}(a_i \,|\, i, I, J) \tag{1.113}$$

where the following normalisation constraint must be verified

$$\sum_j \mathrm{p}(j \,|\, i, I, J) = 1 \tag{1.114}$$

Taking into account the assumptions in Eqs. (1.110), and (1.113), the model probability is given by

$$p_r(\boldsymbol{y} \,|\, \boldsymbol{x}) := \prod_i \sum_j \mathrm{p}(j \,|\, i, I, J)\, \mathrm{p}(y_i \,|\, x_j) \quad . \tag{1.115}$$

Since the model is a hidden variable model, the EM algorithm is used to estimate the parameter set, $\{\mathrm{p}(b \,|\, a), \mathrm{p}(j \,|\, i, I, J)\}$. In order to train this model, firstly, some iterations of the IBM model 1 are performed in order to obtain good dictionary estimates. Afterwards a retraining is performed using the EM update equations for the IBM model 2.

1.3.2 Statistical phrase-based translation systems

The basis of the mainstream and better statistical machine translation models are based on the so-called phrase-based models. The idea of modelling the translation process using phrase dictionaries was firstly introduced in the alignment template approach [Och and Ney, 2004]. In this section we review several proposed phrase-based models.

Generative phrase-based models

We outline here an example of generative phrase-based model that will serve us to present the problems faced by this approach, and to motivate the introduction of heuristically estimated phrase-based systems. We follow the model presented in Zens et al. [2002].

Let $(\boldsymbol{x}, \boldsymbol{y})$ be a pair of source-target sentences, we introduce the conventional conditional probability $p(\boldsymbol{y} \,|\, \boldsymbol{x})$ for the translation model. Let assume that $\boldsymbol{x}$ has been divided into T phrases or segments; and so has $\boldsymbol{y}$. We further assume that each source phrase has been generated by just one target phrase. We unhidde the hidden variable B which is a segmentation of the bilingual segmentation pair $(\boldsymbol{x}, \boldsymbol{y})$

[c] Starting point in which none of the initial parameter values is zero.

into T phrases $(\tilde{\boldsymbol{x}}_1^T, \tilde{\boldsymbol{y}}_1^T)$. Note that, the source segments, $\tilde{\boldsymbol{x}}_1, \ldots, \tilde{\boldsymbol{x}}_T$, are not required to be in the same source order, i.e., $\boldsymbol{x}$ could be different from $\tilde{\boldsymbol{x}}_1^T = \tilde{\boldsymbol{x}}_1 \cdots \tilde{\boldsymbol{x}}_T$ although it must be a reordering of it. Finally, a generative model can be seen as *a full exploration of all possible bilingual segmentation of $\boldsymbol{x}$ and $\boldsymbol{y}$ and all possible alignments between them,*

$$p_r(\boldsymbol{y} \mid \boldsymbol{x}) = \sum_B p_r(\boldsymbol{y}, B \mid \boldsymbol{x}) \tag{1.116}$$

$$= \sum_B p_r(B \mid \boldsymbol{x}) p_r(\boldsymbol{y} \mid B, \boldsymbol{x}) \quad , \tag{1.117}$$

where $p_r(\boldsymbol{y} \mid B, \boldsymbol{x})$ is modelled using a phrase-table

$$p_r(\boldsymbol{y} \mid B, \boldsymbol{x}) := \prod_{t=1}^{T} \mathrm{p}(\tilde{\boldsymbol{y}}_k \mid \tilde{\boldsymbol{x}}_k) \quad , \tag{1.118}$$

whereas the remaining probability in Eq. (1.117), usually ignored, i.e., uniformly modelled for all possible target phrase reordering.

The estimation of a phrase-based model as that presented above is a cumbersome problem that possess not only computational efficiency challenges, but also overwhelming data requirements. One of the main difficulties that phrase-based models have to cope with is the problem of the bilingual segmentation and reordering. In the model proposed above, this segmentation is modelled by the hidden variable B, which leads us to a large combinatorial number of possible segmentations to explore. As can be guessed, these problems are further aggravated with the length of the source and target sentence. Despite this obstacle, there have been several proposals for phrase-based models, from the joint probability model [Birch et al., 2006, Marcu and Wong, 2002], over the HMM phrase-based models [Andrés-Ferrer and Juan, 2007, Deng and Byrne, 2005] to the statistical GIATI model [Andrés-Ferrer et al., 2008].

However, the most popular approach to the development of phrase-based systems has been the log-linear combination of heuristically estimated phrase-based models [Koehn et al., 2003, Och and Ney, 2004], since these systems offer better performance than those based on generative phrase-based models [DeNero et al., 2006].

Heuristic phrase-based models

The heuristic estimation of phrase-based models is grounded on the Viterbi alignments computed as a byproduct of word-based alignment models. The Viterbi alignment is defined as the most probable alignment given the source and target sentences and an estimation of the model parameters $\boldsymbol{\theta}$,

$$\hat{\boldsymbol{a}} = \arg\max_{\boldsymbol{a}} \mathrm{p}_{\boldsymbol{\theta}}(\boldsymbol{a} \mid \boldsymbol{x}, \boldsymbol{y}) \quad , \tag{1.119}$$

can also be rewritten

$$\hat{\boldsymbol{a}} = \arg\max_{\boldsymbol{a}} \mathrm{p}_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{a} \mid \boldsymbol{y}) \quad , \tag{1.120}$$

or

$$\hat{\boldsymbol{a}} = \arg\max_{\boldsymbol{a}} \mathrm{p}_{\boldsymbol{\theta}}(\boldsymbol{y}, \boldsymbol{a} \mid \boldsymbol{x}) \quad . \tag{1.121}$$

For instance, the conventional alignments, those provided by IBM models, disallow the connection of a source word with more than one target word. This unrealistic limitation negates the common linguistic phenomenon in which a word in one language is translated into more than one word in another language. To circumvent this problem, alignments are not only computed from the source language to the target language, but also from the target language to the source language. Doing so, we can reflect the fact that a single word is connected to more than one word.

Once the Viterbi alignments have been computed in both directions, there exist different heuristic algorithms to combine[d] them [Koehn et al., 2003, Och and Ney, 2003]. These algorithms range from the intersection of both alignments in which we have high precision, but low recall alignments, to the union in which we have low precision, but high recall. In between, there are algorithms like the refined method [Och and Ney, 2003] and the *grow-diag-final* [Koehn et al., 2003] that starting from the intersection, heuristically add additional alignment points taken from the union. This is a previous step, before extracting bilingual phrases, to construct a phrase-based system.

Bilingual phrase extraction is based on the concept of *consistency* of a bilingual phrase $(\overline{x}, \overline{y})$ (derived from a bilingual segmentation) with a word alignment a. Formally,

$$\begin{aligned}(\overline{\boldsymbol{x}}, \overline{\boldsymbol{y}}) \text{ consistent with } \boldsymbol{a} \Leftrightarrow \quad & \forall x_j \in \overline{\boldsymbol{x}} : (x_j, y_i) \in \boldsymbol{a} \longrightarrow y_i \in \overline{\boldsymbol{y}} \wedge \\ & \forall y_i \in \overline{\boldsymbol{y}} : (x_j, y_i) \in \boldsymbol{a} \longrightarrow x_j \in \overline{\boldsymbol{x}} \wedge \\ & \exists x_j \in \overline{\boldsymbol{x}},\ y_i \in \overline{\boldsymbol{y}} : (x_j, y_i) \in \boldsymbol{a} \end{aligned} \tag{1.122}$$

basically Eq. (1.122) means that a bilingual phrase is consistent if and only if all the words in the source phrase are aligned to words in the target phrase, and there is at least one word in the source phrase aligned to a word in the target phrase.

Given the definition of consistency, all bilingual phrases (up to a maximum phrase length) that are consistent with the alignment resulting from the symmetrisation process are extracted.

The next step is to define functions that assign a score or a probability to a bilingual phrase in isolation or as part of a sequence of bilingual phrases in a given segmentation. These score functions are integrated in a log-linear fashion under the maximum entropy framework.

The most commonly used score functions are the direct and inverse phrase translation probability estimated as a relative frequency

$$\mathrm{p}_d(\boldsymbol{u} \mid \boldsymbol{v}) = \frac{count(\boldsymbol{u}, \boldsymbol{v})}{\sum\limits_{\boldsymbol{u}'} count(\boldsymbol{u}', \boldsymbol{v})} \qquad \mathrm{p}_i(\boldsymbol{v} \mid \boldsymbol{u}) = \frac{count(\boldsymbol{u}, \boldsymbol{v})}{\sum\limits_{\boldsymbol{v}'} count(\boldsymbol{u}, \boldsymbol{v}')} \tag{1.123}$$

where $\boldsymbol{u}$ stands for a source phrase, and $\boldsymbol{v}$ for a target phrase. A direct and inverse lexical translation probability inspired in the IBM model 1 [Cohn and Lapata, 2007, Koehn et al., 2003] are also used in the log-linear model. Other score functions are related to reordering capabilities, such as the distance-based reordering model [Och and Ney, 2004] and the lexicalised reordering model [Koehn et al., 2005]. Additional score functions are the phrase and the word penalty to control the length of the translated sentence.

The weight of each score function in the log-linear combination is adjusted on a development set with respect to a predefined criterion, usually BLEU. There are two popular techniques in statistical MT to carry out this process, minimum error rate training [Och, 2003] and minimum Bayes risk [Kumar and Byrne, 2004]. Furthermore, the most common approach to the decoding process in log-linear models is the well-known multi-stack decoding algorithm [Koehn, 2004, Och and Ney, 2004, Ortiz et al., 2006]. The Moses toolkit [Koehn et al., 2007], that implements an instantiation of this type of multi-stack decoding algorithms, will be used throughout this thesis to define a baseline reference.

1.3.3 Automatic MT evaluation metrics

In MT, the use of automatic evaluation metrics is imperative due to the high cost of human made evaluations. Also the need of rapid assessment of the translation quality of an MT system during its development and tuning phases is another reason for the usage of automatic metrics. These metrics are

[d] This process is also known as symmetrisation.

used under the assumption that they correlate well with human judgements of translation quality. This arguable statement must be considered bearing in mind the low inter-annotator agreement on translation quality [Callison-Burch et al., 2007]. This fact makes automatic evaluation an open challenge in MT.

In this thesis, we mainly use two conventional translation evaluation metrics, WER and BLEU, although other measures like METEOR [Banerjee and Lavie, 2005] and translation edit rate (TER) [Snover et al., 2006] are becoming more and more popular.

The WER metric [Amengual et al., 2000a, Casacuberta et al., 2004] is defined as the minimum number of word substitution, deletion and insertion operations required to convert the target sentence provided by the translation system into the reference translation, divided by the number of words of the reference translation. It can also be seen as the ratio of the edit distance between the system and the reference translation, and the number of words of the reference translation. This metric will allow us to compare our results to previous work on the same task. Even though the WER metric can value more than 100, it will be expressed as a percentage as it is commonly presented in the SMT literature. The WER metric can also be evaluated with respect to multiple references, however, in this thesis, we have a single reference translation at our disposal.

The BLEU score [Papineni et al., 2001] is the geometric mean of the modified[e] precision for different order of n-grams (usually from unigram up to 4-grams) between the target sentence and the reference translation, multiplied by an exponential brevity penalty (BP) factor that penalises those translations that are shorter than the reference translation. Although some voices have been raised against BLEU as the dominant evaluation methodology over the past years [Callison-Burch et al., 2006], it is still a reference error measure for the evaluation of translation quality in MT systems. The BLEU ranges from 0.0 (worst case) to 1.00 (best case), however, it is a common practice referred as a percentage ranging from 0.0 (worst score) to 100.0 (best score).

[e]The number of occurrences of a word in a target sentence is limited to that of this word in the reference translation.

Bibliography

H. Alshawi, S. Bangalore, and S. Douglas. Learning dependency translation models as collections of finite state head transducers. *Computational Linguistics*, 26, 2000.

J. C. Amengual et al. The EuTrans-I speech translation system. *Machine Translation*, 15:75–103, 2000a.

J.C. Amengual, J.M. Benedí, F. Casacuberta, A. Castaño, A. Castellanos, V. Jiménez, D. Llorens, A. Marzal, M. Pastor, F. Prat, E. Vidal, and J.M. Vilar. The EuTrans-I speech translation system. *Machine Translation*, 15:75–103, 2000b.

J. Andrés-Ferrer and A. Juan. A phrase-based hidden markov model approach to machine translation. In *Proceedings of New Approaches to Machine Translation*, pages 57–62, January 2007. ISBN 978-90-814861-0-1.

J. Andrés-Ferrer, A. Juan, and F. Casacuberta. Statistical estimation of rational transducers applied to machine translation. *Applied Artificial Intelligence*, 22(1-2): 4–22, 2008.

D.J. Arnold et al. *Machine Translation: an Introductory Guide*. Blackwells-NCC, London, 1993.

L. R. Bahl, F. Jelinek, and R. L. Mercer. Some statistical-estimation methods for stochastic. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2:179–190, March 1983.

S. Banerjee and A. Lavie. METEOR: An automatic metric for MT evaluation with improved correlation with human judgments. In *Proc. of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*, pages 65–72, Ann Arbor, Michigan, USA, June 2005. Association for Computational Linguistics.

S. Bangalore and G. Riccardi. A finite-state approach to machine translation. In *Proc. of NAACL'01*, pages 1–8, Morristown, NJ, USA, June 1995. Association for Computational Linguistics.

J.M. Benedí and J.A. Sánchez. Estimation of stochastic context-free grammars and their use as language models. *Computer Speech and Language*, 19(3):249–274, 2005.

Adam L. Berger, Vincent J. Della Pietra, and Stephen A. Della Pietra. A maximum entropy approach to natural language processing. *Comput. Linguist.*, 22 (1):39–71, 1996. ISSN 0891-2017.

A.L. Berger et al. The candide system for machine translation. In *Proc. of HLT'94*, pages 157–162, Morristown, NJ, USA, 1994. Association for Computational Linguistics. ISBN 1-55860-357-3.

A. Birch, C. Callison-Burch, Miles M. Osborne, and P. Koehn. Constraining the phrase-based, joint probability statistical translation model. In *Proceedings on the Workshop on Statistical Machine Translation*, pages 154–157, New York City, New York, USA, June 2006. Association for Computational Linguistics.

P. F. Brown et al. The Mathematics of Statistical Machine Translation: Parameter Estimation. *Computational Linguistics*, 19(2):263–311, 1993a.

Peter F. Brown, John Cocke, Stephen Della Pietra, Vincent J. Della Pietra, Frederick Jelinek, John D. Lafferty, Robert L. Mercer, and Paul S. Rossin. A statistical approach to machine translation. *Computational Linguistics*, 16(2):79–85, 1990.

Peter F. Brown, Stephen Della Pietra, Vincent J. Della Pietra, and Robert L. Mercer. The mathematics of statistical machine translation: Parameter estimation. *Computational Linguistics*, 19(2):263–312, 1993b.

C. Callison-Burch, M. Osborne, and P. Koehn. Re-evaluating the role of bleu in machine translation research. In *Proc. of ACL'06*, pages 249–256, Trento, Italy, April 2006. Association for Computational Linguistics.

C. Callison-Burch et al. (meta-) evaluation of machine translation. In *Proceedings of the Second Workshop on Statistical Machine Translation*, pages 136–158, Prague, Czech Republic, June 2007. Association for Computational Linguistics.

F. Casacuberta and E. Vidal. Machine translation with inferred stochastic finite-state transducers. *Computational Linguistics*, 30(2):205–225, 2004.

F. Casacuberta et al. Some approaches to statistical and finite-state speech-to-speech translation. *Computer Speech and Language*, 18:25–47, 2004.

Stanley Chen and Joshua Goodman. An empirical study of smoothing techniques for language modelling. Technical Report TR-10-98, Harvard University, 1998. See http://research.microsoft.com/ joshuago/tr-10-98.ps.

D. Chiang. Hierarchical phrase-based translation. *Computational Linguistics*, 33(2):201–228, 2007. ISSN 0891-2017.

T. Cohn and M. Lapata. Machine translation by triangulation: Making effective use of multi-parallel corpora. In *Proc. of ACL'07*, pages 728–735, Prague, Czech Republic, June 2007. Association for Computational Linguistics.

A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm (with discussion). *Journal of the Royal Statistical Society B*, 39:1–38, 1977a.

A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *J. Royal Statist. Soc. Ser. B*, 39(1):1–22, 1977b.

J. DeNero, D. Gillick, J. Zhang, and D. Klein. Why generative phrase models underperform surface heuristics. In *Proceedings on the Workshop on Statistical Machine Translation*, pages 31–38, New York City, June 2006. Association for Computational Linguistics.

Y. Deng and W. Byrne. HMM word and phrase alignment for statistical machine translation. In *Proc. of HLT-EMNLP'05*, pages 169–176. Association for Computational Linguistics, October 2005.

Y. Ding and M. Palmer. Machine translation using probabilistic synchronous dependency insertion grammars. In *Proc. of ACL'05*, pages 541–548, Morristown, NJ, USA, 2005. Association for Computational Linguistics.

Y. Ding, D. Gildea, and M. Palmer. An algorithm for word-level alignment of parallel dependency trees. In *Proc. of MT Summit IX*, pages 95–101, September 2003.

Richard O. Duda, Peter E. Hart, and David G. Stork. *Pattern Classification*. John Wiley and Sons, New York, NY, 2nd edition, 2001.

I. García-Varea and F. Casacuberta. Search algorithms for statistical machine translation based on dynamic programming and pruning techniques. In *Proc. of MT Summit VIII*, pages 115–120, Santiago de Compostela, Spain, 2001.

U. Germann et al. Fast decoding and optimal decoding for machine translation. In *Proc. of ACL'01*, pages 228–235, Morristown, NJ, USA, June 2001. Association for Computational Linguistics.

I. J. Good. Population frequencies of species and the estimation of population parameters. *Biometrika*, 40: 237–264, 1953.

Joshua Goodman. A bit of progress in language modeling. *CoRR*, cs.CL/0108005, 2001.

J. Graehl and K. Knight. Training tree transducers. In *Proc. of HLT-NAACL'04*, pages 105–112, Morristown, NJ, USA, May 2004. Association for Computational Linguistics.

G. Heigold, R. Schlüter, and H. Ney. On the equivalence of gaussian hmm and gaussian hmm-like hidden conditional random fields. In *Interspeech*, pages 1721–1724, Antwerp, Belgium, August 2007.

J. Hutchins and H. L. Somers. *An introduction to machine translation*. Academic Press, 1992.

A. Juan and Hermann Ney. Reversing and Smoothing the Multinomial Naive Bayes Text Classifier. In *Proc. of PRIS 2002*, pages 200–212, 2002.

S. Kanthak and H. Ney. FSA: an efficient and flexible C++ toolkit for finite state automata using on-demand computation. In *Proc. of ACL'04*, page 510, Morristown, NJ, USA, July 2004. Association for Computational Linguistics.

Slava M. Katz. Estimation of probabilities from sparse data for the language model component of a speech recognizer. *In IEEE Transactions on Acoustics, Speech and Signal Processing*, ASSP-35:400–401, 1987.

R. Kneser and H. Ney. Improved backing-off for m-gram language modeling. *IEEE Int. Conf. on Acoustics, Speech and Signal Processing*, II:181–184, May 1995.

K. Knight. Decoding complexity in word-replacement translation models. *Computional Linguistics*, 25(4): 607–615, 1999.

P. Koehn. Pharaoh: A Beam Search Decoder for Phrase-Based Statistical Machine Translation Models. In *Proceedings of AMTA'04*, pages 115–124, Washington, District of Columbia, USA, September-October 2004.

P. Koehn. Europarl: A parallel corpus for statistical machine translation. In *Proc. of the MT Summit X*, pages 79–86, September 2005.

P. Koehn, F.J. Och, and D. Marcu. Statistical phrase-based translation. In *Proc. of NAACL'03*, pages 48–54, Morristown, NJ, USA, 2003. Association for Computational Linguistics. doi: http://dx.doi.org/10.3115/1073445.1073462.

P. Koehn et al. Edinburgh system description for the 2005 iwslt speech translation evaluation. In *Proc. of IWSLT'05*, October 2005.

P. Koehn et al. Moses: Open source toolkit for statistical machine translation. In *Proc. of ACL'07: Demo and Poster Sessions*, pages 177–180, Morristown, NJ, USA, June 2007. Association for Computational Linguistics.

S. Kumar and W. J. Byrne. Minimum bayes-risk decoding for statistical machine translation. In *Proc. of HLT-NAACL'04*, pages 169–176, Morristown, NJ, USA, May 2004. Association for Computational Linguistics.

Philippe Langlais, Simona Gandrabur, Thomas Leplus, and Guy Lapalme. The long-term forecast for weather bulletin translation. *Machine Translation*, 19 (1):83–112, March 2005.

David D. Lewis. Naive (Bayes) at forty: The independence assumption in information retrieval. In Claire Nédellec and Céline Rouveirol, editors, *Proceedings of ECML-98, 10th European Conference on Machine Learning*, number 1398, pages 4–15, Chemnitz, DE, 1998. Springer Verlag, Heidelberg, DE.

D. Lin. A path-based transfer model for machine translation. In *Proc. of COLING'04*, page 625, Morristown, NJ, USA, August 2004. Association for Computational Linguistics.

D. Marcu and W. Wong. A phrase-based, joint probability model for statistical machine translation. In *Proc. of EMNLP'02*, pages 133–139, Morristown, NJ, USA, July 2002. Association for Computational Linguistics.

J. B. Mariño et al. N-gram-based machine translation. *Computational Linguistics*, 32(4):527–549, 2006. ISSN 0891-2017.

R.C. Moore. Fast and accurate sentence alignment of bilingual corpora. In *Proc. of AMTA'02*, pages 135–244, October 2002.

R.C. Moore. Improving IBM Word-Alignment Model 1. In *Proc. of ACL'04*, pages 519–524, July 2004.

D.S. Munteanu, A. Fraser, and D. Marcu. Improved machine translation performance via parallel sentence extraction from comparable corpora. In *Proc. of HLT-NAACL'04*, pages 265–272, Morristown, NJ, USA, May 2004. Association for Computational Linguistics.

Kevin P. Murhpy. Hidden semi-Markov Models (HSMMs). Technical report, University of British Columbia, 2007. URL `http://www.cs.ubc.ca/~murphyk/mypapers.html`.

A. Nadas. On turing's formula for word probabilities. *IEEE Trans. Acoustics, Speech, and Signal Proc.*, 33: 1,414–1,416, 1984.

Radford M. Neal and Geoffrey E. Hinton. A view of the em algorithm that justifies incremental, sparse, and other variants. In *Learning in Graphical Models*, pages 355–368. Kluwer Academic Publishers, 1998.

F. Nevado, F. Casacuberta, and E. Vidal. Parallel corpora segmentation using anchor words. In *Proc. of EAMT/CLAW'03*, pages 33–40, May 2003.

H. Ney, S. Martin, and F. Wessel. Statistical language modeling using leaving-one-out. In S. Young and G. Bloothooft, editors, *Corpus-Based Statiscal Methods in Speech and Language Processing.*, pages 174–207. Kluwer Academic Publishers, 1997.

S. Nirenburg et al. *Machine Translation: A Knowledge-based Approach*. Morgan Kaufmann, 1992.

E. H. Nyberg and T. Mitamura. The kant system: fast, accurate, high-quality translation in practical domains. In *Proc. of CL'92*, pages 1069–1073, Morristown, NJ, USA, August 1992. Association for Computational Linguistics.

F. J. Och. Minimum error rate training in statistical machine translation. In *Proc. of ACL'03*, pages 160–167, Morristown, NJ, USA, July 2003. Association for Computational Linguistics.

F. J. Och and H. Ney. A systematic comparison of various statistical alignment models. *Computational Linguistics*, 29(1):19–51, 2003. ISSN 0891-2017.

F. J. Och and H. Ney. The alignment template approach to statistical machine translation. *Computational Linguistics*, 30(4):417–449, 2004. ISSN 0891-2017.

D. Ortiz, I. García-Varea, F. Casacuberta, L. Rodríguez, and J. Tomás. Thot. New features to deal with larger corpora and long sentences. In *TC-STAR OpenLab on Speech Translation Workshop*, Trento (Italy), March 2006.

M. Ostendorf, V. Digalakis, and O. Kimball. From hmms to segment models: a unified view of stochastic modeling for speech recognition. *IEEE Transactions on Acoustics, Speech and Signal Processing*, (4):360–378, 1996.

K. Papineni, S. Roukos, T. Ward, and W. Zhu. BLEU: a Method for Automatic Evaluation of Machine Translation. Technical Report RC22176, Thomas J. Watson Research Center, 2001.

Lawrence Rabiner. A tutorial on hmm and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, February 1989.

M. Snover et al. A study of translation edit rate with targeted human annotation. In *Proc. of AMTA'06*, pages 223–231, Boston, Massachusetts, USA, August 2006. Association for Machine Translation in the Americas.

A. Stolcke. SRILM – an extensible language modeling toolkit. In *Proc. of ICSLP'02*, pages 901–904, September 2002.

Charles Sutton and Andrew McCallum. *Introduction to Statistical Relational Learning*. Lise Getoor and Ben Taskar, 2006. Chapter: An Introduction to Conditional Random Fields for Relational Learning.

C. Tillmann and H. Ney. Word reordering and a dynamic programming beam search algorithm for statistical machine translation. *Computational Linguistics*, 29(1):97–133, March 2003.

R. Udupa and H. K.r Maji. Computational complexity of statistical machine translation. In *Proc. of EACL'06*, April 2006.

N. Ueffing and H. Ney. Word-level confidence estimation for machine translation. *Computational Linguistics*, 33(1):9–40, 2007.

David Vilar, Hermann Ney, A. Juan, and Enrique Vidal. Effect of Feature Smoothing Methods in Text Classification Tasks. In *Proc. of PRIS 2004*, pages 108–117, 2004.

Y. Wang and A. Waibel. Fast decoding for statistical machine translation. In *Proc. of ICSLP'98*, pages 2775–2778, October 1998.

C. F. Jeff Wu. On the convergence properties of the EM algorithm. *The Annals of Statistics*, 11(1):95–103, 1983.

D. Wu. A polynomial-time algorithm for statistical machine translation. In *Proc. of ACL'96*, pages 152–158, Morristown, NJ, USA, June 1996. Morgan Kaufmann / Association for Computational Linguistics.

K. Yamada and K. Knight. A syntax-based statistical translation model. In *Proc. of ACL'01*, pages 523–530, Morristown, NJ, USA, July 2001. Association for Computational Linguistics.

Richard Zens, Franz Josef Och, and Hermann Ney. Phrase-based statistical machine translation. In *KI '02: Proceedings of the 25th Annual German Conference on AI*, pages 18–32, London, UK, 2002. Springer-Verlag. ISBN 3-540-44185-9.

Chapter 2

Constrained-Domain Maximum Likelihood Estimation

“ *If you are out to describe the truth, leave elegance to the tailor.* ” A. EINSTEIN

Contents

2.1 Introduction

Most pattern recognition systems are based on the optimal Bayes' rule (see Section 1.1 and Section 4.2). This rule highly depends on the posterior class probability $p_r(\omega|\boldsymbol{x})$. Provided that the actual posterior probability is not available in real tasks, it is approximated by a model $p_{\boldsymbol{\theta}}(\omega|\boldsymbol{x})$ which is characterised by a parameter set, $\boldsymbol{\theta} \in \boldsymbol{\Theta}$.

The selection of the optimal parameter set $\boldsymbol{\theta}$ depends on the function criterion. As reviewed in Section 1.1.3 Chapter 1, maximum likelihood estimation (MLE) is one of the most widespread criteria. This criterion finds the parameter set $\hat{\boldsymbol{\theta}}$ that maximises the likelihood function which is defined in Section 1.1.3. One of the most important flaws concerning to MLE is that it tends to overfit the parameters to the training data at the expense of reserving small probabilities or even zero probability to the remaining non-training data. This overfitting problem is often a straight consequence of the ratio of the number of parameters to the training size; roughly speaking, the data is scarce for what the model needs to learn.

In order to alleviate the overfitting problem, it is a common approach to distort the optimal parameter set $\hat{\boldsymbol{\theta}}$ obtaining a non-overfitted version of the optimal parameter set, $\tilde{\boldsymbol{\theta}}$. However, on the one hand, several smoothing techniques are heuristic techniques based on practical observation. For instance, such is the case of the interpolate smoothing in which the optimal vector $\hat{\boldsymbol{\theta}}$ is usually interpolated with a uniform distribution. On the other hand, some of the smoothing techniques are based on statistical methods. The maximum a posteriori estimation or the leaving-one-out estimation are examples of such smoothing methods.

In this chapter, we propose a method to avoid the scarce data derived problems such as overfitting. Instead of smoothing the optimal solution obtained by MLE, we introduce the idea of constraining the parametric domain, $\boldsymbol{\Theta}$, before searching for the optimal parameter set. Since, in this way, there is no possible overfitted parameter set in the domain; the optimal parameter set is smoothed in the parametric optimisation. Even more, the optimal parameter set obtained from the constrained domain retains the properties of the MLE whilst the classical smoothed parameter set does not.

We apply the idea of *constrained-domain maximum likelihood estimation* (CDMLE) [Andrés-Ferrer and Juan, 2006, Andrés-Ferrer and Juan, 2009] to the *naive Bayes* text classifier [Juan and Ney, 2002, McCallum and Nigam, 1998, Vilar et al., 2004]. The *naive Bayes* classifier [Andrés-Ferrer and Juan, 2006, Andrés-Ferrer and Juan, 2009] has long been a core technique in information retrieval and, more recently, it has attracted significant interest in pattern recognition and machine learning [Lewis, 1998]. Given the document class and length, this classifier makes the *naive Bayes* assumption that the probability of occurrence of a word does not depend on its position or other words in the document. In spite of being completely unrealistic, this assumption has the advantage of greatly simplifying classifier training. In particular, conventional, maximum likelihood estimation of class-conditional word occurrence probabilities reduces to a simple normalisation of word counts. However, due to data spareness, these estimates suffer from overfitting; i.e. the estimated probabilities memorise the training data and are unable to explain unseen events. Overfitting is usually alleviated using *parameter smoothing*, which is simply a heuristic modification of maximum likelihood estimates to avoid null values [Vilar et al., 2004]. Unfortunately, the resulting *smoothed parameters* are no longer *optimal* in terms of maximum likelihood and thus we cannot attribute to them the desirable properties of maximum likelihood estimators.

The proposed algorithm is described in Section 2.4, after a brief review of the naive Bayes model and its conventional maximum likelihood estimation in the following two Sections. Empirical results and concluding remarks are given in Sections 2.5 and 2.6, respectively.

2.2 Naive Bayes model

We denote the class variable by $c = 1, \ldots, C$ in the remaining of this chapter; the word variable by $d = 1, \ldots, D$; and a document of length L by $\boldsymbol{w}_1^L = w_1 w_2 \cdots w_L$. The joint probability of occurrence of c, L and $\boldsymbol{w}_1^L$ may be written as

$$p_r(c, L, \boldsymbol{w}_1^L) := \mathrm{p}(c)\ \mathrm{p}(L)\ \mathrm{p}_{\boldsymbol{\theta}}(\boldsymbol{w}_1^L \mid c, L) \quad , \tag{2.1}$$

where we have assumed that document length does not depend on the class.

Given the class c and the document length L, the probability of occurrence of any particular document $\boldsymbol{w}_1^L$ can be greatly simplified by making the so-called *naive Bayes* or *independence assumption:* the probability of occurrence of a word w_l in $\boldsymbol{w}_1^L$ does not depend on its position l or other words $w_{l'}$, $l' \neq l$,

$$\mathrm{p}_{\boldsymbol{\theta}}(\boldsymbol{w}_1^L \mid c, L) = \textstyle\prod_{i=1}^{L} \mathrm{p}(w_i \mid c) \quad . \tag{2.2}$$

Using the above assumptions, we may write the *posterior* probability of a document belonging to a class c as:

$$\mathrm{p}_{\boldsymbol{\pi},\boldsymbol{\theta}}(c \mid L, \boldsymbol{w}_1^L) = \frac{\mathrm{p}_{\boldsymbol{\pi},\boldsymbol{\theta}}(c, L, \boldsymbol{w}_1^L)}{\sum_{c'} \mathrm{p}_{\boldsymbol{\pi},\boldsymbol{\theta}}(c', L, \boldsymbol{w}_1^L)} \tag{2.3}$$

$$= \frac{\pi_c \prod_{d=1}^{D} \theta_{cd}^{x_d}}{\sum_{c'} \pi_{c'} \prod_{d=1}^{D} \theta_{c'd}^{x_d}} \tag{2.4}$$

$$:= \ p_{\boldsymbol{\pi},\boldsymbol{\theta}}(c \mid \boldsymbol{x}) \tag{2.5}$$

where x_d is the count of word d in $\boldsymbol{w}_1^L$, $\boldsymbol{x} = (x_1, \ldots, x_D)^t$, and $\boldsymbol{\Theta}$ is the set of unknown parameters, which includes π_c for the class c prior and θ_{cd} for the probability of occurrence of word d in a document from class c. Clearly, these parameters must be non-negative and satisfy the normalisation constraints:

$$\textstyle\sum_c \pi_c = 1 \tag{2.6}$$

$$\textstyle\sum_{d=1}^{D} \theta_{cd} = 1 \qquad (c = 1, \ldots, C) \tag{2.7}$$

The Bayes' decision rule associated with model (2.5) is a log-linear classifier:

$$\mathrm{c}_{\boldsymbol{\pi},\boldsymbol{\theta}}(\boldsymbol{x}) = \arg\max_c \ p_{\boldsymbol{\pi},\boldsymbol{\theta}}(c \mid \boldsymbol{x}) \tag{2.8}$$

$$= \arg\max_c \left\{ \log \pi_c + \sum_d x_d \log \theta_{cd} \right\} \tag{2.9}$$

2.3 Conventional naive Bayes training

Naive Bayes training refers to the problem of deciding (a criterion and) a method to compute an appropriate estimate for $\{\boldsymbol{\pi}, \boldsymbol{\theta}\}$ from a given collection of N labelled training samples $(\boldsymbol{x}_1, c_1), \ldots, (\boldsymbol{x}_N, c_N)$. A conventional training criterion is the *joint* log-likelihood function:

$$\mathrm{LL}(\boldsymbol{\pi}, \boldsymbol{\theta}) = \textstyle\sum_c N_c \log \pi_c + \sum_d N_{cd} \log \theta_{cd} \quad . \tag{2.10}$$

where N_c is the number of documents in class c and N_{cd} is the number of occurrences of word d in training data from class c. It is well-known that the global maximum of this criterion under constraints (2.6)-(2.7) can be computed in closed-form:

$$\hat{\pi}_c = \frac{N_c}{N} \tag{2.11}$$

and

$$\hat{\theta}_{cd} = \frac{N_{cd}}{\sum_{d'} N_{cd'}} \quad . \tag{2.12}$$

Despite the optimality of the estimates (2.12), they are usually *smoothed* (modified) to avoid null estimates originated by data spareness. For instance, in one of the experiments reported in Section 2.5, we face the problem of estimating 1.87M class-conditional word probabilities and only 14.3% of them are non-zero according to (2.12). Thus, without smoothing, the sole occurrence of a rare word in a test document is likely to introduce dominant and underestimated terms in the decision rule (2.8) and, hence, it may certainly be the cause of a classification error.

A popular smoothing method for (2.12) consists of simply adding a "pseudo-count" $\delta > 0$ to every N_{cd} count:

$$\tilde{\theta}_{cd} = \frac{N_{cd} + \delta}{\sum_{d'} (N_{cd'} + \delta)} \quad , \tag{2.13}$$

with $\delta = 1$ as the default value. This method is sometimes referred to as *Laplace smoothing* [McCallum and Nigam, 1998].

Alternatively, as done in the context of *statistical language modelling* for *speech recognition*, we may use the idea of *absolute discounting* to avoid null estimates [Juan and Ney, 2002, Vilar et al., 2004]. Instead of using artificial pseudo-counts, we gain "free" probability mass by discounting a small constant to every count associated with a *seen* event (positive count). The gained probability mass is then distributed among events in accordance with a *generalised distribution* such as the *uniform* distribution,

$$\beta_d = \frac{1}{D} \quad , \tag{2.14}$$

the *unigram* distribution,

$$\beta_d = \frac{\sum_c N_{cd}}{\sum_{d'} \sum_c N_{cd'}} \quad , \tag{2.15}$$

or whatever distribution providing a reliable estimation of class-independent word probabilities. Depending on the set of events that receives the gained probability mass, we distinguish between *back-off* and *interpolation*. Back-off only considers unseen events:

$$\tilde{\theta}_{cd} = \begin{cases} \dfrac{N_{cd} - b}{\sum_{d'} N_{cd'}} & \text{if } N_{cd} > 0 \\ \\ M_c \, \dfrac{\beta_d}{\sum_{d' : N_{cd'} = 0} \beta_{d'}} & \text{if } N_{cd} = 0 \end{cases} \tag{2.16}$$

where the probability mass gained in class c is:

$$M_c = \frac{b \, |\{d' \geq 1 : N_{cd'} > 0\}|}{\sum_{d' \geq 1} N_{cd'}} \quad , \tag{2.17}$$

and the discount b is restricted to the interval $(0, 1)$. In contrast, interpolation distributes the gained probability mass among all events:

$$\tilde{\theta}_{cd} = \max \left\{ 0, \frac{N_{cd} - b}{\sum_{d'} N_{cd'}} \right\} + M_c \, \beta_d \quad , \tag{2.18}$$

where $0 < b \leq 1$.

2.4 Constrained-domain maximum likelihood estimation

As it is said in the introduction, smoothed parameters are no longer optimal in terms of maximum likelihood and thus we cannot attribute to them the desirable properties of maximum likelihood estimators. In this Chapter, we advocate the reduction of the set of feasible parameter estimates, that is, the use of additional constraints on it. In particular, we focus our interest in conventional naive Bayes training, without smoothing, but constrained to class-conditional word probability estimates not smaller than a predefined non-negative constant ϵ. That is to say that we are interested in the maximisation of (2.10) subject to constraints (2.6), (2.7) and

$$\theta_{cd} \geq \epsilon \qquad (c = 1, \ldots, C; d = 1, \ldots, D) \tag{2.19}$$

where ϵ is the minimum probability of occurrence of a word in a document from any class ($0 \leq \epsilon \leq \frac{1}{D}$). Obviously, this is not a value we intend to learn from the data, but a meta-parameter to restrict the set of feasible estimates to "conservative" values. If we choose $\epsilon = 0$, we do not move from conventional naive Bayes training. On the contrary, if $\epsilon = \frac{1}{D}$, the only solution is to set all word probabilities to ϵ. In general, the more training data, the smaller ϵ should be chosen.

2.4.1 Characterisation of the solution

Maximisation of (2.10) subject to constraints (2.6), (2.7) and (2.19) is a convex (concave maximisation) problem with differentiable objective and constraint functions, for which we can find a global maximum using the *Karush-Kuhn-Tucker* (KKT) conditions (see Appendix A). The Lagrangian function is

$$\mathcal{L}(\boldsymbol{\pi}, \boldsymbol{\theta}, \boldsymbol{\lambda}, \boldsymbol{\mu}) = -\operatorname{LL}(\boldsymbol{\pi}, \boldsymbol{\theta}) + \Lambda(\boldsymbol{\pi}, \boldsymbol{\theta}, \boldsymbol{\lambda}) + \Gamma(\boldsymbol{\theta}, \boldsymbol{\mu}) \quad , \tag{2.20}$$

where $\Lambda(\boldsymbol{\pi}, \boldsymbol{\theta}, \boldsymbol{\lambda})$ stands for Lagrangian part corresponding to the equality constraints, i.e.,

$$\Lambda(\boldsymbol{\pi}, \boldsymbol{\theta}, \boldsymbol{\lambda}) = \lambda_0 \left[\sum_c \pi_c - 1 \right] + \sum_c \lambda_c \left[\sum_d \theta_{cd} - 1 \right] \quad , \tag{2.21}$$

where λ_0 and λ_c are Lagrange multipliers associated with constraints (2.6) and (2.7), respectively ($c = 1, \ldots, C$). Conversely, the $\Gamma(\boldsymbol{\theta}, \boldsymbol{\mu})$ function in Eq. (2.20) stands for the Lagrangian part corresponding to the inequality constraint, i.e.,

$$\Gamma(\boldsymbol{\theta}, \boldsymbol{\mu}) = \sum_{c,d} \mu_{cd} \left(\epsilon - \theta_{cd} \right) \quad , \tag{2.22}$$

where μ_{cd} are Lagrange multipliers associated with constraints (2.19), ($c = 1, \ldots, C;\ d = 1, \ldots, D$).

The KKT conditions for a point $\hat{\boldsymbol{\pi}}, \hat{\boldsymbol{\theta}}, \hat{\boldsymbol{\lambda}}, \hat{\boldsymbol{\mu}}$ to be a global maximum are

$$\boldsymbol{\nabla}_{\boldsymbol{\pi},\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\pi}, \boldsymbol{\theta}, \boldsymbol{\lambda}, \boldsymbol{\mu})\big|_{\hat{\boldsymbol{\pi}},\hat{\boldsymbol{\theta}},\hat{\boldsymbol{\lambda}},\hat{\boldsymbol{\mu}}} = \mathbf{0} \tag{2.23}$$

$$\sum_c \hat{\pi}_c = 1 \tag{2.24}$$

$$\sum_d \hat{\theta}_{cd} = 1 \qquad (c = 1, \ldots, C) \tag{2.25}$$

$$\hat{\theta}_{cd} \geq \epsilon \qquad (c = 1, \ldots, C; d = 1, \ldots, D) \tag{2.26}$$

$$\hat{\mu}_{cd}(\epsilon - \hat{\theta}_{cd}) = 0 \qquad (c = 1, \ldots, C; d = 1, \ldots, D) \tag{2.27}$$

$$\hat{\mu}_{cd} \geq 0 \qquad (c = 1, \ldots, C; d = 1, \ldots, D) \tag{2.28}$$

From (2.23) and (2.24) immediately follows that, as in the conventional case, the optimal class priors can be computed in closed-form using (2.11). However, this is not the case of the class-conditional word occurrence probabilities. From (2.23), we have

$$\hat{\theta}_{cd} = \frac{1}{\hat{\lambda}_c + \hat{\mu}_{cd}} N_{cd} \qquad (c = 1, \ldots, C; d = 1, \ldots, D) \tag{2.29}$$

but now we cannot rewrite $\hat{\lambda}_c + \hat{\mu}_{cd}$ in terms of word counts to arrive at a closed-form solution like (2.12). Instead, by some straightforward manipulations, we arrive at the following characterisation for each class c:

$$\hat{\theta}_{cd} = \begin{cases} \epsilon & \text{if } \vartheta_{cd} \leq \epsilon \\ \vartheta_{cd} & \text{if } \vartheta_{cd} > \epsilon \end{cases} \qquad (d = 1, \ldots, D) \tag{2.30}$$

where

$$\vartheta_{cd} = \frac{N_{cd}}{\sum\limits_{d' : \vartheta_{cd'} > \epsilon} N_{cd'}} (1 - M_c) \qquad (d = 1, \ldots, D) \tag{2.31}$$

with

$$M_c = |\{d' : \vartheta_{cd'} \leq \epsilon\}|\, \epsilon \tag{2.32}$$

The idea behind this characterisation is as follows. First note that we distinguish between "rare" words, in the sense that we assign a probability of exactly ϵ to them ($d : \vartheta_{cd} \leq \epsilon$), and "frequent" words, which have probability greater than ϵ ($d : \vartheta_{cd} > \epsilon$). The probability mass allotted to rare words is simply their number times ϵ and is denoted by M_c in (2.32). The remaining probability mass, $1 - M_c$, is distributed among frequent words in accordance with (2.31), which is simply a normalisation of word counts as in the conventional case (2.12). Thus, generally speaking, we proceed as in the conventional case, but using only the probability mass not assigned to words that do not cross the threshold of ϵ.

2.4.2 The algorithm

The above characterisation does not tell us how to partition words into rare and frequent, not even if such a partition exists. Nevertheless, it can be easily shown that a solution exists and can be found iteratively for each class separately. Let c be the current class. The basic algorithm consists in first assuming that the set of rare words is empty, $R_c^{(0)} = \emptyset$; then, in iteration k ($k = 1, 2, \ldots$), the new set of rare words, $R_c^{(k)}$, is obtained from $R_c^{(k-1)}$ by addition of each word d,

$$R_c^{(k)} = R_c^{(k-1)} \cup \{d\} \tag{2.33}$$

which is not in $R_c^{(k-1)}$ but it is actually rare according to our criterion of not having a probability greater than ϵ,

$$\vartheta_{cd}^{(k-1)} \leq \epsilon \tag{2.34}$$

where

$$\vartheta_{cd}^{(k-1)} = \frac{N_{cd}}{\sum\limits_{d' \notin R_c^{(k-1)}} N_{cd'}} (1 - M_c^{(k-1)}) \tag{2.35}$$

with

$$M_c^{(k-1)} = |R_c^{(k-1)}|\, \epsilon \tag{2.36}$$

At the end of iteration k, the algorithm assures that condition (2.34) is satisfied for all words in $R_c^{(k)}$. This condition may be also satisfied by words not in $R_c^{(k)}$ though, in general, it will not be satisfied by most of them.

```
1  Algorithm:  CDMLE
2  Input:
3     C, D                                          // number of classes and words
4     (x_1, c_1), ..., (x_N, c_N)                   // N labelled training samples
5     ε : 0 ≤ ε ≤ 1/D                               // minimum word occurrence probability
6  Output:
7     {θ̂_cd}                                        // solution as characterised by Eqs.(2.30)-(2.32)
8  Variables:
9     {N_cd}  // word counts for each class
10    R', R  // previous and current set of rare words
11    S', S  // previous and current sum of non-rare word counts
12    M', M // previous and current rare words probability mass
13 Method:
14 for c := 1 to C do                               // each class c is processed separately
15    for d := 1 to D do N_cd := 0 endfor
16    for n := 1 to N do
17       if c_n = c then
18          for d := 1 to D do N_cd := N_cd + x_nd endfor
19       endif
20    endfor                                        // word counts for class c computed
21    R := ∅; S := 0; M := 0
22    for d := 1 to D do S := S + N_cd endfor
23    repeat                                        // main loop for class c
24       R' := R; S' := S; M' := M
25       transfers := false
26       for d := 1 to D do if d ∉ R' then
27          θ̂_cd := (N_cd / S') · (1 − M')
28          if θ̂_cd ≤ ε then
29             θ̂_cd := ε                           // d has minimum probability in c
30             R := R ∪ {d}                        // d is a new rare word
31             S := S − N_cd
32             M := M + ε
33             transfers := true
34          endif endif
35       endfor
36    until not transfers
37 endfor
```

Figure 2.1: The Constrained-Domain Maximum Likelihood Estimation (CDMLE) algorithm.

As R_c^0 is empty, $M_c^{(0)}$ is zero and the initial probability estimates, $\vartheta_{cd}^{(0)}$, are exactly those obtained in the conventional case (2.12). Therefore, in the first iteration, we use conventional probability estimates to distinguish between rare and frequent words. Part of the probability mass assigned to frequent words is transferred to rare words for them to arrive at ϵ. The remaining probability mass is redistributed according to (2.35) and, as it is smaller than that distributed before the transference, it may well happen that a frequent word become a new rare word. If it happens, a new iteration is carried out; otherwise, the algorithm stops and returns the desired $\hat{\theta}_{cd}$, as characterised by (2.30).

A detailed description of the basic algorithm described above is given in Fig. 2.1. Given C, D, the training data and ϵ, it returns $\hat{\theta}_{cd}$ for all c and d, as characterised by Eqs.(2.30)-(2.32). The main loop processes each class c at a time (lines 14–37). After computation of word counts (lines 15–20), the optimal CDMLE solution is obtained iteratively (lines 21–36). Initially, no words are considered rare ($R := \emptyset$) and $\hat{\theta}_{cd}$ is computed for all words as in the conventional case (during the first iteration of the loop in lines 23–36). If a word d is found such that $\hat{\theta}_{cd} \leq \epsilon$ (line 28), then d is added to R and a new iteration is executed; otherwise, no transfers to R are carried out and the algorithm stops.

2.4.3 Algorithm correctness and complexity

Let c be the current class and let d be a non-rare word in iteration $k-1$ ($d \notin R_c^{(k-1)}$) for which (2.34) holds. Then, it follows that

$$1 - \frac{\epsilon}{1 - M_c^{(k-1)}} \leq 1 - \frac{N_{cd}}{\sum_{d' \notin R_c^{(k-1)}} N_{cd'}} \tag{2.37}$$

and, rearranging terms,

$$\frac{1 - M_c^{(k-1)} - \epsilon}{\sum_{d' \notin R_c^{(k-1)}} N_{cd'} - N_{cd}} \leq \frac{1 - M_c^{(k-1)}}{\sum_{d' \notin R_c^{(k-1)}} N_{cd'}} \tag{2.38}$$

As $d \notin R_c^{(k-1)}$ but satisfies Eq. (2.34), the algorithm adds d to the set of rare words in iteration k, $R_c^{(k)} = R_c^{(k-1)} \cup \{d\}$. Using this updated set of rare words, Eq. (2.38) can be rewritten as

$$\frac{1 - M_c^{(k)}}{\sum_{d' \notin R_c^{(k)}} N_{cd'}} \leq \frac{1 - M_c^{(k-1)}}{\sum_{d' \notin R_c^{(k-1)}} N_{cd'}} \tag{2.39}$$

from which we have, for any word $d'' \in R_c^{(k)}$,

$$\vartheta_{cd''}^{(k)} \leq \vartheta_{cd''}^{(k-1)} \tag{2.40}$$

by multiplying each side of Eq. (2.39) by $N_{cd''}$. From Eq. (2.40) and the fact that $\vartheta_{cd''}^{(k-1)} \leq \epsilon$ for all $d'' \in R_c^{(k)}$, it follows that $\vartheta_{cd''}^{(k)} \leq \epsilon$ for all $d'' \in R_c^{(k)}$. This means that, in iteration k, word d becomes rare while all rare words in the previous iteration remain rare. Algorithm correctness follows from this result.

The time complexity of the CDMLE algorithm depends on the case. In the best case, no word transfers are done in the repeat-until loop and the algorithm works exactly as the conventional naive Bayes training (without parameter smoothing). More precisely, after the first repeat-until iteration, a second iteration is needed for the algorithm to check that no transfers to the set of rare words are carried out. Then, in the best case, its time complexity is $\Omega(CND)$. On the other hand, the repeat-until loop is executed D times in the worst case, and thus the algorithm has $O(CND + CD^2)$ time complexity.

However, in practice, the repeat-until loop is expected to iterate only a few times. Therefore, the computational behaviour of the CDMLE algorithm is expected to not differ significantly from conventional naive Bayes training.

The previous discussion about the complexity of the CDMLE algorithm only applies to a direct implementation of it, such as that given in Fig. 2.1. However, it is straightforward to derive a refined implementation of $O(CND + CD \log D)$ time complexity. The idea behind this refinement is to apply Eq. (2.33) in non-decreasing order of occurrence probability, as estimated in the conventional case. That is, in iteration k, the next word d to be considered in Eq. (2.33) must have minimum occurrence probability, as given in Eq. (2.12), among all non-rare words. It can be easily checked that, if condition (2.34) does not hold for d, then it will not hold for any other non-rare word and, therefore, the optimal CDMLE solution will have been found.

2.5 Experiments

The proposed approach was empirically compared to the usual practice of simply smoothing relative counts, as described in Section 2.3. This comparison was carried on four text classification data sets (tasks): *Traveller*, *20 Newsgroups*, *Industry Sector* and *Job Category*.

The *Traveller* data set comes from a *limited-domain* Spanish-English machine translation application for human-to-human communication situations in the front-desk of a hotel. It was semi-automatically built from a small "seed" data set of sentence pairs collected from traveller-oriented booklets by four persons; A, F, J and P, each of whom had to cater for a (non-disjoint) subset of subdomains. The *20 Newsgroups* corpus is a collection of approximately $20,000$ newsgroup documents, partitioned (nearly) evenly across 20 different newsgroups. We used the original version of this data set as provided by [Rennie, 2001], in which document headers are discarded but the "From:" and "Subject:" header fields are retained. The *Industry Sector* is a collection of web pages from different companies, divided into a hierarchy of classes. In our experiments, however, we "flattened" this structure, assigning each document a class consisting of the whole path to the document in the hierarchy tree. The *Job Category* data set consist of job titles and descriptions, also organised in a hierarchy of classes. This corpus contains labelled and unlabelled samples and only the former were used in our experiments. Table 2.1 contains a summary with the basic information on these data sets. For further details on them, see [McCallum, 2002, Rennie, 2001, Vidal et al., 2000, Vilar et al., 2004].

Table 2.1: Basic information on the data sets used in the experiments. (*Singletons* are words that occur once; *Class n-tons* refers to words that occur in n classes exactly.)

	Job Category	Industry Sector	20 Newsgroups	Traveller (English)
Type of documents	job titles	web pages	newsgroups	sentences
Number of documents	131 643	9 629	19 974	8 000
Running words	11 221K	1 834K	2 549K	79K
Avg. document length	85	191	128	10
Vocabulary size	84 212	64 551	102 752	391
Singletons (Vocab %)	34.9	41.4	36.0	4
Classes	65	105	20	23.0
Class 1-tons (Vocab %)	49.2	58.7	61.1	74.9
Class 2-tons (Vocab %)	14.0	11.6	12.9	18.3

The *rainbow* toolkit [McCallum, 1998] was used for the preprocessing of all data sets but *Traveller*. We used html skip for web pages and elimination of UU-encoded segments for newsgroup messages. We did not use stop-list removal, stemming or vocabulary pruning by occurrence count.

Figure 2.2 shows the results obtained in each data set. The proposed CDMLE algorithm is compared to:

1. *Laplace:* conventional training and Laplace smoothing,
2. *AD+1gBO:* conventional training and absolute discounting with unigram back-off, and
3. *AD+1gI:* as (2) with unigram interpolation.

Each classification technique considered has its own test-set error rate curve as a function of the discount b:

1. *Laplace:* b refers to δ in Eq. (2.13),
2. *AD+1gBO or 1gI:* b has its usual meaning, as defined in Eq. (2.16), and
3. *CDMLE:* ϵ is defined from b as $\epsilon = 10^{-10\,b} \cdot \frac{1}{D}$ in the Traveller data set and $\epsilon = b \cdot \frac{1}{D}$ in the other data sets.

Each plotted point corresponds to an average error rate obtained from 30 random splits in which 80% documents were used for training while the remaining 20% were held out for testing. Error rate estimates have an approximate 95% confidence interval of $[E\% \pm 1\%]$ ($[E\% \pm 0.4\%]$ for Job Category).

Table 2.2: Summary of the best results.

	Job Category	Industry Sector	20 News	Traveller (English)
Laplace	33.2	38.9	15.0	3.3
AD+1gBO	34.0	38.0	14.9	3.3
AD+1gI	34.2	**37.8**	**14.8**	3.3
CDMLE	**33.0**	38.6	15.3	**3.1**

From the results in Fig. 2.2, it is clear that the CDMLE algorithm performs similarly to the other techniques. In comparison with Laplace, CDMLE provides slightly better results and more stable (flat) error curves in all data sets but 20 Newsgroups. In these data sets, it is indeed much better than Laplace when, as usual with Laplace, the discount factor is simply set to one. In the case of 20 Newsgroups, however, Laplace seems to be a bit better than CDMLE.

In comparison with absolute discounting (AD+1gBO and AD+1gI), it can be said that there is no superiority of one over the other. In Traveller and Job category, the CDMLE algorithm provides better rates than absolute discounting, but the contrary can be observed in the other two data sets. All in all, this is a comparatively good result for CDMLE since, in contrast to absolute discounting with unigram back-off/interpolation, CDMLE does not take advantage of the unigram distribution (2.15) to obtain reliable class-independent word probability estimates. Clearly, this estimates can be used to replace (2.19) by better, word-dependent domain constraints.

A summary of the best results obtained in the experiments is given in Table 2.2. The CDMLE algorithm obtains better results than Laplace and absolute discounting in Job Category and Traveller. However, absolute discounting is better than Laplace and the CDMLE algorithm in Industry Sector and 20 Newsgroups. Note that these differences are significant only to a limited extent.

As said in Section 2.4.3, the time complexity of the CDMLE algorithm is $\Omega(CND)$ in the best case and $O(CND + CD^2)$ in the worst case. More precisely, the difference between these two cases arises from the number of repeat-until iterations executed (lines 23–36 in Fig. 2.1), which may vary

from 2 to D. To study this in the average case, the number of repeat-until iterations was recorded in each CDMLE algorithm execution. On average, it was exactly 2 in the Traveller and 20 Newsgroups data sets, that is, as in the best case. On the other hand, it was only of 2.5 iterations for Industry Sectors and 3.2 for Job category. Therefore, as expected, the repeat-until loop iterates only a few times. That is, in practice, the computational behaviour of the CDMLE algorithm might be considered almost the same as that of conventional naive Bayes training.

2.6 Conclusions

In this chapter, conventional naive Bayes training with parameter smoothing has been restated as a constrained-domain maximum likelihood estimation problem for which an optimal, iterative algorithm has been proposed. The general idea behind our contribution is to avoid parameter estimates that can cause over-fitting while retaining the properties of maximum likelihood estimators. Empirical results on four real text classification tasks have shown that the proposed algorithm provides results similar to those of conventional training and parameter smoothing, with almost the same practical computational requirements.

It is worth noting, however, that smoothing methods have been continuously improved over the years, while our proposal is completely new and thus, there is still room for significant improvements. For instance, the parameter domain might be better adjusted by redefining the constant ϵ introduced in Eq. (2.19) and making it dependent on both the class c and the word d.

We think that the proposed approach is very promising. In general, the idea behind of the proposed approach can be applied to many maximum likelihood estimation problems in pattern recognition. For instance, it can be easily applied to EM-based maximum likelihood estimation of finite mixture models. For these models, it is unclear how to use parameter smoothing in the M step without affecting the EM behaviour. Instead, constrained-domain maximum likelihood estimation can be used without any side effect. Also, this constrained approach might be useful in the case of training criterion other than maximum likelihood such as discriminative training [Juan et al., 2007].

The naive Bayes model follows a Multinomial distribution [Juan and Ney, 2002]; and, hence, the proposed algorithm can be applied for Multinomial estimation. Finally, since the naive Bayes model is also a special case to the n-gram language models, this technique can be extended to higher order n-grams. Specifically, this idea is covered in following chapter.

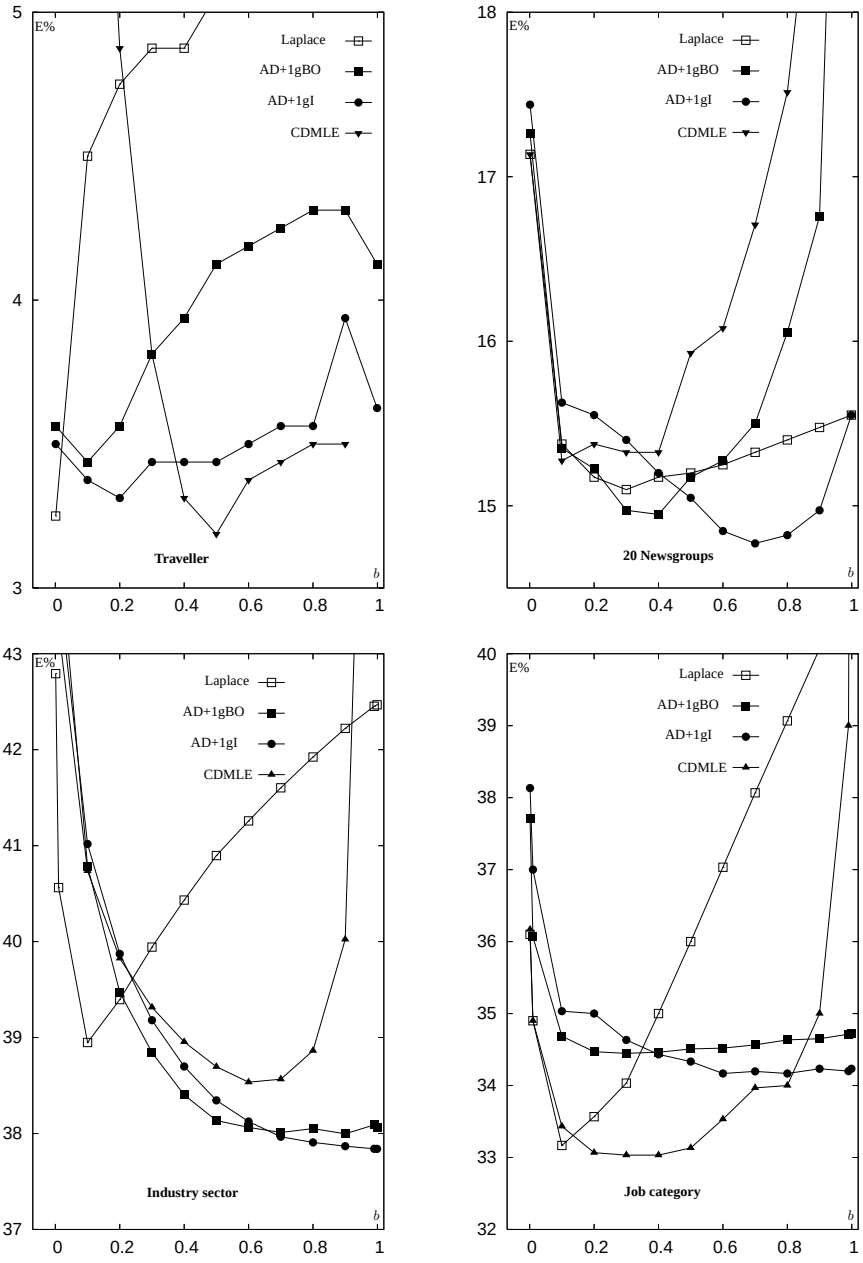


Figure 2.2: Results obtained in the *Traveller*, *20 Newsgroups*, *Industry sector* and *Job category* data sets. Each plot shows the classification error rate as a function of the discount parameter *b*, for the four classification techniques considered (*Laplace*, *AD+1gBO*, *AD+1gI* and *CDMLE*).

Bibliography

J. Andrés-Ferrer and A. Juan. Máxima verosimilitud con dominio restringido aplicada a clasificación de documentos. In *Actas de Campus Multidisciplinar en Percepcion e Inteligencia, CMPI-2006*, volume 2, pages 791–803, 2006. ISBN ISBN 84-689-9560-6 (Obra completa). ISBN 84-689-9562-2 (Volumen II).

Jesús Andrés-Ferrer and A. Juan. Constrained domain maximum likelihood estimation for naive bayes text classification. *Pattern Analysis and Applications (PAA)*, Published online 2009. doi: 10.1007/s10044-009-0149-y.

A. Juan and Hermann Ney. Reversing and Smoothing the Multinomial Naive Bayes Text Classifier. In *Proc. of PRIS 2002*, pages 200–212, 2002.

A. Juan, D. Vilar, and H. Ney. Bridging the gap between Naive Bayes and Maximum Entropy Text Classification. In *Proc. of PRIS'07*, pages 59–65, Funchal, Madeira - Portugal, June 2007.

D. D. Lewis. Naive Bayes at Forty: The Independence Assumption in Information Retrieval. In *Proc. of ECML'98*, pages 4–15, April 1998.

A. McCallum. Industry Sector data set, 2002. `www.cs.umass.edu/~mccallum/code-data.html`.

A. McCallum. Bow: A toolkit for statistical language modeling, text retrieval, classification and clustering, 1998. `www.cs.umass.edu/~mccallum/bow/rainbow`.

A. McCallum and K. Nigam. A Comparison of Event Models for Naive Bayes Text Classification. In *Proc. of AAAI/ICML-98: Workshop on Learning for Text Categorization*, pages 41–48. Morgan Kaufmann, July 1998.

J. Rennie. Original 20 Newsgroups data set, 2001. `people.csail.mit.edu/jrennie/20Newsgroups`.

E. Vidal et al. Example-Based Understanding and Translation Systems (EuTrans). Final Report, ESPRIT project 20268, ITI, 2000.

David Vilar, Hermann Ney, A. Juan, and Enrique Vidal. Effect of Feature Smoothing Methods in Text Classification Tasks. In *Proc. of PRIS 2004*, pages 108–117, 2004.

Chapter 3

Constrained leaving-one-out for language modelling

" *In mathematics the art of asking questions is more valuable than solving problems* "
GEORG FERDINAND LUDWIG PHILIPP CANTOR

Contents

3.1 Introduction

In the previous chapter, we introduced the idea of "smoothing the parametric domain" instead of smoothing the optimal set of parameters. In this way, the optimisation of the training criterion provides an optimal smoothed solution, avoiding the additional training smoothing stage. For introducing this idea we analysed as case of study the naive Bayes classifier for which simple threshold constraints were applied.

In this chapter we further explore the idea of constraining the parametric domain. Specifically, we study two aspects: the estimation criterion (see Section 1.1.2 Chapter 1) and the probabilistic model. On the one hand, the starting point of this chapter is the *leaving-one-out maximum likelihood criterion (LOO)* instead of the (conventional) maximum likelihood criterion (see Section 1.2.3 Chapter 1). On the other hand, we apply the estimation techniques to the most widespread language model, the so called n-gram model (see Section 1.2 Chapter 1).

Unfortunately, due to the large number of free parameters with respect to the training data, the n-gram model needs to resort to smoothing techniques. For instance, for a trigram language model, the events that occur only once or not at all in the training data typically represent a huge percentage of all events. The probabilities of these events are difficult to estimate with conventional methods because they occur few times in the training data. These probabilities are usually referred to as *small probabilities*.

As discussed earlier in Section 1.2 Chapter 1, the best smoothing methods, modified and original Kneser-Ney [Goodman, 2001, Ney et al., 1997], are based on the Turing-Good (TG) counts [Good, 1953, Nadas, 1984]. The back-off parametrisations of these discounts gain a probability mass B and re-distribute it among all the unseen events. The discounted probability mass is obtained subtracting to the actual count r the Turing-Good count $r^\star$ for each seen event (see Eq. (1.92) in Chapter 1).

An outstanding property of the Turing-Good (TG) counts $r^\star$ is that their sparseness is inverse to the (conventional) counts r. The smaller the count r is, the larger the count-of-counts n_r is, and, then, the more confident the estimation of $r^\star$ is. On the one hand, there is the unseen n-grams, $r = 0$; for which the TG counts are well estimated since n_0 and n_1 typically comprise a large amount of events. On the other hand, there is $R-1$, for which n_{R-1} and n_R are typically equal to 1, leading to a poorly estimated TG counts, and consequently, smoothed probabilities.

Unfortunately, the previous property of the leaving-one-out (LOO) smoothed probabilities is also one of their most important weakness, since the LOO probabilities are noisy or badly estimated for larger counts r. Typically, we want the LOO smoothed probabilities $\tilde{p}(w \mid h)$ to be "close" to the relative frequencies (the MLE estimates, $\hat{\mathrm{p}}(w \mid h)$); or, in other words, we want the TG counts to be close to the (conventional) counts. At least, we want the LOO probabilities to retain the same monotonic order that the (conventional) MLE verify; that is to say, if a n-gram hw has occurred more times [a] than other n-gram $h'w'$, $N(w,h) > N(w',h')$, then the probability of the former $\tilde{p}(w \mid h)$ should be larger than the latter, $\tilde{p}(w' \mid h')$. Note that the ML estimates fulfil this requirement whereas the LOO estimates do not ensure it.

The first proposed smoothing based on LOO, Turing-Good method, defines a smoothed language model as a function of unconstrained LOO probability estimates. This method is very noisy since for large values of r, the TG count $r^\star$ is poorly estimated. On the other hand, the Kneser-Ney [Kneser and Ney, 1995] smoothing solved that problem approximating all the probabilities with one parameter, as depicted in Eq. (1.95) in Section 1.2 Chapter 1. Thus, the Turing-Good method and the absolute discounting method represent two extremes, namely either no constraints at all or a heavily constrained model with only a single parameter. We focus, however, on finding a trade-off between the number of

[a] Under the assumption $N(h') = N(h)$; or alternatively comparing the joint probabilities $\tilde{p}(w,h)$ and $\tilde{p}(w',h')$.

free parameters and suitable constraints in order to avoid noisy estimates and achieve optimum performance.

Bare in mind that because of the assumption made when computing the LOO estimates of the discounting parameters $\boldsymbol{\lambda}_1^{R-1}$ (see Eq. (1.83) in Section 1.2 Chapter 1); the probability estimates, $\tilde{p}(w \mid h)$ in Eq. (1.88), obtained with the optimal discounting parameters $\hat{\boldsymbol{\lambda}}_1^{R-1}$, do not necessary fulfil the conditional normalisation constraints,

$$\sum_w \tilde{p}(w \mid h) = 1, \qquad \forall h \in \mathcal{W}^{n-1} \quad , \tag{3.1}$$

although they verify a joint normalisation constraint

$$\sum_w \sum_h \tilde{p}(w, h) = 1, \qquad . \tag{3.2}$$

At this point is important to recall the equivalence between the joint model, $\tilde{p}(w, h)$, and the conditional model, $\tilde{p}(w \mid h)$, which is depicted in Section 1.2 Chapter 1, when some assumption is taken (see Eq. (1.83) in Section 1.2 Chapter 1).

Moreover, due to the way in which we use the parameters $\boldsymbol{\lambda}_1^{R-1}$ when defining the smoothing model as depicted in Eq. (1.88), the discounted probability B_h can be 0 or even negative if $r < r^\star$. In such cases, the heuristic approaches renormalise those parameters adding the negative probability mass plus 1, i.e. $-B_h + 1$, to the total probability amount; or deactivate the smoothing for that n-gram, as it is done in the SRILM toolkit [Stolcke, 2002]. In this chapter, we also try to avoid such problems.

In summary, since the TG counts $r^\star$ are obtained from data, they suffer from similar sparsity problems than that of the original counts r. In this chapter, we present some novel estimation algorithms to avoid the sparsity problems for the LOO estimates while trying to retain an optimal trade-off between the number of parameters and their sparsity. We tackle this problem by constraining the domain, likewise to Chapter 2, in order to force the optimal value to fulfil desirable properties that a not overfitted solution must satisfy. This idea was previously outlined in [Kneser and Ney, 1995], where monotonic and interval constrains were suggested but not applied. Specifically, in this chapter, we present 4 methods that seek to optimise LOO while ensuring monotonicity:

- *Interval constraints [Andrés-Ferrer and Ney, 2009]:* in Section 3.3, we highly constrain the LOO probability estimates ensuring monotonicity.
- *Quasi-monotonic constraints:* this method computes the LOO estimates so that they are monotonic except for probability of the two most frequent probabilities.This exception allow us to find a simpler algorithm in Section 3.4 that paves the way for the following proposed algorithms in Section 3.5.
- *Monotonic constraints with upper bounds:* this method computes the LOO estimates so that they are monotonic and never larger than the MLE estimates. In this way, in Section 3.5, we avoid negative or zero discounted probability mass B_h.
- *Monotonic constraints:* in subsection 3.5.1, we compute the LOO estimates so that they all are monotonic by modifying the algorithm presented in Section 3.5.

Additionally to these 4 methods, we present the *extended and exact Kneser-Ney (eeKN)* [Andrés-Ferrer and Ney, 2009] algorithm in Section 3.6 that computes an exact estimation of the Kenser-Ney (KN) discount and the modified Kneser-Ney (mKN) discount while defining a parameter to fix the number of free probability estimates.

In particular, all the methods except for the exact extended Kneser-Ney (eeKN) enforce the monotonicity of (almost) all the probability estimates. Although the eeKN does not ensures the monotonicity, this method provides a meta-parameter that is used to practically enforce this monotonicity.

The remaining of this chapter is organised as follows. In Section 3.2, we propose a more comfortable formulation for the application of leaving-one-out estimates to smooth a n-gram language model. In Section 3.8 we analyse the proposed methods experimentally, drawing conclusions when possible; and finally, future work and concluding remarks are gathered in the last Section.

3.2 Leaving-one-out for language modelling

Recall from Section 1.2 in Chapter 1 that some smoothings models for n-gram modelling are obtained by computing the smoothing parameters of the model in Eq. (1.93) by leaving-one-out (LOO) . The solution to this model is reviewed in Section 1.2 and is depicted in Eq. (1.85).

However, in order to apply the constraints, there is a more suitable parametrisation for the model defined in Eq. (1.93). There, we presented the smoothing model using the standard *discounting* parametrisation in which a probability mass is subtracted to the conventional ML estimates. In the new parametrisation, the whole discounted probability is regarded as a parameter p_r. Recall that optimising the conditional model under the assumption in Eq. (1.83) is equivalent to directly optimising the joint model, and, hence, we re-parametrise the joint model as follows,

$$\tilde{p}(w,h) := \begin{cases} \frac{R}{N} & N(w,h) = R \\ \mathrm{p}_r & 0 < N(w,h) = r < R \\ n_0\, \mathrm{p}_0\, \beta(w|\bar{h}) & N(w,h) = 0 \end{cases} \tag{3.3}$$

where $\mathbf{p}_0^{R-1}$ or $\mathbf{p}$ is subject to the *joint* normalisation constraint

$$\sum_{r=0}^{R} n_r\, \mathrm{p}_r = 1 \quad . \tag{3.4}$$

Note that this model is equivalent to the joint model in Eq. (1.93). For instance, the discounted probability for a given history h is given by

$$B_h = n_0(h)\, \mathrm{p}_0 \quad , \tag{3.5}$$

whereas the total amount of discounted probability mass independent of any history, is

$$B = \sum_h B_h = n_0\, \mathrm{p}_0 \quad . \tag{3.6}$$

Note that there is a direct conversion between the model parametrised as a function of $\mathbf{p}_0^{R-1}$ and as a function of $\boldsymbol{\lambda}_1^{R-1}$; and vice-versa. This conversion is given by the following expression

$$\mathrm{p}_r = (1-\lambda_r)\frac{r}{N}, \qquad r = 1, \ldots, R-1 \quad . \tag{3.7}$$

In this new model parametrised with $\mathbf{p}_0^{R-1}$, the LOO log-likelihood function is given by

$$\mathrm{F}(\mathbf{p}_0^{R-1}) = \sum_{r=0}^{R-1} (r+1) n_{r+1} \log \mathrm{p}_r + \mathrm{const}(\mathbf{p}_0^{R-1}) \quad . \tag{3.8}$$

In order to obtain the new optimal parameter set $\mathbf{p}_0^{R-1}$, the log-likelihood in Eq. (3.8) is maximised subject to the normalisation constraint in Eq. (3.4). For doing so, we define the Lagrangian function

$$\mathcal{F}(\mathbf{p}_0^{R-1}, \gamma) = \sum_{r=0}^{R-1} (r+1) n_{r+1} \log \mathrm{p}_r - \gamma \left(\sum_{r=0}^{R} n_r\, \mathrm{p}_r - 1 \right) \quad , \tag{3.9}$$

and take the partial derivative with respect to p_r and γ

$$\frac{\partial\,\mathcal{F}(\mathbf{p}_0^{R-1},\gamma)}{\partial\,\mathrm{p}_r} = \frac{(r+1)n_{r+1}}{\mathrm{p}_r} - \gamma n_r\,, \quad r=0,1,\ldots,R-1 \tag{3.10}$$

$$\frac{\partial\,\mathcal{F}(\mathbf{p}_0^{R-1},\gamma)}{\partial\gamma} = \sum_{r=0}^{R} n_r\,\mathrm{p}_r - 1 \quad . \tag{3.11}$$

The optimal set of parameters must verify that the former partial derivatives are equal to 0, from where the optimal solution is obtained

$$\hat{\mathrm{p}}_r = \frac{1}{N}\frac{(r+1)n_{r+1}}{n_r}\left(1 - n_R\frac{R}{N}\right)\,, \qquad r=0,1,\ldots,R-1 \quad . \tag{3.12}$$

Since typically $n_R\frac{R}{N} \ll 1$, it can be approximated as

$$\hat{\mathrm{p}}_r = \frac{1}{N}\frac{(r+1)n_{r+1}}{n_r}, \qquad r=0,1,\ldots,R-1 \quad . \tag{3.13}$$

Note that Turing-Good counts are *generalised* under this parametrisation as

$$r^\star = \mathrm{p}_r\,N \quad , \tag{3.14}$$

for joint models; and as

$$r^\star(h) = \mathrm{p}_r\,N(h) \quad , \tag{3.15}$$

for conditional ones.

The result in Eq. (3.12) is totally equivalent to Eq. (1.85), since if we plug the optimal parameters $\hat{\boldsymbol{\lambda}}_1^{R-1}$ into the Eq. (3.7), then the same optimal parameters $\mathbf{p}_1^{R-1}$ are obtained

$$\begin{aligned}
\hat{\mathrm{p}}_r &= (1-\hat{\lambda}_r)\frac{r}{N} \qquad r=1,\ldots,R-1\,,\\
\hat{\mathrm{p}}_r &= \left(1 - 1 + \frac{n_{r+1}(r+1)}{r n_r}\left(1 - \frac{n_R R}{N}\right)\right)\frac{r}{N}\,, \qquad r=1,\ldots,R-1\\
\hat{\mathrm{p}}_r &= \frac{1}{N}\frac{(r+1)n_{r+1}}{n_r}\left(1 - \frac{n_R R}{N}\right)\,, \qquad r=1,\ldots,R-1 \quad .
\end{aligned}$$

Note that $n_0\,\mathrm{p}_0$ is the probability mass reserved to the unseen events, i.e. B, and hence, p_0 is the most important probability when LOO is used for smoothing, since it gives probability for the unseen events. However, it is important to highlight that we obtain the probability mass to the unseen events at the expense of reducing the probability mass of the seen events. The question tackled in this chapter is how to discount the probability mass from the seen events so that the probabilities are still monotonic and optimal for the LOO criterion.

The discounting idea in the new parametrisation given in Eq. (3.3) is less obvious than in the parametrisation given in Eq. (1.93). However, since we have that all the probabilities must sum up to 1 (constraint in Eq. (3.1))

$$\begin{aligned}
B &= n_0\,\mathrm{p}_0 \qquad (3.16)\\
B &= 1 - \sum_{r=1}^{R} n_r\,\mathrm{p}_r\\
B &= \frac{1}{N}\left(N - \sum_{r=1}^{R} n_r\,\mathrm{p}_r\,N\right) \quad ,
\end{aligned}$$

where taking into account the following property

$$N = \sum_{r=1}^{R} r n_r \quad , \tag{3.17}$$

B is expressed as

$$B = \frac{1}{N}\left(\sum_{r=1}^{R} r n_r - \sum_{r=1}^{R} n_r \, \mathrm{p}_r \, N\right) \quad , \tag{3.18}$$

and grouping common terms

$$B = \frac{1}{N}\sum_{r=1}^{R} n_r \, (r - \mathrm{p}_r \, N) \quad . \tag{3.19}$$

Finally, by using the definition of the TG counts in Eq. (3.14), the following discounting equation is obtained

$$B = \frac{1}{N}\sum_{r=1}^{R} n_r \, (r - r^\star) \quad , \tag{3.20}$$

from where the discounting process is depicted as $r - r^\star$. It is worth noting that a similar expression can be obtained for the conditional normalisation constraint in case of using a conditional smoothing model

$$B_h = \frac{1}{N(h)}\sum_{r=1}^{R} n_r(h) \, (r - r^\star(h)) \qquad \forall h \in \mathcal{W}^{n-1} \quad . \tag{3.21}$$

3.2.1 The smoothing distribution $\beta(w|\bar{h})$

The smoothing distribution $\beta(w|\bar{h})$ can also be estimated by LOO. The result of applying LOO to the estimation of $\beta(w|\bar{h})$ yields the following result [Ney et al., 1997]:

$$\hat{\beta}(w|\bar{h}) = \frac{N_1(w|\bar{h})}{N_1(\bar{h})} \quad , \tag{3.22}$$

which resembles the (conventional) MLE $(n-1)$-gram distribution but defined with the especial counts $\{N_1(w,\bar{h})\}$ instead of the conventional counts $\{N(w,\bar{h})\}$. These especial counts are defined as follows

$$N_1(w|\bar{h}) = \sum_{h \in \bar{h}: N(w,h)=1} 1 \quad , \tag{3.23}$$

where $h \in \bar{h}$ stands for all the n-gram contexts h, that share the same prefix $(n-1)$-gram history, $\bar{h}$; i.e., if $\bar{h} = \boldsymbol{w}_2^{n-1}$, then $h \in \bar{h}$ comprises all h such that $h = w\boldsymbol{w}_2^{n-1}$ for any word $w \in \mathcal{W}$. Finally, the $N_1(\bar{h})$ is the sum over all words of $N_1(w|\bar{h})$

$$N_1(\bar{h}) = \sum_{w} N_1(w|\bar{h}) \quad . \tag{3.24}$$

The smoothing distribution in Eq. (3.22), captures the idea of word coupling. For instance, "New York" is a coupled bi-gram, that means that if "York" is a frequent word so would it be "New" and vice-versa. However, the smoothing distribution should give high probabilities to words that have rarely been

observed in the context. Therefore, if instead of using $N_1(w,\bar{h})$ to estimate the smoothing distribution, we use $N(w,\bar{h})$, the smoothing distribution are over-estimated.

Several works reported [Chen and Goodman, 1998, Ney et al., 1997] that changing the singleton counts $N_1(\cdots)$ by positive counts $N_+(\cdots)$

$$N_+(w|\bar{h}) = \sum_{h\in\bar{h}:N(w,h)\geq 1} 1 \quad , \tag{3.25}$$

incur in better perplexities. Therefore, to counteract the word coupling effect, we recompute counts by the number of different contexts in which the smoothed n-gram occurs. High orders of this smoothing distribution are often recursively smoothed with LOO until the uni-gram distribution following the smoothing scheme presented for the language model.

3.2.2 The interpolated smoothing model

All the previous discussion is focused on the back-off model. This model redistributes the gained probability mass B among the unseen events. Oppositely, the interpolation model redistributes this gained probability mass among all the events, both seen and unseen. Therefore, our joint interpolation model is given by

$$\tilde{p}(w,h) := \begin{cases} \frac{R}{N} + i\beta(w,\bar{h}) & N(w,h) = R \\ \mathrm{p}_r + i\beta(w,\bar{h}) & 0 < N(w,h) = r < R \\ i\beta(w,\bar{h}) & N(w,h) = 0 \end{cases} \tag{3.26}$$

with the discounted probabilities $\mathbf{p}_1^{R-1}$ and the interpolation factor i comprising the parameter set. Note that the smoothing probability distribution $\beta(w,\bar{h})$ must sum up to 1 for all n-grams, and not just the unseen ones, i.e.,

$$\sum_h \sum_w \beta(w,h) = 1 \quad , \tag{3.27}$$

and, finally, the probabilities must sum 1,

$$\sum_{r=1}^{R} n_r \, \mathrm{p}_r + i = 1 \quad . \tag{3.28}$$

Usually this interpolation parameter is estimated in the backing-off model and afterwards assumed to be the same in the interpolation model. That is to say, it is often assumed that

$$i = n_0 \, \mathrm{p}_0 \quad . \tag{3.29}$$

It is worth highlighting that this assumption is actually true if the optimal smoothing probability $\beta(w,\bar{h})$ is known. Although, we focus on this chapter on the backing-off models, we take the previous assumption to compute interpolated smoothing models in the experimental Section 3.8.

3.3 Interval Constraints

The goal of this method is to modify the original MLE probabilities, $\mathrm{p}_r = r/N$, only a little bit. Therefore, we introduce what we call the interval constraints

$$\frac{r-1}{N} \leq \mathrm{p}_r \leq \frac{r}{N}, \qquad r = 1,\ldots,R-1 \quad , \tag{3.30}$$

and

$$\mathrm{p}_0 \leq \frac{1}{N} \quad . \tag{3.31}$$

Recall that the upper bound in Eq. (3.30) is the MLE estimate.

As presented in the model in Eq. (3.3), the probability p_R is not used in the LOO log-likelihood function and its value is fixed to the MLE.

Therefore, mathematically, we want to maximise the LOO log-likelihood obtained in Eq. (3.8)

$$F(\mathbf{p}_0^{R-1}) = \sum_{r=0}^{R-1} (r+1) n_{r+1} \log \mathrm{p}_r \tag{3.32}$$

constrained to Eqs. (3.31), (3.30) and the normalisation constraint

$$\sum_{r=0}^{R} n_r \, \mathrm{p}_r = 1 \quad , \tag{3.33}$$

and recalling that p_R is fixed to the MLE, i.e.

$$\mathrm{p}_R = \frac{R}{N} \quad .$$

The idea of applying these constraints was previously outlined in [Kneser and Ney, 1995], where an heuristic and not optimal solution was proposed. In order to obtain an optimal solution to the problem, we use the Karush-Kuhn-Tucker (KKT) conditions (see Appendix A). In this case, the Lagrangian function is instanced to

$$\mathcal{L}(\mathbf{p}_0^{R-1}, \lambda, \boldsymbol{\mu}, \boldsymbol{\nu}) = \sum_{r=0}^{R-1} (r+1) n_{r+1} \log \mathrm{p}_r - \Lambda(\mathbf{p}_0^{R-1}, \lambda) - \Psi(\mathbf{p}_0^{R-1}, \boldsymbol{\mu}, \boldsymbol{\nu}) \quad ,$$

where $\Lambda(\mathbf{p}_0^{R-1}, \lambda)$ is the Lagrangian part for the normalisation constraint

$$\Lambda(\mathbf{p}_0^{R-1}, \lambda) = \lambda \left(\sum_{r=0}^{R} n_r \, \mathrm{p}_r - 1 \right) \quad ,$$

and where $\Psi(\mathbf{p}_0^{R-1}, \boldsymbol{\mu}, \boldsymbol{\nu})$ is the part generated by the inequality constraints

$$\Psi(\mathbf{p}_0^{R-1}, \boldsymbol{\mu}, \boldsymbol{\nu}) = \sum_{r=2}^{R-1} \mu_r \left(\frac{r-1}{N} - \mathrm{p}_r \right) + \sum_{r=1}^{R-1} \nu_r \left(\mathrm{p}_r - \frac{r}{N} \right) + \nu_0 \left(\mathrm{p}_0 - \frac{1}{N} \right) \quad .$$

In this case the KKT conditions are reduced to

$$\frac{(r+1)n_{r+1}}{\mathrm{p}_r} - n_r\lambda + \mu_r - \nu_r = 0 \qquad r = 1,\ldots,R-1 \tag{3.34}$$

$$\frac{n_1}{\mathrm{p}_0} - n_0\lambda - \nu_0 = 0 \tag{3.35}$$

$$\sum_{r=0}^{R} n_r\,\mathrm{p}_r = 1 \tag{3.36}$$

$$\mu_r\left(\frac{r-1}{N} - \mathrm{p}_r\right) = 0 \qquad r = 2,\ldots,R-1 \tag{3.37}$$

$$\nu_r\left(\mathrm{p}_r - \frac{r}{N}\right) = 0 \qquad r = 1,\ldots,R-1 \tag{3.38}$$

$$\nu_0\left(\mathrm{p}_0 - \frac{1}{N}\right) = 0 \tag{3.39}$$

$$\mu_r \geq 0 \qquad r = 2,\ldots,R-1 \tag{3.40}$$

$$\nu_r \geq 0 \qquad r = 0,\ldots,R-1 \tag{3.41}$$

together with constraints in Eqs. (3.30) and (3.31).

In the previous KKT conditions, μ_r stands for the "strictness" of the lower bound for each probability p_r, and ν_r is the corresponding "strictness" for the upper bound. Therefore, if μ_r is greater than 0 then the lower constraint is (strictly) active, i.e. the bound is verified by equality and p_r has the value of the lower bound. Oppositely, if μ_r is equal to 0, then the lower bound is not (strictly) active and p_r can take values larger than it. This interpretation of the Lagrangian multipliers allows us to make a case analysis depending on the multipliers for a given probability p_r with $r > 1$:

- *"Lower bound is active"*, then $\mu_r > 0$ and $\nu_r = 0$
- *"Upper bound is active"*, then $\mu_r = 0$ and $\nu_r > 0$
- *"Unbound case"*, then $\mu_r = 0$ and $\nu_r = 0$

Note that we have omitted the case $\mu_r > 0$ and $\nu_r > 0$, since it implies that both constrains are active, which is impossible by definition.

"Lower bound is active"

In this case, since $\nu_r = 0$ we can work out the value of μ_r from Eq. (3.34)

$$\mu_r = -\left(\frac{(r+1)n_{r+1}}{\mathrm{p}_r} - n_r\lambda\right) \quad . \tag{3.42}$$

As result of Eq. (3.40) and (3.42) we obtain

$$\mathrm{p}_r \geq \frac{1}{\lambda}\frac{(r+1)n_{r+1}}{n_r} \quad , \tag{3.43}$$

and using the constraints in Eq.(3.30)

$$\mathrm{p}_r \geq \frac{r-1}{N} \quad , \tag{3.44}$$

we obtain a solution by plugging previous Eqs.(3.43) and (3.44) into the constraint in Eq. (3.37)

$$\mathrm{p}_r(\lambda) = \max\left\{\frac{r-1}{N}, \frac{1}{\lambda}\frac{(r+1)n_{r+1}}{n_r}\right\} \quad , \tag{3.45}$$

which depends on a normalisation constant λ.

"Upper bound is active"

In this case, since $\mu_r = 0$ we can again work out the value of ν_r from Eq. (3.34)

$$\nu_r = \frac{(r+1)n_{r+1}}{\mathrm{p}_r} - n_r\lambda \quad . \tag{3.46}$$

As result of Eq. (3.41) and (3.46) we obtain

$$\mathrm{p}_r \leq \frac{1}{\lambda}\frac{(r+1)n_{r+1}}{n_r} \quad , \tag{3.47}$$

and using the constraints in Eq.(3.30), specifically

$$\mathrm{p}_r \leq \frac{r}{N} \quad , \tag{3.48}$$

we obtain a solution by means of the constraint in Eq. (3.38)

$$\mathrm{p}_r(\lambda) = \min\left\{\frac{r}{N}, \frac{1}{\lambda}\frac{(r+1)n_{r+1}}{n_r}\right\} \quad , \tag{3.49}$$

which depends on a normalisation constant λ.

"Unbound case"

In this case since $\mu_r = 0$ and $\nu_r = 0$, the value of p_r is straightly worked out from Eq. (3.34)

$$\mathrm{p}_r(\lambda) = \frac{1}{\lambda}\frac{(r+1)n_{r+1}}{n_r} \quad . \tag{3.50}$$

The actual solution

Finally, taking into account the result for each case depicted in Eqs. (3.45), (3.49),and (3.50), we obtain a solution dependent on a normalisation constant λ

$$\mathrm{p}_r(\lambda) = \max\left\{\frac{r-1}{N}, \min\left\{\frac{1}{\lambda}\frac{(r+1)n_{r+1}}{n_r}, \frac{r}{N}\right\}\right\} \quad . \tag{3.51}$$

Note that this solution is valid for $r = 2, \ldots, R-1$.

An analogous procedure can be carried out for the special cases $r = 0, 1$ yielding the following result

$$\mathrm{p}_r(\lambda) = \min\left\{\frac{1}{\lambda}\frac{(r+1)n_{r+1}}{n_r}, \frac{1}{N}\right\} \quad .$$

The interpretation of the solution in Eq. (3.51) is as follows. We compute the unconstrained LOO estimate $\mathrm{p}_r = \frac{1}{\lambda}\frac{(r+1)n_{r+1}}{n_r}$, with the unknown normalisation constant λ. This estimate is then compared with the lower and upper bound and it is clipped if necessary. Now the only remaining problem is that this comparison requires the normalisation constant to be known. To this purpose we introduce the λ depending *normalisation function*,

$$Q(\lambda) = \sum_{r=0}^{R} n_r\,\mathrm{p}_r(\lambda) \quad . \tag{3.52}$$

This way, the normalisation constraint is reformulated as $Q(\lambda) = 1$. Since $Q(\lambda)$ is a monotonically decreasing function, the value for λ can be easily computed.

Note that in order to ensure monotonicity the following constraint must be added to the algorithm

$$\mathrm{p}_0 \leq \mathrm{p}_1 \quad . \tag{3.53}$$

The addition of this constrain does not significantly modify the algorithm, though it becomes more awkward. The solution with the additional constraint in Eq. (3.53) can be obtained in the same way that in the proposed method but it becomes more cumbersome. Specifically, if we define

$$\bar{\mathrm{p}}_{01}(\lambda) = \frac{1}{\lambda}\frac{2n_2 + n_1}{n_1 + n_0} \quad , \tag{3.54}$$

and

$$\bar{\mathrm{p}}_{0}(\lambda) = \frac{1}{\lambda}\frac{n_1}{n_0} \quad , \tag{3.55}$$

and

$$\bar{\mathrm{p}}_{1}(\lambda) = \frac{1}{\lambda}\frac{2n_2}{n_1} \quad , \tag{3.56}$$

then the solution to the interval constraints with the additional constraint in Eq. (3.53) is given by Eq. (3.51) for all the r values but for 0 and 1 which are equal to

$$\mathrm{p}_0(\lambda) = \begin{cases} \frac{1}{N} & \bar{\mathrm{p}}_{01}(\lambda) \geq \frac{1}{N} \text{ and } \bar{\mathrm{p}}_{0}(\lambda) \geq \bar{\mathrm{p}}_{1}(\lambda) \\ \bar{\mathrm{p}}_{01}(\lambda) & \bar{\mathrm{p}}_{01}(\lambda) < \frac{1}{N} \text{ and } \bar{\mathrm{p}}_{0}(\lambda) \geq \bar{\mathrm{p}}_{1}(\lambda) \\ \bar{\mathrm{p}}_{0}(\lambda) & \text{otherwise} \end{cases} \quad , \tag{3.57}$$

and

$$\mathrm{p}_1(\lambda) = \begin{cases} \frac{1}{N} & \bar{\mathrm{p}}_{01}(\lambda) \geq \frac{1}{N} \text{ and } \bar{\mathrm{p}}_{0}(\lambda) \geq \bar{\mathrm{p}}_{1}(\lambda) \\ \bar{\mathrm{p}}_{01}(\lambda) & \bar{\mathrm{p}}_{01}(\lambda) < \frac{1}{N} \text{ and } \bar{\mathrm{p}}_{0}(\lambda) \geq \bar{\mathrm{p}}_{1}(\lambda) \\ \bar{\mathrm{p}}_{1}(\lambda) & \text{otherwise} \end{cases} \quad , \tag{3.58}$$

respectively. Let it be as it were, the constraint in Eq. (3.53) is always verified in practice, and hence, this constraint becomes useless.

3.4 Quasi-monotonic constraints

A natural requirement is that the probability estimates, p_r, should be a monotonic function of r. This is a more natural requirement than the interval constraints. The monotonic requirement is specified by the following set of constraints

$$\mathrm{p}_r \leq \mathrm{p}_{r+1}, \qquad r = 0, 1, \ldots, R-2 \quad . \tag{3.59}$$

Similar constraints were previously proposed in [Kneser and Ney, 1995] but no algorithm or solution to computed them was given. Note that, we have intentionally omitted the last monotonic constraint

$$\mathrm{p}_{R-1} \leq \mathrm{p}_R = \frac{R}{N} \quad . \tag{3.60}$$

Obviously this makes the following discourse not to ensure monotonicity, and that is why we will refer to this algorithm as *quasi-monotonic* algorithm. This assumption has two main motivations. On the one hand, dropping this last bound incur in a simpler algorithm that will allow us to introduce another algorithm in the following Section 3.5. On the other hand, the algorithm for obtaining the totally monotonic solution is a special case of the algorithm proposed in the mentioned Section 3.5. Furthermore, this constraint is (almost) always verified in practice by the current algorithm.

Thereby, for the remaining of this section, we wish to optimise the model in Eq. (3.3), constrained by the monotonic requirements in Eq. (3.59). By applying the KKT conditions (see Appendix A), only a characterisation of the solution is obtained,

$$\underbrace{\mathrm{p}_0 = \ldots = \mathrm{p}_{r_1}}_{q_0} < \underbrace{\mathrm{p}_{r_1+1} = \ldots = \mathrm{p}_{r_2}}_{q_1} < \ldots < \underbrace{\mathrm{p}_{r_{K-1}+1} = \ldots = \mathrm{p}_{r_K}}_{q_{K-1}}, \; q_K = \frac{R}{N} \tag{3.61}$$

Since either $\mathrm{p}_{r-1} < \mathrm{p}_r$ or $\mathrm{p}_{r-1} = \mathrm{p}_r$ must be verified; the solution is a structure of $K+1$ segments of probabilities with the set of boundaries $\boldsymbol{r}_0^{K+1}$

$$-1 := r_0 < r_1 < \ldots < r_K := R-1, \quad r_{K+1} := R \quad . \tag{3.62}$$

Inside the k-th segment, all the probabilities share the very same probability q_k. The number of segments range from 2 to R. For $K=1$, there are just two segments: one segment contains just one index, R, and the other segment contains the remaining indexes, $r = 0, \ldots, R-1$; which share the very same probability, q_0. For $K = R-1$, each segment is made up of one probability.

In order to simplify notation, we express the constraints in terms of the segment probabilities $\boldsymbol{q}_0^K := q_0, ..., q_K$. By defining

$$m_k = \sum_{r=r_k+1}^{r_{k+1}} n_r \quad , \tag{3.63}$$

the normalisation constraint is rewritten as

$$\sum_{k=0}^{K} m_k q_k = 1 \quad , \tag{3.64}$$

and the monotonicity constraints are summarised as

$$q_{k-1} < q_k \qquad k = 1, \ldots, K-1 \quad . \tag{3.65}$$

Recall that q_K is as usual fixed to the relative frequencies. i.e.

$$q_K = \frac{R}{N} \quad . \tag{3.66}$$

In order to obtain the solution, we start by assuming that the segmentation $\boldsymbol{r}_0^{K+1}$ is given. In such case, the LOO log-likelihood function is

$$F(\boldsymbol{q}_0^{K-1}) = \sum_{k=0}^{K-1} A_k \log q_k \quad , \tag{3.67}$$

with A_k defined as

$$A_k = \sum_{r=r_k+1}^{r_{k+1}} (r+1) n_{r+1} \quad , \tag{3.68}$$

subject to the normalisation constraint in Eq. (3.64). The Lagrangian function for this optimisation problem is

$$\mathcal{L}(\boldsymbol{q}_0^{K-1}, \lambda) = F(\boldsymbol{q}_0^{K-1}) - \Lambda(\lambda, \boldsymbol{q}_0^{K-1}) \quad , \tag{3.69}$$

with

$$\Lambda(\lambda, \boldsymbol{q}_0^{K-1}) = \lambda \left(\sum_{k=0}^{K} m_k q_k - 1 \right) \quad . \tag{3.70}$$

Constraining the partial derivatives of the Lagrangian to be equal to 0 yields the optimal solution for a given segmentation

$$q_k = \frac{1}{\lambda}\frac{A_k}{m_k} \tag{3.71}$$

where the normalisation constant λ is independent of the segmentation

$$\lambda = \frac{N}{1 - m_K q_K} = \frac{N}{1 - n_R \, \mathrm{p}_R} \tag{3.72}$$

Since we have assumed the segmentation given, whether the solution verifies the constraints in Eq. (3.65) or not depends on if the segmentation is optimal or not. Therefore, in order to obtain the boundaries of the segmentation, we should find the boundaries $\boldsymbol{r}_0^{K-1}$ that maximise the log-likelihood in Eq. (3.67) while satisfying the monotonic constraints.

This is efficiently solved by dynamic programming using the following recurrence

$$F(r) = \underset{r' < r \,:\, \mathrm{p}_{r'} < \mathrm{p}_r}{\arg\max} \{F(r'-1) + A(r', r) \log q(r', r)\} \quad , \tag{3.73}$$

with

$$A(r', r) = \sum_{s=r'}^{r} (s+1) n_{s+1} \quad , \tag{3.74}$$

and with

$$q(r', r) = \frac{1}{\lambda}\frac{A(r', r)}{\sum_{s=r'}^{r} n_s} \quad . \tag{3.75}$$

Note that $F(r)$ is the log-likelihood for the partial segmentation that ends at count r. As usual with dynamic programming, the optimal solution is obtained by tracing back the decisions made during the recurrence in Eq. (3.73).

3.5 Monotonic Constraints with Upper Bounds

In Section 3.4, we analysed the quasi-monotonic constraints. This constraints involve practical problems since the probability for an n-gram that has occurred r times can be larger than r/N, leading to conditional probabilities that may not verify the conditional normalisation constraints in Eq. (3.1). This problem is one of the model deficiencies outlined in Section 3.1, and it is derived from the assumption made in Eq. (1.83) in Chapter 1.

In order to avoid those problems, we could add another set of constraints to the formulation in Section 3.4

$$\mathrm{p}_r \leq \frac{r}{N}, \qquad r = 1, 2, \ldots, R \quad . \tag{3.76}$$

Our aim is, then, to maximise Eq. (3.8) with the normalisation constraint in Eq. (3.33), with the monotonic constraints in Eqs. (3.59) and (3.60); and with the upper boundaries defined in Eq. (3.76). Similarly to Section 3.4, if we apply the KKT conditions to the maximisation, we obtain the characterisation of the solution. The solution structure is similar to the structure in Eq. (3.61) as follows

$$\underbrace{\mathrm{p}_0 = \ldots = \mathrm{p}_{r_1}}_{q_0} < \underbrace{\mathrm{p}_{r_1+1} = \ldots = \mathrm{p}_{r_2}}_{q_1} < \ldots < \underbrace{\mathrm{p}_{r_K+1} = \ldots = \mathrm{p}_{r_{K+1}}}_{q_K} \tag{3.77}$$

with the following additional constraints

$$q_k \leq \frac{r_{k+1}}{N} \qquad k = 0, \ldots, K-1 \quad . \tag{3.78}$$

In this case, we proceed in a similar fashion to the previous section. Therefore, if the segmentation $\boldsymbol{r}_0^{K+1}$ is given, then we have to maximise Eq. (3.67) subject to the normalisation constraint in Eq. (3.64), and constrained by Eq.(3.78). Applying KKT conditions to this problem requires the definition of the following Lagrangian function

$$\mathcal{L}(\boldsymbol{q}_0^{K-1}, \lambda) = \sum_{k=0}^{K-1} A_k \log q_k - \Lambda(\lambda, \boldsymbol{q}_0^{K-1}) - \Psi(\boldsymbol{\nu}, \boldsymbol{q}_0^{K-1}) \quad , \tag{3.79}$$

with $\Lambda(\lambda, \boldsymbol{q}_0^{K-1})$ being the Lagrangian terms induced by the normalisation constraint,

$$\Lambda(\lambda, \boldsymbol{q}_0^{K-1}) = \lambda \left(\sum_{k=0}^{K} m_k q_k - 1 \right) \quad , \tag{3.80}$$

and with $\Psi(\boldsymbol{\nu}, \boldsymbol{q}_0^{K-1})$ being the Lagrangian terms concerning to the upper bounds in Eq. (3.78),

$$\Psi(\boldsymbol{\nu}, \boldsymbol{q}_0^{K-1}) = \sum_{k=0}^{K-1} \nu_k \left(\frac{q_k - r_{k+1}}{N} \right) \quad . \tag{3.81}$$

Recall that A_k is defined in Eq. (3.68).

The KKT conditions in this case are the followings

$$\frac{A_k}{q_k} - \lambda m_k + \nu_k = 0 \tag{3.82}$$

$$\nu_k \left(\frac{r_{k+1}}{N} - q_k \right) = 0 \tag{3.83}$$

$$\frac{r_{k+1}}{N} \geq q_k \tag{3.84}$$

$$\nu_k \geq 0 \tag{3.85}$$

Similarly to Section 3.3, we proceed by cases:

- *Bound is active*, then $\nu_k > 0$.
- *Bound is not active*, then $\nu_k = 0$.

On the one hand, if the bound is active then, using Eq. (3.83) we get the value of q_k

$$q_k = \frac{r_{k+1}}{N} \quad . \tag{3.86}$$

On the other hand, if the bound is not active, we work out the value of q_k from Eq.(3.82)

$$q_k = \frac{1}{\lambda} \frac{A_k}{m_k} \quad . \tag{3.87}$$

From Eqs. (3.83), (3.86), and (3.87), we obtain a solution depending on a normalisation constant, λ,

$$q_k(\lambda) = \min \left\{ \frac{1}{\lambda} \frac{A_k}{m_k}, \frac{r_{k+1}}{N} \right\}, \quad k = 0, \ldots, K-1 \quad , \tag{3.88}$$

where the normalisation constant λ depends on the segmentation

$$\lambda = \lambda(\boldsymbol{r}_0^{K+1}) \quad .$$

Recall that m_k is defined in Eq. (3.63) and A_k is defined in Eq. (3.68).

Therefore, if the unknown normalisation constant $\lambda(\boldsymbol{r}_0^{K+1})$ were known, then the question of whether the solution in Eq. (3.88) verifies the monotonic constraints in Eq. (3.59) or not depends on the segmentation boundaries, $\boldsymbol{r}_0^{K-1}$, that maximise Eq. (3.67), constrained by Eq.(3.78) and by Eq. (3.64). Therefore, if the normalisation constant λ is given, then we can compute the segmentation that maximises Eq. (3.67) with the following recurrence

$$F'(r) = \underset{r'\leq r\,:\,\mathrm{p}_{r'}<\mathrm{p}_r}{\arg\max} \{F'(r'-1)+A(r',r)\log(q_\lambda(r',r))\} \quad , \tag{3.89}$$

with

$$q_\lambda(r',r) = \min\left\{\frac{1}{\lambda}\frac{A(r',r)}{\sum_{s=r'}^{r} n_s}, \frac{r}{N}\right\} \quad . \tag{3.90}$$

Recall that $A(r',r)$ is defined in Eq. (3.74).

After evaluating the recurrence in Eq. (3.89) for $r = 0, ..., R$; we trace back the decisions made in the evaluation to recover the optimum segmentation. Given the optimal segmentation, it is straightforward to compute the optimal probabilities $\hat{q}_k$ for $k = 0, ..., K$.

In a fashion similar to the interval constraints in Section 3.3, we define a λ dependent normalisation function

$$Q'(\lambda) = \sum_{k=0}^{K} m_k \hat{q}_k(\lambda) \quad , \tag{3.91}$$

and reformulate the normalisation constraint in Eq. (3.64) as

$$Q'(\lambda) = 1 \quad . \tag{3.92}$$

Note that the function $Q'(\lambda)$ is defined using the probabilities of the optimal segmentation for λ.

Since the function $Q'(\lambda)$ is monotonically decreasing the normalisation constraint can be found using algorithms similar to the ones used for $Q(\lambda)$ in Section 3.3.

3.5.1 Monotonic constraints

In order to make the quasi-monotonic algorithm fully monotonic, we can develop a similar training scheme to that obtained for computing the solution to the monotonic constraints with upper bounds. We start with the quasi-monotonic wording but also adding the left-out constraint in Eq. (3.60). Afterwards, if we apply the techniques used for obtaining the solution to the monotonic constraints with upper bounds, then we obtain a solution where the only difference is in the recursion in Eq. (3.89) which now is given by

$$F''(r) = \underset{r'\leq r\,:\,\mathrm{p}_{r'}<\mathrm{p}_r}{\arg\max} \{F''(r'-1)+A(r',r)\log(q_\lambda(r',r))\} \quad , \tag{3.93}$$

where $q_\lambda(r',r)$ is defined by cases as follows

$$q_\lambda(r',r) = \begin{cases} \frac{1}{\lambda}\frac{A(r',r)}{\sum_{s=r'}^{r} n_s} & r < R-1 \\ \\ \min\left\{\frac{A(r',R-1)}{\sum_{s=r'}^{R-1} n_s}, \frac{R}{N}\right\} & r = R-1 \end{cases} \quad . \tag{3.94}$$

Unlike the quasi-monotonic case, the addition of the constraint in Eq. (3.60), makes the normalisation constant λ, not to be independent of the segmentation. Therefore it is necessary to use a scheme similar to the monotonic with upper bounds, restating the normalisation constraint as a normalisation function $Q''(\lambda)$ and requiring it to be 1.

3.6 Exact extended Kneser-Ney smoothing

The exact extended Kneser-Ney smoothing [Andrés-Ferrer and Ney, 2009] method reduces the number of free parameters in the LOO estimation by using an absolute discounting model for counts larger than a given discounting threshold S, mathematically expressed as

$$\mathrm{p}_r(\boldsymbol{p}_0^{S-1}, d) = \begin{cases} p_r & r < S \\ \frac{r-d}{N} & r \geq S \end{cases} \quad , \tag{3.95}$$

where the parameter d is the so-called discounting parameter. Obviously, this method does not guarantee that the remaining probabilities p_r for $r = 0, 1, ..., S-1$ are monotonic. Whether monotonicity is satisfied or not depends on the training data and the chosen discounting threshold S.

This estimation technique was initially presented with a fixed discounting threshold, $S = 1$ [Kneser and Ney, 1995], and afterwards extended to $S = 3$ [Chen and Goodman, 1998]. Nevertheless, no exact solution was given for the estimation of $S > 1$. In this section, we analyse the exact solution for this approach using the LOO log-likelihood criterion.

We wish to optimise the model in Eq. (3.3), but with the probabilities, p_r, depending on d as expressed in Eq. (3.95), for counts r larger or equal to the threshold S. Therefore, the log-likelihood function in Eq. (3.32) is rewritten by

$$F(\boldsymbol{p}_0^{S-1}, d) = \sum_{r=0}^{S-1} (r+1) n_{r+1} \log p_r + \sum_{r=S}^{R} (r+1) n_{r+1} \log \frac{r-d}{N} \quad , \tag{3.96}$$

subject to the normalisation constraint in Eq. (3.33) rewritten as

$$\sum_{r=0}^{S-1} n_r p_r + \sum_{r=S}^{R} n_r \frac{r-d}{N} = 1 \quad . \tag{3.97}$$

The optimal parameter set must maximise Eq. (3.96) subject to the normalisation constraint in Eq. (3.97). The Lagrangian function of such mathematical problem is

$$\mathcal{L}(\boldsymbol{p}_0^{S-1}, d, \lambda) = F(\boldsymbol{p}_0^{S-1}, d) - \Lambda(\boldsymbol{p}_0^{S-1}, d, \lambda) \quad , \tag{3.98}$$

with

$$\Lambda(\boldsymbol{p}_0^{S-1}, d, \lambda) = \lambda \left(\sum_{r=0}^{S-1} n_r p_r + \sum_{r=S}^{R} n_r \frac{r-d}{N} - 1 \right) \quad , \tag{3.99}$$

and where $F(\boldsymbol{p}_0^{S-1}, d)$ is defined in Eq. (3.96).

As usual convex optimisation problems, it is needed to compute the gradient of the Lagrangian function with respect to d, and p_r,

$$\frac{\mathcal{L}(\boldsymbol{p}_0^{S-1}, d, \lambda)}{\partial p_r} = \frac{(r+1) n_r}{p_r} - \lambda n_r, \quad r = 0, 1, \ldots, S-1 \tag{3.100}$$

$$\frac{\mathcal{L}(\boldsymbol{p}_0^{S-1}, d, \lambda)}{\partial d} = - \sum_{r=S+1}^{R} \frac{r n_r}{r-1-d} + \lambda \sum_{r=S}^{R} \frac{n_r}{N} \quad , \tag{3.101}$$

and equalling them to 0 allow us to work out the value of the optimal probability estimates, $\hat{p}_r$,

$$\hat{p}_r(d) = \frac{1}{\lambda(d)} \frac{(r+1) n_{r+1}}{n_r}, \quad r = 0, \ldots, S-1 \quad , \tag{3.102}$$

where the normalisation constant depends on d as follows

$$\lambda(d) = \left(\sum_{r=S+1}^{R} \frac{r n_r}{r-1-d} \right) \left(\sum_{r=S}^{R} \frac{n_r}{N} \right)^{-1} \quad . \tag{3.103}$$

Similarly to Sections 3.3 and 3.5, we reformulate the normalisation constraint in Eq. (3.97) by defining a normalisation function $Q'''(d)$

$$Q'''(d) = \sum_{r=0}^{S-1} n_r \hat{p}_r(d) + \sum_{r=S}^{R} n_r \frac{r-d}{N} \quad , \tag{3.104}$$

and requiring it to be equal to 1, $Q'''(d) = 1$.

The function $Q'''(d)$ is again monotonically decreasing, and therefore it is straightforward to find the optimal value $\hat{d}$ such that $Q'''(\hat{d}) = 1$

Unlike original and modified Kneser-Ney, we have not made any approximation in order to obtain the exact value for $\hat{d}$ and p_0. Additionally, the threshold count S is not fixed beforehand to be either 1 (Kneser-Ney), or 3 (modified Kneser-Ney).

Note also that although these estimation techniques were firstly introduced by defining p_r for $r < S$ as a function of a discount parameter d_r, both parametrisation are equivalent by means of the following equation

$$p_r = \frac{r - d_r}{N} \quad .$$

3.7 A word on time complexity

Until now, we have not analysed the time requirements of the proposed methods compared with the standard Kneser-Ney (KN) and modified Kneser-Ney (mKN). Obviously, all the proposed methods need to compute both the conventional n-gram counts $N(w,h)$ and the counts-of-counts (COC) n_r for $r = 0, 1, \ldots, R-1$. Therefore, we omit the time complexity required to compute those counts.

The standard KN and mKN smoothings require a time complexity of $O(R)$, since only one or three discounting parameters must be computed in order to define the probabilities $\tilde{\mathrm{p}}(w|h)$ of the model in Eq (1.75) in Chapter 1. For instance, the Kneser-Ney makes use only of n_1, n_2 and n_3 for defining just one discounting parameter b as expressed in Eq. (1.96) Chapter 1.

The proposed methods are split into two groups:

Iterative methods: comprising the constrained methods that need to perform iterations to compute the normalisation constant λ by means of a decreasingly monotonic normalisation function $Q(\lambda)$, that is to say the following methods: interval constraints, monotonic constraints with and without upper bounds; and exact extended Kneser-Ney (eeKN).

Non-iterative methods: made up of the methods that do not require to iterate, i.e., the quasi-monotonic method.

For the latter we give the total time complexity. Specifically, the quasi-monotonic constraint need to compute the recursion specified in Eq. (3.73), which can be computed in $O(R^2)$.

However, for the iterative methods, we give the time complexity for each iteration. On the one hand, the monotonic constraints with and without upper bounds, need to compute the recursions in Eqs. (3.89) and (3.93) which are similar to the recursion for the quasi-monotonic constrains requiring a time complexity of $O(R^2)$ for each iteration. Therefore, these two methods require $O(R^2 I)$ in order to find the solution where I stands for the total amount of iterations needed to find the normalisation constant.

On the other hand, the interval constraints and eeKN can compute the probability estimates in a time complexity of $O(R)$ for a given normalisation constant. This leads to a total time complexity of $O(RI)$ where I stands for the total amount of iterations needed to find the normalisation constant.

The number of iterations I varies depending on the data, the model order, n; and the smoothing algorithm. However, in all the experimentation we have carried out in the following section, it usually lays in between 10 and 50.

3.8 Experiments

Experiment setup

In this section, the practical performance of all the proposed smoothing techniques is analysed from the LM point of view. The perplexity (see Section 1.2 Chapter 1) on a test set will be used to compare all the techniques. The less the perplexity is, the better the model is. Furthermore, we also use a modified version of the perplexity, the so-called *joint perplexity* for evaluating some of the proposed methods. This is motivated by the assumption made in the modelling that makes equivalent the optimisation of the joint and conditional models. The joint perplexity is defined as the (conditional) perplexity but for the probabilities which are joint probabilities $\mathrm{p}(w,h)$ instead of the conditional probabilities $\mathrm{p}(w|h)$ as follows

$$\mathrm{PP}(T) = 2^{\frac{1}{W}\sum_{m=1}^{M}\sum_{t=1}^{T_m}\log_2 p(s_{ml},h)} \quad , \tag{3.105}$$

with the testing data S comprising a set of evaluation sentences $\{\boldsymbol{s}_1,\ldots,\boldsymbol{s}_M\}$ each of length T_m; and where W stands for the total amount of words, i.e.,

$$W = \sum_{m=1}^{M} T_m \quad .$$

In order to quantify the behaviour of each techniques, we have compared all the the proposed techniques with the baseline perplexity given by the modified Kneser-Ney [Chen and Goodman, 1998] and original Kneser-Ney [Kneser and Ney, 1995]. In order to obtain the baseline, we have used the standard SRILM toolkit [Stolcke, 2002]. Additionally, the experiments have been obtained using the smoothed back-off model in Eq. (3.3) unless it is otherwise specified.

For analysing the different smoothing techniques two corpora has been used: the English part of the Europarl v3 [Callison-Burch et al., 2007] and the Wall Street Journal (WSJ) [Kneser and Ney, 1995]. Table 3.1 summarises some statistics about the two corpora. As previously discussed, it is observed that the percentage of singletons[b] is very high for 3-grams comprising the 28.5% of the total 3-gram occurrences.

Table 3.2 contains some statistics for the testing data. The test set for the Europarl is defined in the shared task [Callison-Burch et al., 2007]; on the other hand, for the WSJ, we have selected an small percentage of paragraphs from all the years, in order to gain independence on the test set with respect to time factors.

In order to analyse the behaviour of all the techniques as a function of the training size, we have split the training into increasing sizes starting from 200K sentences and doubling the size until the full corpus, i.e., 200K, 400K, 800K, and full corpus (≈ 1.6M for WSJ and ≈ 1.4M for Europarl).

An important problem in LM evaluation is how to handle the *out of vocabulary (OOV)* words. If the OOV are not handled properly, then misleading conclusions could be drawn from evaluation. Some works [Kneser and Ney, 1995], tackled the problem by selecting the most frequent words, for instance 20K, from training and tagging the remaining words as unknown words. Then the unknown

[b] By singleton we denote here, an event such as a word or a n-gram that has occurred just once in the corpus.

Table 3.1: Table with some statistics of the both corpora used in the experiments.

Training	Europarl	WSJ
sentences	1.40M	1.62M
avg. length	24.6	26.0
running words	34.4M	42.12M
vocab. size	280.5K	200.1K
n_1/N (1-gram)	0.4 %	0.2%
n_1/N (2-gram)	18.9 %	18.1%
n_1/N (3-gram)	28.5 %	28.5%

Table 3.2: Some statistics of the both test sets.

Test	Europarl	WSJ
sentences	2K	12.5K
avg. length	26.8	26.1
running words	53.6K	326.3K

word becomes a very common event, such that small variations in its probability could dominate the perplexity differences among systems. For instance, if we take the extreme example in which all the vocabulary words are unknown, then the perplexity of any text will be 1; although the meaning of this perplexity is misleading since it does not mean that given a previous history of words the LM is able to predict the following word. Instead, this perplexity means that our model is able to predict that the following word will be *unknown* to the LM, which is useless for most of the applications.

In order to avoid these misleading conclusions, two steps have been taken. On the one hand, we report perplexity results skipping OOV n-grams. On the other hand, we have performed experiments increasing the size of the vocabulary from 10% of the vocabulary until the 100% in steps of 10%. For the 100% case in which all the vocabulary words are considered, we have reserved the smoothing probability mass for the unseen uni-grams in order to give probability to the unknown words. For doing so, the full vocabulary size must be known, however, any sensible estimation of the size suffices, specifically, we have extrapolated the number of unseen words in the vocabulary from the seen words. Bare in mind that we also report perplexities ignoring OOV words to quantify the influence of our estimation for unknown events. Table 3.4 reflects the percentage of OOV in test as a function of the training size. Furthermore, in table 3.3 the sizes of the vocabulary partitions are detailed.

In the remaining of this section, we have obtained results mainly for bi-grams and tri-grams language models. For some experiment configurations we have also computed 4-grams results. In all cases the conclusions are consistent with results obtained for the tri-gram language model.

Theoretical properties in practice

We can gain some insights into the constrained technique by analysing the TG counts $r^\star$. Therefore, in Figure 3.1 the TG counts, $r^\star = \mathrm{p}_r\, N$, are plotted as a function of the original counts r. The plots were obtained using a 4-gram language model in the 200k partition of the WSJ corpus and using the

Table 3.3: Percentage of out of vocabulary words (OOV) in test as a function of the training size and the percentage of vocabulary size. The figures represent percentages, i.e., 5.4 stands for 5.4% words.

Voc. Pctg.	Corpus	200K	400K	800K	full size
10%	Europarl	5.4	4.0	2.9	2.1
	WSJ	6.3	4.9	3.7	2.8
20%	Europarl	3.0	2.1	1.5	1.1
	WSJ	3.4	2.5	1.8	1.4
30%	Europarl	2.1	1.4	1.0	0.8
	WSJ	2.3	1.6	1.2	0.8
40%	Europarl	1.7	1.2	0.9	0.6
	WSJ	1.7	1.2	0.8	0.6
50%	Europarl	1.4	1.0	0.7	0.5
	WSJ	1.4	0.9	0.7	0.4
60%	Europarl	1.3	0.9	0.7	0.5
	WSJ	1.2	0.8	0.5	0.3
70%	Europarl	1.2	0.9	0.6	0.5
	WSJ	1.0	0.7	0.5	0.3
80%	Europarl	1.1	0.8	0.6	0.5
	WSJ	1.0	0.6	0.4	0.3
90%	Europarl	1.1	0.7	0.5	0.5
	WSJ	0.8	0.6	0.4	0.2
100%	Europarl	1.0	0.7	0.5	0.4
	WSJ	0.7	0.5	0.3	0.2

full vocabulary. It is worth noting that the original TG counts wildly oscillate for larger values of r, i.e.m the estimation of $r^\star$ is *noisy*. Recall that we have already outlined this property of the TG counts when analysing the LOO smoothing model deficiencies in Section 3.1.

It is valuable to mention, the way in which each of the proposed methods counteract such overtraining. The probability estimates obtained with the interval constraints tend to strictly verify one of the constraints, either the upper or the lower. The (quasi-)monotonic constraints[c], howeverer, tend to produce strips with the same TG count. These strips are sometimes larger than the count r itself. Oppositely, the monotonic constraints with upper bounds method avoids this undesirable result by splitting these strips whenever necessary. Finally, in the case of the eeKN and for the chosen discounting threshold ($S = 30$), it can be seen that the TG counts are not monotonic. However, if we had chosen $S = 10$ instead, then the counts would have been monotonic, since all the noisy counts are approximated by one discount as depicted in Figure 3.1.

The figure 3.2 is the analogous version of figure 3.1 but for using a 3-gram model instead of 4-gram model. When comparing figures 3.1 and 3.2, we observe that when the n-gram order, i.e. n, is increased the counts oscillate more wildly.

[c]Since the only difference between the quasi-monotonic and the monotonic constraints is the upper constraint $\mathrm{p}_{R-1} \leq R/N$; their plots are virtually the same.

Table 3.4: Vocabulary size as a function of the training size and the percentage of the full vocabulary. The figures represent Kilo-words, i.e., 8.7 stands for 8.7K words.

Voc. Pctg.	Corpus	200K	400K	800K	full size
10%	Europarl	8.7	11.7	15.3	20.0
	WSJ	10.2	14.6	20.9	28.1
20%	Europarl	17.4	23.3	30.7	40.0
	WSJ	20.4	29.2	41.8	56.1
30%	Europarl	26.2	35.0	46.0	60.0
	WSJ	30.7	44.0	62.7	84.2
40%	Europarl	34.9	46.6	61.3	80.0
	WSJ	40.9	58.3	83.7	112.2
50%	Europarl	43.6	58.3	76.7	100.0
	WSJ	51.1	73.0	104.6	140.3
60%	Europarl	52.3	70.0	92.0	120.1
	WSJ	61.3	87.5	125.5	168.3
70%	Europarl	61.1	81.6	102.4	140.1
	WSJ	71.6	102.1	146.4	196.4
80%	Europarl	69.8	93.3	122.7	160.1
	WSJ	81.8	116.7	167.3	224.4
90%	Europarl	78.5	104.9	138.1	180.1
	WSJ	92.0	131.3	188.2	252.5
100%	Europarl	87.2	116.6	153.5	200.0
	WSJ	102.3	145.8	209.1	280.5

Finally, the figure 3.3 depicts one of the normalisation functions, $Q(\lambda)$. Specifically, we have selected the normalisation function for the interval constraints. In this plot, it is observed that the function is monotonically decreasing. Note that the total number of seen n-grams, N, is equal to $5\,210\,341$ and the optimal normalisation constant $\hat{\lambda}$ takes the value of $5\,244\,023$. The normalisation functions of the other proposed smoothings show a similar behaviour of that depicted in figure 3.3 but for the eeKN smoothing.

For the eeKN case, we have plotted the normalisation function in figure 3.4. Recall that, the normalisation function for the eeKN smoothing is parametrised depending on the discounting parameter d instead of a normalisation constant λ. In the figure 3.4, we have plotted the normalisation function $Q(d)$ for several n-gram models in the case of eeKN with $S = 3$. The larger the n-gram order is, the larger the discounting parameter d is. Specifically, the discounting parameter takes the value of 0.61, 1.01, 1.13 and 1.20 respectively for uni-gram, bi-gram, tri-gram and four-gram models.

Backing-off results

Firstly, in the figures 3.5 and 3.6 we analyse the practical behaviour of all the methods involving monotonic constraints: quasi-monotonic, monotonic with upper bounds and monotonic. The first surprising result is that the (conventional) Kneser-Ney (KN) outperforms the modified Kneser-Ney (mKN). Re-

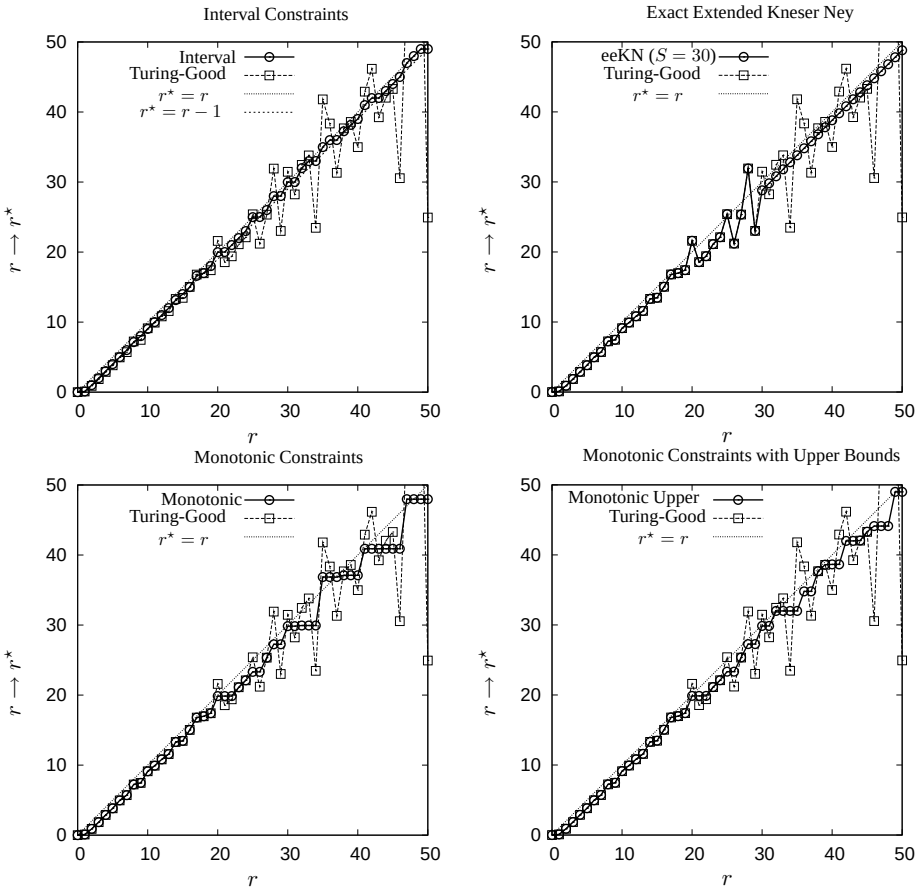


Figure 3.1: The 4-gram modified count $r^\star$ as a function of the original count r for the 4 proposed techniques obtained with 200K sentences partition of WSJ and full vocabulary size.

sults in [Chen and Goodman, 1996] report that the mKN outperforms the KN smoothing, however, this result are obtained using a linear discounting instead of a backing-off discount. Afterwards, we analyse the behaviour of linear discounting smoothings, and, then, we will see that mKN outperforms the KN for interpolated discounting methods.

In figure 3.5, the (conditional) perplexity ignoring OOV events is plotted as a function of the percentage of most frequent words from the vocabulary. The three techniques show virtually the same behaviour which slightly improves the best baseline, KN. This improvement, is systematic and grows with the vocabulary size. Although, the plot in figure 3.5 was obtained with a 3-gram language model, other orders such as 2-gram or 4-gram, obtain similar plots. For the case of the perplexity without ignoring the OOV the plots show a similar behaviour. Therefore, we do not include the plots for the full perplexity (with OOV) since the behaviour is similar but with a slightly smaller gap.

Figure 3.6, comprises two plots with the perplexity as a function of the training size for the WSJ

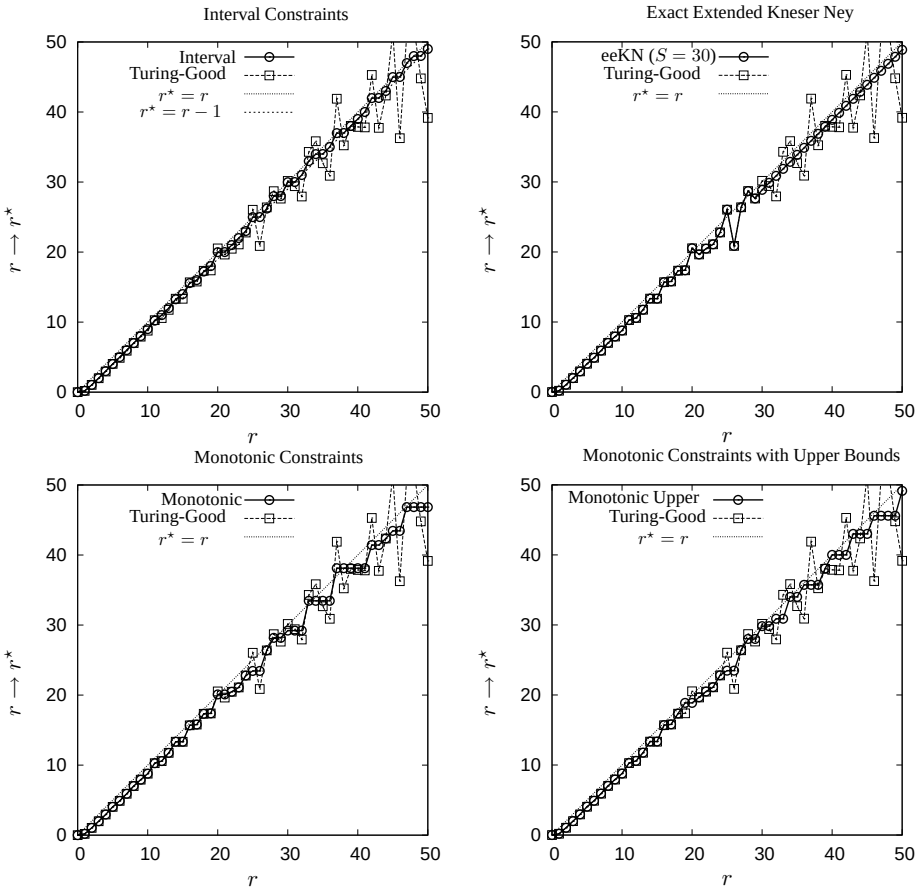


Figure 3.2: The 3-gram modified count $r^\star$ as a function of the original count r for the 4 proposed techniques obtained with 200K sentences partition of WSJ and full vocabulary size.

corpus. The top plot, shows the performance of the monotonic smoothing techniques for a 2-gram language model. The bottom plot is the analogous plot for 3-gram language model. There is not any difference between the performance of the monotonic models which obtain slightly better results than the KN baseline. It is observed that the improvement gap is larger for low order models.

Since we have not observed any significant difference between the monotonic approaches, we will henceforth take monotonic with upper bounds as the representative of these smoothing techniques. Furthermore, the KN smoothing is taken as the baseline since it obtains better results for a back-off smoothing scheme than the mKN.

The behaviour of the exact and extended Kneser-Ney (eeKN) is depicted in figure 3.7. It is observed that the best perplexity results are obtained with $S = 1$, that is to say, with just 2 free parameters: p_0 and d. Note that in this case the eeKN is just a different estimation of the conventional KN smoothing. Anyway, as the training data increases, the differences between different choices of this discounting

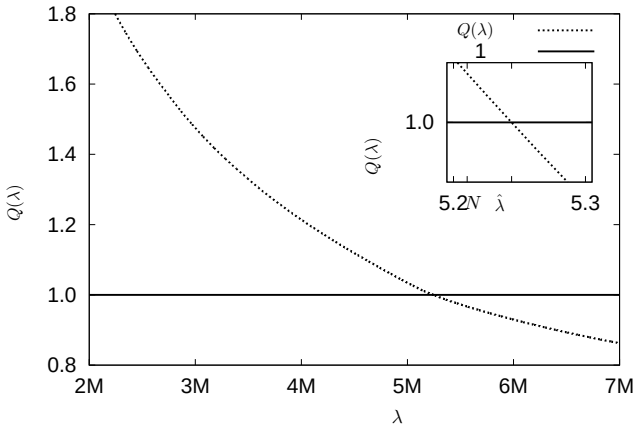


Figure 3.3: The normalisation function $Q(\lambda)$ for a interval constraint 3-gram model computed with the 200K sentences partition of WSJ and full vocabulary.

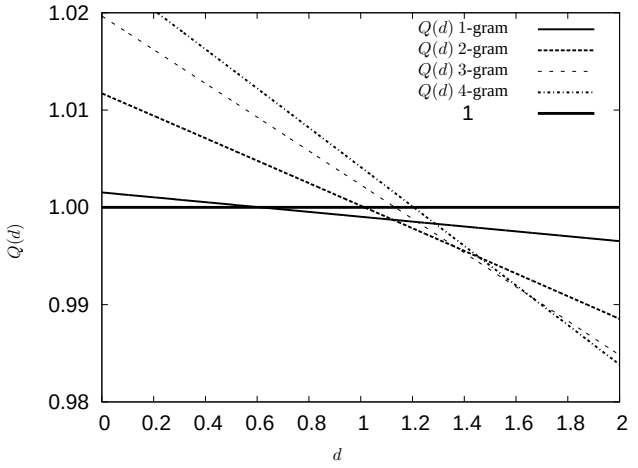


Figure 3.4: The normalisation function $Q(d)$ for the eeKN ($S = 3$) smoothing computed with the 200K sentences partition of WSJ and full vocabulary size.

threshold do not significantly modify the result. It is of worth noting that for the tested n-grams ($n = 2, 3, 4$), the eeKN always outperforms the KN smoothing. The more scarce training data is, the larger the improvement is.

In Table 3.5, we compare the perplexities obtained in both corpora (full vocabulary) using a trigram

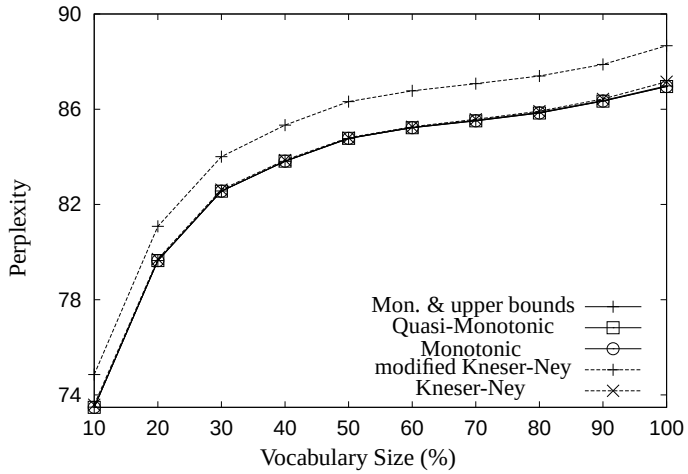


Figure 3.5: Perplexity skipping OOV events as a function of vocabulary size (in %) with the full size partition of the Europarl corpus for a 3-gram language model smoothed with all the monotonic approaches and the standard smoothings Kneser-Ney (KN) and modified Kneser-Ney (mKN).

language model. We can conclude that all the proposed techniques perform at least as the baseline, being better in certain circumstances. It is observed that eeKN outperforms the baseline in all the circumstances. It is also observed that all the proposed monotonic constraints yield the same result.

In the first place, we had expected better results with the monotonic approaches since they are less restrictive than the interval constraints. Recall from Section 1.2.3 in Chapter 1, that for optimising the probability estimates, we have smoothed the conditional probabilities $\tilde{p}(w|h)$ with the model in Eq. (1.75) and in order to optimise the parameters, we have taken the assumption in Eq. (1.83) which leads to the optimisation of the joint model in Eq. (1.93). Therefore, we may loose performance by the assumption made in Eq. (1.83). Figure 3.8 shows the *joint* perplexity as defined in Eq. (3.105) for the the different approaches using the full size partition of the WSJ corpus. We have plotted only the monotonic constraints with upper bounds since all the monotonic constraints obtain the very same joint perplexities as well as conditional perplexities.

Two conclusions can be drawn from figure 3.8. On the one hand, the joint perplexity always increases with the training size since it is computed with the joint probabilities and, hence, the more n-grams we observe the smaller the probabilities are in average. On the other hand, it clearly shows that the monotonic approaches are significantly influenced by the assumption in Eq. (1.83) which make us to optimise a joint model instead of a conditional model. Actually, all the proposed smoothings but for the monotonic constrained, have almost the same behaviour according to the joint perplexity. This common behaviour is also shared by the KN. Therefore, the behaviour observed with the conditional perplexity is due to the fact that the smoothings are degraded when passing from the joint model to the conditional model. The models which obtain better results are less degraded than the others. This fact is somehow surprising and inspiring for future work (see Section 3.9).

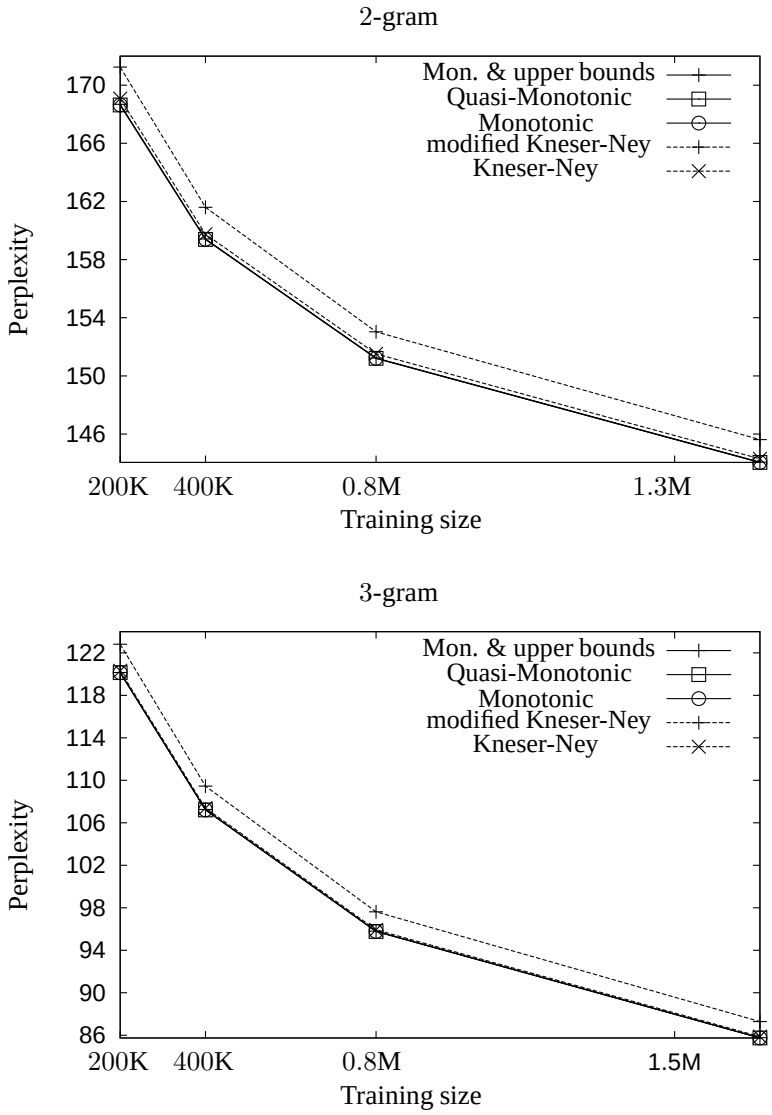


Figure 3.6: Perplexity a function of the WSJ training size with full vocabulary, and for all the monotonic approaches and the standard smoothings Kneser-Ney (KN) and modified Kneser-Ney (mKN).

Linear interpolation results

Although the theory and the proposed methods are aimed at a backing-off smoothing model, we can experimentally use the smoothings in a linear interpolation model as discussed in Section 3.2.2.

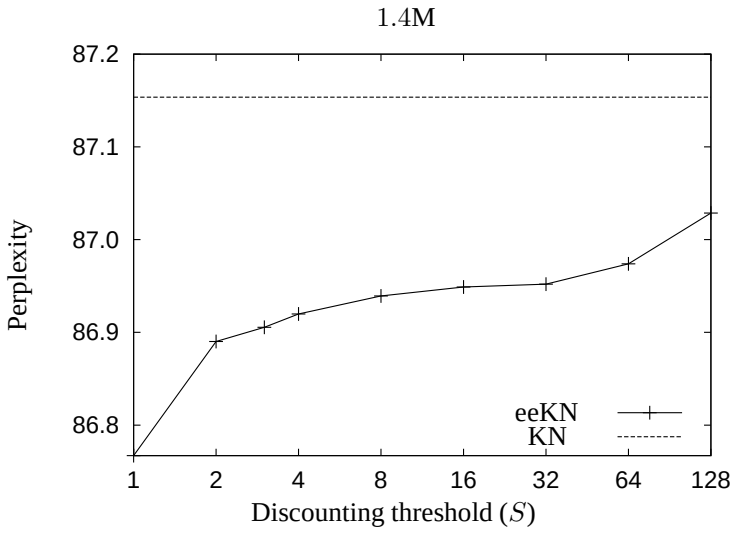


Figure 3.7: Perplexity ignoring OOV events as a function of the eeKN discounting threshold (S) in logarithmic scale computed with the Europarl corpus and using a 3-gram language model.

As can be seen in table 3.6, the mKN outperforms the KN as expected in this case. In general, all the proposed smoothing techniques are significantly degraded when used in a interpolation smoothing, obtaining similar results to that of the mKN. This degradation is also observed for 4-grams and 2-gram.

It is also observed that for the interpolation case the discounting threshold of the eeKN seems to play an important role. In Fig. 3.9, it is clearly observed that the best result is obtained with the discounting threshold, $S = 3$.

If we compare tables 3.5 and 3.6, it is observed that in general the interpolated smoothing obtains better results.

3.9 Conclusions and future work

Standard discounting models based on leaving-one-out estimates represent two extremes. On the one extreme the absolute discounting (Kneser-Ney) reduces the number of parameters to estimate to one. On the other extreme the Turing-Good smoothing estimates all the LOO probabilities, producing small probabilities and over-fitting problems.

In this chapter, we have developed novel discounting methods that are less restrictive than absolute discounting approaches, but more restrictive than Turing-Good method. Therefore, we try to optimise the trade-off between the number of parameters and the data scarcity.

Specifically, we have proposed five novel discounting methods based on constraining leaving-one-out estimates: interval constraints, quasi-monotonic constraints, monotonic constraints, monotonic constraints with upper bound and the exact extended Kneser-Ney smoothing. The associated estimation algorithms are also derived in order to compute the discounted estimates in an efficient way. We have also

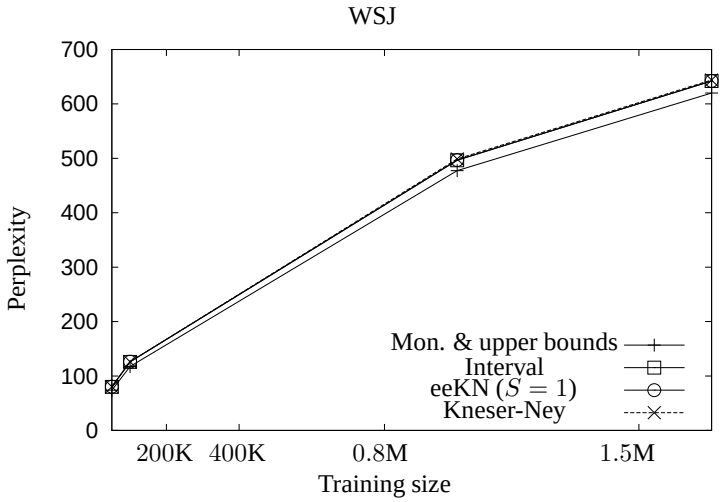


Figure 3.8: *Joint perplexity* as a function of WSJ training size for several smoothing techniques applied to a 3-gram language model.

performed systematic experiments for two language modelling tasks, comparing the proposed methods with other standard discounting methods. This experimentation reports slight improvements over the baseline of the KN/mKN method under some circumstances, specially for scarce data. However, an improvement in terms of perplexity does not always imply an improvement in terms of word error rate (WER). As future work we intend to check if the perplexity improvement are transferred to the system performance.

We have found several surprising and interesting conclusions. Firstly, all the monotonic-tagged methods (quasi-monotonic, monotonic, and monotonic with upper bounds) behave similarly. This is very surprising, since except for the monotonic with upper bound, all monotonic smoothing models have normalisation deficiencies. As discussed in Section 3.1, due to the way in which we use the joint probability estimates p_r to define the conditional smoothing model as depicted in Eq. (3.3), the discounted probability B_h can be 0 or even negative if $r < r^\star$. In such cases, the heuristic approaches renormalise those parameters adding the negative probability mass plus 1, i.e. $-B_h + 1$, to the total probability amount. We thought that this arbitrary renormalisation was distorting the probability estimates degrading the system performance. Oppositely, we found that fixing theoretically that problem by adding upper boundaries to the probability estimates p_r, do not incur in any profit when compared with deficient models fixed heuristically.

For eeKN case, we found that several values of the discounting threshold S, obtain worse results than that of the minimum value $S = 1$ for a backing-off smoothing. From previous works, where the mKN obtained better results than KN; this conclusion was unexpected since the KN can be understood as an alternative estimation of the case $S = 1$ and the mKN as an alternative estimation for the case $S = 3$. However, in a interpolation model, we have found that eeKN with $S = 3$ outperforms $S = 1$, which is consistent with the fact that mKN outperforms KN with interpolation smoothing. Hence, it seems that the fact of redistribution the discounted probability mass over all the events has a positive effect, however in this case is important to use 3 ($S = 3$) different discounting parameters oppositely

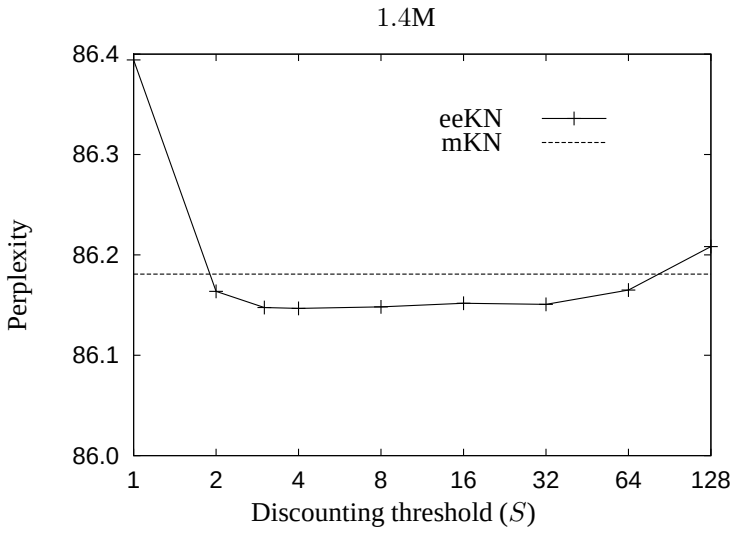


Figure 3.9: Perplexity ignoring OOV events as a function of the eeKN discounting threshold (S) in logarithmic scale computed with the Europarl corpus and using a interpolated smoothed 3-gram language model.

to back-off smoothing.

Another interesting observation is that directly applying the optimal smoothing parameters for the backing-off smoothing model to the interpolated model degrades all the smoothings. It would be interesting to apply the proposed theory to an interpolated smoothing model, in order to see whether the proposed smoothings improve the interpolation baseline or not.

Finally, the most surprising conclusion is that the monotonic-tagged smoothings do not report an improvement with respect to the interval constraint in terms of (conditional) perplexity. However, if we define a joint version of such perplexity then these models obtain a higher performance, as expected. Therefore, the assumption of optimising a joint model instead of the conditional model has some important and negative repercussions. Actually, all the discounting methods but for the monotonically-tagged methods, obtain almost the same results in terms of joint perplexity. This make us think that the main difference among the smoothings methods is the way in which each smoothing is degraded when passing from a joint model to a conditional one.

From the discussion above, we expect to obtain improvements by directly optimising a conditional smoothing model without any assumption. As future work, we intend to optimise conditional probabilities avoiding the map to a joint model.

Europarl								
Training Size	200K		400K		800K		Full Size	
	All	Sk OOV	All	Sk OOV	All	Sk OOV	All	Sk OOV
mKN	117.7	117.0	106.0	105.2	96.3	95.7	89.3	88.7
KN	115.3	114.0	104.1	103.0	94.8	93.9	88.0	87.2
Monotonic Upper	115.3	113.7	104.0	102.7	94.6	93.6	87.9	87.0
Quasi-Monotonic	115.3	113.7	104.0	102.7	94.6	93.6	87.9	87.0
Monotonic	115.3	113.7	104.0	102.7	94.6	93.6	87.9	87.0
Interval	115.1	113.6	103.8	102.6	94.5	93.6	87.8	86.9
eeKN ($S = 1$)	**114.8**	**113.2**	**103.6**	**102.3**	**94.4**	**93.4**	**87.6**	**86.7**
eeKN ($S = 3$)	115.1	113.6	103.8	102.6	94.5	93.5	87.8	86.9
Wall Street Journal (WSJ)								
mKN	120.3	118.6	107.4	106.0	95.9	94.8	85.9	85.2
KN	120.3	118.6	107.4	106.0	95.9	94.8	85.9	85.2
Monotonic Upper	120.2	118.2	107.2	105.7	95.8	94.6	85.8	85.0
Quasi-Monotonic	120.1	118.2	107.2	105.7	95.8	94.6	85.7	85.0
Monotonic	120.1	118.2	107.2	105.7	95.8	94.6	85.7	85.0
Interval	119.9	118.0	107.1	105.6	95.7	94.5	85.7	84.9
eeKN ($S = 1$)	**119.7**	**117.7**	**106.9**	**105.3**	**95.6**	**94.4**	**85.6**	**84.8**
eeKN ($S = 3$)	120.0	118.0	107.1	105.6	95.7	94.5	85.7	84.9

Table 3.5: Perplexities on the corpora for a *backing-off* smoothed 3-gram language model. *Sk OOV* column stands for the perplexity skipping the OOV, while the *All* column accumulates all the events (OOV and known).

Europarl								
Training Size	200K		400K		800K		Full Size	
	All	Sk OOV	All	Sk OOV	All	Sk OOV	All	Sk OOV
mKN	123.9	111.6	109.3	101.3	97.9	92.7	**90.4**	86.2
KN	124.2	111.6	109.6	101.5	98.2	92.9	90.8	86.4
Monotonic Upper	123.9	111.3	109.3	101.2	98.0	**92.6**	90.5	86.2
Quasi-Monotonic	124.0	111.3	109.3	101.2	98.0	**92.6**	90.5	86.2
Monotonic	124.0	111.3	109.3	101.2	98.0	**92.6**	90.5	86.2
Interval	124.0	111.3	109.3	101.2	97.9	**92.6**	90.5	**86.1**
eeKN ($S = 1$)	124.3	111.5	109.6	101.4	98.2	92.9	90.8	86.4
eeKN ($S = 3$)	**123.8**	**111.2**	**109.2**	**101.1**	**97.2**	**92.6**	90.5	**86.1**
Wall Street Journal (WSJ)								
mKN	**126.6**	116.3	**110.7**	104.5	**97.6**	**93.9**	**86.8**	**84.5**
KN	127.1	116.6	111.2	104.8	98.1	94.2	87.1	84.9
Monotonic Upper	126.8	**116.2**	110.9	104.5	97.8	**93.9**	86.8	84.6
Quasi-Monotonic	126.8	**116.2**	110.9	104.5	97.8	**93.9**	86.9	84.6
Monotonic	126.8	**116.2**	110.9	104.5	97.8	**93.9**	86.9	84.6
Interval	126.8	116.3	110.9	104.5	97.8	**93.9**	86.9	84.6
eeKN ($S = 1$)	127.3	116.6	111.4	104.9	98.2	94.3	87.2	84.9
eeKN ($S = 3$)	126.7	**116.2**	110.9	**104.4**	97.7	**93.9**	**86.8**	**84.5**

Table 3.6: Perplexities on the corpora for a *linear interpolation* smoothed 3-gram language model. *Sk OOV* column stands for the perplexity skipping the OOV, while the *All* column accumulates all the events (OOV and known).

Bibliography

Jesús Andrés-Ferrer and Hermann Ney. Extensions of absolute discounting (kneser-ney method). In *IEEE International Conference on Acoustics, Speech, and Signal Processing*, Taipei, Taiwan, 2009. Association for Computational Linguistics.

Chris Callison-Burch, Philipp Koehn, Cameron Shaw Fordyce, and Christof Monz, editors. *Proceedings of the Second WSMT*. Association for Computational Linguistics, Prague, Czech Republic, June 2007. URL `http://www.aclweb.org/anthology/W/W07/W07-02`.

S. F. Chen and J. Goodman. An empirical study of smoothing techniques for language modeling. In *Proc. of ACL'96*, pages 310–318, Morristown, NJ, USA, June 1996. Association for Computational Linguistics.

Stanley Chen and Joshua Goodman. An empirical study of smoothing techniques for language modelling. Technical Report TR-10-98, Harvard University, 1998. See http://research.microsoft.com/ joshuago/tr-10-98.ps.

I. J. Good. Population frequencies of species and the estimation of population parameters. *Biometrika*, 40: 237–264, 1953.

Joshua Goodman. A bit of progress in language modeling. *CoRR*, cs.CL/0108005, 2001.

R. Kneser and H. Ney. Improved backing-off for m-gram language modeling. *IEEE Int. Conf. on Acoustics, Speech and Signal Processing*, II:181–184, May 1995.

A. Nadas. On turing's formula for word probabilities. *IEEE Trans. Acoustics, Speech, and Signal Proc.*, 33: 1,414–1,416, 1984.

H. Ney, S. Martin, and F. Wessel. Statistical language modeling using leaving-one-out. In S. Young and G. Bloothooft, editors, *Corpus-Based Statiscal Methods in Speech and Language Processing.*, pages 174–207. Kluwer Academic Publishers, 1997.

A. Stolcke. SRILM – an extensible language modeling toolkit. In *Proc. of ICSLP'02*, pages 901–904, September 2002.

Chapter 4

The loss function in statistical pattern recognition

" *Life's most important questions are, for the most part, nothing but probability problems.* "
PIERRE-SIMON LAPLACE

Contents

4.1 Introduction

Statistical pattern recognition is a well-founded discipline that allow us to solve many practical classification problems. A classification problem is stated as the problem of choosing to which class a given object belongs. Let $\boldsymbol{X}$ be the domain of the objects that a classification system might observe; and Ω the set of possible classes ($\{\omega_1, \omega_2, \ldots, \omega_C\}$) to which an object may belong to. A classification system is characterised by a function that maps each object to one class, the so-called *classification function (*$\mathrm{c} : \boldsymbol{X} \rightarrow \Omega$*)* [Duda et al., 2001].

The performance of a classification system is usually measured as a function of the classification error. However, there are problems in which all the classification mistakes or misclassifications do not have the same repercussions. Therefore, a criterion that ranks these mistakes should be provided. The *loss function,* $\mathrm{l}(\omega_p|\boldsymbol{x}, \omega_c)$, evaluates the *loss* in which the classification system incurs when classifying the object $\boldsymbol{x}$ into the class ω_p, knowing that the correct class is ω_c [Duda et al., 2001]. It is well known that, if a 0–1 loss function is provided, then the optimal system minimises the classification error rate [Duda et al., 2001].

This chapter is mainly devoted to design loss functions that should improve system performance while keeping the simplicity of 0–1 optimal classification system. In [R. Schlüter and Ney, 2005] complex classification rules were analysed using a *metric loss function.* Some other works, for instance [Ueffing and Ney, 2004], analyse the most general loss functions. However, we focus on other loss functions which are not restricted by the metric requirements at the expense of ignoring the class proposed by the system, i.e. ω_p.

The remainder of this chapter is organised as follows. In Section 4.2 pattern recognition problems are analysed from a decision theory point of view. In Section 4.3, we introduce statistical machine translation as a case of study. Finally, concluding remarks are summarised in Section 4.4.

4.2 Bayes Decision Theory

In this Section, we review and develop some of the ideas introduced in Section 1.1 Chapter 1. A classification problem is an instance of a decision problem. From this point of view, a classification problem is composed of three different items:

1. A set of *Objects* ($\boldsymbol{X}$) the system might observe and has to classify.
2. A set of classes ($\Omega = \{\omega_1, \ldots, \omega_C, \ldots\}$) in which the system has to classify each observed object $\boldsymbol{x} \in \boldsymbol{X}$.
3. A *Loss function,* $\mathrm{l}(\omega_p|\boldsymbol{x}, \omega_c)$, used to weight the classification actions. This function evaluates the loss of classifying an observed object $\boldsymbol{x}$ into a class, $\omega_p \in \Omega$, knowing that the *optimal class* for the object $\boldsymbol{x}$ is $\omega_c \in \Omega$.

Recall that a classification system is characterised by the classification function, which maps each object to one class [Duda et al., 2001]

$$\mathrm{c} : \boldsymbol{X} \rightarrow \Omega \quad . \tag{4.1}$$

Therefore, when an object $\boldsymbol{x} \in \boldsymbol{X}$ is observed in a classification system, the system should choose the "correct" class from all possible classes. Taking this framework into account, we define the risk of a system when classifying an object $\boldsymbol{x}$, the so-called *conditional risk given* $\boldsymbol{x}$, as

$$R(\omega_p|\boldsymbol{x}) = \sum_{\omega_c \in \Omega} \mathrm{l}(\omega_p|\boldsymbol{x}, \omega_c)\, p_r(\omega_c|\boldsymbol{x}) \quad . \tag{4.2}$$

Note that the conditional risk is the expected value of the loss function, $\mathrm{l}(\omega_p|\boldsymbol{x}, \omega_c)$, with respect to the actual probability distribution, $p_r(\omega|\boldsymbol{x})$.

Using the conditional risk, we define the *the global risk* [Duda et al., 2001] as the contribution of all objects to the classifier performance, i.e.

$$R(\mathrm{c}) = \mathrm{E}_{\boldsymbol{x}}[R(\mathrm{c}(\boldsymbol{x})|\boldsymbol{x})] = \int_{\mathcal{X}} R(\mathrm{c}(\boldsymbol{x})|\boldsymbol{x})\, p_r(\boldsymbol{x}) d\boldsymbol{x} \quad , \tag{4.3}$$

where $R(\mathrm{c}(\boldsymbol{x})|\boldsymbol{x})$ is the conditional risk given $\boldsymbol{x}$, as defined in Eq. (4.2).

We wish to design the classification function that minimises the global risk. Since minimising the conditional risk for each object $\boldsymbol{x}$ is a sufficient condition to minimise the global risk, without any loss of generality, the optimal classification rule, namely *minimum Bayes' risk*, is the one that minimises the conditional risk, i.e.

$$\hat{\mathrm{c}}(\boldsymbol{x}) = \arg\min_{\omega \in \Omega} R(\omega|\boldsymbol{x}) \quad . \tag{4.4}$$

Therefore, depending which loss function the system design is based on, there is a different optimal classification rule.

The algorithms that perform the minimisation in previous Eq. (4.4), are often called *decoding algorithms* or *search algorithms*. Consequently, the problem of designing an algorithm that perform such minimisation is called *the decoding problem* or *the search problem*.

Throughout this chapter we focus on the way of building the optimal classification system with the best possible model. We do not intend to discuss about which training criterion, method or algorithm is better for improving the system performance. Instead, we deal with the following stage in the design of the system. Once we have the best possible approximation to the actual probability distributions, we answer the question of which the best decoding strategy is.

In practice, we also need to compare among systems. In order to do so, we need to compare the global risk of those systems. The global risk in Eq. (4.3), can be understood as the expected loss with respect to the object-class joint probability distribution

$$R(\mathrm{c}) = \mathrm{E}_{\boldsymbol{x}}[R(\mathrm{c}(\boldsymbol{x})|\boldsymbol{x})] = \int_{\mathcal{X}} \sum_{\omega \in \Omega} \mathrm{l}(\mathrm{c}(\boldsymbol{x})|\boldsymbol{x}, \omega)\, p_r(\omega, \boldsymbol{x}) d\boldsymbol{x} \quad , \tag{4.5}$$

with $p_r(\omega, \boldsymbol{x}) = p_r(\omega|\boldsymbol{x})\, p(\boldsymbol{x})$. Therefore, using the law of great numbers for a given test set, $T = \{(\boldsymbol{x}_n, \omega_n)\}_{n=1}^{N}$, i.i.d. according to $p_r(\omega, \boldsymbol{x})$, the global risk can be approximated by

$$\bar{R}_T(\mathrm{c}) = \frac{1}{N} \sum_{n=1}^{N} \mathrm{l}(\mathrm{c}(\boldsymbol{x}_n)|\boldsymbol{x}_n, \omega_n) \quad . \tag{4.6}$$

We call this approximation the *empirical risk* on the test set T.

There is not a unique best loss function for any system, since the loss depends on the characterisation of the task that we want to solve. The classical and most common approach is to consider that each misclassification has the same impact. Therefore, a priori we distinguish two sorts of actions: wrong classifications (loss of 1) and correct classifications (zero loss), i.e.,

$$\mathrm{l}(\omega_p|\boldsymbol{x}, \omega_c) = \begin{cases} 0 & \omega_p = \omega_c \\ 1 & \text{otherwise} \end{cases} \tag{4.7}$$

This loss function is known as the *0–1 loss function*.

Minimising the risk when the loss function is the *0–1 loss function*, is equivalent to minimise the classifying errors. When Eq. (4.7) is used, the minimum Bayes' risk in Equation (4.4) is simplified yielding the well-known optimal Bayes' classification rule [Duda et al., 2001],

$$\mathrm{c}(\boldsymbol{x}) = \arg\max_{\omega \in \Omega} p_r(\omega \,|\, \boldsymbol{x}) \quad , \tag{4.8}$$

where $\boldsymbol{x}$ is the object to be classified, and ω denotes a possible proposed class.

However, while the 0–1 loss function is adequate for many problems, which have a small set of classes; there are problems where a more appropriate loss function should be defined. For example, if the system classifies diseases, it may be worse to classify an ill person as a healthy one than vice-versa. This distinction is made independently of the illness probability, and depending upon the repercussions of the wrong actions, i.e., depending on the following questions: is the illness treatment dangerous? is the illness deadly?, and so on. Another important example is the case in which the set of classes is large, or even infinite (but still enumerable). In such a case, as the set of all possible classes is huge, it is not appropriate to penalise all wrong classes with the same weight. In other words, since it is impossible to define a uniform distribution when the number of classes is infinite, it does not make sense to define a uniform loss function in the infinite domain because there are objects that are more probable than others, and the error will be increased if the system fails in probable objects. Instead of using the 0–1 loss function, it would be better to highly penalise the domain zones where the probability is high. In this way, the system will avoid mistakes on probable objects at the expense of making mistakes on unlikely objects. Consequently, the error will be decreased since unlikely objects occur fewer times in comparison with probable objects. Note that we are dealing with infinite enumerable sets in this example, and, therefore, this is a classification problem and not a linear regression problem. An example of this idea is plotted at Fig. 4.1

The most general loss function that can be defined makes use of the three variables: the object to classify $\boldsymbol{x}$, the proposed class ω_p and the correct class ω_c. In general, it is useless to define a non-zero loss function when the proposed class and the correct class are equal. Therefore, we define the *error function* $\epsilon(\boldsymbol{x}, \omega_p, \omega_c)$ as the value of the loss function when $\omega_p \neq \omega_c$. For each error function we define a loss function in the following way

$$\mathrm{l}(\omega_p|\boldsymbol{x}, \omega_c) = \begin{cases} 0 & \omega_p = \omega_c \\ \epsilon(\boldsymbol{x}, \omega_p, \omega_c) & \text{otherwise} \end{cases} \tag{4.9}$$

The error function must verify the following finiteness property,

$$\sum_{\omega_c \in \Omega} p_r(\omega_c|\boldsymbol{x})\ \epsilon(\boldsymbol{x}, \omega_p, \omega_c) < \infty \quad , \tag{4.10}$$

since the conditional risk defined $R(\omega_p|\boldsymbol{x})$ in Eq. (4.2) must exist.

The optimal Bayes' classification rule corresponding to the previous loss function in Eq. (4.9) is

$$\mathrm{c}(\boldsymbol{x}) = \arg\min_{\omega_p \in \Omega} \sum_{\omega_c \neq \omega_p} p_r(\omega_c|\boldsymbol{x})\, \epsilon(\boldsymbol{x}, \omega_p, \omega_c) \quad . \tag{4.11}$$

The previous classification rule in Eq. (4.11), has a great disadvantage. In order to classify an object we have to perform the minimisation which also implies a sum over all classes. If we compare the rules in Eq. (4.11), and the rule in Eq. (4.8), it is clear that in the former case, the cost is higher since the sum over all possible correct classes should be performed. This sum is not important if the number of classes is small, however, in several appealing language problems such as statistical machine translation or speech recognition the number of classes is huge or even infinite (enumerable). In those cases, the sum inside the minimisation could be even unfeasible.

The loss functions in Eq. (4.9) and in Eq. (4.7) represent two extremes of the loss function possibilities. On the one hand, the 0–1 loss function yields the simplest and fastest classification rule. On the other hand, the general loss function in Eq. (4.11), is the most general loss but also the slowest one.

There is another category of loss functions which represent a trade-off between computational cost and generality. This category is characterised by the property of ignoring the proposed class ω_p in the

error function. Therefore, if we define the following loss function,

$$\mathrm{l}(\omega_p|\boldsymbol{x},\omega_c) = \begin{cases} 0 & \omega_p = \omega_c \\ \epsilon(\boldsymbol{x},\omega_c) & \text{otherwise} \end{cases} \quad , \tag{4.12}$$

where the dependence of $\epsilon(\cdots)$ on the proposed class ω_p is dropped, and, then, the optimal classification rule is given by

$$\mathrm{c}(\boldsymbol{x}) = \arg\min_{\omega_p \in \Omega} \sum_{\omega_c \neq \omega_p} p_r(\omega_c|\boldsymbol{x})\ \epsilon(\boldsymbol{x},\omega_c) \quad . \tag{4.13}$$

Applying some basic arithmetic operations to the classification rule in previous Eq. (4.13), the classification rule is significantly simplified, i.e.,

$$\mathrm{c}(\boldsymbol{x}) = \arg\min_{\omega_p \in \Omega} \left\{ \sum_{\omega_c \neq \omega_p} p_r(\omega_c|\boldsymbol{x})\, \epsilon(\boldsymbol{x},\omega_c) \right\} \tag{4.14}$$

$$= \arg\min_{\omega_p \in \Omega} \{ -p_r(\omega_p|\boldsymbol{x})\ \epsilon(\boldsymbol{x},\omega_p) + S(\boldsymbol{x}) \} \tag{4.15}$$

$$= \arg\max_{\omega_p \in \Omega} \{ p_r(\omega_p|\boldsymbol{x})\ \epsilon(\boldsymbol{x},\omega_p) \} \tag{4.16}$$

with $S(\boldsymbol{x}) = \sum_{\omega \in \Omega} \epsilon(\omega,\boldsymbol{x})\, p_r(\omega|\boldsymbol{x})$.

Note that comparing Eqs. (4.16) and (4.8), it is observed that the cost is almost the same, except for the computation of $\epsilon(\boldsymbol{x},\omega_p)$. Actually, all the constant error functions, i.e. $\epsilon(\boldsymbol{x},\omega_p) = c$, lead to the same classification rule than the 0–1 loss function in Eq. (4.8). Therefore, the 0–1 loss function is the simplest error function of this category. If we further compare Eq. (4.16) with Eq. (4.11), it is seen that the former is fastest that the latter, since in the former a sum over all the class domain must be computed for each candidate class in the minimisation search. We classify the loss functions into two categories:

- The *general loss functions* characterised in Eq. (4.9) that require to scan the set of classes twice: one to compute the minimisation and another scan in order to compute the sum for each candidate in the former minimisation (see Eq. (4.11)).
- The *simplified loss functions* that drop the dependence on the proposed class defined as detailed in Eq. (4.12), and that only require to scan the set of classes once for computing the maximisation (see Eq. (4.16)).

Analysing the Eq. (4.12), the question of which the best error function is, raises immediately. The answer is not easy, and it depends on the task and problem for which we are designing the classification system. For instance, if the number of classes is huge or even infinite, a good approximation is to use the probability distribution over the classes, i.e. $\epsilon(\boldsymbol{x},\omega_c) = p_r(\omega_c)$. Figure 4.1 plots this idea. Note that since there are classes in the domain with a small probability of occurrence, it is useless to uniformly distribute the loss. For instance, let assume that ω_h is the most probable class and that ω_l is one of the less probable classes. We further assume that for a given object $\boldsymbol{x}$, the loss of classifying it in each of both classes, say ω_h and ω_l, is the same. If $\boldsymbol{x}$ belongs to ω_l, then we could misclassify it by assigning it to the class ω_h and vice-versa. Since the class ω_h is more probable, the system could fail more times than if the loss of misclassifying an object of the class ω_h were the highest at the expense of reducing the loss of misclassifying objects belonging to class ω_l. Note that this fact is independent of the quality of the models used to approximate the actual probabilities. This idea is analysed into detail for the statistical machine translation problem in Section 4.3.

According to previous argument, if the loss were $\epsilon(\boldsymbol{x},\omega_c) = p_r(\omega_c,\boldsymbol{x})$ then we should expect that the system would work even better. The difference between the marginal probability and the joint

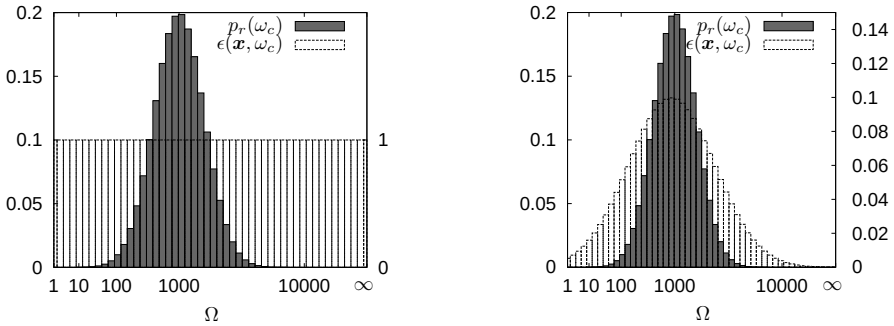


Figure 4.1: Difference of using a 0–1 loss function (on the left) and an approximation to the true class probability as the loss function (on the right) when using a loss function of the sort of Eq. (4.12). The left-scale of the y axis shows a possible actual probability over the target sentences. The right-scale of the y axis shows the value of the loss function when a mistake is made. Finally, the x axis is an infinite enumeration of the numerical identifiers corresponding to the infinite enumerable set of possible classes (or target sentences in the SMT case).

probability is that we can modify the loss on the correct class depending on each object. Obviously, this refines the accuracy of the the loss.

A more general approach can be used for mixing different models and information sources. It consists in defining an additional training step to optimise a parametrised loss function. We start by defining a family of error functions, Υ, and identifying each function in the family with some vector of parameters, say $\boldsymbol{\lambda}$. Then, we use another function criterion, say $C(\epsilon_{\boldsymbol{\lambda}}(\boldsymbol{x}, \omega_c))$, in order to range between the classification systems. Afterwards, with the help of an optimisation method, either theoretical or practical, the vector $\boldsymbol{\lambda}$ is optimised. In practice, this is used to approximate a generic error function $\epsilon(\boldsymbol{x}, \omega_c, \omega_p)$ with a faster error function that drops the dependence on the proposed class, i.e. $\epsilon_{\boldsymbol{\lambda}}(\boldsymbol{x}, \omega_c)$. In this way, we design a fast classification rule, that approximate our real classification risk. In order to perform the minimisation, a validation set is typically used. This idea is further explored in Section 4.3.3 under the view of statistical machine translation.

4.3 Statistical Machine Translation

In this section, we propose and analyse different loss functions which are eligible for substituting the 0–1 loss function in pattern recognition problems. Since, this substitution is specially appealing when the set of classes is infinite, we focus on the real scenario of *statistical machine translation (SMT) [Brown et al., 1993]*.

In Chapter 1, we stated the SMT problem as the problem of finding the translation $\mathbf{y}$ for a given source sentence $\mathbf{x}$. SMT is a specific instance of a classification problem where the set of possible classes is the set of all the possible sentences that might be written in a target language, i.e. $\Omega = \boldsymbol{Y}^*$, where $\boldsymbol{Y}$ is the target lexicon. Likewise, the objects to be classified[a] are sentences of a source language, i.e. $\mathbf{x} \in \boldsymbol{X}^*$, where $\boldsymbol{X}$ is the source lexicon.

[a] In this context to classify an object $\mathbf{x}$ in the class ω_c is a way of expressing that ω_c is the translation of $\mathbf{x}$.

Recall from Section 1.3 that, the SMT systems are based on the Bayes' classification rule for the 0–1 loss function depicted in Eq. (4.8). Usually, the class posterior probability is decomposed using Bayes' theorem into two probabilities,

$$\hat{\mathbf{y}} = \hat{\mathrm{c}}(\mathbf{x}) = \underset{\boldsymbol{y}_p \in \boldsymbol{Y}^*}{\arg\max} \{ p_r(\mathbf{x}|\boldsymbol{y}_p)\, p_r(\boldsymbol{y}_p) \} \quad , \tag{4.17}$$

which is known as the *inverse translation rule (ITR)* since the translation probability, $p_r(\mathbf{x}|\boldsymbol{y}_p)$, is defined in an inverse way, i.e., we define a probability distribution over the source sentence $\boldsymbol{x}$ which is the information that is "given" to the system. On the other hand, a direct model distributes the probability among the target sentences $\boldsymbol{y}_p$ conditionally to the given information $\boldsymbol{x}$, $p_r(\boldsymbol{y}_p|\boldsymbol{x})$. Note that we used $\boldsymbol{y}_p$ instead of $\boldsymbol{y}$ to highlight the fact that $\boldsymbol{y}_p$ plays the role of the proposed translation in the definition of the loss function.

Equation (4.17) implies that the system has to search the target string $\hat{\boldsymbol{y}}$ that maximises the product of both, the target language model $p_r(\mathbf{y})$ and the inverse translation model $p_r(\mathbf{x}|\mathbf{y})$. Nevertheless, using this rule implies, in practice, changing the distribution probabilities as well as the models through which the probabilities are approached. This is exactly the advantage of this approach, as it allows the modelling of the direct translation probability $p_r(\mathbf{y}|\mathbf{x})$ with two models: an inverse translation model that approximates the direct probability distribution $p_r(\mathbf{x}|\mathbf{y})$; and a language model that approximates the language probability $p_r(\mathbf{y})$.

This approach has a strong practical drawback: the search problem. This search is known to be an NP-hard problem [Knight, 1999, Udupa and Maji, 2006]. However, several approximate search algorithms have been proposed in the literature to solve this problem efficiently [Al-Onaizan et al., 1999, Brown et al., 1990, García-Varea and Casacuberta, 2001, Germann et al., 2001, Jelinek, 1969, Tillmann and Ney, 2003, Wang and Waibel, 1997].

Another drawback of the ITR, is that it is obtained using the 0–1 loss function. As stated in Section 4.2, this loss function is not particularly appropriate when the number of classes is huge as it happens in SMT problems. Specifically, if the correct translation for the source sentence $\mathbf{x}$ is $\mathbf{y}_c$, and the hypothesis of the translation system is $\mathbf{y}_p$; then using the 0-1 loss function (Eq. (4.7)) has the consequence of penalising the system in the same way, independently of which translation the system proposes $\mathbf{y}_p$ and which the correct translation $\mathbf{y}_c$ is.

4.3.1 General error functions

As stated above, the most generic loss functions depicted in Eq (4.9), produce minimisations which require the computation of a sum over all the set of classes. Machine translation is a classification problem with a huge set of classes. Hence, the most generic loss functions yield difficult search algorithms which are approximated. There are some works that have already explored this kind of loss functions [R. Schlüter and Ney, 2005, Ueffing and Ney, 2004].

The more appealing application of this loss functions is the use of a metric loss function [R. Schlüter and Ney, 2005]. For instance, in machine translation one widespread metric is the WER (see Section 1.3 for a definition), since the loss function in Equation (4.12) depends on both, the proposed translation and the reference translation, the WER can be used as loss function [Ueffing and Ney, 2004]. Nevertheless, due to the high complexity, the use of these general loss functions, is only feasible in constrained situations like n-best lists [Kumar and Byrne, 2004].

4.3.2 Simplified error functions

The search algorithms generated by the classification rule in Eq. (4.12) have the same asymptotic cost than 0–1 loss function, at the expense of dropping the dependence on the proposed class. As stated in

Section 4.2, a more suitable loss function than the 0–1 loss, is obtained using as the error function the target sentence probability, $\epsilon(\mathbf{x}, \mathbf{y}_c) = p_r(\mathbf{y}_c)$,

$$\mathrm{l}(\mathbf{y}_p|\mathbf{x}, \mathbf{y}_c) = \begin{cases} 0 & \mathbf{y}_p = \mathbf{y}_c \\ p_r(\mathbf{y}_c) & \text{otherwise} \end{cases} \tag{4.18}$$

This loss function seems to be more appropriate than the 0-1. This is due to the fact that if the system misclassifies some sentences of a given test set, this loss function tries to force the system to fail in the source sentence $\mathbf{x}$ of which correct translation[b] $\mathbf{y}_c$ is one of the least probable in the target language. Thus, the system will fail in the least probable translations, whenever it gets confused; and therefore, the *Global Risk* will be reduced.

The associated Bayes' rule for loss function in Eq. (4.18) is

$$\hat{\boldsymbol{y}}(\mathbf{x}) = \underset{\boldsymbol{y}_p \in \boldsymbol{Y}^*}{\arg\max} \left\{ p_r(\boldsymbol{y}_p|\mathbf{x}) p_r(\boldsymbol{y}_p) \right\} \quad . \tag{4.19}$$

Note that we used $\boldsymbol{y}_p$ instead of $\boldsymbol{y}$ to highlight the fact that $\boldsymbol{y}_p$ plays the role of the proposed translation in the definition of the loss function in Eq. (4.18).

Previous Eq. (4.19) is known as the *direct translation rule (DTR)* since the translation probability, $p_r(\boldsymbol{y}_p|\mathbf{x})$, is defined in an direct way. The direct translation rule was heuristically introduced into the scope of machine translation in order to alleviate the search problem by many of the current SMT systems [Koehn et al., 2003, Och and Ney, 2004a, Och et al., 1999, Zens, 2008]. Note that the DTR was introduced as a heuristic version of the ITR in Eq. (4.17), where $p_r(\mathbf{x}|\mathbf{y})$ is substituted by $p_r(\mathbf{y}|\mathbf{x})$. This rule allows an easier search algorithm for some of the translation models. Although the DTR has been widely used, its statistical theoretical foundation has not been clear for long time, as it seemed to be against the Bayes' classification rule. As stated above, the direct translation rule is the Bayes' optimal classification rule *if the loss function in Eq.* (4.18) *is used* [Andrés-Ferrer et al., 2008].

Since the DTR uses the target language probability as the error function, it should work better than the ITR, from a theoretical point of view. Nevertheless, the statistical models used for approximate the translation probabilities may not be good enough. Thus, the modelling error, which is the error made when approximating the actual probability with a model, could be more important than the advantage obtained from the use of a more appropriate loss function. Therefore, it seems a good idea to use the direct rule in the equivalent inverse form so that the translation system will be the same and then these asymmetries will be reduced. By simply applying the Bayes' theorem to Eq. (4.19), we obtain a equivalent rule

$$\hat{\mathbf{y}} = \underset{\mathbf{y} \in \boldsymbol{Y}^*}{\arg\max} \left\{ p_r(\mathbf{y})^2 p(\mathbf{x}|\mathbf{y}) \right\} \quad . \tag{4.20}$$

Theoretically, rules in Eq. (4.19) and Eq (4.20) are equivalent and must give the same solution. Therefore, the difference between the Eq. (4.19) and Eq (4.20) measure the asymmetries of the translation models as well as the error in the modelling. Bare in mind that the language models also presents some modelling errors and, hence, this last approach assumes that the language model is a very good approximation to the actual probability distribution, due to the fact that the "direct weight" has passed from the direct translation model $p_r(\mathbf{y}|\mathbf{x})$ to the language model. Whether the direct model or the inverse model is better for the translation task depends on the model properties, the estimation technique and the training data.

[b] Here lies the importance of distinguishing between the translation proposed by the system $\mathbf{y}_p$ and the correct translation $\mathbf{y}_c$ for a given source sentence $\mathbf{x}$.

As stated in Section 4.2 a refined loss function is designed using the joint probability as the error function, $\epsilon(\mathbf{x}, \mathbf{y}_j) = p_r(\mathbf{x}, \mathbf{y}_j)$,

$$\mathrm{l}(\mathbf{y}_p|\mathbf{x}, \mathbf{y}_c) = \begin{cases} 0 & \mathbf{y}_p = \mathbf{y}_c \\ p_r(\mathbf{x}, \mathbf{y}_c) & \text{otherwise} \end{cases} \tag{4.21}$$

which leads to

$$\hat{\boldsymbol{y}} = \arg\max_{\mathbf{y} \in \boldsymbol{Y}^*} \{p_r(\mathbf{x}, \mathbf{y})\, p_r(\boldsymbol{y}|\boldsymbol{x})\} \quad . \tag{4.22}$$

Depending on how we model probabilities in Eq. (4.22), several optimal classification rules are obtained. Specially if the joint probability ($p(\mathbf{x}, \mathbf{y})$) is modelled with an inverse translation probability plus a target language probability, then, the *inverse and direct translation rule (I&DTR)*, is obtained

$$\hat{\mathbf{y}} = \arg\max_{\mathbf{y} \in \boldsymbol{Y}^*} \{p_r(\mathbf{y})\, p_r(\mathbf{x} \,|\, \mathbf{y})\, p_r(\mathbf{y} \,|\, \mathbf{x})\} \quad . \tag{4.23}$$

The interpretation of this rule is a refinement of the direct translation rule. In this case, if the system makes a mistake, then it tends to be done in the least probable pairs $(\mathbf{x}, \mathbf{y})$ in terms of $p_r(\mathbf{y}, \mathbf{x})$.

4.3.3 Approximation to general error functions

As stated in Section 4.2, the loss functions of the kind in Eq. (4.12), are usually faster than the general loss functions depicted in Eq. (4.9) since the former scans the possible translations once and the latter twice (one for the minimisation and another for the sum). The loss function in Eq. (4.12) sacrifices the proposed translation in order to speed up the search process. Unfortunately, the automatic evaluation metrics used to quantify the translation systems require both, the proposed and the correct translation. Therefore, with the fastest loss functions we are not able to minimise the evaluation metrics, which in principle is what we expect from our best system. However, by defining a family of simple error functions depending on a parametric vector, say $\boldsymbol{\lambda}$, we are able to approximate the evaluation metric, such as the BLEU or WER.

One way to approximate this general error function is to use a set of features, $f_k(\boldsymbol{x}, \boldsymbol{y}_c)$, that depend on both the source sentence and its correct target translation. Then we define the following error function

$$\epsilon_{\boldsymbol{\lambda}}(\boldsymbol{x}, \boldsymbol{y}_c) = \prod_{k=1}^{K} f_k(\boldsymbol{x}, \boldsymbol{y}_c)^{\lambda_k} \quad . \tag{4.24}$$

If our actual evaluation error function is

$$\epsilon(\boldsymbol{x}, \boldsymbol{y}_p, \boldsymbol{y}_c) = 1 - \mathrm{BLEU}(\boldsymbol{y}_p, \boldsymbol{y}_c) \quad , \tag{4.25}$$

then using a validation set $D = \{(\boldsymbol{x}_n, \boldsymbol{y}_n)\}_{n=1}^{N}$ we can use any optimisation algorithm to minimise our actual error function in Eq. (4.25). For instance, the maximum entropy algorithm [Berger et al., 1996] is typically applied to find the optimal parameter vector $\boldsymbol{\lambda}$.

The error function defined in Eq. (4.24) leads to the following classification rule

$$\hat{\boldsymbol{y}}_{\boldsymbol{\lambda}}(\boldsymbol{x}) = \arg\max_{\boldsymbol{y}_c \in \boldsymbol{Y}^*} \; p_r(\boldsymbol{y}_c|\boldsymbol{x}) \prod_{k=1}^{K} f_k(\boldsymbol{x}, \boldsymbol{y}_c)^{\lambda_k} \quad . \tag{4.26}$$

If we extend the feature vector, $\boldsymbol{f}$, by adding the conditional probability, $p_r(\boldsymbol{y}_c|\boldsymbol{x})$ as a new feature with a new parameter [c] λ_{K+1}; then the classification rule expressed in terms of the extended feature vector,

[c] In the case that there existed a feature, say $f_l(\cdot)$, which already is the conditional probability, then the new feature vector remains the same and the new parameter vector is the previous one but for the component l which is increase by one, i.e. $\bar{\lambda}_l = \lambda_l + 1$.

$\bar{\boldsymbol{f}}$ and the extended parametric vector $\bar{\boldsymbol{\lambda}}$ is

$$\hat{\boldsymbol{y}}_{\bar{\boldsymbol{\lambda}}}(\boldsymbol{x}) = \arg\max_{\boldsymbol{y}_c \in \boldsymbol{Y}^*} \prod_{k=1}^{K} \bar{f}_k(\boldsymbol{x}, \boldsymbol{y}_c)^{\bar{\lambda}_k} \quad . \tag{4.27}$$

If we apply the logarithm to the previous Eq. (4.27) we obtain the equivalent expression

$$\hat{\boldsymbol{y}}_{\bar{\boldsymbol{\lambda}}}(\boldsymbol{x}) = \arg\max_{\boldsymbol{y}_c \in \boldsymbol{Y}^*} \sum_{k=1}^{K} \bar{\lambda}_k \log \bar{f}_k(\boldsymbol{x}, \boldsymbol{y}_c) \quad . \tag{4.28}$$

If we define $\bar{\boldsymbol{h}} = \log \bar{\boldsymbol{f}}$ then the classification rule in Eq. (4.28) is expressed as follows

$$\hat{\boldsymbol{y}}_{\bar{\lambda}}(\boldsymbol{x}) = \arg\max_{\boldsymbol{y}_c \in \boldsymbol{Y}^*} \sum_{k=1}^{K} \bar{\lambda}_k \bar{h}_k(\boldsymbol{x}, \boldsymbol{y}_c) \quad , \tag{4.29}$$

where $\bar{\boldsymbol{\lambda}}$ stands for the extended extended parameter vector as in Eq. (4.27).

Most of the state-of-the-art systems use this idea, although they present it as if they were using a log-linear model [Mariño et al., 2006, Och and Ney, 2004a]. Specifically, if in Eq. (4.8) we model the direct probability as a log linear model

$$p_{\boldsymbol{\lambda}}(\boldsymbol{y}|\boldsymbol{x}) = \frac{1}{\mathrm{Z}_{\boldsymbol{\lambda}}(\boldsymbol{x})} \exp(\sum_{k=1}^{K} \lambda_k h_k(\boldsymbol{x}, \boldsymbol{y})) \quad , \tag{4.30}$$

with

$$\mathrm{Z}_{\boldsymbol{\lambda}}(\boldsymbol{x}) = \sum_{\boldsymbol{y} \in \boldsymbol{Y}^*} \exp(\sum_{k=1}^{K} \lambda_k h_k(\boldsymbol{x}, \boldsymbol{y})) \tag{4.31}$$

then using the model in Eq. (4.30) in the rule in Eq. (4.8), we obtain the following rule

$$\hat{\boldsymbol{y}}_{\boldsymbol{\lambda}}(\boldsymbol{x}) = \arg\max_{\boldsymbol{y} \in \boldsymbol{Y}^*} \frac{1}{\mathrm{Z}_{\boldsymbol{\lambda}}(\boldsymbol{x})} \exp(\sum_{k=1}^{K} \lambda_k h_k(\boldsymbol{x}, \boldsymbol{y})) \tag{4.32}$$

$$= \arg\max_{\boldsymbol{y} \in \boldsymbol{Y}^*} \exp(\sum_{k=1}^{K} \lambda_k h_k(\boldsymbol{x}, \boldsymbol{y})) \tag{4.33}$$

$$= \arg\max_{\boldsymbol{y} \in \boldsymbol{Y}^*} \sum_{k=1}^{K} \lambda_k h_k(\boldsymbol{x}, \boldsymbol{y}) \quad . \tag{4.34}$$

Note that if you compare Eqs. (4.34) and (4.29), they are equivalent.

Although the log-linear explanation of the process yields the same classification rule, it is not satisfactory in the sense that the log-linear model in Eq. (4.30) is never trained in its full form and, its *normalisation weight* $\mathrm{Z}_{\boldsymbol{\lambda}}(\boldsymbol{x})$ *is ignored*. This ellipsis can be done in the decoding process but *cannot be done in training*. In other words, the log-linear model in Eq. (4.30) is only trained in the form of classification rule (4.34) so that it minimises the general loss function in Eq. (4.25) by using the loss function in Eq. (4.12) with the error function in Eq. (4.24). Therefore, the state-of-art log-linear models are a *log-lineal loss function* trained to resemble the general loss function in Eq. (4.25).

Typical features used by the state-of-the-art systems range among [Mariño et al., 2006, Och and Ney, 2004a] the followings:

- *Direct translation models:* a typical feature is to use a direct translation model

$$h_k(\boldsymbol{x}, \boldsymbol{y}) = \log p_{\boldsymbol{\theta}}(\boldsymbol{y}|\boldsymbol{x}), \quad f_k(\boldsymbol{x}, \boldsymbol{y}) = p_{\boldsymbol{\theta}}(\boldsymbol{y}|\boldsymbol{x}) \quad . \tag{4.35}$$

  The most used models are the IBM model 1 and the phrase-based models.

- *Inverse translation models:* a typical feature is to use a inverse translation model

$$h_k(\boldsymbol{x}, \boldsymbol{y}) = \log p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{y}), \quad f_k(\boldsymbol{x}, \boldsymbol{y}) = p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{y}) \quad . \tag{4.36}$$

  The most used models are the IBM model 1 and the phrase-based models.

- *Joint translation models:* a typical feature is to use a stochastic finite transducer,

$$h_k(\boldsymbol{x}, \boldsymbol{y}) = \log p_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{y}), \quad f_k(\boldsymbol{x}, \boldsymbol{y}) = p_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{y}) \quad . \tag{4.37}$$

- *An n-gram language model:* that is to say

$$h_k(\boldsymbol{x}, \boldsymbol{y}) = \log p_{\boldsymbol{\theta}}(\boldsymbol{y}), \quad f_k(\boldsymbol{x}, \boldsymbol{y}) = p_{\boldsymbol{\theta}}(\boldsymbol{y}) \quad . \tag{4.38}$$

- *Word bonus:* it is a well know problem of the n-gram language models that they give more probability to short sentences. Additionally, the translation models tend to distribute the probability among ill-formed sentences as the length of the sentence increases [Brown et al., 1993]. Therefore, in order to keep the translation systems from always producing poor translations because of trying to shorten them, the following feature is used

$$h_k(\boldsymbol{x}, \boldsymbol{y}) = \log \exp(|\boldsymbol{y}|), \quad f_k(\boldsymbol{x}, \boldsymbol{y}) = \exp(|\boldsymbol{y}|) \quad . \tag{4.39}$$

4.3.4 Experiments

The aim of this Section is to show experimentally how the theory stated in this work can be used to improve the performance of a translation system. Therefore, the objective is not to obtain a competitive system, but rather to analyse the previously stated properties in practice.

In order to analyse the theory, we have used two set of experiments. For the former set we used a semi-synthetic corpora and a simple translation model, the IBM model II (see Section 1.3.1). For the latter, two real tasks are used whilst the translation models used were the phrase-based models (see Section 1.3.2). Through both experiments a n-gram language model is used to approximate the language probability distributions, i.e. $p_r(\mathbf{y})$. Specifically, the language model was trained using a 5-gram model obtained with the SRILM toolkit [Stolcke, 2002].

Similarly to Germann et al. [2001], we defined two error measures: *search error* and *model error*. These error measures are inspired on the idea that when a SMT system proposes a wrong translation, it is due to of one of the following reasons: either the suboptimal search algorithm has not been able to find a good translation or the model is not able to make up a good translation, and hence it is impossible to find it. A translation error is a *search error (SE)* if the probability of the proposed translations is less than a reference translation; otherwise it is a *model error*, i.e., the probability of the proposed translations is greater than the reference translation. Although a model error always has more probability than the reference translation, this does not excludes the fact that a much better translation maybe found.

In order to evaluate the translation quality, we used the following well-known automatically computable measures: *word error rate (WER), bilingual evaluation understudy (BLEU), position independent error rate (PER),* and *sentence error rate (SER).*

4.3.5 Corpora

Three different corpora were used for the experiments that were carried out in this chapter: Eutrans-I (Tourist), Europarl and Xerox.

Table 4.1 summarises some of the statistics of the Tourist corpus [Amengual et al., 1996]. The Spanish-English sentence pairs correspond to human-to-human communication situations at the front-desk of a hotel which were semi-automatically produced using a small seed corpus compiled from travel guides booklets.

Table 4.1: Basic statistics of the Spanish-English TOURIST task.

		Spanish	**English**
Training	Sentences	170K	
	Running Words	2.2K	2.2M
	Vocabulary	688	540
	Avg. sentence length	12.9	13.0
Test	Sentences	1K	
	Running Words	12.7K	12.6K
	Perplexity	3.6	2.9

Table 4.2 shows some statistics of the Europarl corpus [Koehn, 2005]. Specifically, this is the version that was used in the shared task of the NAACL 2006 Workshop on SMT [NAACL 2006]. Europarl corpus is extracted from the proceedings of the European Parliament, which are written in the different languages of the European Union. There are different versions of the Europarl corpus depending on the pair of languages that are used. In this work, only the English-Spanish version was used. As can be observed in Table 4.2, the Europarl corpus contains a great number of sentences and large vocabulary sizes. These features are common to other well-known corpora described in the literature.

Table 4.2: Statistics of the Europarl corpus

		Spanish	**English**
Training	Sentences	730K	
	Running Words	15.7M	15.2M
	Vocabulary	102.9K	64.1K
	Avg. sentence length	21.5	20.8
Test	Sentences	3.1K	
	Running Words	91.7K	85.2K
	Perplexity	102	120

Table 4.3 reports some statistics of the Xerox corpus [Atos Origin, Instituto Tecnológico de Informática, RWTH Aachen, RALI Laboratory, Celer Soluciones and Société Gamma and Xerox Research Centre Europe, 2001]. This corpus involves the translation of technical Xerox Manuals from English to Spanish, French and German, and vice-versa. In this work, only the English-Spanish version was used. As can be observed in Table 4.3, the Xerox corpus contains a considerable number of sentences and medium-size vocabularies.

Table 4.3: Statistics of the Xerox corpus

		Spanish	**English**
Training	Sentences	55.7K	
	Running Words	0.75M	0.67M
	Vocabulary	11.0K	8.0K
	Avg. sentence length	13.5	11.9
Test	Sentences	1.1K	
	Running Words	10.1K	8.4K
	Perplexity	35	47

Word Based Translation experiments

In this section, the IBM Model 2 [Brown et al., 1993] is used to approximate the translation probability distributions. Together with the IBM Model 2 [Brown et al., 1993], its corresponding search algorithms are used to carry out the experiments in this Section. This choice was motivated by several reason. Firstly, the simplicity of the translation model allows us to obtain a good estimation of the model parameters. Secondly, there are several models that are initialised using the alignments and dictionaries of the IBM model 2, for instance, the IBM HMM [Och et al., 1999] or the phrase-based models can be initialised by the IBM model 2. Finally, the search problem can be solved exactly using dynamic programming for the case of the direct translation rule depicted in Eq. (4.19).

In order to train the IBM Model 2, we used the standard tool *GIZA++* [Och, 2000]. We re-implemented the algorithm presented in [García-Varea and Casacuberta, 2001] to perform the search process for the ITR. Even though this search algorithm is not optimal, we configured the search parameters in order to minimise the search errors, so that most of the errors should be model errors. In addition, we implemented the corresponding version of this algorithm for the DTR and for the I&DTR. All these algorithms were developed by dynamic programming. For the I&DTR, we implemented two versions of the search: one guided by the direct model (a non-optimal search algorithm, namely I&DTR-D) and the other guided by the inverse translation model (which is also non-optimal but more accurate, namely I&DTR-I).

In order to have an experimentation as close as possible to a theoretical scenario, we selected the Spanish-English TOURIST task (see Section 4.3.5). The parallel corpus consisted of $171,352$ different sentence pairs, where 1K sentences were randomly selected from testing, and the rest (in sets of exponentially increasing sizes: 1K, 2K, 4K, 8K, 16K, 32K, 64K, 128K and 170K sentences pairs) for training. All the figures show the confidence interval at 95%.

Figure 4.2 shows the differences in terms of the WER among all the mentioned forms of the DTR: "IFDTR" (Eq. 4.20), and "DTR" (Eq. 4.19). Since the IBM Model 2 (in its direct version) tries to provide very short translations, we implemented a normalised length version of the DTR. In the figure this normalised version is referred "DTR-N". Note the importance of the model asymmetry in the obtained results. The best results were the ones obtained using the inverse form of the DTR. This behaviour is not surprising, since the only mechanism that the IBM Model 2 has to ensure that all sources words are translated is a length distribution that usually allows the model to ommit the translation of a few words. Anyway, the "DTR" and "DTR-N" performed worse than the ITR (Table 4.4).

Figure 4.3 shows the results achieved with search algorithms base on the most important classification rules. All the I&DTR obtain similar results to the ITR. Nevertheless, the non-optimal search algorithm guided by the direct model ("I&DTR-D") was an order of magnitude faster than the more accurate one ("I&DTR-I") and the "ITR". The inverse form of the DTR ("IFDTR") behaved similarly

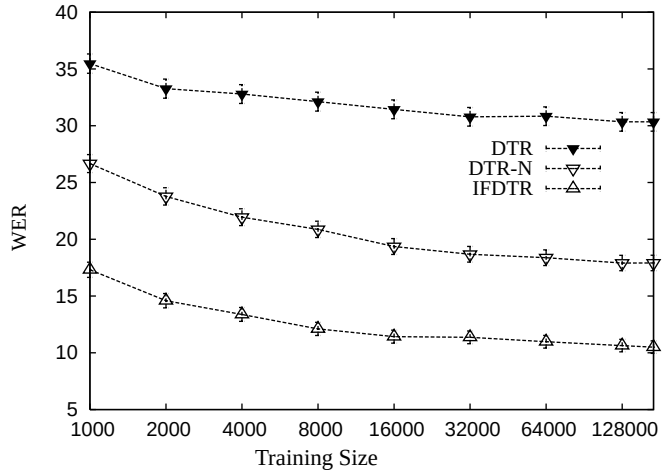


Figure 4.2: Asymmetry of the IBM Model 2 measured with the respect to the WER for the TOURIST test set for different training sizes.

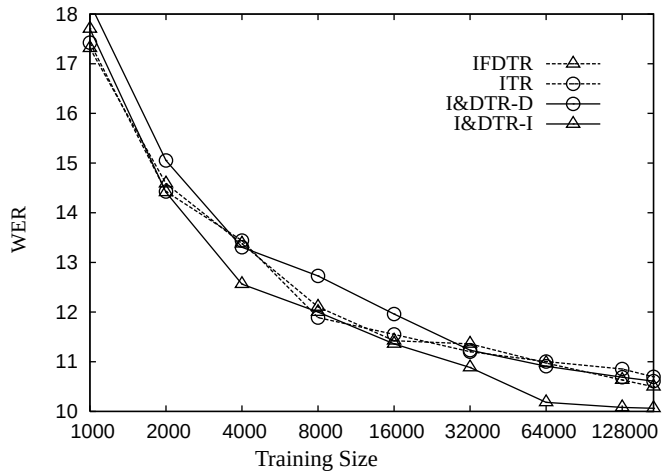


Figure 4.3: WER results for the TOURIST test set for different training sizes and different classification rules.

to these, significantly improving the results reported by DTR. There are no significant differences between the rules analysed in terms of WER. However, the execution times were significantly reduced by the direct guided search in comparison with the other searches. Table 4.4 shows these execution times and the figures with the maximum training size.

The different search algorithms (based on loss functions) do not convey a significant improvement

Table 4.4: Translation quality results with different translation rules for TOURIST test set for a training set of 170K sentences. Where T is the time expressed in seconds and SE stands for the percentage of *search errors*.

Model	WER	SER	BLEU (%)	SE (%)	T
I&DTR I	**10.0**	**49.2**	**84.7**	1.3	34
I&DTR D	10.6	51.6	84.4	9.7	2
IFDTR	10.5	60.0	83.7	2.7	35
ITR	10.7	58.1	84.3	1.9	43
DTR N	17.9	74.1	75.0	0.0	2
DTR	30.3	92.4	53.5	0.0	2

in WER in Figure 4.3. Note that the loss function only evaluates the SER, i.e. the loss function minimises the SER, and does not try to minimise the WER. Thus, changing the loss function, does not necessarily decrease the WER.

In order to check this hypothesis, Figure 4.4 shows the analogous version of Figure 4.3 but with SER instead of WER. It should be noted that as the training size increases, there is a difference in the behaviour between the ITR and both I&DTR. Consequently, the use of these rules provides better SER, and this difference becomes statistically significant as the estimation of the parameters improve. In the case of the inverse form of the DTR ("IFDTR"), as the training size increases, the error tends to decrease and approximate the ITR error. However, the differences are not statistically significant and both methods are equivalent from this point of view.

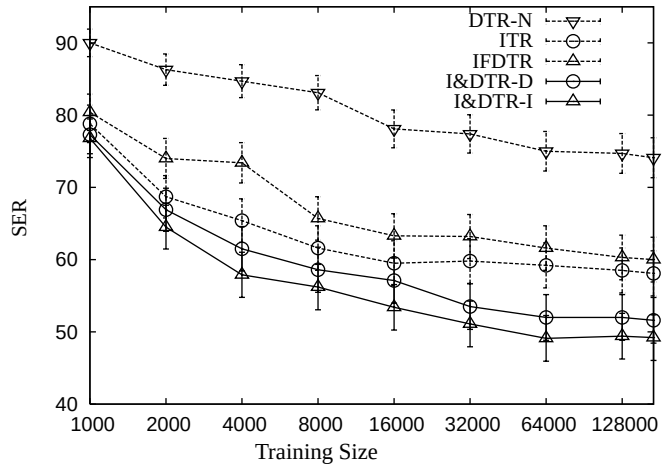


Figure 4.4: SER results for the TOURIST test set for different training sizes and different classification rules.

In conclusion, there are two sets of rules: the first set is made up of IFDTR and ITR, and the second

Table 4.5: The results of translation quality obtained using the proposed variety of loss functions with the Europarl test set.

Spanish → English				
Rule	Formula	BLEU (%)	WER	PER
ITR	$p_r(\mathbf{x}\|\mathbf{y})p_r(\mathbf{y})$	26.8	61.1	45.2
DTR	$p_r(\mathbf{y}\|\mathbf{x})p_r(\mathbf{y})$	20.6	61.1	48.9
I&DTR	$p_r(\mathbf{y}\|\mathbf{x})p_r(\mathbf{x}\|\mathbf{y})p_r(\mathbf{y})$	**28.1**	**59.0**	**43.3**
IFDTR	$p_r(\mathbf{x}\|\mathbf{y})[p_r(\mathbf{y})]^2$	22.2	62.5	48.3
English → Spanish				
Rule	Formula	BLEU	WER	PER
ITR	$p_r(\mathbf{x}\|\mathbf{y})p_r(\mathbf{y})$	25.7	60.7	45.8
DTR	$p_r(\mathbf{y}\|\mathbf{x})p_r(\mathbf{y})$	19.9	62.0	51.3
I&DTR	$p_r(\mathbf{y}\|\mathbf{x})p_r(\mathbf{x}\|\mathbf{y})p_r(\mathbf{y})$	**26.0**	**59.4**	**45.1**
IFDTR	$p_r(\mathbf{x}\|\mathbf{y})[p_r(\mathbf{y})]^2$	21.5	62.7	49.4

is composed by the two versions of the I&DTR. The first set reports worse SER than the the second set. However, the I&DTR guided with the direct model ("I&DTR-D") has many good properties in practice. Note that for real tasks and state-of-the-art systems, it is expected that the behaviour of the rules correspond to the result obtained with the smallest corpus size, where no significant difference exists among the systems in terms of SER.

Phrase-based translation experiments

In the case of phrase-based translation (PBT) different models (for the two tasks considered) were estimated. The training of these models were carried out in the following way:

- First, a word-level alignment of all the sentence pairs in the training corpus was carried out. This alignment was performed for the Spanish-to-English and English-to-Spanish directions, using a standard GIZA++ [Och, 2000] training, with the standard training scheme $1^5 2^5 3^1 4^5$.
- Then, a symmetrisation of both alignment matrices was built, using the THOT toolkit [Ortiz et al., 2005]. Specifically, the refined symmetrisation method was used [Och and Ney, 2004b].
- Finally, a phrase-based model was estimated, using the THOT toolkit [Ortiz et al., 2005].

With respect to the decoding process, we implemented our own phrase-based decoder. Specifically, the decoder implements an $A\star$ algorithm which is very similar to that described in the literature [Germann et al., 2001, Ortiz et al., 2003] for single-word models. The decoder was adapted to deal with the different translation rules (or equivalently, the different loss functions) proposed here. These decoders verbatim the unknown words to the output, since our model is not fine-grained and its basic units are words.

Tables 4.5 and 4.6 show the translation quality measures for the Europarl and Xerox tasks, respectively, for the different loss functions proposed in Section 4.2. The DTR and FIRTD behaves similarly. As expected, the D&ITR obtains the best performance. The differences between the FIRTD and the DTR (which are theoretically equivalent) are not too great, so the under-performance of the DTR compared with the ITR is not due to model asymmetries. If the translations given by the DTR are compared

Table 4.6: Translation quality results obtained, using the proposed variety of loss functions, with the test set of Xerox task.

Spanish → English				
Rule	Rule (Search Alg.)	BLEU (%)	WER	PER
ITR	$p_r(\mathbf{x}\|\mathbf{y})p_r(\mathbf{y})$	61.7	25.9	17.6
DTR	$p_r(\mathbf{y}\|\mathbf{x})p_r(\mathbf{y})$	59.0	27.0	18.9
I&DTR	$p_r(\mathbf{y}\|\mathbf{x})p_r(\mathbf{x}\|\mathbf{y})p_r(\mathbf{y})$	**61.6**	**25.9**	**17.5**
IFDTR	$p_r(\mathbf{x}\|\mathbf{y})[p_r(\mathbf{y})]^2$	60.6	26.2	18.0
English → Spanish				
Rule	Rule (Search Alg.)	BLEU	WER	PER
ITR	$p_r(\mathbf{x}\|\mathbf{y})p_r(\mathbf{y})$	63.6	25.6	18.5
DTR	$p_r(\mathbf{y}\|\mathbf{x})p_r(\mathbf{y})$	62.8	26.0	19.1
I&DTR	$p_r(\mathbf{y}\|\mathbf{x})p_r(\mathbf{x}\|\mathbf{y})p_r(\mathbf{y})$	**64.6**	**25.1**	**18.1**
IFDTR	$p_r(\mathbf{x}\|\mathbf{y})[p_r(\mathbf{y})]^2$	62.8	26.2	19.0

with the ITR, it can be observed that the DTR tends to generate shorter translations. This result is expected since the error function of the DTR, $p_r(\boldsymbol{y})$, is modelled using a n-gram language model, and it is well-known that n-gram language models give more probability to short sentences, that is to say, the resulting systems tends to shorten translations.

Tables 4.5 and 4.6 show that the theoretically expected increase of the translation performance in terms of WER and BLEU, is apparently not achieved for the DTR and both corpora. Although in the Xerox corpus the improved performance for the DTR is achieved, the differences between the systems are not very high. However, figures 4.5 and 4.6 show that, in fact, the DTR rule outperforms the ITR, but also provides shorter translations. Note that the longer the sentences are the worse the *brevity penalty (BP) of the BLEU score* is and consequently the worse the BLEU is (Fig. 4.6). Note that in Fig. 4.5, the DTR incurs in a WER which is in all cases smaller than the WER performed by ITR. Again this is due to the n*-gram* model which is used to model the language model, i.e. the error function of the DTR. The I&DTR had the same brevity penalty problem, however, in this case the problem was not so important since the rule includes the inverse translation model, which counteracts the problem.

Table 4.7 shows some translations obtained using both DTR and ITR. As can be seen, DTR tends to produce shorter translations than ITR, which typically produces more translation errors. For instance, in the first sentence, *the European agency* is translated as *the agency* by the DTR; this is due to the fact that although the first translation is more precise, the language model (the loss function for the DTR) scores the second as a more probable sentence. Oppositely, the DTR correctly translates *must* in the first sentence but the ITR translates it as *should*. Most of the common mistakes shared for both rules are syntactic errors, although semantic errors can be found, as well.

In conclusion, the DTR and I&DTR, obtain better results with short sentences due to a bias in the language model, although the precision of such sentences is better. Nevertheless, the I&DTR is not dramatically affected by an increase in the sentence length. As future work, we intend to solve the language model bias to short sentence in some way, perhaps by introducing a length normalisation in the loss function or in the models.

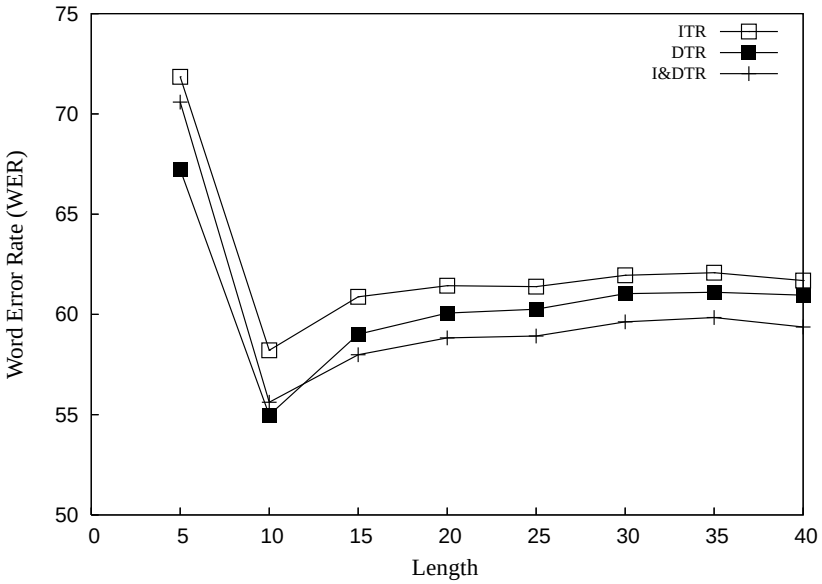


Figure 4.5: The WER results obtained for the Europarl test set (Spanish to English) with the length of the reference sentences restricted to be less than the value of the x-axis.

Table 4.7: Differences between some translation examples obtained using DTR and ITR. Bold words highlight the differences between the two proposed translations. REF stands for the reference translation.

SRC: *en segundo lugar , la agencia europea debe ser completamente independiente .* REF: *secondly , the European agency must be completely independent*
DTR: secondly , the **agency must** be **totally** independent ITR: secondly , the **European agency should** be **completely** independent .
SRC: *es crucial que los consumidores acepten el euro .* REF: *it is crucial for consumers to accept the euro .*
DTR: it is crucial that consumers **to accept** the euro . ITR: it is crucial that consumers **acceptance of** the euro .
SRC: *de modo que me siento reacio a ir más lejos en materia de comercio o a aconsejar medidas suplementarias en materia de comercio o inversión .* REF: *so i am reluctant to go further on trade or to advise further action on trade or investment .*
DTR: i am reluctant to go further trade or advise **further steps** trade or investment . ITR: i am reluctant to go further **on** trade or **to** advise **additional efforts on** trade or investment .

4.4 Conclusions

The analysis of the loss function is an appealing issue. The results of analysing different loss functions range from allowing to use metric loss functions such as BLEU, or WER; to proving the properties

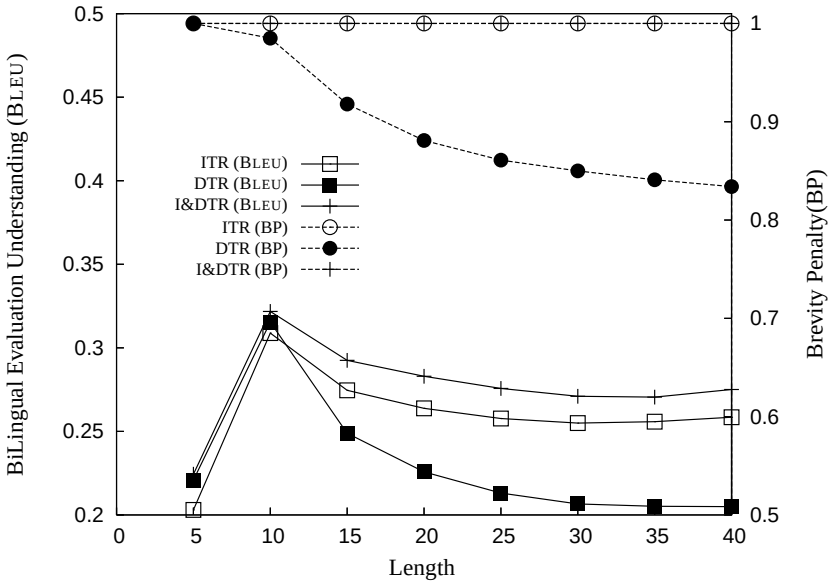


Figure 4.6: The BLEU results (on the left y-scale) and the brevity penalty (BP) of BLEU score (on the right y-scale) obtained for the Europarl test set (Spanish to English) with the length of the reference sentences restricted to be less than the value of the x-axis.

of some outstanding classification rules such as the direct translation rule, the inverse translation rule or even the maximum entropy rule. For each different error function $\epsilon(\mathbf{x}, \mathbf{y}_j, \mathbf{y}_k)$ in the general loss function of Eq. (4.9), there is a different optimal Bayes' rule. The point of using one specific rule is an heuristic and practical issue.

An interesting focus of study is the use of metrics such as BLEU, or WER; as the loss function. Nevertheless due to the high complexity, it is only feasible on constrained situations like n-best lists.

The work developed in this chapter is focused on the study of loss functions that have a linear complexity and that are outstanding due to historical or practical reasons. This work explores the direct translation rule, the inverse translation rule, and the direct and inverse translation rule. In this sense, we have provided a theoretical approach based on decision theory which explains the differences and resemblances between the Direct and the Inverse Translation rules. We have also given insights into the practical differences of these two rules, which are widely used. For instance, this theoretical frame predicts an improvement (in terms of SER), an improvement that has been confirmed in practice for simple words models. In conclusion, according to the experimental results, the DTR outperforms the ITR when short sentences are provided to the system.

The proposed modifications to the 0–1 loss function depicted in Eq. (4.12) can handle the intuitive idea of penalising a wrong action based on the repercussions of the correct action. For instance, if the correct translation, $\boldsymbol{y}_c$, of a source sentence, $\boldsymbol{x}$, is a very unlikely sentence, failure in the translation of such a sentence is not important. Oppositely, failure in the translation of a likely sentence is an important mistake. It is important to note the fact that the proposed loss functions cannot handle significant cases. For example, it is not the same to make an incorrect translation due to grammar errors than to make

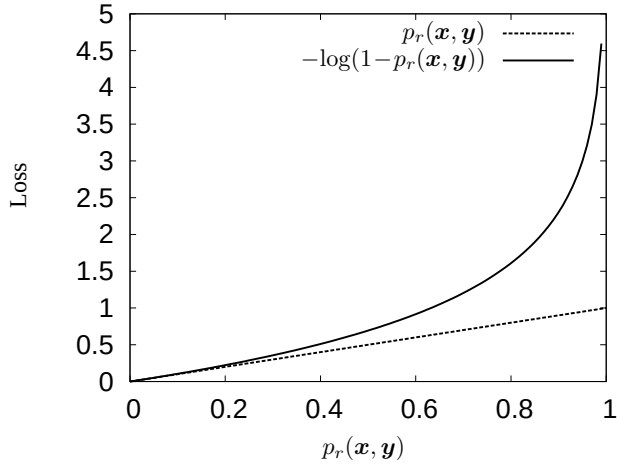


Figure 4.7: Difference between the remaining information and the probability as error functions.

an incorrect translation due to semantic errors. In order to take into account such cases, it is necessary to work with general loss functions of the sort in Eq. (4.9) despite of its cost. However, the idea of penalising the mistakes proportionally to the probability of the correct translation can also be used in case of dealing with more complicated decision rules and, eventually, with more complicated search algorithms.

Note that though we have focused our analysis to error functions which are a probability distribution, the error function $\epsilon(\cdot)$ does not necessary have to be a probability distribution. This idea brings up the question of which the best loss function is. For instance, a confidence measure could even be used to define error functions. Maybe the growing of the loss function should better be non-lineal with the probability. In this sense more interesting loss functions could be obtained using information theory. For instance, we can penalise the system with the *remaining information*. That is, if we know $p_r(\boldsymbol{x},\boldsymbol{y})$, then the information associated with a target sentence $\mathbf{y}_c$ is $-\log(p_r(\boldsymbol{x},\boldsymbol{y}_c))$. The remaining information, or the information that the system has learnt when it fails is given by

$$-\log(\sum_{(\boldsymbol{x}',\boldsymbol{y}')\neq(\boldsymbol{x},\boldsymbol{y}_c)} p_r(\boldsymbol{x}',\boldsymbol{y}')) = -\log(1-p_r(\boldsymbol{x},\mathbf{y}_c)) \quad ,$$

leading to the the error function

$$\epsilon(\boldsymbol{x},\boldsymbol{y}_c) = -\log(1-p(\boldsymbol{x},\boldsymbol{y}_c)) \quad . \tag{4.40}$$

Figure 4.7, shows the remaining information of a probability function. Note that the remaining information has a singularity at 1, i.e. if the system has not been able to learn a sure event, which has probability of 1, then the loss is infinity. Note that this loss can be defined for any probability such as $p_r(\boldsymbol{y})$ or $p_r(\boldsymbol{x},\boldsymbol{y})$.

Another very interesting research line is derived from approximating complex loss functions in Eq. (4.9) with simple loss functions in Eq. (4.12). Although, many of the state-of-art SMT systems

indirectly make use of this idea, as analysed in Section 4.2 (page 90), this idea may be exploited from the point of view presented in this chapter.

Bibliography

Y. Al-Onaizan et al. Statistical Machine Translation: Final Report. Technical report, Johns Hopkins University 1999 Summer Workshop on Language Engineering, Center for Language and Speech Processing, Baltimore, MD, USA, 1999.

J.C. Amengual, J.M. Benedí, M.A. CastaÃśo, A. Marzal, F. Prat, E. Vidal, J.M. Vilar, C. Delogu, A. di Carlo, H. Ney, and S. Vogel. Definition of a machine translation task and generation of corpora. Technical report d4, Instituto Tecnológico de Informática, September 1996. ESPRIT, EuTrans IT-LTR-OS-20268.

J. Andrés-Ferrer, D. Ortiz-Martínez, I. García-Varea, and F. Casacuberta. On the use of different loss functions in statistical pattern recognition applied to machine translation. *Pattern Recognition Letters*, 29(8):1072–1181, 2008.

Atos Origin, Instituto Tecnológico de Informática, RWTH Aachen, RALI Laboratory, Celer Soluciones and Société Gamma and Xerox Research Centre Europe. TransType2 - Computer Assisted Translation. Project Technical Annex., 2001.

Adam L. Berger, Vincent J. Della Pietra, and Stephen A. Della Pietra. A maximum entropy approach to natural language processing. *Comput. Linguist.*, 22 (1):39–71, 1996. ISSN 0891-2017.

Peter F. Brown, John Cocke, Stephen Della Pietra, Vincent J. Della Pietra, Frederick Jelinek, John D. Lafferty, Robert L. Mercer, and Paul S. Rossin. A statistical approach to machine translation. *Computational Linguistics*, 16(2):79–85, 1990.

Peter F. Brown, Stephen Della Pietra, Vincent J. Della Pietra, and Robert L. Mercer. The mathematics of statistical machine translation: Parameter estimation. *Computational Linguistics*, 19(2):263–312, 1993.

Richard O. Duda, Peter E. Hart, and David G. Stork. *Pattern Classification*. John Wiley and Sons, New York, NY, 2nd edition, 2001.

I. García-Varea and F. Casacuberta. Search algorithms for statistical machine translation based on dynamic programming and pruning techniques. In *Proc. of MT Summit VIII*, pages 115–120, Santiago de Compostela, Spain, 2001.

U. Germann et al. Fast decoding and optimal decoding for machine translation. In *Proc. of ACL'01*, pages 228–235, Morristown, NJ, USA, June 2001. Association for Computational Linguistics.

F. Jelinek. A fast sequential decoding algorithm using a stack. *IBM Journal of Research and Development*, 13:675–685, 1969.

Kevin Knight. Decoding complexity in word-replacement translation models. *Computational Linguistics*, 25(4):607–615, 1999.

P. Koehn. Europarl: A parallel corpus for statistical machine translation. In *Proc. of the MT Summit X*, pages 79–86, September 2005.

P. Koehn, F.J. Och, and D. Marcu. Statistical phrase-based translation. In *Proc. of NAACL'03*, pages 48–54, Morristown, NJ, USA, 2003. Association for Computational Linguistics. doi: http://dx.doi.org/10.3115/1073445.1073462.

S. Kumar and W. Byrne. Minimum bayes-risk decoding for statistical machine translation, 2004.

J. B. Mariño et al. N-gram-based machine translation. *Computational Linguistics*, 32(4):527–549, 2006. ISSN 0891-2017.

NAACL 2006. `http://nlp.cs.nyu.edu/hlt-naacl06/`.

F. J. Och. GIZA++: Training of statistical translation models, 2000. `http://www-i6.informatik.rwth-aachen.de/\~och/software/GIZA++.html`.

F. J. Och and H. Ney. The alignment template approach to statistical machine translation. *Computational Linguistics*, 30(4):417–449, 2004a. ISSN 0891-2017.

F. J. Och, Christoph Tillmann, and Hermann Ney. Improved alignment models for statistical machine translation. In *Proc. of the Joint SIGDAT Conf. on Empirical Methods in Natural Language Processing and Very Large Corpora*, pages 20–28, University of Maryland, College Park, MD, June 1999.

F.J. Och and H. Ney. The alignment template approach to statistical machine translation. *Computational Linguistics*, 30(4):417–449, 2004b.

D. Ortiz, Ismael García-Varea, and Francisco Casacuberta. An empirical comparison of stack-based decoding algorithms for statistical machine translation. In *New Advance in Computer Vision*, Lecture Notes in Computer Science. Springer-Verlag, 2003. 1st Iberian Conference on Pattern Recongnition and Image Analysis -IbPRIA2003- Mallorca. Spain. June.

D. Ortiz, I. García-Varea, and F. Casacuberta. Thot: a toolkit to train phrase-based statistical translation models. In *Tenth Machine Translation Summit*, pages 141–148, Phuket, Thailand, September 2005.

V. Steinbiss R. Schlüter, T. Scharrenbach and H. Ney. Bayes risk minimization using metric loss functions. In *Proceedings of the European Conference on Speech Communication and Technology, Interspeech*, pages 1449–1452, Lisbon, Portugal, September 2005.

A. Stolcke. SRILM – an extensible language modeling toolkit. In *Proc. of ICSLP'02*, pages 901–904, September 2002.

C. Tillmann and H. Ney. Word reordering and a dynamic programming beam search algorithm for statistical machine translation. *Computational Linguistics*, 29(1):97–133, March 2003.

Raghavendra Udupa and Hemanta K. Maji. Computational complexity of statistical machine translation. In *Proceedings of the Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, pages 25–32. Trento, Italy, 2006.

N. Ueffing and H. Ney. Bayes decision rules and confidence measures for statistical machine translation. In *EsTAL - Espa for Natural Language Processing*, pages 70–81, Alicante, Spain, October 2004. Springer Verlag, LNCS.

Y. Wang and A. Waibel. Decoding algorithm in statistical translation. In *Proc. of ACL'97*, pages 366–372, Morristown, NJ, USA, July 1997. Morgan Kaufmann / Association for Computational Linguistics.

R. Zens. *Phrase-based Statistical Machine Translation: Models, Search, Training*. PhD thesis, RWTH Aachen University, Aachen, Germany, February 2008.

Chapter 5

Statistical stochastic finite state transducers

" *An expert is someone who knows some of the worst mistakes that can be made in his subject, and how to avoid them* " W. HEISENBERG

Contents

5.1 Introduction

As stated in Section 1.3 Chapter 1, the machine translation process is a classification problem. Therefore, multiplying the maximisation Eq. (1.9) Chapter 1 by $\mathrm{p}_r(\boldsymbol{x})$, which is a constant for the maximisation, the optimal translation is the one that maximises the following equation

$$\hat{\boldsymbol{y}}(x) = \arg\max_{\boldsymbol{y}} \{p_r(\boldsymbol{y}, \boldsymbol{x})\} \tag{5.1}$$

where the joint probability $p_r(\boldsymbol{y}, \boldsymbol{x})$ can be modelled by a *stochastic finite state transducer (SFST)*. Note that Eq. (5.1) is equal that Eq. (1.11) but instantiated to the MT notation.

SFSTs constitute an important family of translation models within the theory of formal languages [Vidal et al., 2005a]. Even though these models are much more limited than other more powerful ones, the computational costs of the algorithms that are needed to deal with them are much lower. SFSTs also permit a simple integration with other information sources, which makes it easy to apply SFSTs to more difficult tasks such as speech translation [Casacuberta et al., 2004]. SFSTs and the corresponding training and search techniques have been studied by several authors, in many cases explicitly motivated by MT applications [E. Vidal and Segarra, 1989, Oncina et al., 1993, Knight and Al-Onaizan, 1998, Mäkinen, 1999, Amengual et al., 2000, Alshawi et al., 2000a, Casacuberta, 2000a, Vilar, 2000, Vogel and Ney, 2000, Picó and Casacuberta, 2001, Bangalore and Riccardi, 2003, Kumar and Byrne, 2003, Casacuberta and Vidal, 2004, Tsukada and Nagata, 2004, Casacuberta et al., 2005, Kumar et al., 2006, Casacuberta and Vidal, 2007, Mariòo et al., 2006]. There are other statistical models for MT that are based on alignments between words (*statistical word-alignment models*) [Brown et al., 1993] or between word sequences (*phrase-based models or alignment templates*) [Och and Ney, 2004, Zens, 2008]. Some of these models (*monotone phrase-based models* [Zens, 2008]) are closely related to SFST. Other translation models, which can be considered generalisations of SFSTs, are the *inversion transduction grammars* [Wu, 1995] and the *head transducers* [Alshawi et al., 2000b]. These models are theoretically more powerful than SFSTs, but in general, they require higher computational costs.

The GIATI technique [Casacuberta and Vidal, 2007] has been applied to machine translation [Casacuberta and Vidal, 2004], speech translation [Casacuberta et al., 2004] and computed-assisted translation [Barrachina et al.]. The results obtained using GIATI suggest that, among all the SFST learning techniques tested, GIATI is the only one that can cope with translation tasks under real conditions of vocabulary sizes and amounts of training data available. However, as the task complexity increases, GIATI tends to fall behind other approaches that more explicitly rely on statistics [Casacuberta and Vidal, 2007, Mariòo et al., 2006].

5.2 Stochastic finite-state transducers (SFST)

Stochastic finite-state transducers (SFST) are similar to stochastic finite-state grammars or automata [Vidal et al., 2005b], but in this case two different alphabets are involved: source (input) and target (output) alphabets. Each transition in a SFST has attached a source word[a] and a (possible empty) string of target words [Vidal et al., 2005a].

Definition 1 *A SFST $\mathcal{T}$ is defined as a tuple $\{\boldsymbol{X}, \boldsymbol{Y}, Q, q_0, t, f\}$, where $\boldsymbol{X}$ is a finite set of source words; $\boldsymbol{Y}$ is a finite set of target words; Q is a finite set of states; $q_0 \in Q$ is the initial state; $p : Q \times \boldsymbol{X}^\star \times \boldsymbol{Y}^\star \times Q \to [0,1]$ is a transition probability function and $f : Q \to [0,1]$ is a final-state probability function. The functions t and f must verify:*

$$\forall q \in Q, \quad f(q) \quad + \sum_{(\bar{\boldsymbol{x}}, \bar{\boldsymbol{y}}, q') \in \boldsymbol{X} \times \boldsymbol{Y}^\star \times Q} t(q, \bar{\boldsymbol{x}}, \bar{\boldsymbol{y}}, q') = 1 \quad . \tag{5.2}$$

[a] The term "word" is used to refer a single token as in MT i.e. a "symbol" in formal language theory.

The non-probabilistic counterpart of a given a SFST $\mathcal{T}$, called *characteristic finite-state transducer of* $\mathcal{T}$ (FST), can be defined. The *transitions* are those tuples in $Q \times \boldsymbol{X}^\star \times \boldsymbol{Y}^\star \times Q$ with probability greater than zero and the *set of final states* are those states in Q with final-state probability greater than zero.

Given $\mathcal{T}$, a *translation form* with J (the number of words or symbols in the source sentence) transitions associated with the *translation pair* $(\boldsymbol{x}, \boldsymbol{y}) \in \boldsymbol{X}^* \times \boldsymbol{Y}^*$ is a sequence of transitions $\phi = (q_0, \bar{\boldsymbol{x}}_1, \bar{\boldsymbol{y}}_1, q_1)\ (q_1, \bar{\boldsymbol{x}}_2, \bar{\boldsymbol{y}}_2, q_2)\ (q_2, \bar{\boldsymbol{x}}_3, \bar{\boldsymbol{y}}_3, q_3) \dots\ (q_{J-1}, \bar{\boldsymbol{x}}_J, \bar{\boldsymbol{y}}_J, q_J)$, such that $\bar{\boldsymbol{x}}_1 \bar{\boldsymbol{x}}_2 \dots \bar{\boldsymbol{x}}_J = \boldsymbol{x}$ and $\bar{\boldsymbol{y}}_1 \bar{\boldsymbol{y}}_2 \dots \bar{\boldsymbol{y}}_J = \boldsymbol{y}$. Its probability is the product of the corresponding transition probabilities, times the final-state probability of the last state in the sequence, that is to say,

$$p_{\mathcal{T}}(\phi) = \prod_{j=1}^{J} t(q_{j-1}, \bar{\boldsymbol{x}}_j, \bar{\boldsymbol{y}}_j, q_j)\, f(q_J) \quad . \tag{5.3}$$

The set of translation forms associated with a translation pair $(\boldsymbol{x}, \boldsymbol{y})$ with probability higher than zero is denoted as $\Phi(\boldsymbol{x}, \boldsymbol{y})$.

The *probability of a translation pair* $(\boldsymbol{x}, \boldsymbol{y})$ according to $\mathcal{T}$ is then defined as the sum of the probabilities of all the translation forms associated with $(\boldsymbol{x}, \boldsymbol{y})$, i.e.,

$$p_{\mathcal{T}}(\boldsymbol{x}, \boldsymbol{y}) = \sum_{\forall \phi \in \Phi(\boldsymbol{x}, \boldsymbol{y})} p_{\mathcal{T}}(\phi) \quad . \tag{5.4}$$

If $\mathcal{T}$ has no useless states [Vidal et al., 2005a], $p_{\mathcal{T}}(\boldsymbol{x}, \boldsymbol{y})$ describes a probability distribution on $\boldsymbol{X}^\star \times \boldsymbol{Y}^\star$ which is called *stochastic finite-state translation*. Recall that this distribution is used to model the joint probability introduced in Eq. (5.1). The terms *regular* or, more properly, *rational* translations are also often used in the scientific literature to refer to (the non-probabilistic counterpart of) these mappings [Berstel, 1979].

A SFST has two embedded stochastic regular languages, one for the source alphabet and another for the target alphabet. These languages correspond to the two marginals (p^i and p^o) of the joint distribution modelled by the SFST as follows

$$p^i_{\mathcal{T}}(\boldsymbol{x}) = \sum_{\boldsymbol{y} \in \boldsymbol{Y}^\star} p_r(\boldsymbol{x}, \boldsymbol{y}), \quad p^o_{\mathcal{T}}(\boldsymbol{y}) = \sum_{\boldsymbol{x} \in \boldsymbol{X}^\star} p(\boldsymbol{x}, \boldsymbol{y}) \quad . \tag{5.5}$$

In practice, the corresponding source or target finite-state grammars are obtained from the finite-state transducer by dropping the target or source words of each transition, respectively.

SFSTs exhibit properties and problems similar to those exhibited by stochastic regular languages. One of these properties is the formal basis of the GIATI technique for transducer inference. It can be stated as the following theorem [Casacuberta et al., 2005]: *Every stochastic finite-state translation can be obtained from a stochastic regular language and two morphisms*. This is a weaker version of the stochastic extension of a classical morphism theorem [Berstel, 1979]: *Every rational translation can be obtained from a local language and two alphabetic morphisms*, where a *local language* is defined by a set of permitted two-word segments (and therefore a *stochastic local language* is equivalent to a *bigram distribution* [Vidal et al., 2005a]). In both cases, the morphisms allow us to build the components of a pair of the finite-state translation from a string of the corresponding local language [Casacuberta et al., 2005].

A SFST $\mathcal{T}$ can be used to approximate the joint probability in Eq. (5.1), $p_r(\boldsymbol{x}, \boldsymbol{y})$, obtaining

$$\hat{\boldsymbol{y}} = \arg\max_{\boldsymbol{y} \in \boldsymbol{Y}^\star} p_{\mathcal{T}}(\boldsymbol{x}, \boldsymbol{y}). \tag{5.6}$$

That is, given $\mathcal{T}$ and $\boldsymbol{x} \in \boldsymbol{X}^*$, search for a target string $\hat{\boldsymbol{y}}$ which maximises $p_{\mathcal{T}}(\boldsymbol{x}, \boldsymbol{y})$.

While this *SFST stochastic translation problem* is proved to be **NP**-Hard [Casacuberta, 2000b], a generally good approximation can be obtained in polynomial time through a simple extension of the Viterbi algorithm [Picó and Casacuberta, 2001]. This approximation consists in replacing the sum operator in Eq. (5.4) with the maximum operator as follows

$$p_{\mathcal{T}}(\boldsymbol{x},\boldsymbol{y}) \approx \widehat{p}_{\mathcal{T}}(\boldsymbol{x},\boldsymbol{y}) = \max_{\forall\phi\in\Phi(\boldsymbol{x},\boldsymbol{y})} p_{\mathcal{T}}(\phi). \tag{5.7}$$

This approximation to the SFST stochastic translation problem permits the computation of the optimal translation form (with respect to Eq. (5.7)) in linear time with the number of source words. The translation of the given source sentence is then approached as the sequence of target strings which appear in this optimal translation form.

5.2.1 Grammatical inference and alignments for transducer inference (GIATI)

The morphism theorems stated in Section 5.2 suggest a technique [Casacuberta et al., 2005] to infer an SFSTs, the so-called *grammatical inferences and alignments for transducer inference* (GIATI) approach [Casacuberta, 2000b]. Therefore, the GIATI technique has a strong and solid theoretical foundation. This technique has been applied to machine translation [Casacuberta and Vidal, 2007, Mariòo et al., 2006], speech translation [Casacuberta et al., 2004] and computed-assisted translation [Barrachina et al.]. The results obtained using GIATI suggest that, among all the SFST learning techniques tested, GIATI is the only one that can cope with translation tasks under real conditions of vocabulary sizes and amounts of training data available. However, as the task complexity increases, GIATI tends to fall behind other approaches that more explicitly rely on statistics.

Given a finite sample of pairs $D = \{(\boldsymbol{x}_n,\boldsymbol{y}_n)\}_{n=1}^{N}$ of string pairs from $\boldsymbol{X}^\star \times \boldsymbol{Y}^\star$ (*a parallel corpus*), the GIATI approach works as follows,

1. Each training pair $(\boldsymbol{x}_n,\boldsymbol{y}_n)$ from D is transformed into a string $\boldsymbol{z}_n$ from an *extended alphabet* Γ to obtain a sample D' of strings ($D' \subset \Gamma^\star$).
2. A stochastic finite-state *grammar* [Vidal et al., 2005b], G, is inferred from D'.
3. The symbols (from Γ) of the *grammar transitions* are transformed into input/output symbols ($\boldsymbol{X}^\star \times \boldsymbol{Y}^\star$).

The main problem of this procedure is to define the set of the extended symbols. The transformation of the training pairs must capture the correspondences between words of the input and the output sentences and must permit the implementation of the inverse transformation of the third step. This is achieved with the help of bilingual segmentations [Casacuberta and Vidal, 2004].

In order to illustrate this first step, we will use the Spanish-English pair ("*una habitación doble*" , "*a double room*"). A suitable word-alignment would align "*una*" with "*a*", "*habitación*" with "*room*" and "*doble*" with "*double*". From this alignment, a possible string could be "*(una,a) (habitación,room) (doble,double)*". However, this would imply a reordering of the words "*double*" and "*room*", that is difficult to model in the finite-state framework. The key idea is to avoid a reordering, for example, the alignment can be used to produce a left-to-right bilingual segmentation into two segments: ("*una*" , "*a*") and ("*habitación doble*" , "*double room*"). This segmentation directly yields the single string and the corresponding extended alphabet required by GIATI. In the first version of GIATI, empty target segments were allowed, in this case, a simpler segmentation which has proved equivalently adequate in practice is: "*(una,a) (habitación,-) (doble,double room)*".

One of the shortcomings of GIATI comes from the fact that heuristically it needs "*external*" statistical techniques to preprocess the training pairs. Actually, the bilingual segmentation of first step in the

original GIATI technique was based on statistical alignment models [Brown et al., 1992]. The probabilities associated with the transitions of a SFST learnt in this way are just those of the corresponding stochastic finite-state grammar inferred in *step 2*. Therefore, an interesting feature of GIATI was that it can readily make use of all the smoothing techniques known for n-grams language models (see Section 1.2 Chapter 1) and for stochastic finite-state grammars [Llorens et al., 2002]. Note, however, that GIATI extended alphabets are typically very large and this fact hardens the data-sparseness problems.

Clearly, for a given translation pair, there are many possible bilingual segmentations; but the original version of GIATI did not take advantage of this fact, thereby making less profit from the (always scarce) training data. In the next section, a new GIATI version is introduced [Andrés-Ferrer et al., 2008]. This version is more explicitly based on statistical estimation procedures and would not suffer from this shortcoming. These procedures require an initialisation that can be random or based on the above segmentation obtained using statistical alignment models. Another interesting feature of the new GIATI version is that the *step 2* is embedded in the estimation procedure itself.

5.3 Statistical GIATI (SGIATI)

Our new, statistical version of GIATI, SGIATI [Andrés-Ferrer et al., 2008], is based on a rather simple probabilistic model for segment-based (phrase-based) statistical machine translation. Given a translation pair, $(\boldsymbol{x}, \boldsymbol{y})$, we assume that both sentences can be segmented into a certain number of segments, say C, which are monotonically aligned one-to-one, to produce the desired segment-based translation of $\boldsymbol{x}$ into $\boldsymbol{y}$. This is illustrated in the example shown in Figure 5.1, where three possible segmentations of a given pair are considered. Note that we use $\boldsymbol{j}$ and $\boldsymbol{i}$ to define the precise limits of the segments in $\boldsymbol{x}$ and $\boldsymbol{y}$, respectively.

por favor	,	súbanos	nuestros bultos	a	la	habitación	.
please	,	send up	our luggage	to	the	room	.

por favor ,	súbanos nuestros bultos	a la	habitación .
please ,	send up our luggage	to the	room .

por favor , súbanos nuestros bultos	a la	habitación .
please ,	send up our	luggage to the room .

Figure 5.1: Three possible segmentations of the translation pair "`por favor , súbanos nuestros bultos a la habitación .`" and "`please , send up our luggage to the room .`". The used segmentations are: $\boldsymbol{j} = (0, 2, 3, 4, 6, 7, 8, 9, 10)$ and $\boldsymbol{i} = (0, 1, 2, 4, 6, 7, 8, 9, 10)$ for the first segmentation; $\boldsymbol{j} = (0, 3, 6, 8, 10)$ and $\boldsymbol{i} = (0, 2, 6, 8, 10)$ for the second; and $\boldsymbol{j} = (0, 6, 8, 10)$ and $\boldsymbol{i} = (0, 2, 5, 10)$ for the third.

Uncovering the *hidden* random variables for the number of segments, C, and those for the segmentations of $\boldsymbol{x}$ and $\boldsymbol{y}$, $\boldsymbol{j}$ and $\boldsymbol{i}$ respectively; the probability of observing a given translation pair, $p_r(\boldsymbol{x}, \boldsymbol{y})$, is written as follows

$$p_r(\boldsymbol{x}, \boldsymbol{y}) = \sum_{C} \sum_{\boldsymbol{j}, \boldsymbol{i}} p_r(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{j}, \boldsymbol{i}, C) \quad , \tag{5.8}$$

where $\boldsymbol{j}$ and $\boldsymbol{i}$ range over the set of all possible segmentations of $\boldsymbol{x}$ and $\boldsymbol{y}$

$$\boldsymbol{j} = (j_0, j_1, j_2, \ldots, j_C), \quad j_l < j_{l+1} \text{ with } l \in \{1, \ldots, C-1\} \text{ and } j_C = J,\ j_0 = 0 \tag{5.9}$$

$$\boldsymbol{i} = (i_0, i_1, i_2, \ldots, i_C), \quad i_l < i_{l+1} \text{ with } l \in \{1, \ldots, C-1\} \text{ and } i_C = I,\ i_0 = 0 \tag{5.10}$$

with $J = |\boldsymbol{x}|$ and $I = |\boldsymbol{y}|$. For brevity, we use $\boldsymbol{j}_k^l$ to denote the sub-vector of $\boldsymbol{j}$ from position k to l; i.e., $\boldsymbol{j}_k^l = (j_k, j_{k+1}, \ldots, j_{l-1}, j_l)$.

Our probabilistic model for $(\boldsymbol{x}, \boldsymbol{y})$, *completed* with $\boldsymbol{j}$, $\boldsymbol{i}$ and C, is decomposed left-to-right as follows

$$p_r(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{j}, \boldsymbol{i}, C) = \prod_{c=1}^{C} p_r(\boldsymbol{x}(c), \boldsymbol{y}(c), j_c, i_c \,|\, H(c-1)) p_r(\$, \$, C \,|\, H(C)) \quad , \tag{5.11}$$

where we have used the notation $\boldsymbol{x}(c)$ for c-th segment of $\boldsymbol{x}$, i.e., $\boldsymbol{x}_{j_{c-1}+1}^{j_c}$; $\boldsymbol{y}(c)$ is similarly used for $\boldsymbol{y}$; and $H(c-1)$ denotes the history of $c-1$ previous segments,

$$H(c-1) = \{\boldsymbol{x}(\boldsymbol{j}_0^{c-1}), \boldsymbol{y}(\boldsymbol{i}_0^{c-1}), \boldsymbol{j}_1^{c-1}, \boldsymbol{i}_1^{c-1}\} \quad , \tag{5.12}$$

where $\boldsymbol{x}(\boldsymbol{j}_0^{c-1})$ stands for the $c-1$ segments $\boldsymbol{x}_{j_0+1}^{j_1}, \boldsymbol{x}_{j_1+1}^{j_2}, \ldots, \boldsymbol{x}_{j_{c-2}+1}^{j_{c-1}}$ and similarly does $\boldsymbol{y}(\boldsymbol{i}_0^{c-1})$; and where $H(0)$ is defined as the sure event and, hence, $p_r(\boldsymbol{x}(1), \boldsymbol{y}(1), j_1, i_1 \,|\, H(0))$ is equal to $p_r(\boldsymbol{x}(1), \boldsymbol{y}(1), j_1, i_1)$; and, finally, the length distribution $p_r(\$, \$, C \,|\, H(C)) = 0$ if C differs from the length of $\boldsymbol{j}$ or $\boldsymbol{i}$.

Note that $H(c-1)$ can be approximated with the n more recent segmentations, similarly to n-gram language modelling. For simplicity, the probability of the current segment given the history of the $c-1$ previous segments is approximated using a first-order Markovian assumption ($n = 2$). That is to say,

$$H(c-1) \approx H^2(c-1) = \{(\boldsymbol{x}_{j_{c-2}+1}^{j_{c-1}}), (\boldsymbol{y}_{i_{c-2}+1}^{i_{c-1}}), j_{c-2}, j_{c-1}, i_{c-2}, i_{c-1}\} \quad . \tag{5.13}$$

In this way, our complete probabilistic model in Eq. (5.11) is approximated as follows

$$p_r(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{j}, \boldsymbol{i}, C) := \prod_{c=1}^{C} p_r(\boldsymbol{x}(c), \boldsymbol{y}(c), j_c, i_c \,|\, H^2(c-1)) p_r(\$, \$ \,|\, H^2(C)) \quad . \tag{5.14}$$

The model in Eq. (5.14) is still difficult to learn due to the inclusion of absolute segment boundaries while computing the probability of the current segment. Instead, to ease parameter estimation, each absolute boundary is rewritten relative to its previous boundary,

$$p_r(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{j}, \boldsymbol{i}, C) := \prod_{c=1}^{C} p_r(\boldsymbol{x}(c), \boldsymbol{y}(c), j_c - j_{c-1}, i_c - i_{c-1} \,|\, H^2(c-1)) p_r(\$, \$ \,|\, H^2(C)) \tag{5.15}$$

$$:= \prod_{c=1}^{C} \mathrm{p}(\boldsymbol{x}(c), \boldsymbol{y}(c) \,|\, H^2(c-1)) \, \mathrm{p}(\$, \$ \,|\, H^2(C)) \quad , \tag{5.16}$$

where, in Eq. (5.16), we assume that segment probabilities are null when the relative boundaries $j_c - j_{c-1}$ and $i_c - i_{c-1}$ disagree with their corresponding segment lengths; and where from Eq. (5.15) to Eq. (5.16) we have changed the probability distributions $p_r(\cdots)$ by the model parameters $\mathrm{p}(\cdots)$. Therefore, our final model is parametrised with the following parameter set:

$$\Theta = \{\mathrm{p}(\boldsymbol{u}, \boldsymbol{v}), \mathrm{p}(\boldsymbol{u}, \boldsymbol{v} \,|\, \boldsymbol{u}', \boldsymbol{v}') \,|\, \forall \boldsymbol{u}, \boldsymbol{u}' \in \mathbf{X}^*, \boldsymbol{v}, \boldsymbol{v}' \in \mathbf{Y}^*\} \quad , \tag{5.17}$$

where all $\boldsymbol{u}', \boldsymbol{v}'$ verify the following normalisation property

$$\sum_{\boldsymbol{u} \in \mathbf{X}^*, \boldsymbol{v} \in \mathbf{Y}^*} \mathrm{p}(\boldsymbol{u}, \boldsymbol{v} \,|\, \boldsymbol{u}', \boldsymbol{v}') = 1 \quad . \tag{5.18}$$

For clarity shake, consider as an example, $\boldsymbol{x} = x_1x_2x_3$ and $\boldsymbol{y} = y_1y_2$. Using Eq. (5.8), the joint probability of observing $\boldsymbol{x}$ and $\boldsymbol{y}$ can be written as follows

$$p_r(\boldsymbol{x},\boldsymbol{y}) = p_r(\boldsymbol{x},\boldsymbol{y},(0,3),(0,2),1) + p_r(\boldsymbol{x},\boldsymbol{y},(0,1,3),(0,1,2),2) + p_r(\boldsymbol{x},\boldsymbol{y},(0,2,3),(0,1,2),2) \tag{5.19}$$

According to Eqs. (5.14) and (5.15), the probabilities in the right-hand side of Eq. (5.19) are approximated by

$$p_r(\boldsymbol{x},\boldsymbol{y},(0,3),(0,2),1) := p_r(\boldsymbol{x}_1^3,\boldsymbol{y}_1^2,3,2)p_r(\$,\$\,|\,\boldsymbol{x}_1^3,\boldsymbol{y}_1^2) \tag{5.20}$$

$$p_r(\boldsymbol{x},\boldsymbol{y},(0,1,3),(0,1,2),2) := p_r(x_1,y_1,1,1)p_r(\boldsymbol{x}_2^3,y_2,2,1\,|\,x_1,y_1)p_r(\$,\$\,|\,\boldsymbol{x}_2^3,y_2) \tag{5.21}$$

$$p_r(\boldsymbol{x},\boldsymbol{y},(0,2,3),(0,1,2),2) := p_r(\boldsymbol{x}_1^2,y_1,2,1)p_r(x_3,y_2,1,1\,|\,\boldsymbol{x}_1^2,y_1)p_r(\$,\$\,|\,x_3,y_2) \quad . \tag{5.22}$$

From Eqs. (5.19)–(5.22), and applying Eq. (5.16), the joint probability of observing $\boldsymbol{x}$ and $\boldsymbol{y}$ is written in terms of model parameters

$$\begin{aligned} p_r(\boldsymbol{x},\boldsymbol{y}) := {} & \mathrm{p}(\boldsymbol{x}_1^3,\boldsymbol{y}_1^2)\,\mathrm{p}(\$,\$\,|\,\boldsymbol{x}_1^3,\boldsymbol{y}_1^2) \\ & + \mathrm{p}(x_1,y_1)\,\mathrm{p}(\boldsymbol{x}_2^3,y_2\,|\,x_1,y_1)\,\mathrm{p}(\$,\$\,|\,\boldsymbol{x}_2^3,y_2) \\ & + \mathrm{p}(\boldsymbol{x}_1^2,y_1)\,\mathrm{p}(x_3,y_2\,|\,\boldsymbol{x}_1^2,y_1)\,\mathrm{p}(\$,\$\,|\,x_3,y_2) \end{aligned} \quad . \tag{5.23}$$

Note that the source and target segmentations in Eqs. (5.19)–(5.22) do not explicitly appear in Eq. (5.23), though they are implicitly taken into account since they guide the probability decompositions in Eqs. (5.20)–(5.22).

5.4 Useful recurrences

Given a bilingual pair $(\boldsymbol{x},\boldsymbol{y})$ and a parameter set $\boldsymbol{\theta}$, the joint probability $\mathrm{p}_{\boldsymbol{\theta}}(\boldsymbol{x},\boldsymbol{y})$ is efficiently computed by means of any of the following two recurrences: the *forward-like* and *backward-like* recurrences.

Roughly speaking, the forward recursion efficiently computes the probability of a given prefix. More precisely, given the source boundaries l',l and the target boundaries m',m; the forward recurrence is defined as the probability of the prefix $\boldsymbol{x}_1^l$ and $\boldsymbol{y}_1^m$ to occur knowing that the previous segment ended at positions l' and m',

$$\alpha_{l'lm'm} = \alpha_{l'lm'm}(\boldsymbol{x},\boldsymbol{y}) = p_{\boldsymbol{\theta}}(\boldsymbol{x}_1^l,\boldsymbol{y}_1^m,l',m') \quad , \tag{5.24}$$

where $0 < l' < l \leq J$ and $0 < m' < m \leq I$.

The forward-like recursion is efficiently computed by the following recursive equation

$$\alpha_{l'lm'm} = \begin{cases} 1 & l' = l = m = m' = 0 \\ \mathrm{p}(\boldsymbol{x}_1^l,\boldsymbol{y}_1^m) & l' = m' = 0, \\ & l > 0, m > 0 \\ \displaystyle\sum_{l''=0}^{l'-1}\sum_{m''=0}^{m'-1} \alpha_{l''l'm''m'}\,\mathrm{p}(\boldsymbol{x}_{l'+1}^l,\boldsymbol{y}_{m'+1}^m\,|\,\boldsymbol{x}_{l''+1}^{l'},\boldsymbol{y}_{m''+1}^{m'}) & \text{otherwise} \end{cases} \tag{5.25}$$

The backward counterpart efficiently computes the probability of a given suffix. Specifically, given the source boundaries l'',l' and the target boundaries m'',m' the forward recurrence is defined as the probability of the suffix $\boldsymbol{x}_{l'+1}^J$ and $\boldsymbol{y}_{m'+1}^I$ given that the previous segments were $\boldsymbol{x}_{l''+1}^{l'}$ and $\boldsymbol{y}_{m''+1}^{m'}$, in other words

$$\beta_{l''l'm''m'} = \beta_{l''l'm''m'}(\boldsymbol{x},\boldsymbol{y}) = p_{\boldsymbol{\theta}}(\boldsymbol{x}_{l'+1}^J,\boldsymbol{y}_{m'+1}^I\,|\,\boldsymbol{x}_{l''+1}^{l'},\boldsymbol{y}_{m''+1}^{m'}) \quad , \tag{5.26}$$

where $0 < l'' < l' \leq J$ and $0 < m'' < m' \leq I$.

Again, the backward-like recursion is efficiently computed by the application of the following recursive equation

$$\beta_{l''l'm''m'} = \begin{cases} 1 & l' = l = J, m = m' = I \\ \mathrm{p}(\$, \$ \mid \boldsymbol{x}_{l''+1}^{J}, \boldsymbol{y}_{m''+1}^{I}) & l' = J, m' = I, \\ & l'' < J, m'' < I \\ \displaystyle\sum_{l=l'+1}^{J} \sum_{m=m'+1}^{I} \beta_{l'lm'm}\, \mathrm{p}(\boldsymbol{x}_{l'+1}^{l}, \boldsymbol{y}_{m'+1}^{m} \mid x_{l''+1}^{l'}, \boldsymbol{y}_{m''+1}^{m'}) & \text{otherwise} \end{cases} \tag{5.27}$$

Using the forward-like recurrence, the joint probability $\mathrm{p}_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{y})$ is computed as follows

$$\mathrm{p}_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{y}) = \sum_{l,m} \alpha_{lJmI} \quad . \tag{5.28}$$

Alternatively, the joint probability can also be computed with the backward recursion as follows

$$\mathrm{p}_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{y}) = \beta_{0000} \quad . \tag{5.29}$$

The total time complexity required to compute both recurrences is $O(J^3 I^3)$, and two matrices of size $O(J^2 I^2)$ are needed to store them. It is important to highlight that if we use a Markovian approximation of order n, higher than 2, then the recurrence tables will need $O(J^n I^n)$ elements and a time complexity of $O(J^{n+1} I^{n+1})$.

The probability of using a given source and target segment positions l'', l', l and m'', m', m, respectively, is defined as follows

$$\gamma_{l''l'lm''m'm} = \frac{\alpha_{l''l'm''m'}\, \mathrm{p}(\boldsymbol{x}_{l'+1}^{l}, \boldsymbol{y}_{m'+1}^{m} \mid x_{l''+1}^{l'}, \boldsymbol{y}_{m''+1}^{m'}) \beta_{l'lm'm}}{\mathrm{p}_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{y})} \quad , \tag{5.30}$$

with $0 \leq l'' < l' < l \leq J$ and $0 \leq m'' < m' < m \leq I$.

Finally, we will henceforth use the notation $\gamma_{nl''l'lm''m'm}$ to refer to $\gamma_{l''l'lm''m'm}$ for the n-th outcome of a given collection of training translation pairs $\{(\boldsymbol{x}_1, \boldsymbol{y}_1), \ldots, (\boldsymbol{x}_N, \boldsymbol{y}_N)\}$. This notation is also expanded to the corresponding forward $\alpha_{nl'lm'm}$, and backward $\beta_{nl'lm'm}$ recurrences.

5.5 Maximum likelihood estimation of SGIATI

Since our model is based on a hidden bilingual segmentation variable, it is necessary to use some approximate inference algorithm. Specifically, we use the EM algorithm introduced in Section 1.1.4 in order to estimate the parameters in Eq. (5.17) w.r.t. a collection of training translation pairs $\{(\boldsymbol{x}_1, \boldsymbol{y}_1)\}_{n=1}^{N}$. The *(incomplete)* log-likelihood function is given by

$$\mathrm{LL}(\boldsymbol{\theta}) = \sum_{n=1}^{N} \log \sum_{C_n} \sum_{\boldsymbol{j}_n, \boldsymbol{i}_n} \mathrm{p}_{\boldsymbol{\theta}}(\boldsymbol{x}_n, \boldsymbol{y}_n, \boldsymbol{j}_n, \boldsymbol{i}_n, C_n) \quad , \tag{5.31}$$

with

$$\mathrm{p}_{\boldsymbol{\theta}}(\boldsymbol{x}_n, \boldsymbol{y}_n, \boldsymbol{j}_n, \boldsymbol{i}_n, C_n) = \prod_{c=1}^{C_n} \mathrm{p}(\boldsymbol{x}_n(c), \boldsymbol{y}_n(c) \mid H_n^2(c-1))\ \mathrm{p}(\$, \$ \mid H_n^2(C_n)) \quad . \tag{5.32}$$

However, the EM algorithm maximises $\mathrm{LL}(\cdot)$ by iteratively maximising a variational function $\mathcal{L}(\mathrm{q}, \boldsymbol{\theta})$ as reviewed in Section 1.1.4 Chapter 1. Let $\boldsymbol{\theta}^{(k-1)}$ be a guess of the optimal parameters obtained from previous iterations. Then, in the E-step all the sufficient statistics needed to compute $\mathrm{q}^{(k-1)}(\boldsymbol{j}_n, \boldsymbol{i}_n) = \mathrm{p}_{\boldsymbol{\theta}^{(k-1)}}(\boldsymbol{j}_n, \boldsymbol{i}_n \,|\, \boldsymbol{x}_n, \boldsymbol{y}_n)$ are calculated. Specifically, we compute the probabilities $\gamma_{nl''l'lm''m'm}$, for all $0 \leq l'' < l' < l \leq J$ and $0 \leq m'' < m' < m \leq I$. Note that in order to efficiently compute and store $\gamma_{nl''l'lm''m'm}$, both recurrences $\alpha_{nl'lm'm}$ and $\beta_{nl'lm'm}$ are computed and stored.

In the M-step, the parameter set $\boldsymbol{\theta}^{(k)}$ that maximises $\mathcal{L}(\mathrm{q}^{(k)}, \boldsymbol{\theta})$ are computed as follows

$$\mathrm{p}^{(k)}(\boldsymbol{u}, \boldsymbol{v} \,|\, \boldsymbol{u}', \boldsymbol{v}') = \frac{N^{(k-1)}(\boldsymbol{u}, \boldsymbol{v}; \boldsymbol{u}', \boldsymbol{v}')}{\sum_{\boldsymbol{u}'', \boldsymbol{v}''} N^{(k-1)}(\boldsymbol{u}'', \boldsymbol{v}''; \boldsymbol{u}', \boldsymbol{v}')} \quad , \tag{5.33}$$

where we have used the definition

$$N^{(k-1)}(\boldsymbol{u}, \boldsymbol{v}; \boldsymbol{u}', \boldsymbol{v}') = \sum_{n} \sum_{\substack{l''<l'<l \\ m''<m'<m}} \gamma_{nl''l'lm''m'm}\, \delta_{l''l'lm''m'm}(\boldsymbol{x}_n, \boldsymbol{y}_n, \boldsymbol{u}', \boldsymbol{u}, \boldsymbol{v}', \boldsymbol{v}) \quad ,$$

which is the expected value of the occurrences of the event $(\boldsymbol{u}, \boldsymbol{v}; \boldsymbol{u}', \boldsymbol{v}')$ in the training data. The expression $\delta_{l''l'lm''m'm}(\boldsymbol{x}_n, \boldsymbol{y}_n, \boldsymbol{u}', \boldsymbol{u}, \boldsymbol{v}', \boldsymbol{v})$ is a predicate that is 1 if the segment boundaries l'', l', l, and m'', m', m, and the source and target phrases are compatible, i.e.,

$$\delta_{l''l'lm''m'm}(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{u}', \boldsymbol{u}, \boldsymbol{v}', \boldsymbol{v}) = \begin{cases} 1 & \boldsymbol{x}_{l''+1}^{l'} = \boldsymbol{u}', \boldsymbol{x}_{l'+1}^{l} = \boldsymbol{u}, \boldsymbol{y}_{m''+1}^{m'} = \boldsymbol{v}', \boldsymbol{y}_{m'+1}^{m} = \boldsymbol{v} \\ 0 & \text{otherwise} \end{cases} \tag{5.34}$$

In order to implement the re-estimation Eq. (5.33), it is only needed to compute the forward $\alpha_{nl'lm'm}$ and backwards $\beta_{nl'lm'm}$ for all samples and for all values of l, m, l' and m'. Afterwards, the expected counts given by $\gamma_{nl''l'lm''m'm}$, are efficiently computed using the previously computed forward and backward recursions.

As we have discussed in Section 1.1.3, the maximum likelihood estimation technique tends to underestimate the probability of the unseen events. The EM algorithm is not an exception, and, therefore, we need to resort to smoothing techniques. Since the SGIATI techniques is highly inspired in n-gram models, it seems sensible to extend the leaving-one-out smoothing estimation techniques discussed in Chapter 3. Therefore, we use the following backing-off smoothing

$$\tilde{\mathrm{p}}(\boldsymbol{u}, \boldsymbol{v} \,|\, \boldsymbol{u}', \boldsymbol{v}') = \begin{cases} \mathrm{p}(\boldsymbol{u}, \boldsymbol{v} \,|\, \boldsymbol{u}', \boldsymbol{v}')(1 - \phi(\boldsymbol{u}', \boldsymbol{v}')) & \text{if } (\boldsymbol{u}, \boldsymbol{v}) \in \mathcal{V}(\boldsymbol{u}', \boldsymbol{v}') \\ \mathrm{p}_{bo}(\boldsymbol{u}', \boldsymbol{v}')\phi(\boldsymbol{u}', \boldsymbol{v}') & \text{if } (\boldsymbol{u}, \boldsymbol{v}) \notin \mathcal{V}(\boldsymbol{u}', \boldsymbol{v}') \end{cases} \tag{5.35}$$

where $\mathcal{V}(\boldsymbol{u}', \boldsymbol{v}') \subset \mathbf{X}^\star \times \mathbf{Y}^\star$ is the set of segments that have a not-null probability given the history $(\boldsymbol{u}', \boldsymbol{v}')$; $\mathrm{p}_{bo}(\boldsymbol{u}', \boldsymbol{v}')$ is a probability distribution defined over all unseen segmentations pairs, i.e. $(\boldsymbol{u}, \boldsymbol{v}) \notin \mathcal{V}(\boldsymbol{u}', \boldsymbol{v}')$; and, finally, $\phi(\boldsymbol{u}', \boldsymbol{v}')$ stands for the probability mass discounted from the seen events that occur after the previous history $(\boldsymbol{u}', \boldsymbol{v}')$.

Since in this model we are using fractional occurrence counts $N^{(k-1)}(\boldsymbol{u}, \boldsymbol{v}; \boldsymbol{u}', \boldsymbol{v}')$, instead of actual counts; it is not possible to apply leaving-one-out to obtain a closed form solution to the discounted probability mass $\phi(\boldsymbol{u}', \boldsymbol{v}')$ as it is done in n-gram language models [Ney et al., 1997]. In practice, we have fixed it to a constant value $\phi(\boldsymbol{u}', \boldsymbol{v}') = \epsilon$.

5.6 Preliminary experiments

In this section, some preliminary experiments were carried out to asses the formal derivation of the current GIATI version, SGIATI, and to compare it with the previous heuristic version of GIATI.

	Test Set		Train Set	
	Spanish	English	Spanish	English
sentences	2K		10K	
avg. length	12.7	12.6	12.9	13.0
vocabulary	611	468	686	513
singletons	63	49	8	10
running words	35.0K	35.6K	97.2K	99.2K
perplexities (3-gram)	-	-	5.4	3.8

Table 5.1: Basic statistics of the Spanish-English EUTRANS-I task, where *singletons* stands for the words occurring once, and *running words* denotes the total amount of word occurrences.

	WER		BLEU		SER	
Order (n)	1	2	1	2	1	2
GIATI	20.4	8.3	63.2	87.3	80.4	44.2
SGIATI	13.0	7.7	77.4	88.5	65.3	41.1
Moses	11.7		88.3		42.1	

Table 5.2: Results obtained with the EUTRANS-I task for different algorithms: SGIATI (the EM version), and GIATI, which corresponds to the model obtained by counting the occurrences of each segment and then re-normalising by the sum of all counts.

The experiments were carried out using the Spanish-English EUTRANS-I task [Amengual et al., 2000]. The Spanish-English sentence pairs correspond to human-to-human communication situations at the front-desk of a hotel which were semi-automatically produced using a small seed corpus compiled from travel guides booklets. The corpus comprises several domains and 4 persons each of which was in charge of a (non-disjoint) subset of sub-domains. The basic statistics of this corpus are shown in Table 5.1.

Since the size of recurrence tables grow exponentially with the length of the history size, we only report results for the bigram and unigram case. Moreover, in the bigram case we used the smoothing detailed in Eq. (5.35).

Table 5.2 summarises some results. The GIATI denotes the model obtained by counting the occurrences of each segment and then re-normalising by the sum of all counts, i.e. previous GIATI estimation technique. The SGIATI stands for the training algorithm presented in this chapter. Finally, *Moses* stands for the Moses system [Koehn et al., 2007] that were trained performing the MERT in a validation set. We have fixed the maximum phrase length to 7 words for all the systems. The proposed statistical estimation provides an increase of performance with respect to the GIATI version for both history sizes. The high increase obtained for the unigram model is due to the fact that there is only one state and the probability mass can be readjusted properly. This re-estimation meaningfully differs from GIATI parameters. In the bigram case, the average of segment pairs per state is over 5, which means that on average the EM can only redistribute the probability mass among few segments (over 5) for a given previous history. Both GIATI algorithms are highly dependent on the quality of the selected segments used to initialise the algorithm.

5.7 Conclusions

In this chapter, we have proposed a new statistical estimation for stochastic finite-state transducers. Specifically, a segmental extension to these models obtained via GIATI methodology have been described. This statistical framework is more independent with respect to alignment methods. The results reported show that the new technique increases the system performance with respect to (the conventional) GIATI in a small translation task.

However, the proposed statistical approach presents two great disadvantages: the complexity and the memory requirements. These requirements, which are generated by the recurrences, make the generalisation of this technique with complex data and with longer histories unfeasible. For example, the algorithm needs about 7GB in order to store the recurrence tables for trigrams and sentences no longer than 100 words. If a maximum memory is given for training the SGIATI model, then the longer the n of the Markovian approximation is, the shorter the sentences have to be. Additionally, we have observed that the improvements obtained using the proposed SGIATI model are not larger enough for justifying the memory and time complexity that this model requires.

Another great disadvantage is that, unlike the GIATI model [Casacuberta et al., 2005], the SGIATI model cannot directly take profit from the n-gram smoothing techniques based on leaving-one-out. Therefore, we probably loose more performance than what we can gain by using SGIATI.

Finally, since the SGIATI is a joint model $\mathrm{p}_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{y})$, we are modelling more than what we need, i.e., a conditional probability model $\mathrm{p}_{\boldsymbol{\theta}}(\boldsymbol{y} \mid \boldsymbol{x})$. In a joint model, not only the translation correspondence between words is learnt but also their occurrence frequency. The problem generated by this fact can be easily understood with the following example. We assume that we have observed a bilingual pair just once, but that the translation of this pair is unique. For instance, we might have observed "My room is 217" and its translation "Mi habitación es la 217". We further assume that the numbers "217" have only occurred in this outcome. From this example it is clear that a good conditional model will assign a high probability to the event $(217 \mid 217)$, i.e., $\mathrm{p}(217 \mid 217) \approx 1$. However, since the event $(217, 217)$, has only occurred once, a joint model will give it a very small probability, i.e., $\mathrm{p}(217 \mid 217) \approx 0$. This small probability accounts for two facts: that the pair $(217, 217)$ is a good translation phrase or/and that the pair $(217, 217)$ is not frequent; and there is no way to differentiate between them. In the following chapter, we propose a conditional model based on this SGIATI model that fixes some of the deficiencies of this model.

Bibliography

E. Vidal, F. Thollard, F. Casacuberta C. de la Higuera, and R. Carrasco. Probabilistic finite-state machines - part II. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27 (7):1026–1039, 2005a.

F. Casacuberta et al. Some approaches to statistical and finite-state speech-to-speech translation. *Computer Speech and Language*, 18:25–47, 2004.

P. García E. Vidal and E. Segarra. "inductive learning of finite-state transducers for the interpretation of unidimensional objects". *Structural Pattern Analysis*, pages 17–35, 1989.

J. Oncina, P. García, and E. Vidal. Learning subsequential transducers for pattern recognition interpretation tasks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15: 448–458, 1993.

K. Knight and Y. Al-Onaizan. Translation with finite-state devices. In E. Hovy D. Farwell, L. Gerber, editor, *Proc. of AMTA'98*, volume 1529, pages 421–437, London, UK, October 1998. Springer-Verlag. ISBN 3-540-65259-0.

E. Mäkinen. Inferring finite transducers. Technical Report A-1999-3, University of Tampere, 1999.

Juan C. Amengual, José M. Benedí, Asunción Castano, Antonio Castellanos, Víctor M. Jiménez, David Llorens, Andrés Marzal, Moisés Pastor, Federico Prat, Enrique Vidal, and Juan M. Vilar. The EuTrans-I speech translation system. *Machine Translation*, 15:75–103, 2000.

H. Alshawi, S. Douglas, and S. Bangalore. Learning dependency translation models as collections of finite-state head transducers. *Computational Linguistics*, 26(1):45–60, 2000a. ISSN 0891-2017.

F. Casacuberta. Inference of finite-state transducers by using regular grammars and morphisms. In *Grammatical Inference: Algorithms and Applications. Procedings of the 5th. ICGI*, volume 1891 of *Lecture Notes in Computer Science*, pages 1–14. Springer-Verlag, 2000a.

J. M. Vilar. Improve the learning of subsequential transducers by using alignments and dictionaries. In *Grammatical Inference: Algorithms and Applications*, volume 1891 of *Lecture Notes in Artificial Intelligence*, pages 298–312. Springer-Verlag, 2000.

S. Vogel and H. Ney. Translation with cascaded finite state transducers. In *Proceedings of the 38th Annual meeting of the Association for Computational Linguistics*, Hong Kong, October 2000.

David Picó and Francisco Casacuberta. Some statistical-estimation methods for stochastic finite-state transducers. *Machine Learning*, 44: 121–142, July-August 2001.

S. Bangalore and G. Riccardi. Stochastic finite-state models for spoken language machine translation. *Machine Translation*, 17(3):165–184, 2003.

S. Kumar and W. Byrne. A weighted finite state transducer implementation of the alignment template model for statistical machine translation. In *Proceedings of the 2003 Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*, pages 63–70, Edmonton, May-June 2003.

F. Casacuberta and E. Vidal. Machine translation with inferred stochastic finite-state transducers. *Computational Linguistics*, 30(2):205–225, 2004.

H. Tsukada and M. Nagata. Efficient decoding for statistical machine translation with a fully expanded wfst model. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*, pages 427–433, Barcelona, july 2004.

F. Casacuberta, E. Vidal, and D. Picó. Inference of finite-state transducers from regular languages. *Pattern Recognition*, 38:1431–1443, 2005.

S. Kumar, Y. Deng, and W. Byrne. A weighted finite state transducer translation template model for statistical machine translation. *Natural Language Engineering*, 2006. In press.

F. Casacuberta and E. Vidal. Learning finite-state models for machine translation. *Machine Learning*, 66(1):69–91, 2007.

José B. Mariòo, Rafael E. Banchs, Josep M. Crego, Adrià de Gispert, Patrik Lambert, José A. R. Fonollosa, and Marta R. Costa-jussà. N-gram-based machine translation. *Comput. Linguist.*, 32(4):527–549, 2006. ISSN 0891-2017. doi: http://dx.doi.org/10.1162/coli.2006.32.4.527.

P. F. Brown et al. The Mathematics of Statistical Machine Translation: Parameter Estimation. *Computational Linguistics*, 19(2):263–311, 1993.

F.J. Och and H. Ney. The alignment template approach to statistical machine translation. *Computational Linguistics*, 30(4):417–449, 2004.

R. Zens. *Phrase-based Statistical Machine Translation: Models, Search, Training*. PhD thesis, RWTH Aachen University, Aachen, Germany, February 2008.

D. Wu. Stochastic inversion transduction grammars, with application to segmentation, bracketing, and alignment of parallel corpora. In *Proceedings of 14th International Joint Conference on Artificial Intelligence*, pages 1328–1335, Montreal, Canada, August 1995.

H. Alshawi, S. Bangalore, and S. Douglas. Learning dependency translation models as collections of finite state head transducers. *Computational Linguistics*, 26, 2000b.

S. Barrachina et al. Statistical approaches to computer-assisted translation. *Computational Linguistics*, page In press.

E. Vidal, F. Thollard, F. Casacuberta C. de la Higuera, and R. Carrasco. Probabilistic finite-state machines - part I. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27 (7):1013–1025, 2005b.

J. Berstel. *"Transductions and conteex-free languages*. B. G. Teubner Stuttgart, 1979.

F. Casacuberta. Inference of finite-state transducers by using regular grammars and morphisms. In A.L. Oliveira, editor, *Grammatical Inference: Algorithms and Applications*, volume 1891 of *Lecture Notes in Computer Science*, pages 1–14. Springer-Verlag, 2000b. 5th International Colloquium Grammatical Inference -ICGI2000-. Lisboa. Portugal.

P. F. Brown et al. Class-based n-gram models of natural language. *Computational Linguistics*, 18(4):467–479, 1992.

D. Llorens, J. M. Vilar, and F. Casacuberta. Finite state language models smoothed using n-grams. *International Journal of Pattern Recognition and Artificial Intelligence*, 16(3):275–289, 2002.

J. Andrés-Ferrer, A. Juan, and F. Casacuberta. Statistical estimation of rational transducers applied to machine translation. *Applied Artificial Intelligence*, 22(1-2):4–22, 2008.

H. Ney, S. Martin, and F. Wessel. Statistical language modeling using leaving-one-out. In S. Young and G. Bloothooft, editors, *Corpus-Based Statiscal Methods in Speech and Language Processing.*, pages 174–207. Kluwer Academic Publishers, 1997.

P. Koehn et al. Moses: Open source toolkit for statistical machine translation. In *Proc. of ACL'07: Demo and Poster Sessions*, pages 177–180, Morristown, NJ, USA, June 2007. Association for Computational Linguistics.

Chapter 6

A phrase-based hidden Markov model for monotone machine translation

“ *The existing scientific concepts cover always only a very limited part of reality, and the other part that has not yet been understood is infinite.* ” W. HEISENBERG

Contents

6.1 Introduction

As previously discussed, the machine translation problem is stated as the problem of translating a *source (input)* sentence, $\boldsymbol{x}$, into a *target (output)* sentence, $\boldsymbol{y}$. Typically, at least one inverse translation model is needed to approximate the probability $p_r(\boldsymbol{x} \mid \boldsymbol{y})$ in Eq. (1.102).

In the Chapter 1, we discussed that the first proposed models, the so-called *IBM translation models* [Brown et al., 1993], tackled the problem with word-level dictionaries plus alignments between words. However, current systems model the inverse conditional probability using *phrase dictionaries*. This phrase-based methodology stores specific sequences of target words (*target phrase*) into which a sequence of source words (*source phrase*) is translated. A key concept of this approach is the procedure through which these phrase pairs are inferred.

A popular, phrase-based technique consists in using the IBM alignment models [Brown et al., 1993] to obtain a symmetrised alignment matrix from which *coherent* phrases are extracted (see Section 1.3 Chapter 1). Then, an approximate and heuristically motivated count normalisation is carried out in order to obtain a conditional phrase dictionary [Koehn et al., 2003].

Alternatively, some approaches have been described in the last few years in which phrase dictionaries are statistically inferred. In particular, the SGIATI model presented in Chapter 5 defines a joint model with its algorithm for estimating phrase-based probabilities. Another joint probability model for phrase-based estimation was proposed in [Marcu and Wong, 2002], however, this model is a particular case of SGIATI in which the previous history is ignored ($n = 1$) and where the monotonicity constraint has been removed. In the work by Marcu and Wong [2002], all possible segmentations are extracted using the EM algorithm [Dempster et al., 1977], without any matrix alignment constraint, in contrast to the approach followed in Och and Ney [2004]. Based on this work, Birch et al. [2006], constrained the EM to only consider phrases which agree with the alignment matrix, thus reducing the size of the phrase dictionaries (or tables).

A drawback of the above phrase-based models is that they are not conditional, but joint models that need to be renormalised in order to make them conditional. Recall that in the previous Chapter 5, we outlined some of the problems of using a joint model. In this chapter, however, we introduce a direct, conditional phrase-based approach for monotone translation [Andrés-Ferrer and Juan, 2007]. Monotonicity allows us to derive a relatively simple statistical model which is properly described as a *phrase-based hidden Markov model.*

In the remaining of this chapter, we first introduce our model in Section 6.2, and then their associated training recurrences in Section 6.3. The decoding algorithm is explained in the following Section 6.4. EM-based maximum likelihood estimation of the model parameters is described in Section 6.5. Empirical results are reported in Section 6.6 and then some concluding remarks are given.

6.2 Phrase-based hidden Markov model

Let $\boldsymbol{x}$ and $\boldsymbol{y}$ be a pair of source and target sentences of known length, J and I. In order to define our phrase-based hidden Markov model for $\mathrm{p}_{\boldsymbol{\theta}}(\boldsymbol{x} \mid \boldsymbol{y}, J)$, it is first convenient to introduce our definition of monotone segmentation, both for the monolingual and bilingual cases.

A monotone, monolingual segmentation of $\boldsymbol{x}$ into a given number of segments, T, is any sequence of indexes $\boldsymbol{j} = (j_0, j_1, \ldots, j_T)$ such that $1 = j_0 < j_1 < \cdots < j_T = J$. Similarly, a monotone, segmentation of $\boldsymbol{y}$ into T segments is any sequence of indexes $\boldsymbol{i} = (i_0, i_1, \ldots, i_T)$ such that $1 = i_0 < i_1 < \cdots < i_T = I$. Given two monotone, monolingual segmentations of $\boldsymbol{x}$ and $\boldsymbol{y}$ into T segments, $\boldsymbol{j}$ and $\boldsymbol{i}$, their associated *bilingual* segmentation of $\boldsymbol{x}$ and $\boldsymbol{y}$ is defined as $\boldsymbol{s} = s_1 s_2 \cdots s_T$ with $s_t = (j_{t-1} + 1, j_t, i_{t-1} + 1, i_t)$, $t = 1, \ldots, T$. Reciprocally, given a monotone, *bilingual* segmentation of $\boldsymbol{x}$ and $\boldsymbol{y}$, we can easily extract their associated monolingual counterparts.

Figure 6.1 shows an example in which all possible bilingual segmentations for $J = 4$ and $I = 5$ are represented as paths in a directed, multi-stage graph. The initial stage of the graph has a single, artificial node labelled as "init", which is only included to point to the initial segments of all the possible segmentations. There are 12 of such initial segments, vertically aligned on the first stage. Similarly, there are 15, 3 and 13 segments aligned on the second, third and final stages, respectively. The total number of segments is then 43. There is a unique segmentation of unit length, $\boldsymbol{s} = s_1 = (1415)$, which is represented by the rightmost path, but there are 12, 18 and 4 segmentations of length 2, 3 and 4, respectively; comprising 35 segmentations in total. As empty segments are not allowed, segmentation lengths range from one to the length of the shortest sentence. Note that segments on the first stage can only appear in the first position of a segmentation. Also, segments on the second and final stages can only appear on analogous positions in a segmentation. However, those three on the third stage (i.e. (3334), (3333) and (3344)) may appear in the second or third positions, although they cannot end any segmentation. For instance, (3334) appears in the second position of $((1212), (3334), (4455))$ and also in the third position of $((1111), (2222), (3334), (4455))$.

Note that we are using the terms segment and segmentation only for positions in the input and output sentences. We reserve the term *phrase* for actual portions of the given sentences. For instance, the bilingual segmentation $((1212), (3334), (4455))$ of $\boldsymbol{x}_1^4$ and $\boldsymbol{y}_1^5$ results in the bilingual phrases $(\boldsymbol{x}_1^2, \boldsymbol{y}_1^2)$, $(x_3, \boldsymbol{y}_3^4)$ and (x_4, y_5).

In what follows, we will write $\boldsymbol{x}(s_t)$ to denote the portion of $\boldsymbol{x}$ delimited by (the input part of) segment s_t; more generally, $\boldsymbol{x}(\boldsymbol{s}_{t'}^t)$ will denote the concatenation $\boldsymbol{x}(s_{t'})\boldsymbol{x}(s_{t'+1}) \cdots \boldsymbol{x}(s_t)$. Analogous notation will be used for $\boldsymbol{y}$, i.e., $\boldsymbol{y}(s_t)$ and $\boldsymbol{y}(\boldsymbol{s}_{t'}^t)$.

Now, we can define our inverse translation model for $p_r(\boldsymbol{x} \mid \boldsymbol{y})$ as a full exploration of all bilingual segmentations of $\boldsymbol{x}$ and $\boldsymbol{y}$,

$$p_r(\boldsymbol{x} \mid \boldsymbol{y}) = p_r(\boldsymbol{x} \mid \boldsymbol{y}, J) = \sum_{T=1}^{\min(J,I)} \sum_{\boldsymbol{s}} p_r(\boldsymbol{x}, \boldsymbol{s}, T \mid \boldsymbol{y}, J) \quad , \tag{6.1}$$

where the second sum is defined over all possible bilingual segmentations of length T; and as most of the literature in SMT, although we do not explicitly the dependence on the source sentence length J, it is assumed to be known.

To compute $p_r(\boldsymbol{x}, \boldsymbol{s}, T \mid \boldsymbol{y}, J)$ in (6.1), we use the following decomposition

$$p_r(\boldsymbol{x}, \boldsymbol{s}, T \mid \boldsymbol{y}, J) = p_r(T \mid \boldsymbol{y}, J) p_r(\boldsymbol{s} \mid \boldsymbol{y}, T, J) p_r(\boldsymbol{x} \mid \boldsymbol{y}, \boldsymbol{s}, T, J) \quad ,$$

where $p_r(\boldsymbol{s} \mid \boldsymbol{y}, T, J)$ is modelled as a first-order Markovian process

$$p_r(\boldsymbol{s} \mid \boldsymbol{y}, T, J) := \prod_{t=1}^{T} p_r(s_t \mid s_{t-1}) \quad , \tag{6.2}$$

with s_0 :="init", i.e. the initial state used to model the probability of each state to be the first in the sequence of states; and $p_r(\boldsymbol{x} \mid \boldsymbol{y}, \boldsymbol{s}, T, J)$ is modelled as composed of independent bilingual phrases

$$p_r(\boldsymbol{x} \mid \boldsymbol{y}, \boldsymbol{s}, T, J) := \prod_{t=1}^{T} p_r(\boldsymbol{x}(s_t) \mid \boldsymbol{y}(s_t), s_t) \quad . \tag{6.3}$$

Clearly, the above modelling assumptions lead to a phrase-based HMM-like model. Its set of states is that of all possible bilingual segments, while its set of transitions includes all pairs $\langle q', q \rangle$ in which the state (segment) q is a successor of q', $q \in \mathrm{Succ}(q')$. For each state q, we will have a different emission probability for each target segment $\boldsymbol{y}(q)$.

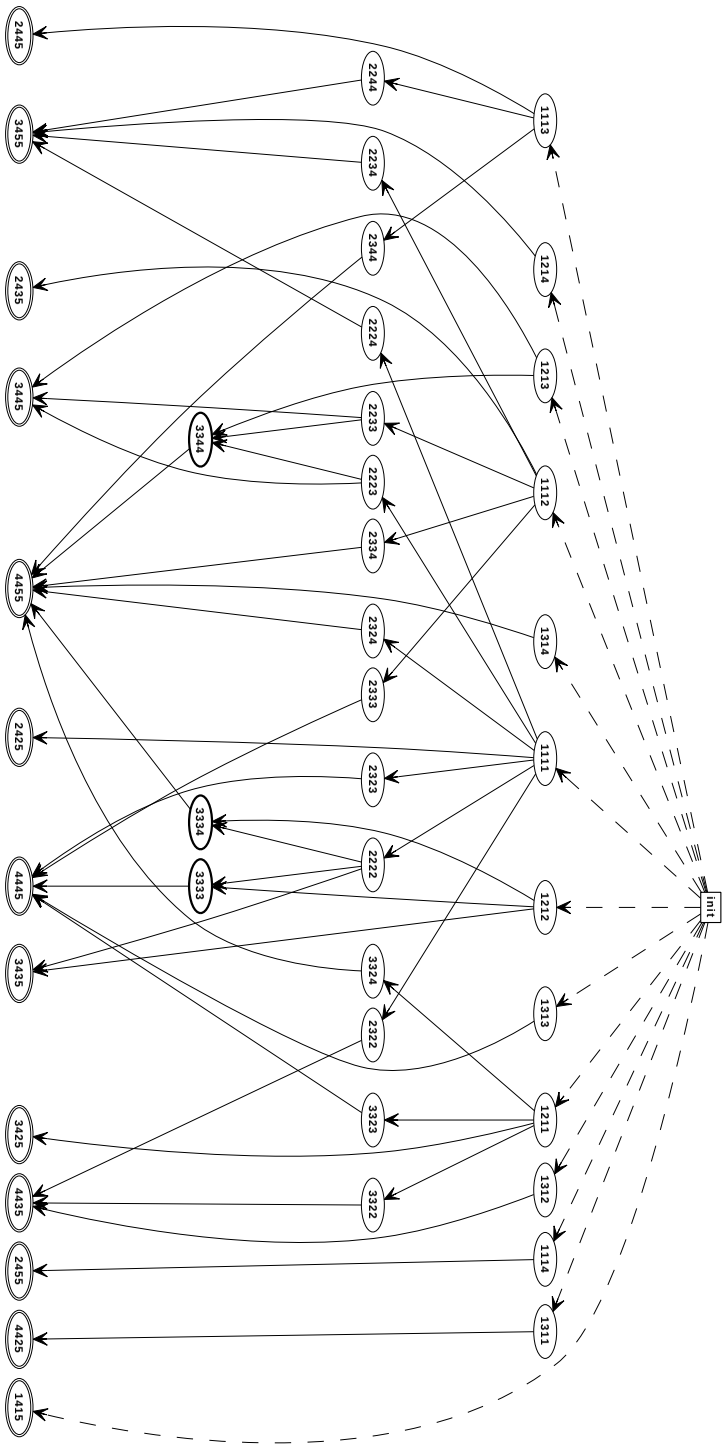


Figure 6.1: Directed, multi-stage graph representing all possible bilingual segmentations for an input sentence of length 4 and an output sentence of length 5. Each node defines a different segment; the first two digits of the node label are the segment limits in the input sentence, while the other two digits correspond to the output sentence.

For efficiency and simplicity, we will further assume in this chapter that both initial and transition state probabilities are uniformly distributed; hence, for each q and q', including "init" for q',

$$p_r(q \,|\, q') := \begin{cases} \frac{1}{|\mathrm{Succ}(q')|} & \text{if } q \in \mathrm{Succ}(q') \\ 0 & \text{otherwise} \end{cases} \tag{6.4}$$

Also, T is assumed to be uniformly distributed,

$$p_r(T \,|\, \boldsymbol{y}, J) := \frac{1}{\min(I, J)} \tag{6.5}$$

and the phrase translation probabilities are assumed to be stored in a single, state-independent table

$$p_r(\boldsymbol{x}(s_t) \,|\, \boldsymbol{y}(s_t), s_t) := \mathrm{p}(\boldsymbol{x}(s_t) \,|\, \boldsymbol{y}(s_t)) \quad .$$

Using the above assumptions, our model (6.1) can be rewritten as follows

$$\mathrm{p}_{\boldsymbol{\theta}}(\boldsymbol{x} \,|\, \boldsymbol{y}, J) := \frac{1}{\min(J, I)} \sum_{|\boldsymbol{s}| \leq \min(J,I)} \prod_{t=1}^{|\boldsymbol{s}|} \frac{\mathrm{p}(\boldsymbol{x}(s_t) \,|\, \boldsymbol{y}(s_t))}{|\mathrm{Succ}(s_{t-1})|} \quad . \tag{6.6}$$

The vector of parameters governing this model only includes a table of phrase translation probabilities,

$$\Theta = \{\mathrm{p}(\boldsymbol{u} \,|\, \boldsymbol{v}) : (\boldsymbol{u}, \boldsymbol{v}) \text{ bilingual phrase}\} \quad .$$

6.3 Forward and backward probabilities

As usual with HMMs (see Section 1.1.5 Chapter 1), we will discuss here the so-called *forward* and *backward* probabilities for efficient computation of the model probabilities, as given in Eq. (6.6). To fix ideas, consider $\boldsymbol{x}$ and $\boldsymbol{y}$ to be two arbitrary sentences for which we have to compute Eq. (6.6). Given a segmentation length and position, T and t, and a state q, the forward probability is defined as the following prefix joint probability

$$\alpha_{tq}^T := \mathrm{p}_{\boldsymbol{\theta}}(\boldsymbol{x}(\boldsymbol{s}_1^t), s_t = q \,|\, \boldsymbol{y}, T) \quad ,$$

where s_1^t is any partial segmentation, from positions 1 to t, such that $s_t = q$. This probability can be recursively computed by dynamic programming, using the so-called *forward recurrence*,

$$\alpha_{tq}^T = \sum_{q' \,:\, q \in \mathrm{Succ}(q')} \alpha_{t-1q'}^T p_r(q \,|\, q') \,\mathrm{p}(\boldsymbol{x}(q) \,|\, \boldsymbol{y}(q)) = \sum_{q' \,:\, q \in \mathrm{Succ}(q')} \alpha_{t-1q'}^T \frac{\mathrm{p}(\boldsymbol{x}(q) \,|\, \boldsymbol{y}(q))}{|\mathrm{Succ}(q')|} \quad , \tag{6.7}$$

with the base case $\alpha_{tq}^T = 1$ for $t = 0$ and $q =$"init"; 0 otherwise. Note that in Eq. (6.7), we have decomposed the case α_{tq}^T in terms of a smaller case $\alpha_{t-1\,q'}^T$.

The backward probability also depends on a given segmentation length T and position t; and a state q. It is defined as the following suffix probability

$$\beta_{tq}^T := \mathrm{p}_{\boldsymbol{\theta}}(\boldsymbol{x}(\boldsymbol{s}_{t+1}^T) \,|\, \boldsymbol{y}, T, s_t = q) \quad ,$$

where $\boldsymbol{s}_{t+1}^T$ is any partial segmentation, from positions $t+1$ to T, that might follow the state q in position t. As before, it can be efficiently computed by dynamic programming, using a "reverse" version of the forward recurrence called *backward recurrence*,

$$\beta_{tq}^T = \sum_{q' \in \mathrm{Succ}(q)} \beta_{t+1\,q'}^T p_r(q' \,|\, q) \,\mathrm{p}(\boldsymbol{x}(q') \,|\, \boldsymbol{y}(q')) = \sum_{q' \in \mathrm{Succ}(q)} \beta_{t+1\,q'}^T \frac{\mathrm{p}(\boldsymbol{x}(q') \,|\, \boldsymbol{y}(q'))}{|\mathrm{Succ}(q)|} \quad , \tag{6.8}$$

with the base case $\beta_{Tq}^{T} = 1$ for any *terminal state* $q = (\cdot, I, \cdot, J)$ and any t; and 0 otherwise. Note that similarly to the forward recurrence, in Eq. (6.8), we have decomposed the case β_{tq}^{T} in terms of the simpler case $\beta_{t+1\,q'}^{T}$.

Finally, Eq. (6.6) can be computed using (6.7) as

$$\mathrm{p}_{\boldsymbol{\theta}}(\boldsymbol{x} \,|\, \boldsymbol{y}) = \frac{1}{\min(J, I)} \sum_{T} \sum_{q=(\cdot,I,\cdot,J)} \alpha_{Tq}^{T} \quad ,$$

or using (6.8) as

$$\mathrm{p}_{\boldsymbol{\theta}}(\boldsymbol{x} \,|\, \boldsymbol{y}) = \frac{1}{\min(J, I)} \sum_{T} \beta_{0\text{“init”}}^{T} \quad .$$

An efficient implementation of both recurrences requires two tables of $O(IJ \min(J, I))$ values and a computational complexity of $O(I^2 J^2 \min(J, I))$.

6.4 Decoding and Viterbi recurrence

The Viterbi recursion, efficiently computes the most likely state sequence that can emit a given output sequence. We introduce this recursion here as a prelude to the search recurrence. This recursion is defined as the most likely state sequence of length t that ends in the state q, i.e.

$$\delta_{q\,t} = \max_{\boldsymbol{s}_1^t \,:\, s_t = q} \{\mathrm{p}_{\boldsymbol{\theta}}(\boldsymbol{x}(\boldsymbol{s}_1^t), \boldsymbol{s}_1^t \,|\, \boldsymbol{y})\} \quad , \tag{6.9}$$

note that the last state s_t is required to be q.

The Viterbi recursion in Eq. (6.9) is efficiently computed by the following recurrence

$$\delta_{q\,t} = \max_{q'} \left\{ \mathrm{p}(q \,|\, q') \, \mathrm{p}(\boldsymbol{x}(q) \,|\, \boldsymbol{y}(q)) \max_{\boldsymbol{s}_1^{t-1} \,:\, s_{t-1} = q'} \{\mathrm{p}_{\boldsymbol{\theta}}(\boldsymbol{x}(\boldsymbol{s}_1^{t-1}), \boldsymbol{s}_1^{t-1} \,|\, \boldsymbol{y}(\boldsymbol{s}_1^{t-1}))\} \right\} \quad , \tag{6.10}$$

where by applying the Viterbi's definition yields

$$\delta_{q\,t} = \max_{q'} \{\mathrm{p}(q \,|\, q') \, \mathrm{p}(\boldsymbol{x}(q) \,|\, \boldsymbol{y}(q)) \delta_{q'\,t-1}\} = \max_{q'} \left\{ \frac{\mathrm{p}(\boldsymbol{x}(q) \,|\, \boldsymbol{y}(q))}{|\mathrm{Succ}(q')|} \delta_{q'\,t-1} \right\} \quad . \tag{6.11}$$

Finally, the probability of the Viterbi's segmentation is given by

$$\mathrm{p}_{\boldsymbol{\theta}}(\hat{\boldsymbol{s}}, \boldsymbol{x} \,|\, \boldsymbol{y}) = \max_{T, q=(\cdot,J,\cdot,I)} \delta_{q,T} \quad . \tag{6.12}$$

As usually, tracing back the decisions made during the maximisation process yields the maximum segmentation, $\hat{\boldsymbol{s}}$.

The Viterbi recursion shares the same asymptotic requirements than that of the forward and backward recursions.

6.4.1 The decoding process

The decoding process is stated as the problem of finding the most likely target sentence $\hat{\boldsymbol{y}}$ for given a source sentence $\boldsymbol{x}$. According to our model the decoding problem is stated as

$$\hat{\boldsymbol{y}} = \arg\max_{\boldsymbol{y}} \{\mathrm{p}_{\boldsymbol{\theta}}(\boldsymbol{x} \,|\, \boldsymbol{y}) \; \mathrm{p}(\boldsymbol{y})\} \quad . \tag{6.13}$$

During the decoding process, an additional problem with respect to the Viterbi recursion must be kept in mind, that is to say, the target sentence $\boldsymbol{y}$ is unknown. Therefore, we want to find the following maximum probability

$$\mathrm{p}_{\boldsymbol{\theta}}(\boldsymbol{x}, \hat{\boldsymbol{y}}) = \max_{\boldsymbol{y}} \left\{ \sum_{T=1}^{\min\{I,J\}} \sum_{\boldsymbol{s}} \mathrm{p}_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{s}_1^T, T \,|\, \boldsymbol{y})\, \mathrm{p}(\boldsymbol{y}) \right\} \quad , \tag{6.14}$$

since the fundamental equation of machine translation introduced in Eq. (1.102) Chapter 1 requires a language model.

In order to solve Eq. (6.14) it is usually assumed a Viterbi-like approach approximating each sum by their maximum value as follows,

$$\hat{\mathrm{p}} = \mathrm{p}_{\boldsymbol{\theta}}(\boldsymbol{x}, \hat{\boldsymbol{y}}) = \max_{\boldsymbol{y}, \boldsymbol{s}_1^T, T} \{ \mathrm{p}_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{s}_1^T, T \,|\, \boldsymbol{y})\ \mathrm{p}(\boldsymbol{y}) \} \quad , \tag{6.15}$$

where recall that $\hat{\boldsymbol{y}}$ stands for the target sentence that maximises this probability.

Provided that we only use n-gram language models in this thesis, we further assume that the language model probability of a given target phrase $\mathrm{p}(\boldsymbol{y}(s_t) \,|\, \boldsymbol{y}(s_1^{t-1}))$ only depends on the $(n-1)$-most recent words, i.e.

$$\mathrm{p}(\boldsymbol{y}(s_t) \,|\, \boldsymbol{y}(s_1^{t-1})) := \mathrm{p}(\boldsymbol{y}(s_t) \,|\, \operatorname*{suf}_{n-1}(\boldsymbol{y}(s_1^{t-1}))) \quad , \tag{6.16}$$

where $\operatorname{suf}_{n-1}(\cdots)$ stands for the $(n-1)$-most recent words.

In order to perform this maximisation, we define a decoding recurrence

$$\sigma_{q,\boldsymbol{v}}(\boldsymbol{x}) = \max_{\substack{t, \boldsymbol{s}_1^t, \boldsymbol{y}(\boldsymbol{s}_1^t) \\ s_t = q,\, \operatorname{suf}_{|\boldsymbol{v}|}(\boldsymbol{y}(\boldsymbol{s}_1^t)) = \boldsymbol{v}}} \left\{ \mathrm{p}_{\boldsymbol{\theta}}(\boldsymbol{x}(\boldsymbol{s}_1^t), \boldsymbol{s}_1^t \,|\, \boldsymbol{y}(\boldsymbol{s}_1^t))\ \mathrm{p}(\boldsymbol{y}(\boldsymbol{s}_1^t)) \right\} \quad , \tag{6.17}$$

where by $\operatorname{suf}_{|\boldsymbol{v}|}(\boldsymbol{y}(\boldsymbol{s}_1^t)) = \boldsymbol{v}$ we denote the fact that the suffix of $\boldsymbol{y}(\boldsymbol{s}_1^t)$ must be equal to $\boldsymbol{v}$. We further assume that if $\boldsymbol{v} = \star$ then this constraint is ignored, i.e.,

$$\sigma_{q,\star}(\boldsymbol{x}) = \sigma_q(\boldsymbol{x}) = \max_{\substack{t, \boldsymbol{s}_1^t, \boldsymbol{y}(\boldsymbol{s}_1^t) \\ s_t = q}} \left\{ \mathrm{p}_{\boldsymbol{\theta}}(\boldsymbol{x}(\boldsymbol{s}_1^t), \boldsymbol{s}_1^t \,|\, \boldsymbol{y}(\boldsymbol{s}_1^t))\ \mathrm{p}(\boldsymbol{y}(\boldsymbol{s}_1^t)) \right\} \quad . \tag{6.18}$$

Note that $\sigma_{q,\boldsymbol{v}}$ can be recursively expressed in terms of a more basic case of itself as follows

$$\sigma_{q,\boldsymbol{v}} = \max_{\substack{q', \boldsymbol{v}', \boldsymbol{h} \\ \operatorname{suf}_{|\boldsymbol{v}|}(\boldsymbol{v}'\boldsymbol{h}) = \boldsymbol{v}, |\boldsymbol{v}'\boldsymbol{h}| \geq |\boldsymbol{v}|}} \{ \mathrm{p}(q \,|\, q')\ \mathrm{p}(\boldsymbol{x}(q) \,|\, \boldsymbol{v}')\ \mathrm{p}(\boldsymbol{v}' \,|\, \boldsymbol{h})\, \sigma_{q',\boldsymbol{h}} \} \quad , \tag{6.19}$$

where note that $\boldsymbol{v}'$ plays the role of $\boldsymbol{y}(s_t)$ and q the role of s_t; and where $\mathrm{p}(q \,|\, q')$ is uniformly distributed as shown in Eq. (6.4).

In this way the probability of the desired target string is computed using the search recurrence as follows

$$\mathrm{p}_{\boldsymbol{\theta}}(\boldsymbol{x}, \hat{\boldsymbol{y}}) = \max_{I} \{ \max_{q=(\cdot,I,\cdot,J)} \{ \sigma_{q,\star} \} \} \quad . \tag{6.20}$$

As usually, tracing back the decisions made during the recurrence computation provides the optimal solution defined in Eq. (6.15).

However, although the recursion $\sigma_{q,\boldsymbol{v}}$ speeds up the search problem, it is still a hard problem. For this reason, we still need to perform an approximate decoding in which we use a maximum number of hypothesis for each state q, say M, and also a beam pruning [Wang and Waibel, 1997, 1998]. That is to say, instead of using Eq. (6.19), we use the following approximated version

$$\sigma^{\star}_{q,\boldsymbol{v}} = \max^{\star}_{\substack{q', \boldsymbol{v}', \boldsymbol{h} \\ \operatorname{suf}_{|\boldsymbol{v}|}(\boldsymbol{v}'\boldsymbol{h}) = \boldsymbol{v}, |\boldsymbol{v}'\boldsymbol{h}| \geq |\boldsymbol{v}|}} \{ \mathrm{p}(q \,|\, q')\ \mathrm{p}(\boldsymbol{x}(q) \,|\, \boldsymbol{v}')\ \mathrm{p}(\boldsymbol{v}' \,|\, \boldsymbol{h})\, \sigma^{\star}_{q',\boldsymbol{h}} \} \quad , \tag{6.21}$$

where $\overset{\star}{\max}$ stands for an approximate version of $\max$ where we have applied several heuristics, such as beam search or histogram pruning.

6.5 Maximum likelihood estimation

As discussed in section 6.2, the unknown vector of parameters of our phrase-based HMM model only includes a table of phrase translation probabilities (see Eq. (6.2)). We will describe here its EM-based maximum likelihood estimation with respect to a collection of training translation pairs $\{(\boldsymbol{x}_1, \boldsymbol{y}_1), \ldots, (\boldsymbol{x}_N, \boldsymbol{y}_N)\}$.

The log-likelihood function of $\boldsymbol{\theta}$ is:

$$\begin{aligned} \mathrm{LL}(\boldsymbol{\theta}) &= \sum_n \log p_r(\boldsymbol{x}_n \,|\, \boldsymbol{y}_n) \\ &= \sum_n \log \frac{1}{\min(J_n, I_n)} \sum_{\substack{\boldsymbol{s} \\ |s| \leq \min(J_n, I_n)}} \prod_{t=1}^{|\boldsymbol{s}|} \frac{\mathrm{p}(\boldsymbol{x}_n(s_t) \,|\, \boldsymbol{y}_n(s_t))}{|\mathrm{Succ}(s_{t-1})|} \end{aligned} \tag{6.22}$$

Remember from Section 1.1.4 in Chapter 1, that the EM algorithm maximises $\mathrm{LL}(\cdot)$ by iteratively maximising a variational function $\mathcal{L}(\mathrm{q}, \boldsymbol{\theta})$ through the application of two basic steps in each iteration: the E(xpectation) step and the M(aximisation) step.

Let $\boldsymbol{\theta}^{(k-1)}$ be a guess of the optimal parameters obtained from previous iterations; then, in this case, the E step requires the computation, for each pair $(\boldsymbol{x}_n, \boldsymbol{y}_n)$, of the sample versions of (6.7) and (6.8), as well as the following joint probability

$$\xi^T_{ntq'q} := \mathrm{p}_{\boldsymbol{\theta}^{(k-1)}}(\boldsymbol{x}_n, s_{t-1} = q', s_t = q \,|\, \boldsymbol{y}_n, T) \quad ,$$

which can be efficiently computed as

$$\xi^T_{ntq'q} = \frac{\alpha^T_{nt-1q'} \;\mathrm{p}(\boldsymbol{x}(q) \,|\, \boldsymbol{y}(q))\, \beta^T_{ntq}}{p_r(\boldsymbol{x}_n \,|\, \boldsymbol{y}_n)\, |\mathrm{Succ}(q')|} \quad . \tag{6.23}$$

On the other hand, the M step re-estimates the table of phrase translation probabilities,

$$\mathrm{p}^{(k)}(\boldsymbol{u} \,|\, \boldsymbol{v}) = \frac{N^{(k-1)}(\boldsymbol{u}, \boldsymbol{v})}{\sum_{\boldsymbol{u}'} N^{(k-1)}(\boldsymbol{u}', \boldsymbol{v})} \quad , \tag{6.24}$$

where $N^{(k-1)}(\boldsymbol{u}, \boldsymbol{v})$ is the expected number of occurrences of the the pair $(\boldsymbol{u}, \boldsymbol{v})$; i.e.

$$N^{(k-1)}(\boldsymbol{u}, \boldsymbol{v}) = \sum_n \frac{1}{\min(J_n, I_n)} \sum_{q'\,q} \sum_{T\;t} \xi^T_{ntq'q}\, \delta_{nq}(\boldsymbol{u}, \boldsymbol{v}) \quad , \tag{6.25}$$

with $\delta_{nq}(\boldsymbol{u}, \boldsymbol{v})$ defined as 1 if $\boldsymbol{u} = \boldsymbol{x}_n(q)$ and $\boldsymbol{v} = \boldsymbol{y}_n(q)$; 0 otherwise.

6.6 Experiments

6.6.1 Corpora

The proposed phrase-based hidden Markov model was assessed on two different corpora: the EuTrans-I dataset [Amengual et al., 2000] and the Europarl-10. The former dataset comprises 12 000 bilingual

EUTRANS-I	Train Set		Test Set	
	Spanish	English	Spanish	English
sentences	10K		2K	
avg. length	12.9	13.0	12.7	12.6
vocabulary	686	513	611	468
running words	97.2K	99.2K	35.0K	35.6K
perplexities (3-gram)	-	-	5.4	3.8

Table 6.1: Basic statistics of the Spanish-English EUTRANS-I task, where *running words* denotes the total amount of word occurrences.

EUROPARL-10	Train set		Test set	
	English	Spanish	English	Spanish
sentences	76,996		5,000	
avg. length	7.01	7.0	7.2	7.0
voc. size	16K	22K	4.1K	5.2K
running words	546K	540K	35.8K	3.91M
perplexities (3-gram)	-	-	77.6	86.8

Table 6.2: Basic statistics of the EUROPARL-10 corpus where *running words* denotes the total amount of word occurrences.

sentence pairs from a limited-domain Spanish-English machine translation application for human-to-human communication situations in the front-desk of a hotel. It has also been used in previous Chapter 5. The latter comprises all the sentences of the English-Spanish Europarl-v2 [Koehn, 2005] with length equal or less than 10. We have randomly selected 5K sentences for testing. Some basic statistics are shown in Table 6.1 and 6.2.

6.6.2 Results

Two basic experiments were carried out with the Eutrans-I corpus. In the first experiment, we used Moses [Koehn et al., 2007] to obtain a baseline of the corpus using an inverse phrased-based probability model and a 4-gram language model, i.e. without MERT training, as discussed in Section 6.4. For this experiment, we have used a maximum phrase length of 15 words and we have used only the inverse translation model and the language model, i.e. we have not perform the MERT training. For evaluating the performance we used *word error rate* (WER) and *bilingual evaluation understudy* (BLEU) measures. We obtained a WER of 7.7% and a BLEU score of 89.1%. These are relatively good results since, recall that in general low values of WER and high values of BLEU are a clear indication of high quality translations. Additionally, we compared this Moses baseline with the proposed phrase-based hidden Markov model to better train the phrase translation table. We proceeded as in the baseline model although, now, the phrase table obtained before was used to initialise the EM algorithm proposed in Section 6.5 for parameter training in accordance with criterion in Eq. (6.22). In this case, we obtained a WER of 7.8% and BLEU of 88.5%.

Obviously, the result obtained with our model was not better than that obtained with the baseline approach. In analysing the phrase table provided by our model, we found that the EM algorithm prefers

long to short phrases; that is, given a target phrase, long source phrases are favoured with higher probabilities. To empirically check this hypothesis, we repeated the two basic experiments described above by first discarding training phrases longer that a given maximum threshold. For the most restrictive thresholds, however, phrases longer than the threshold were not discarded so as to ensure full coverage of the training data. The results are shown in Figure 6.2 in terms of BLEU.

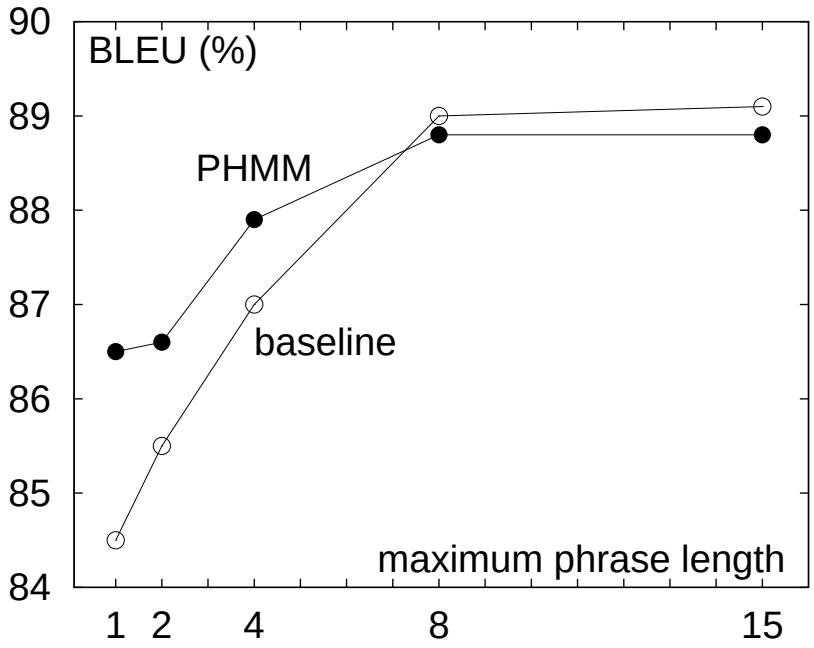


Figure 6.2: BLEU (%) as a function of the maximum phrase length threshold, for the baseline approach and our phrase-based HMM (PHMM).

The results in Figure 6.2 confirm our hypothesis on the bias to long phrases in our model. A possible solution to this problem is to refine our phrase-based HMM with inclusion of length models to penalise long phrases.

In the case of the Europarl-10 corpus, we carried out one experiment similar to the first experiment with the Eutrans-I corpus. In this case we fixed the phrase length to 7. The Moses baseline scored 50.0% points of WER and 32.7% of BLEU; whilst the proposed model scored 54.6% points of WER and 26.7% of BLEU; which clearly worsens the baseline. It seems that this negative result is also due to the overfitting tendency of the MLE, since we did not propose a smoothing model.

6.7 Conclusions

A phrase-based hidden Markov model has been proposed for statistical machine translation. We have described the forward and backward recurrences for efficient computation of the model and its EM-based parameter re-estimation algorithm. Empirically results have been reported comparing the proposed model with a baseline system. It has been found that our model is biased to long phrases and

tends to quickly overfit the system. Due to this overfitting the systems does not outperforms the baseline system.

We have identified some problems. Firstly, a classical HMM is not fully adequate for processing both input and output strings, since the handling of the segmentation variables requires speciallised states. Secondly, the computational time is too expensive to make this model useful with big corpora. Actually, we need a matrix $O(IJ\min\{I,J\})$; however the third dimension $\min\{I,J\}$ accounts for the number of phrases that have been used and does not introduce any relevant statistical information to the training process. Finally, the model tends to get overfitted biasing the long phrases.

In the following chapter, we extend and modify this model by making use of a ***hidden semi-Markov model (HSMM)*** formalism. This allow us to amend the three previously mentioned problems.

Bibliography

Juan C. Amengual, José M. Benedí, Asunción Castano, Antonio Castellanos, Víctor M. Jiménez, David Llorens, Andrés Marzal, Moisés Pastor, Federico Prat, Enrique Vidal, and Juan M. Vilar. The EuTrans-I speech translation system. *Machine Translation*, 15:75–103, 2000.

J. Andrés-Ferrer and A. Juan. A phrase-based hidden markov model approach to machine translation. In *Proceedings of New Approaches to Machine Translation*, pages 57–62, January 2007. ISBN 978-90-814861-0-1.

A. Birch, C. Callison-Burch, Miles M. Osborne, and P. Koehn. Constraining the phrase-based, joint probability statistical translation model. In *Proceedings on the Workshop on Statistical Machine Translation*, pages 154–157, New York City, New York, USA, June 2006. Association for Computational Linguistics.

P. F. Brown et al. The Mathematics of Statistical Machine Translation: Parameter Estimation. *Computational Linguistics*, 19(2):263–311, 1993.

A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *J. Royal Statist. Soc. Ser. B*, 39(1):1–22, 1977.

P. Koehn. Europarl: A parallel corpus for statistical machine translation. In *Proc. of the MT Summit X*, pages 79–86, September 2005.

P. Koehn, F.J. Och, and D. Marcu. Statistical phrase-based translation. In *Proc. of NAACL'03*, pages 48–54, Morristown, NJ, USA, 2003. Association for Computational Linguistics. doi: http://dx.doi.org/10.3115/1073445.1073462.

P. Koehn et al. Moses: Open source toolkit for statistical machine translation. In *Proc. of ACL'07: Demo and Poster Sessions*, pages 177–180, Morristown, NJ, USA, June 2007. Association for Computational Linguistics.

D. Marcu and W. Wong. A phrase-based, joint probability model for statistical machine translation. In *Proc. of EMNLP'02*, pages 133–139, Morristown, NJ, USA, July 2002. Association for Computational Linguistics.

F.J. Och and H. Ney. The alignment template approach to statistical machine translation. *Computational Linguistics*, 30(4):417–449, 2004.

Y. Wang and A. Waibel. Decoding algorithm in statistical translation. In *Proc. of ACL'97*, pages 366–372, Morristown, NJ, USA, July 1997. Morgan Kaufmann / Association for Computational Linguistics.

Y. Wang and A. Waibel. Fast decoding for statistical machine translation. In *Proc. of ICSLP'98*, pages 2775–2778, October 1998.

Chapter 7

A phrase-based hidden semi-Markov model for monotone machine translation

" *Then there was the man who drowned crossing a stream with an average depth of six inches.* "
W. I. E. GATES

Contents

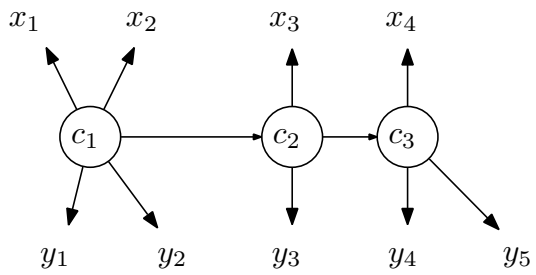


Figure 7.1: A generative example of the hidden semi-Markov model approach to machine translation, in which a source string $\boldsymbol{x}_1^4$ is translated to a target string $\boldsymbol{y}_1^5$ through a segmentation of both sentences into 4 concepts.

7.1 Introduction

In previous chapter, we introduced a novel approach to machine translation based on the hidden Markov model. One drawback of this model, is the computational complexity associated with its training algorithm. In this section, by making use of the hidden semi-Markov formalism, we successfully amend the computation problem of the previous model. With this new formalism, we properly define a phrase-based model eligible for being theoretically expanded without the problems derived from the heuristically computed phrase dictionaries. Finally, and in order to avoid the harmful overfitting problems, we resort to a smoothing word-based IBM model 1.

We begin this chapter with the description of our new proposal in Section 7.2. Likewise to classical HSMM, in Section 7.3 the well-known forward and backward recurrences are described into detail. The training algorithms are explained in Section 7.4. The practical behaviour of the PBHSMM is analysed in Section 7.6. Afterwards, concluding remarks are gathered in Section 7.7.

7.2 The phrase-based hidden semi-Markov model

Inspired on the HSMM described in Section 1.1.6 Chapter 1, we define here our *phrase-based hidden semi-Markov model (PBHSMM)* for monotone machine translation [Andrés-Ferrer and Juan, 2009]. Let $\boldsymbol{x} \in \boldsymbol{X}^\star$ be the source sentence and $\boldsymbol{y} \in \boldsymbol{Y}^\star$ the target sentence, we model the conditional translation probability, $p_r(\boldsymbol{x} \mid \boldsymbol{y}, J)$ by assuming that the monotonic translation process has been carried out from left to right in segments of words or *phrases*. For this purpose, both sentences should be segmented in the same amount of phrases. Figure 7.1, depicts an example of a possible monotonic bilingual segmentation in which the source sentence comprises 4 words, $\boldsymbol{x}_1^4$; whereas the target sentence is made up of 5 words, $\boldsymbol{y}_1^5$. Note that each bilingual phrase forms a *concept* [Brown et al., 1993]; for instance c_1, c_2 and c_3 are concepts in fig. 7.1. In order to represent the segmentation process, we use two segmentation variables for both source, $\boldsymbol{l}$, and target, $\boldsymbol{m}$, sentences.

To better understand our monotone translation model $\mathrm{p}_{\boldsymbol{\theta}}(\boldsymbol{x} \mid \boldsymbol{y}, J)$, it is first convenient to fully understand how the segmentation process is represented using the formerly mentioned segmentation variables: the source segmentation variable, $\boldsymbol{l}$; and the target segmentation variable, $\boldsymbol{m}$.

On the one hand, the target segmentation variable $\boldsymbol{m}$ stores each target segment length at the position at which the segment begins. Therefore, if the target segmentation variable $\boldsymbol{m}$ has a value greater

than 0 at position i, then a segment with length m_i starts at such position i. Note that this notation differs from that of Chapter 6, since now the length is specified at the segment boundaries. For instance, the target segmentation represented in Figure 7.1 is given by $\boldsymbol{m} = \boldsymbol{m}_1^5 = (2, 0, 1, 2, 0)$. Therefore, values for the segmentation variable such as $\boldsymbol{m} = (2, 1, 0, 3, 0)$ or $\boldsymbol{m} = (2, 2, 0, 3, 0)$, *are out-of-domain*, and, hence, invalid. The Table 7.1 enumerates in the second column all the target segment variable domain for the case of a target sentence of 5 words. In the same table, the third column corresponds to the induced target segmentation for the value of $\boldsymbol{m}$ specified in the first column. Note that the domain of the target segmentation ranges among all the possible segmentation lengths.

On the other hand, the source counterpart of the target segmentation variable is the source segmentation variable $\boldsymbol{l}$. The source segmentation random variable accounts for the length of each *source segment* at the position at which its corresponding *target segment* begins. If the source segmentation variable $\boldsymbol{l}$ has a value greater than 0 at position i, then the length of the source segment corresponding to the target phrase that starts at position i, is l_i. Recall that the length of the target segment starting at position i is m_i. For instance, in Figure 7.1 the source segmentation variable is $\boldsymbol{l} = \boldsymbol{l}_1^5 = (2, 0, 1, 1, 0)$.

Table 7.1 enumerates all possible values of both segmentations variables, $\boldsymbol{m}$ and $\boldsymbol{l}$ for a source sentence of 4 words and a target sentence of 5; and the segmentation they induce in both source and target sentences. It is valuable to mention, that the possible values of $\boldsymbol{l}$ depend on both $\boldsymbol{m}$ and J. There is only one bilingual segmentation with unit length; but there are 12, 18 and 4 segmentations of length 2, 3 and 4, respectively. Note that in the special case in which $\boldsymbol{m}$ splits the target sentence $\boldsymbol{y}_1^5$ in 5 segments; there is no possible value for $\boldsymbol{l}$ and no segmentation is induced in $\boldsymbol{x}_1^4$.

Now, we can mathematically define our inverse translation model, depicted in Figure 7.1, as a full exploration of all segmentations

$$p_r(\boldsymbol{x} \mid \boldsymbol{y}, J) = \sum_{\boldsymbol{m}} \sum_{\boldsymbol{l}} p_r(\boldsymbol{x}, \boldsymbol{l}, \boldsymbol{m} \mid \boldsymbol{y}, J) \quad , \tag{7.1}$$

where $\boldsymbol{m}$ ranges among all the possible target segment values for $\boldsymbol{y}$, and $\boldsymbol{l}$ ranges only on those values that are in accordance with $\boldsymbol{m}$ and J.

The complete model in Eq. (7.1) is decomposed as follows

$$p_r(\boldsymbol{x}, \boldsymbol{l}, \boldsymbol{m} \mid \boldsymbol{y}, J) = p_r(\boldsymbol{m} \mid \boldsymbol{y}, J) p_r(\boldsymbol{l} \mid \boldsymbol{m}, \boldsymbol{y}, J) p_r(\boldsymbol{x} \mid \boldsymbol{l}, \boldsymbol{m}, \boldsymbol{y}, J) \quad . \tag{7.2}$$

All the probabilities in Eq. (7.2) are being decomposed left-to-right. We explain into detail the decomposition of the target segment length probability model since the extension of this technique to the source length and the emission probabilities is straightforward and can make the discussion cumbersome.

To simplify notation, we need to give some additional definitions, before decomposing the target length probability $p_r(\boldsymbol{m} \mid \boldsymbol{y}, J)$. Given a target segmentation variable, say $\boldsymbol{m}$, we define its prefix counterpart, $\bar{\boldsymbol{m}}$ as follows

$$\bar{m}_i = \sum_{k=1}^{i} m_k \quad i = 0, 1, \ldots, I \quad . \tag{7.3}$$

Similarly, for the source segmentation variable $\boldsymbol{l}$ we can define its prefix counterpart $\bar{\boldsymbol{l}}$ as follows

$$\bar{l}_i = \sum_{k=1}^{i} l_k \quad i = 0, 1, \ldots, I \quad . \tag{7.4}$$

For instance, in Figure 7.1, the prefix segments lengths are $\bar{\boldsymbol{m}} = \bar{\boldsymbol{m}}_0^5 = (0, 2, 2, 3, 5, 5)$ and $\bar{\boldsymbol{l}} = \bar{\boldsymbol{l}}_0^5 = (0, 2, 2, 3, 4, 4)$, for target and source segmentation variables respectively.

# segments	$\boldsymbol{m}$	$\boldsymbol{y}_1^5$ segments	$\boldsymbol{l}$	$\boldsymbol{x}_1^4$ segments
1	$(5,0,0,0,0)$	$\boldsymbol{y}_1^5$	$(4,0,0,0,0)$	$\boldsymbol{x}_1^4$
2	$(4,0,0,0,1)$	$\boldsymbol{y}_1^4, y_5$	$(3,0,0,0,1)$	$\boldsymbol{x}_1^3, x_4$
			$(2,0,0,0,2)$	$\boldsymbol{x}_1^2, \boldsymbol{x}_2^4$
			$(1,0,0,0,3)$	$x_1, \boldsymbol{x}_2^4$
	$(3,0,0,2,0)$	$\boldsymbol{y}_1^3, \boldsymbol{y}_4^5$	$(3,0,0,1,0)$	$\boldsymbol{x}_1^3, x_4$
			$(2,0,0,2,0)$	$\boldsymbol{x}_1^2, \boldsymbol{x}_2^4$
			$(1,0,0,3,0)$	$x_1, \boldsymbol{x}_2^4$
	$(2,0,3,0,0)$	$\boldsymbol{y}_1^2, \boldsymbol{y}_3^5$	$(3,0,1,0,0)$	$\boldsymbol{x}_1^3, x_4$
			$(2,0,2,0,0)$	$\boldsymbol{x}_1^2, \boldsymbol{x}_2^4$
			$(1,0,3,0,0)$	$x_1, \boldsymbol{x}_2^4$
	$(1,4,0,0,0)$	$y_1, \boldsymbol{y}_2^5$	$(3,1,0,0,0)$	$\boldsymbol{x}_1^3, x_4$
			$(2,2,0,0,0)$	$\boldsymbol{x}_1^2, \boldsymbol{x}_2^4$
			$(1,3,0,0,0)$	$x_1, \boldsymbol{x}_2^4$
3	$(3,0,0,1,1)$	$\boldsymbol{y}_1^3, y_4, y_5$	$(2,0,0,1,1)$	$\boldsymbol{x}_1^2, x_3, x_4$
			$(1,0,0,2,1)$	$x_1, \boldsymbol{x}_2^3, x_4$
			$(1,0,0,1,2)$	$x_1, x_2, \boldsymbol{x}_3^4$
	$(2,0,2,0,1)$	$\boldsymbol{y}_1^2, \boldsymbol{y}_3^4, y_5$	$(2,0,1,0,1)$	$\boldsymbol{x}_1^2, x_3, x_4$
			$(1,0,2,0,1)$	$x_1, \boldsymbol{x}_2^3, x_4$
			$(1,0,1,0,2)$	$x_1, x_2, \boldsymbol{x}_3^4$
	$(2,0,1,2,0)$	$\boldsymbol{y}_1^2, y_3, \boldsymbol{y}_4^5$	$(2,0,1,1,0)$	$\boldsymbol{x}_1^2, x_3, x_4$
			$(1,0,2,1,0)$	$x_1, \boldsymbol{x}_2^3, x_4$
			$(1,0,1,2,0)$	$x_1, x_2, \boldsymbol{x}_3^4$
	$(1,3,0,0,1)$	$y_1, \boldsymbol{y}_2^4, y_5$	$(2,1,0,0,1)$	$\boldsymbol{x}_1^2, x_3, x_4$
			$(1,2,0,0,1)$	$x_1, \boldsymbol{x}_2^3, x_4$
			$(1,1,0,0,2)$	$x_1, x_2, \boldsymbol{x}_3^4$
	$(1,2,0,2,0)$	$y_1, \boldsymbol{y}_2^3, \boldsymbol{y}_4^5$	$(2,1,0,1,0)$	$\boldsymbol{x}_1^2, x_3, x_4$
			$(1,2,0,1,0)$	$x_1, \boldsymbol{x}_2^3, x_4$
			$(1,1,0,2,0)$	$x_1, x_2, \boldsymbol{x}_3^4$
	$(1,1,3,0,0)$	$y_1, y_2, \boldsymbol{y}_3^5$	$(2,1,1,0,0)$	$\boldsymbol{x}_1^2, x_3, x_4$
			$(1,2,1,0,0)$	$x_1, \boldsymbol{x}_2^3, x_4$
			$(1,1,2,0,0)$	$x_1, x_2, \boldsymbol{x}_3^4$
4	$(2,0,1,1,1)$	$\boldsymbol{y}_1^2, y_3, y_4, y_5$	$(1,0,1,1,1)$	x_1, x_2, x_3, x_4
	$(1,2,0,1,1)$	$y_1, \boldsymbol{y}_2^3, y_4, y_5$	$(1,1,0,1,1)$	x_1, x_2, x_3, x_4
	$(1,1,2,0,1)$	$y_1, y_2, \boldsymbol{y}_3^4, y_5$	$(1,1,1,0,1)$	x_1, x_2, x_3, x_4
	$(1,1,1,2,0)$	$y_1, y_2, y_3, \boldsymbol{y}_4^5$	$(1,1,1,1,0)$	x_1, x_2, x_3, x_4
5	$(1,1,1,1,1)$	y_1, y_2, y_3, y_4, y_5	$\emptyset$	$\emptyset$

Table 7.1: A full domain specification for both segmentation variables, $\boldsymbol{m}$ and $\boldsymbol{l}$, in the case of a source sentence of 4 words and a target sentence of 5 words. For better understanding of these variables, the induced segmentation in both source and target sentences is also provided in columns 3 and 5. Although there is a possible segmentation of the target sentence $\boldsymbol{y}$ into 5 segments (last row), it is not the case for the source sentence $\boldsymbol{x}$.

The probability of the target segmentation variable is given by

$$p_r(\boldsymbol{m} \,|\, \boldsymbol{y}, J) = \prod_{i=1}^{I} p_r(m_i \,|\, \boldsymbol{m}_1^{i-1}, \boldsymbol{y}, J) \quad . \tag{7.5}$$

At first stage, we assume that each partial probability in Eq. (7.5) does not depend neither on $\boldsymbol{y}$, nor on both lengths (I and J) and, hence, the probability $p_r(m_i \,|\, \boldsymbol{m}_1^{i-1}, \boldsymbol{y}, J)$ is modelled as follows

$$p_r(m_i \,|\, \boldsymbol{m}_1^{i-1}, \boldsymbol{y}, J) := \begin{cases} \mathrm{p}(m_i) & \bar{m}_{i-1} + 1 = i,\ m_i > 0 \\ 1 & \bar{m}_{i-1} + 1 \neq i,\ m_i = 0 \end{cases} \tag{7.6}$$

Note that the first case in Eq. (7.6) is satisfied by the positions i in which a segment begins, whereas the other case is satisfied by the positions that lay inside a segment (execept for the boundaries).

Finally the segmentation probability can be expressed as follows

$$p_r(\boldsymbol{m} \,|\, \boldsymbol{y}, J) := \prod_{i \in \mathcal{Z}(\boldsymbol{m})} 1 \prod_{i \notin \mathcal{Z}(\boldsymbol{m})} \mathrm{p}(m_i) \tag{7.7}$$

where $\mathcal{Z}(\boldsymbol{m})$ or simply $\mathcal{Z}$ stands for the set of positions i for which m_i is 0. For instance, in the example in Figure 7.1, $\mathcal{Z}$ is instanced to $\mathcal{Z}(\boldsymbol{m}) = \{2, 5\}$.

Provided that one of the two products in Eq. (7.7) simplifies to 1, the segmentation probability is expressed as

$$p_r(\boldsymbol{m} \,|\, \boldsymbol{y}, J) := \prod_{i \notin \mathcal{Z}} \mathrm{p}(m_i) \quad . \tag{7.8}$$

Since explicitly showing these details makes the discourse to be awkward, we will henceforth omit them abusing of notation whenever it does not entail confusion. Hence, we will use equations similar to the following

$$p_r(\boldsymbol{m} \,|\, \boldsymbol{y}, J) := \prod_{t} \mathrm{p}(m_t) \quad , \tag{7.9}$$

where we have explicitly omitted that $t \in \mathcal{Z}$, but we keep the subindex t instead of i for subtly highlighting this modelling process. Note that decomposition is similar to the usual state probability decomposition used in hidden semi-Markov models (see Section 1.1.6 Chapter 1).

Similarly to the target segmentation modelling, the source segmentation yields the following equation

$$p_r(\boldsymbol{l} \,|\, \boldsymbol{m}, \boldsymbol{y}, J) := \prod_{t} \mathrm{p}(l_t \,|\, m_t) \quad , \tag{7.10}$$

where we have assumed that the t-th source segment length l_t depends only on the corresponding t-th target segment length m_t; and hence, it is independent of the remaining target segment lengths as well as independent of the previous $t-1$ source segment lengths $\boldsymbol{l}_1^{t-1}$.

Finally, knowing the segmentation variables, the emission probability is also decomposed left-to-right yielding

$$p_r(\boldsymbol{x} \,|\, \boldsymbol{l}, \boldsymbol{m}, \boldsymbol{y}, J) := \prod_{t} \mathrm{p}(\boldsymbol{x}(t) \,|\, \boldsymbol{y}(t)) \quad , \tag{7.11}$$

where we have assumed that the emission of the source phrase $\boldsymbol{x}(t)$ only depends on $\boldsymbol{y}(t)$; and where $\boldsymbol{x}(t)$ stands for $\boldsymbol{x}_{\bar{l}_{t-1}+1}^{\bar{l}_t}$ and $\boldsymbol{y}(t)$ for $\boldsymbol{y}_t^{t+m_t-1}$; i.e., the t-th "emitted" source phrase and its respective t-th target phrase. Note that $\bar{l}_t$ is equal to $\bar{l}_{t-1} + l_t$ provided that position t is a starting position for a target segment.

Summarising, the proposed (complete) conditional translation model is defined as follows

$$p_r(\boldsymbol{x}, \boldsymbol{l}, \boldsymbol{m} \,|\, \boldsymbol{y}, J) := \prod_t \mathrm{p}(m_t)\, \mathrm{p}(l_t \,|\, m_t)\, \mathrm{p}(\boldsymbol{x}(t) \,|\, \boldsymbol{y}(t)) \quad , \tag{7.12}$$

and, hence, the incomplete model introduced in Eq. (7.1) is parametrised as follows

$$\mathrm{p}_{\boldsymbol{\theta}}(\boldsymbol{x} \,|\, \boldsymbol{y}, J) := \sum_{\boldsymbol{m}} \sum_{\boldsymbol{l}} \mathrm{p}_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{l}, \boldsymbol{m} \,|\, \boldsymbol{y}) \quad , \tag{7.13}$$

where $\mathrm{p}_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{l}, \boldsymbol{m} \,|\, \boldsymbol{y})$ is given by the following expression

$$\mathrm{p}_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{l}, \boldsymbol{m} \,|\, \boldsymbol{y}) = \prod_t \mathrm{p}(m_t)\, \mathrm{p}(l_t \,|\, m_t)\, \mathrm{p}(\boldsymbol{x}(t) \,|\, \boldsymbol{y}(t)) \quad , \tag{7.14}$$

with the following parameter set $\boldsymbol{\theta}$

$$\boldsymbol{\theta} = \{\mathrm{p}(m), \mathrm{p}(l \,|\, m), \mathrm{p}(\boldsymbol{u} \,|\, \boldsymbol{v}) \,|\, \forall l > 0, \forall m > 0, \forall \boldsymbol{u} \in \boldsymbol{X}^\star, \forall \boldsymbol{u} \in \boldsymbol{Y}^\star\} \quad . \tag{7.15}$$

In the *phrase-based hidden semi-Markov model (PBHSMM)* defined in this section, we can understand each target phrase $\boldsymbol{y}(t)$ as the "state" of a HSMM in which the source phrase $\boldsymbol{x}(t)$ is emitted. Note that the output sentence probability, $\mathrm{p}(\boldsymbol{y})$, plays the role of the state sequence probability in a HSMM. Such probability $\mathrm{p}(\boldsymbol{y})$, is better modelled by a specific language model such as a n-gram, due to the nature of such variable. Obviously this is not a pure HSMM in which we have a latent state variable. The omission of this latent variable is more an assumption than a requirement. Recall that in Figure 7.1 we have depicted each bilingual phrase pair being emitted by a *concept* which could represent a latent state. We have proposed this extension in the conclusion section as a future research line.

Since the model assumes that the segmentation variables are not given in the training data, some approximate inference algorithm such as the EM (see Section 1.1.4 Chapter 1) is needed. In the following section, the standard recurrences needed in HSMM training are adapted to the proposed translation model.

7.3 Recurrences

As it is common in hidden Markov models and specifically in hidden semi-Markov models, some helpful recurrences are defined in order to efficiently obtain the answer to some common questions. We focus, in this section, on 3 selected questions among many others:

- Which is the probability for a given bilingual pair $(\boldsymbol{x}, \boldsymbol{y})$?
- Which is the best segmentation for a given bilingual pair $(\boldsymbol{x}, \boldsymbol{y})$?
- Which is the best parameter set $\boldsymbol{\theta}$ given a training set $\{(\boldsymbol{x}_n, \boldsymbol{y}_n)\}_{n=1}^N$?

7.3.1 Forward recurrence

The forward recurrence α_{tl} is defined as the following prefix probability

$$\alpha_{tl} = \alpha_{tl}(\boldsymbol{x}, \boldsymbol{y}) := \mathrm{p}_{\boldsymbol{\theta}}(\boldsymbol{x}_1^l, \bar{l}_t = l, \bar{m}_t = t \,|\, \boldsymbol{y}) \quad , \tag{7.16}$$

where the events $\bar{l}_t = l$ and $\bar{m}_t = t$, imply that a source (or target) phrase ends at position l (or t) in the input (or output, respectively).

The prefix probability in Eq. (7.16) is recursively computed as follows

$$\alpha_{tl} = \begin{cases} 1 & t = 0, l = 0 \\ \sum_{t'=0}^{t-1} \sum_{l'=0}^{l-1} \alpha_{t'l'} \, \mathrm{p}(t'-t) \, \mathrm{p}(l'-l \,|\, t'-t) \, \mathrm{p}(\boldsymbol{x}_{l'+1}^{l} | \boldsymbol{y}_{t'+1}^{t}) & \begin{matrix} 0 < t \leq I \\ 0 < l \leq J \end{matrix} \\ 0 & \text{otherwise} \end{cases} \tag{7.17}$$

In order to compute the forward recurrence in Eq. (7.16), a matrix of $O(IJ)$ elements is needed. The computational complexity required to fill such a matrix is $O(I^2J^2)$. However, if the phrases are constrained to a maximum source and target phrase length, L and M respectively, then the complexity is reduced to $O(IJML)$.

Furthermore, a detailed analysis of the forward algorithm unveils that not all the elements of α_{tl} must be computed. The elements excluded do not verify one of the following requirements: that both source and target sentences must be segmented in the same amount of phrases; or that both source and target phrases must be smaller than L and M respectively. For instance, in Figure 7.2, we have highlighted which elements must be computed for two sentences of length 20 and 22. The remaining values are useless, and we should save the time needed to compute them. Note that the longer the sentences are, the more effective this optimisation is. Specifically, in the previous example the number of elements to compute approximately account for the 50% of the total values. This reduces the computational complexity in a ratio of 2. Additionally, it is possible to add some heuristics to the process such as beam pruning [Wang and Waibel, 1997, 1998].

Finally, the answer to the first question, i.e., how to compute the probability of a given pair, is given by means of the forward recurrence as follows

$$\mathrm{p}_{\boldsymbol{\theta}}(\boldsymbol{x} \,|\, \boldsymbol{y}) = \alpha_{IJ} \quad . \tag{7.18}$$

7.3.2 Backward recurrence

The backward recurrence β_{tl} is defined as the suffix probability

$$\beta_{tl} = \beta_{tl}(\boldsymbol{x}, \boldsymbol{y}) = \mathrm{p}_{\boldsymbol{\theta}}(\boldsymbol{x}_{l+1}^{J} | \bar{l}_t = l, \bar{m}_t = t, \boldsymbol{y}) \quad , \tag{7.19}$$

where $\bar{l}_t = l$ and $\bar{m}_t = t$, implies that a source (or target) phrase ended/started at position l (or t) of the input (or output, respectively).

The suffix probability in Eq. (7.19) is recursively computed as follows

$$\beta_{tl} = \begin{cases} 1 & t = I, l = J \\ \sum_{t'=t+1}^{I} \sum_{l'=l+1}^{J} \beta_{t'l'} \, \mathrm{p}(t'-t) \, \mathrm{p}(l'-l \,|\, t'-t) \, \mathrm{p}(\boldsymbol{x}_{l+1}^{l'} \,|\, \boldsymbol{y}_{t+1}^{t'}) & \begin{matrix} 0 \leq t < I \\ 0 \leq l < J \end{matrix} \\ 0 & \text{otherwise} \end{cases} \tag{7.20}$$

The computational complexity, both in terms of memory and time, of the backward recurrence are the same of that of the forward recurrence. Furthermore, the same optimisations applied to the forward recursion are also eligible to be applied in the backward recursion.

Analogously to the forward recursion, the probability of a given bilingual pair of sentences $(\boldsymbol{x}, \boldsymbol{y})$ can be efficiently computed as follows

$$\mathrm{p}_{\boldsymbol{\theta}}(\boldsymbol{x} \,|\, \boldsymbol{y}) = \beta_{00} \quad . \tag{7.21}$$

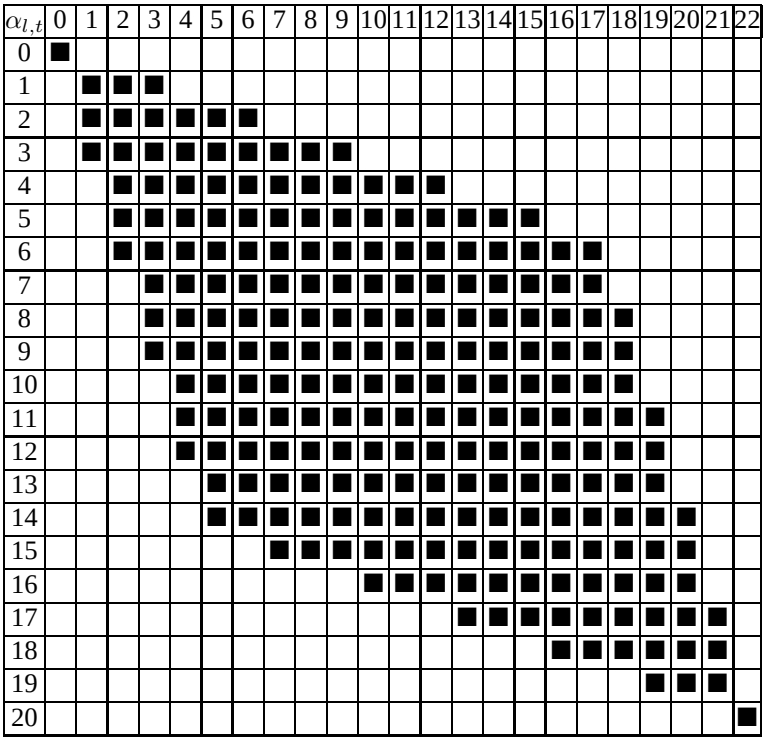


Figure 7.2: The relevant values that should be computed for the forward recurrence in the case of a source sentence of 20 words and a target sentence of 22 words. The maximum phrase length is assumed to be 3 for both source and target phrases. Finally, the black squares (■) stand for the α_{tl} values that must be computed while the remaining points are not needed.

7.3.3 Viterbi recursion

The second question proposed at the beginning of the section is stated as finding the best segmentation for a given bilingual pair $(\boldsymbol{x}, \boldsymbol{y})$, i.e.

$$(\hat{\boldsymbol{l}}, \hat{\boldsymbol{m}}) = \arg\max_{\boldsymbol{l},\boldsymbol{m}} \{\mathrm{p}_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{l}, \boldsymbol{m} \,|\, \boldsymbol{y}, J)\} \quad . \tag{7.22}$$

In order to efficiently answer this question, we define the Viterbi recursion as follows

$$\delta_{tl} = \max_{\substack{T, \boldsymbol{l}_1^T, \boldsymbol{m}_1^T \\ l_T = l, m_T = t}} \{\mathrm{p}_{\boldsymbol{\theta}}(\boldsymbol{x}_1^l, \boldsymbol{l}_1^T, \boldsymbol{m}_1^T \,|\, \boldsymbol{y}, J)\} \quad , \tag{7.23}$$

where note that $\boldsymbol{l}, \boldsymbol{m}$ are required to end at positions l and t respectively.

The Viterbi recursion in Eq. (7.23) is efficiently computed by the following recurrence

$$\delta_{tl} = \begin{cases} 1 & t = 0, l = 0 \\ \max\limits_{t', l'} \{\delta_{t'l'} \, \mathrm{p}(t' - t) \, \mathrm{p}(l' - l \,|\, t' - t) \, \mathrm{p}(\boldsymbol{x}_{l'+1}^{l} | \boldsymbol{y}_{t'+1}^{t})\} & \begin{matrix} 0 < t \leq I \\ 0 < l \leq J \end{matrix} \\ 0 & \text{otherwise} \end{cases} \tag{7.24}$$

A traceback of the decisions made to compute δ_{IJ} provides the maximum segmentation $\hat{\boldsymbol{m}}$ and $\hat{\boldsymbol{l}}$, i.e. the solution to the Eq. (7.22).

The Viterbi recursion shares the computational requirements of the forward and backward recurrence. Furthermore, the same optimisations applied to the forward and backward recursion should also be adapted for computing the Viterbi recursion.

7.4 Training

Since the proposed PBHSMM assumes that the segment length variables are not given in the training data, an approximate inference algorithm such as the EM is needed. We give here the description of the common training algorithms with respect to a collection of training translation pairs $\{(\boldsymbol{x}_1, \boldsymbol{y}_1), \ldots, (\boldsymbol{x}_N, \boldsymbol{y}_N)\}$, that is to say: the Baum-Welch algorithm [Rabiner, 1989], and the Viterbi algorithm [Rabiner, 1989]. Both algorithms are instantiations of the EM algorithm as discussed in Section 1.1.4 Chapter 1.

The log-likelihood function as a function of the parameters $\boldsymbol{\theta}$ is

$$\begin{aligned}\mathrm{LL}(\boldsymbol{\theta}) &= \sum_n \log p_r(\boldsymbol{x}_n \,|\, \boldsymbol{y}_n) \\ &= \sum_n \log \sum_{\boldsymbol{l},\boldsymbol{m}} \mathrm{p}_{\boldsymbol{\theta}}(\boldsymbol{x}_n, \boldsymbol{l}, \boldsymbol{m} \,|\, \boldsymbol{y}_n) \\ &= \sum_n \log \sum_{\boldsymbol{l},\boldsymbol{m}} \prod_t \mathrm{p}(m_t)\, \mathrm{p}(l_t \,|\, m_t)\, \mathrm{p}(\boldsymbol{x}_n(t) \,|\, \boldsymbol{y}_n(t)) \quad . \end{aligned} \tag{7.25}$$

However, recall that the EM algorithm maximises a lower bound to the log-likelihood function, $\mathrm{LL}(\cdot)$, by iteratively maximising a variational function $\mathcal{L}(\mathrm{q}, \boldsymbol{\theta})$ through the application of two basic steps in each iteration: the E(xpectation) step and the M(aximisation) step.

7.4.1 Fractional counts

Using the previously defined forward and backward recursions, we can compute the probability of using the source phrase $\boldsymbol{x}_{l+1}^{l'}$ and the target phrase $\boldsymbol{y}_{t+1}^{t'}$ when segmenting a given sample $(\boldsymbol{x}, \boldsymbol{y})$,

$$\begin{aligned}\gamma_{tlt'l'} &= \mathrm{p}_{\boldsymbol{\theta}}(\boldsymbol{x}_{l+1}^{l'}, \bar{l}_t = l', \bar{l}_{t-1} = l, \bar{m}_t = t', \bar{m}_{t-1} = t \,|\, \boldsymbol{y}) \\ &= \frac{\alpha_{tl}\, \mathrm{p}(t'-t)\, \mathrm{p}(l'-l \,|\, t'-t)\, \mathrm{p}(\boldsymbol{x}_{l+1}^{l'} \,|\, \boldsymbol{y}_{t+1}^{t'}) \beta_{t'l'}}{\mathrm{p}_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{y})} \quad , \end{aligned} \tag{7.26}$$

with $l < l'$ and $t < t'$. This fractional counts are very useful for training the parameters of the model in the following two Sections 7.4.2 and 7.4.3.

7.4.2 Baum-Welch training

Let $\boldsymbol{\theta}^{(k)}$ be a guess of the optimal parameters obtained from previous iterations. In this case, the E step requires the computation, for each pair $(\boldsymbol{x}_n, \boldsymbol{y}_n)$, of the sample versions of (7.16), say $\alpha_{ntl}^{(k)}$, and (7.19), say $\beta_{ntl}^{(k)}$, as well as the fractional counts per sample, say $\gamma_{ntlt'l'}^{(k)}$. These sufficient statistics are computed using the parameters obtained from previous iteration, $\boldsymbol{\theta}^{(k)}$. Recall that these sufficient statistics summarise the optimal function $\mathrm{q}^{(k)}$ obtained in the E-step, by storing the relevant information; in other words, computing these recurrences is equivalent to compute $\mathrm{q}^{(k)}$.

Afterwards, in the M step, a new set of parameters $\boldsymbol{\theta}^{(k+1)}$ is estimated from the recurrences computed in the E step. The new set of the parameters includes the three probabilities of the model: the

phrase dictionary $\mathrm{p}^{(k+1)}(\boldsymbol{u} \,|\, \boldsymbol{v})$, the target length probability $\mathrm{p}^{(k+1)}(m)$ and the source length conditional probability $\mathrm{p}^{(k+1)}(m \,|\, l)$.

The phrase dictionary is estimated as follow

$$\mathrm{p}^{(k+1)}(\boldsymbol{u} \,|\, \boldsymbol{v}) = \frac{N^{(k)}(\boldsymbol{u}, \boldsymbol{v})}{\sum_{\boldsymbol{u}'} N^{(k)}(\boldsymbol{u}', \boldsymbol{v})} \quad , \tag{7.27}$$

with

$$N^{(k)}(\boldsymbol{u}, \boldsymbol{v}) = \sum_{n} \sum_{l,l'} \sum_{t,t'} \gamma^{(k)}_{ntlt'l'} \delta(\boldsymbol{x}_n(l,l'), \boldsymbol{u}) \delta(\boldsymbol{y}_n(t,t'), \boldsymbol{v}) \quad , \tag{7.28}$$

where we use the notation $\boldsymbol{z}(l,l')$ to refer to $\boldsymbol{z}_{l+1}^{l'}$, and where $\delta(a,b)$ stands for the Kronecker delta function which evaluates to 1 if $a = b$ and 0 otherwise.

For the target phrase length probabilities, we obtain the following re-estimation equation

$$\mathrm{p}^{(k+1)}(m) = \frac{N^{(k)}(m)}{\sum_{m'} N^{(k)}(m')} \quad , \tag{7.29}$$

with

$$N^{(k)}(m) = \sum_{n} \sum_{l,l'} \sum_{t} \gamma^{(k)}_{nt,l,(t+m),l'} \quad , \tag{7.30}$$

where m is a target phrase length.

Finally, the source phrase length probabilities are re-estimated as follows

$$\mathrm{p}^{(k+1)}(l \,|\, m) = \frac{N^{(k)}(l,m)}{\sum_{l'} N^{(k)}(l',m)} \quad , \tag{7.31}$$

with

$$N^{(k)}(l,m) = \sum_{n} \sum_{l'} \sum_{t} \gamma^{(k)}_{t,l,(t+m),(l'+l)} \quad , \tag{7.32}$$

where l denotes a source phrase length, and m a target phrase length.

If an initial parameter set is given $\boldsymbol{\theta}^{(0)}$; we can iteratively refine our initial guess by alternatively applying the E-step and the M-step until convergence. The convergence criteria is given by either reaching a maximum number of iterations or increasing the log-likelihood under a given threshold. Since the log-likelihood function in Eq. (7.25) is not convex, the Baum-Welch training only provides a local optimum after convergence. Therefore, the reader should bare in mind that a wrong initial guess $\boldsymbol{\theta}^{(0)}$ can ruin the system performance.

7.4.3 Viterbi training

Let $\boldsymbol{\theta}^{(k)}$ be a guess of the optimal parameters obtained from previous iterations. In this case, the E-step requires the computation of the maximum segmentation $(\boldsymbol{l}_n^{(k)}, \boldsymbol{m}_n^{(k)})$, as defined in Eq. (7.22), for each pair $(\boldsymbol{x}_n, \boldsymbol{y}_n)$. In order to efficiently compute the optimal segmentation, the Viterbi recursion in Eq. (7.24) is computed for each sample.

Afterwards, in the M-step finding the parameter set $\boldsymbol{\theta}^{(k)}$ that maximises $\mathcal{L}(\mathrm{q}, \boldsymbol{\theta})$ is equivalent to find the parameter set that maximises the following function

$$Q(\boldsymbol{\theta} \,|\, \boldsymbol{\theta}^{(k)}) = \sum_{n} \sum_{t} \log \mathrm{p}(\boldsymbol{m}_{nt}^{(k)}) + \log \mathrm{p}(\boldsymbol{l}_{nt}^{(k)} \,|\, \boldsymbol{m}_{nt}^{(k)}) + \log \mathrm{p}(\boldsymbol{x}_n(t) \,|\, \boldsymbol{y}_n(t)) \quad . \tag{7.33}$$

Rearranging terms in Eq. (7.33), we obtain

$$Q(\boldsymbol{\theta} \,|\, \boldsymbol{\theta}^{(k)}) = \sum_{m} M^{(k)}(m) \log \mathrm{p}(m) + \sum_{m} \sum_{l} M^{(k)}(l, m) \log \mathrm{p}(l \,|\, m) + \sum_{\boldsymbol{u}} \sum_{\boldsymbol{v}} M^{(k)}(\boldsymbol{u}, \boldsymbol{v}) \log \mathrm{p}(\boldsymbol{u} \,|\, \boldsymbol{v}) \quad , \tag{7.34}$$

where $M^{(k)}(e)$ stands for the number of times the event e has occurred in the sample completed with the length variables, i.e., $\{(\boldsymbol{x}_n, \boldsymbol{y}_n, \boldsymbol{l}_n^{(k)}, \boldsymbol{m}_n^{(k)})\}_{n=1}^{N}$. Specifically, $M^{(k)}(m)$ is defined as follows

$$M^{(k)}(m) = \sum_{n} \sum_{t} \delta(m_{nt}^{(k)}, m) \quad , \tag{7.35}$$

and $M^{(k)}(l, m)$ is given by

$$M^{(k)}(l, m) = \sum_{n} \sum_{t} \delta(m_{nt}^{(k)}, m) \delta(l_{nt}^{(k)}, l) \quad , \tag{7.36}$$

and finally the phrase counts are defined as

$$M^{(k)}(\boldsymbol{u}, \boldsymbol{v}) = \sum_{n} \sum_{t} \delta(\boldsymbol{x}_n^{(k)}(t), \boldsymbol{u}) \delta(\boldsymbol{y}_n^{(k)}(t), \boldsymbol{v}) \quad , \tag{7.37}$$

where $\boldsymbol{x}_n^{(k)}(t)$ and $\boldsymbol{y}_n^{(k)}(t)$ stand for the t-th source and target phrase induced by $\boldsymbol{l}_n^{(k)}$ and $\boldsymbol{m}_n^{(k)}$, respectively; and where the expression $\delta(a, b)$ is the Kronecker's delta function.

The Viterbi training described here is also an iterative training process, since it is another instantiation of the EM algorithm (see Section 1.1.4 Chapter 1). Therefore, if an initial parameter set is given $\boldsymbol{\theta}^{(0)}$; we can iteratively refine our initial guess by alternatively applying the E-step and the M-step until convergence. Similarly to the Baum-Welch training, the Viterbi training only provides a local optimum after converge. However, since the Viterbi algorithm constraints the family of functions in the E-step, i.e. $\mathrm{q}^{(k)}(\boldsymbol{l}, \boldsymbol{m}) = \delta(\boldsymbol{l}, \boldsymbol{l}_n^{(k)}) \delta(\boldsymbol{m}, \boldsymbol{m}_n^{(k)})$, the optimal parameter set obtained after a Viterbi training is typically worse than that of the Baum-Welch training [Rabiner, 1989].

The main advantage of the Viterbi training with respect to Baum-Welch training is that we only need to compute one recurrence, the Viterbi recurrence, in contrast to the two recurrences, the forward and backward recurrences, needed in the Baum-Welch training. Additionally, since the Viterbi algorithm only takes into account the most probable segmentation in each iteration, a given outcome has effect on less parameters, speeding up the algorithm. Therefore, the Viterbi training is at least twice times faster than the Baum-Welch training.

7.4.4 The model smoothing

A well-known drawback of the EM algorithm is that it tends to overfit the models. Moreover, the HMM-based model discussed in Chapter 6, shown severe overfitting problems. In order to alleviate these problems, we smoothed the phrase table $\tilde{\mathrm{p}}(\boldsymbol{u} \,|\, \boldsymbol{v})$ with a IBM model 1 [Brown et al., 1993] as follows

$$\tilde{\mathrm{p}}(\boldsymbol{u} \,|\, \boldsymbol{v}) := (1 - \epsilon)\, \hat{\mathrm{p}}(\boldsymbol{u} \,|\, \boldsymbol{v}) + \epsilon\, \mathrm{p}_{IBM1}(\boldsymbol{u} \,|\, \boldsymbol{v}) \quad , \tag{7.38}$$

where $\tilde{\mathrm{p}}(\boldsymbol{u} \,|\, \boldsymbol{v})$ stands for the smoothed phrase table, $\hat{\mathrm{p}}(\boldsymbol{u} \,|\, \boldsymbol{v})$ stands for the optimal phrase table (not smoothed) obtained after the EM training and $\mathrm{p}_{IBM1}(\boldsymbol{u} \,|\, \boldsymbol{v})$ stands for the probability of the IBM model 1 *without the null word*, i.e.,

$$\mathrm{p}_{IBM1}(\boldsymbol{u} \,/\, \boldsymbol{v}, |\boldsymbol{u}|, |\boldsymbol{v}|) := \sum_{i=1}^{|\boldsymbol{v}|} \prod_{j}^{|\boldsymbol{u}|} \frac{1}{|\boldsymbol{v}|} \mathrm{p}(u_j \,|\, v_i) \quad . \tag{7.39}$$

The IBM model 1 performs accurately when deciding whether a set of words contains the translations of another set of words or not; even though it is unable to learn the order of such words.

7.5 Decoding Recurrence

In this section, we explain the recurrence used to perform the model search or decoding process inside a system based on the fundamental equation of machine translation introduced in Section 1.3 (Eq. (1.102)). The search problem in such a system is stated as the problem of finding the maximum probable target sentence as follows

$$\hat{\boldsymbol{y}} = \arg\max_{\boldsymbol{y}} \{ \mathrm{p}_{\boldsymbol{\theta}}(\boldsymbol{x} \,|\, \boldsymbol{y})\, \mathrm{p}(\boldsymbol{y}) \} \quad , \tag{7.40}$$

where the $\mathrm{p}_{\boldsymbol{\theta}}(\boldsymbol{x} \,|\, \boldsymbol{y})$ is modelled according to the PBHSMM proposed in Eq. (7.13), i.e.,

$$\hat{\boldsymbol{y}} = \arg\max_{\boldsymbol{y}} \left\{ \sum_{\boldsymbol{m}} \sum_{\boldsymbol{l}} \mathrm{p}_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{l}, \boldsymbol{m} \,|\, \boldsymbol{y})\, \mathrm{p}(\boldsymbol{y}) \right\} \quad . \tag{7.41}$$

In order to cope with Eq. (7.41) a Viterbi-like approximation is taken for all the sums. In this way the search problem is reduced to

$$\hat{\boldsymbol{y}} = \arg\max_{\boldsymbol{y}} \left\{ \max_{\boldsymbol{m}, \boldsymbol{l}} \{ \mathrm{p}_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{l}, \boldsymbol{m} \,|\, \boldsymbol{y})\, \mathrm{p}(\boldsymbol{y}) \} \right\} \quad . \tag{7.42}$$

For sake of simplicity, we focus on finding the probability of the maximum probable translation $\hat{\boldsymbol{y}}$, i.e.,

$$\mathrm{p}_{\boldsymbol{\theta}}(\boldsymbol{x} \,|\, \hat{\boldsymbol{y}}) = \max_{\boldsymbol{y}, \boldsymbol{l}, \boldsymbol{m}} \{ \mathrm{p}_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{l}, \boldsymbol{m} \,|\, \boldsymbol{y})\, \mathrm{p}(\boldsymbol{y}) \} \quad . \tag{7.43}$$

Provided that we only use n-gram language models, we assume that the language model probability for a given target phrase $\mathrm{p}(\boldsymbol{v} \,|\, \boldsymbol{y}_1^m)$ only depends on the $(n-1)$-most recent words, i.e.

$$\mathrm{p}(\boldsymbol{v} \,|\, \boldsymbol{y}_1^m) := \mathrm{p}(\boldsymbol{v} \,|\, \underset{n-1}{\mathrm{suf}}(\boldsymbol{y}_1^m)) \quad , \tag{7.44}$$

where $\mathrm{suf}_{n-1}(\cdots)$ stands for the $(n-1)$-most recent words.

In order to perform the maximisation in Eq. (7.43) we define a Viterbi-like decoding recurrence as follows

$$\sigma_{l,\boldsymbol{v}}(\boldsymbol{x}) = \max_{\substack{t, \boldsymbol{l}_1^t, \boldsymbol{m}_1^t, \boldsymbol{y}_1^{m_t} \\ l_t = l, \mathrm{suf}_{|\boldsymbol{v}|}(\boldsymbol{y}_1^{m_t}) = \boldsymbol{v}}} \left\{ \mathrm{p}_{\boldsymbol{\theta}}(\boldsymbol{x}_1^l, \boldsymbol{l}_1^t, \boldsymbol{m}_1^t \,|\, \boldsymbol{y}_1^{m_t})\, \mathrm{p}(\boldsymbol{y}_1^{m_t}) \right\} \quad , \tag{7.45}$$

where by $\mathrm{suf}_{|\boldsymbol{v}|}(\boldsymbol{y}_1^{m_t}) = \boldsymbol{v}$ we denote that the suffix of $\boldsymbol{y}_1^{m_t}$ must be equal to $\boldsymbol{v}$. We further assume that if $\boldsymbol{v} = \star$, then this constraint is ignored, i.e.,

$$\sigma_{l,\star}(\boldsymbol{x}) = \sigma_l(\boldsymbol{x}) = \max_{\substack{t, \boldsymbol{l}_1^t, \boldsymbol{m}_1^t, \boldsymbol{y}_1^{m_t} \\ l_t = l}} \left\{ \mathrm{p}_{\boldsymbol{\theta}}(\boldsymbol{x}_1^l, \boldsymbol{l}_1^t, \boldsymbol{m}_1^t \,|\, \boldsymbol{y}_1^{m_t})\, \mathrm{p}(\boldsymbol{y}_1^{m_t}) \right\} \quad . \tag{7.46}$$

The search recurrence $\sigma_{l,\boldsymbol{v}}$ can be recursively expressed in terms of a simpler case of itself as follows

$$\sigma_{l,\boldsymbol{v}}(\boldsymbol{x}) = \max_{\substack{l', \boldsymbol{v}', \boldsymbol{h} \\ \mathrm{suf}_{|\boldsymbol{v}|}(\boldsymbol{v}'\boldsymbol{h}) = \boldsymbol{v} \\ l' < l, |\boldsymbol{v}'\boldsymbol{h}| \geq |\boldsymbol{v}|}} \left\{ \mathrm{p}(l)\, \mathrm{p}(|\boldsymbol{v}'| \,/\, l)\, \mathrm{p}(\boldsymbol{x}_{l'+1}^l \,|\, \boldsymbol{v}')\, \mathrm{p}(\boldsymbol{v}' \,|\, \boldsymbol{h}) \sigma_{l',\boldsymbol{h}}(\boldsymbol{x}) \right\} \quad , \tag{7.47}$$

where note that $\boldsymbol{v}$ is split into two parts $\boldsymbol{v}'$ and $\boldsymbol{h}$. The first part is used in the translation of the source phrase whereas the second part $\boldsymbol{h}$, is used as the prefix of the language model.

Finally, the solution to the search problem in Eq. (7.43) is computed using the search recurrence as follows

$$\mathrm{p}_{\boldsymbol{\theta}}(\boldsymbol{x} \,|\, \hat{\boldsymbol{y}}) = \sigma_{J,\star} \quad . \tag{7.48}$$

Note that once it is known how to compute $\mathrm{p}_{\boldsymbol{\theta}}(\boldsymbol{x} \,|\, \hat{\boldsymbol{y}})$, the optimal $\hat{\boldsymbol{y}}$ is obtained by tracing back the decisions made during its computation.

However, although the recursion $\sigma_{l,\boldsymbol{v}}$ speeds up the search problem, it is still a hard problem. For this reason, we still need to perform an approximate decoding in which we use a maximum number of hypothesis for each source position l, say M, and also a beam pruning [Wang and Waibel, 1997, 1998]. That is to say, instead of using Eq. (7.47), we use the following approximated version

$$\sigma^{\star}_{l,\boldsymbol{v}}(\boldsymbol{x}) = \mathop{\overset{\star}{\max}}_{\substack{l',\boldsymbol{v}',\boldsymbol{h} \\ \mathrm{suf}_{|\boldsymbol{v}|}(\boldsymbol{v}'\boldsymbol{h})=\boldsymbol{v}, |\boldsymbol{v}'\boldsymbol{h}|\geq|\boldsymbol{v}|}} \Big\{ \mathrm{p}(l)\, \mathrm{p}(|\boldsymbol{v}'| \,|\, l)\, \mathrm{p}(\boldsymbol{x}_{l'+1}^{l} \,|\, \boldsymbol{v}')\, \mathrm{p}(\boldsymbol{v}' \,|\, \boldsymbol{h}) \sigma^{\star}_{l',\boldsymbol{h}}(\boldsymbol{x}) \Big\} \quad , \tag{7.49}$$

where $\overset{\star}{\max}$ stands for an approximate version of $\max$ where we have applied several heuristics, such as beam search or histogram pruning.

In the experimental section, we have also used the proposed model inside a log-linear loss function (see Section 1.3 of Section 4.3.3 for further details). For this aim, we have used the Moses system [Koehn et al., 2007], and added our model as a feature inside its search. Should the reader be interested in the details of this search, please refer to Koehn et al. [2007].

7.6 Experiments

We have carried out two types of experiments. The first set of experiments [Andrés-Ferrer and Juan, 2009] were designed to analyse the properties of the proposed model when used in a classical phrase-based model that is based on the fundamental equation of statistical machine translation defined in Eq. (1.102). The second set of experiments were designed to analyse the behaviour of the improvements obtained in the first experiment when passed as a feature in a log-linear model based on Eq. (1.103). To evaluate the quality of the translations, we used two error measures: bilingual evaluation understudy (BLEU) [Papineni et al., 2001], and translation edit rate (TER) [Snover et al., 2006].

7.6.1 Classical phrase-based models.

For the first set of experiments, we tested our model in two corpora: the Europarl-10 and the Europarl-20. The former comprises all the sentences from the English-to-Spanish part of Europarl-v3 [Koehn, 2005] with length equal to or less than 10. The latter is made up of all the English-to-Spanish Europarl-v3 sentences with length equal to or less than 20. For both corpora we randomly selected $5\,000$ sentences to test the algorithms. However, since the Europarl-10 has several repeated sentences, we avoided repeated sentences in the test. Note that since we wanted to perform detailed experimentation, we constrained the training length because of the time requirement for training the proposed PBHSMM. Table 7.2 shows some basic statistics of the training part for both corpora; Table 7.3 summarises some statistics from the testing part.

All the experiments were carried out using a 4-gram language model computed with the standard tool SRILM [Stolcke, 2002] and a modified Kneser-Ney smoothing. We used two systems: the proposed PBHSMM with the search algorithm depicted in Section 7.5; and the Moses system [Koehn et al., 2007] but constraining the model to a classical SMT system based on Eq.(1.102) in Chapter 1 (a phrase-based inverse model and a n-gram language model). We used this constrained version of Moses instead of the

Training	Europarl-10		Europarl-20	
Language	En	Sp	En	Sp
sentences	$76,996$		$306,897$	
avg. length	7.01	7.0	12.6	12.7
running words	546K	540K	3.86M	3.91M
voc. size	15.5K	22.1K	37.1K	57.8K

Table 7.2: Basic statistics of the training sets.

Test	Europarl-10		Europarl-20	
Language	En	Sp	En	Sp
sentences	$5,000$		$5,000$	
avg. length	7.2	7.0	12.6	12.9
running words	35.8K	35.2K	63.0K	63.8K
ppl (4-gram)	48.3	56.9	62.3	69.1

Table 7.3: Basic statistics of the test sets.

full log-linear model in order to define a fair translation baseline. However, in the following subsection, we will compare both systems inside a log-linear model.

The proposed training algorithms require an initial guess. To this aim, we computed the IBM word alignment models with GIZA++ [Och and Ney, 2003], for both translation directions. We computed the symmetrisation heuristic [Och and Ney, 2004] and extracted all the *consistent* phrases [Och and Ney, 2004]. Afterwards, we computed our initial guess by counting the occurrences of each bilingual phrase and then normalising the counts. Instead of using the Moses system to perform this initialisation task, we have implemented our own version of this process.

Since the training algorithm highly depends on the maximum phrase length, for most of the experimentations we limited it to 4 words. Table 7.4 summarises the results obtained for both translation directions with the Europarl-10 corpus. Surprisingly, Viterbi training obtains almost the same results as the Baum-Welch training; this is probably because most of the sentences accumulate all the probability mass in just one possible segmentation. Maybe that is why our algorithm is not able to obtain a large improvement with respect to the initialisation. Note that since the proposed system and the Moses system use different phrase-tables, these two numbers should not be compared. Therefore, the Moses baseline is only given as a reference and not as a system to improve. The important question is whether the model produces an improvement with respect to the initialisation, i.e., the result on iteration 0. Note that this corpus is small; therefore, although its complexity allow us to check some properties of the algorithm, we cannot draw further conclusions. Moreover, recall that we have erased the repetitions from the this test set.

Table 7.5 is the counterpart of Table 7.4 but for the Europarl-20. It can be observed that Baum-Welch training has no advantage with respect to Viterbi training. Typically, approximately 4 iterations suffice to avoid overfitting which maximises the system performance. The results show a small improvement over the initialisation. Although the improvement is very small, its magnitude is similar to the improvement obtained when extending the maximum phrase length as shown in Table 7.6. For instance, as the table shows, extending the maximum phrase length from 4 to 5 incurs in the same improvement as performing 4 Viterbi iterations in the model. Finally, in most of the cases, the Viterbi training improves the translation quality in terms of TER and/or BLEU.

	En → Sp		Sp → En	
	TER	BLEU	TER	BLEU
Moses $\mathrm{p}(\boldsymbol{x} \mid \boldsymbol{y})\,\mathrm{p}(\boldsymbol{y})$	50.0	32.9	47.2	32.7
Iterations	PBHSMM (Baum-Welch training)			
0	51.4	31.9	48.2	33.2
1	51.4	31.9	47.9	33.1
2	51.5	31.9	47.9	33.1
4	51.2	32.6	48.1	33.1
8	51.4	31.8	48.0	33.0
Iterations	PBHSMM (Viterbi training)			
0	51.4	31.9	48.2	33.2
1	51.4	31.9	47.9	33.1
2	51.1	32.6	48.0	33.2
4	51.2	32.6	48.0	33.0
8	51.4	31.8	48.0	33.0

Table 7.4: Results for the Europarl-10 corpus with a maximum phrase length of 4.

	En → Sp		Sp → En	
	TER	BLEU	TER	BLEU
Moses $\mathrm{p}(\boldsymbol{x} \mid \boldsymbol{y})\,\mathrm{p}(\boldsymbol{y})$	57.3	23.5	55.1	24.10
Iterations	PBHSMM (Baum-Welch training)			
0	57.7	25.0	56.0	26.0
1	57.7	25.1	55.8	26.4
2	57.7	25.1	55.9	26.4
4	57.7	25.2	55.8	26.5
8	57.7	25.2	55.8	26.5
Iterations	PBHSMM (Viterbi training)			
0	57.7	25.0	56.0	26.0
1	57.7	25.1	55.8	26.4
2	57.7	25.1	55.9	26.4
4	57.7	25.2	55.8	26.5
8	57.7	25.2	55.8	26.5

Table 7.5: Results for the Europarl-20 corpus with a maximum phrase length of 4.

Even though, the training does not incur in a significant improvement over the baseline in terms of BLEU and/or TER; in practice, the quality of the translations is increased by the training. Table 7.7 shows some translation examples. A detailed analysis of the proposed translations suggest that most cases belong to case A, case B or case D, and few translations belong to case C.

Iterations	En → Sp		Sp → En	
	TER	BLEU	TER	BLEU
Iterations	Maximum phrase length 2			
0	60.5	21.2	57.9	23.5
4	60.5	21.2	58.1	23.5
Iterations	Maximum phrase length 3			
0	58.6	24.1	56.1	25.7
4	58.3	24.1	56.4	25.5
Iterations	Maximum phrase length 4			
0	57.7	25.0	56.0	26.0
4	57.7	25.1	55.8	26.5
Iterations	Maximum phrase length 5			
0	57.7	25.1	55.8	26.6
4	57.4	25.3	55.3	26.9
Iterations	Maximum phrase length 6			
0	57.7	25.4	55.9	26.6
4	57.3	25.6	55.4	26.8

Table 7.6: Results for the Europarl-20 corpus with several phrase length.

7.6.2 Log-linear models.

In this case, we also used two corpora: the Europarl-20, and Europarl-v3. The former has already been described in the previous section. In this set of experiments, we randomly selected 2 000 sentences from the Europarl-20 training set as the development set to train the log-linear weights. The latter corpus is the standard dataset used in Koehn and Monz [2006]. Table 7.8 summarises the properties of the first corpus and Table 7.9 summarises the porperties of the second corpus.

We compared 3 systems: Moses, log-PBHSMM, and log-PBHSMM+Moses. All the systems used the Moses decoder [Koehn et al., 2007] to perform the decoding process. Therefore, the differences among the three systems lay in the features used in the log-linear model. The first system is the standard log-linear system trained with Moses [Koehn et al., 2007], where the following features were used:

- Direct phrase-based translation model
- Inverse phrase-based translation model
- Direct lexicon model
- Inverse lexicon model
- A phrase penalty 2.718
- A word penalty e
- A 5-gram language model smoothed with the modified Kneser-Ney smoothing

The second model, the log-PBHSMM, has the same features as the Moses system, but we replaced both direct and inverse phrase-based models with our trained PBHSMM systems. Finally, for the third system, the *log-PBHSMM+Moses* system, we added both direct and inverse phrase-based probabilities trained with our PBHSMM system to the Moses system. Additionally, we obtained results with the systems in a monotone way (not using reodering) and using the standard distance-based reordering implemented in the Moses decoder [Koehn et al., 2007]. The log-linear weights of the three systems

Case A	Training improves evaluation measures
REF.	I sincerely believe that the aim of the present directive is a step in the right direction .
IT. 0	I am convinced that the aim of this directive is a step in the right direction .
IT. 4	I sincerely believe that the aim of the directive before us is a step in the right direction .
MOSES	I sincerely believe that the aim behind the directive is also a step in the right direction .
Case B	Training improves translation but not evaluation measures
REF.	Mr president , i wish to endorse mr posselt 's comments .
IT. 0	Mr president , i support for to our .
IT. 4	Mr president , i join in good faith to our colleague , mr posselt .
MOSES	mr president , i would like to join in good faith in the words of our colleague , mr robig .
Case C	Training degrades the system
REF.	BSE has already cost the uk gbp 1.5 billion in lost exports .
IT. 0	BSE has cost the uk 1.5 million losses exports .
IT. 4	BSE already has cost in the uk alone 1500 million pounds into loss of exports .
MOSES	BSE has already claimed to britain 1500 million pounds into loss of trade .
Case D	Other cases
REF.	I will finish by telling you a story .
IT. 0	I will history .
IT. 4	To conclude a story .
MOSES	I shall conclude a history .
REF.	Are there any objections to amendment nos 3 and 14 being considered as null and void from now on ?
IT. 0	Are there any objections to give amendments nos 3 and 14 .
IT. 4	Are there any objections to adopt amendments nos 3 and 14 ?
MOSES	Are there any objections to consider amendments nos 3 and 14 ?

Table 7.7: Some translation examples (Sp → En) before and after training the phrase table, 4 iterations with the Viterbi training, and maximum phrase length of 4 words.

	Training		Development		Test	
Language	En	Sp	En	Sp	En	Sp
sentences	304 897		2 000		5 000	
avg. length	12.7	12.6	12.8	12.6	12.6	12.8
running words	3.83M	3.88M	25.1K	25.5K	63.8K	63.0K
voc. size	37.0K	57.7K	3.9K	4.7K	6.3K	8.1K
ppl (5-gram)	–	–	62.2	67.2	63.3	69.2

Table 7.8: Basic statistics of Europarl-20 with development set.

were trained in the development set of each corpus performing *Minimum Error Rate Training (MERT)* in terms of BLEU.

Table 7.10 shows the results in terms of BLEU and TER for these systems using the Europarl-20 training corpus. Instead of computing a single figure, we computed the confidence interval at 95% as described in Koehn [2004]. In this case we constrained the maximum phrase length to 4 words, so that

	Training		Development		Test	
Language	En	Sp	En	Sp	En	Sp
sentences	730 740		2 000		2 000	
avg. length	20.8	21.5	29.3	30.3	30.0	30.2
running words	15.2M	15.7M	58.7K	60.6K	58.0K	60.3K
voc. size	72.7M	113.9K	6.5K	8.2K	6.5K	8.3K
ppl (5-gram)	–	–	79.6	78.8	78.3	79.8

Table 7.9: Basic statistics of Europarl-v3.

System	En → Sp		Sp → En	
Distance-based reordering				
	TER	BLEU	TER	BLEU
Moses	56.7 ± 0.7	27.7 ± 0.7	54.2 ± 0.7	28.5 ± 0.7
log-PBHSMM	56.4 ± 0.7	28.1 ± 0.7	53.8 ± 0.7	28.7 ± 0.6
Moses + log-PBHSMM	56.4 ± 0.7	28.3 ± 0.7	53.4 ± 0.7	28.8 ± 0.7
Monotone				
	TER	BLEU	TER	BLEU
Moses	58.6 ± 0.7	26.1 ± 0.6	55.1 ± 0.7	27.3 ± 0.7
log-PBHSMM	57.6 ± 0.7	26.6 ± 0.6	54.4 ± 0.7	27.9 ± 0.6
Moses + log-PBHSMM	58.6 ± 0.7	26.4 ± 0.7	54.2 ± 0.6	28.0 ± 0.7

Table 7.10: Results for several translation systems on the Europarl-20 corpus.

these results were comparable with the results obtained in the previous experimental setup. It can be observed, that the log-PBHSMM obtains an improvement over the monotonic baseline, though it is not statistically significant.

Table 7.11 is the same as Table 7.10 but with the Europarl-v3 corpus. In this case, we constrained the maximum phrase length to the standard length of 7 word. It can be observed, that, in this case, our proposed model PBHSMM is not better than the standard Moses baseline. However, this corpus has the peculiarity that the development and test set are not distributed according to the training set probability distribution. This can be easily checked in Table 7.9 by comparing the average sentence lengths in each partition. Therefore, the fact that the proposed PBHSMM works slightly worse than the standard phrase-tables is not surprising. The only reason for providing these results is because it is a standard corpus.

7.7 Conclusions

In this chapter, we have presented a phrase-based hidden semi-Markov model for machine translation inspired on both phrase-based models and classical hidden semi-Markov models. The idea behind this model is to provide a well-defined monotonic formalism that, while remaining close to the phrase-based model, explicitly introduces the statistical dependencies needed to define the monotonic translation process with theoretical correctness.

Although the proposed model does not take full advantage from the HSMM formalism, we could not ignore some previous negative results [DeNero et al., 2006] when conditional phrase-based models

System	En → Sp		Sp → En	
Distance-based reordering				
	TER	BLEU	TER	BLEU
Moses	55.0 ± 0.8	29.9 ± 0.9	53.6 ± 0.9	30.5 ± 1.0
log-PBHSMM	55.6 ± 0.9	29.3 ± 0.9	53.8 ± 0.9	30.1 ± 0.9
Moses+log-PBHSMM	54.9 ± 0.9	29.9 ± 0.8	53.5 ± 1.0	30.6 ± 0.9
Monotone				
	TER	BLEU	TER	BLEU
Moses	55.6 ± 0.9	29.1 ± 0.8	53.8 ± 0.9	30.2 ± 0.9
log-PBHSMM	56.0 ± 0.9	28.9 ± 0.9	54.3 ± 0.9	29.8 ± 0.9
Moses+log-PBHSMM	55.6 ± 0.8	29.2 ± 0.9	54.0 ± 0.9	30.1 ± 0.9

Table 7.11: Results for several translation systems on the Europarl-v3 corpus.

are trained statistically. In that work, DeNero et al. [2006] concluded that a statistical (conditional) phrase-based model worsens the translation performance of a phrase-based system because statistical systems peak the phrase table probabilities. Consequently, we have forced our PBHSMM to be as close as possible to a phrase-based model to check whether DeNero's conclusion was extensible to this formalism or not. In contrast to DeNero et al. [2006], our experimental analysis has shown a slight improvement in some cases by applying the estimation algorithm with respect to the baseline, though this improvement is not statistically significant. Furthermore, we have surprisingly found that both training algorithms, Viterbi and Baum-Welch, obtain the same practical behaviour. Hence, we advocate for the use of Viterbi training.

We consider that the addition of a well defined training procedure would allow us to improve the system with future extensions. For instance, we could have assumed that the t-th source segment length depends on the t-th target segment length and on the t-th target segment $\boldsymbol{y}(t)$, that is to say,

$$p_r(\boldsymbol{l} \,|\, \boldsymbol{m}, \boldsymbol{y}, J) := \prod_t \mathrm{p}(l_t \,|\, m_t, \boldsymbol{y}(t)) \quad . \tag{7.50}$$

In order to fully take advantage of the HSMM theoretical framework, one outstanding and simple extension to the proposed model is to "unhide" the *concept* variable by having a mixture of phrase-based dictionaries. Hence, the model proposed in Section 7.2 would be given by

$$p_r(\boldsymbol{x} \,|\, \boldsymbol{y}, J) := \sum_{\boldsymbol{c}} \sum_{\boldsymbol{l}} \sum_{\boldsymbol{m}} \prod_t \mathrm{p}(c_t \,|\, c_{\pi_t}) \, \mathrm{p}(m_t \,|\, c_t) \, \mathrm{p}(l_t \,|\, m_t, c_t) \, \mathrm{p}(\boldsymbol{x}(t) \,|\, \boldsymbol{y}(t), c_t) \quad , \tag{7.51}$$

where $\mathrm{p}(\boldsymbol{x}(t) \,|\, \boldsymbol{y}(t), c_t)$ stands for a phrase-table that depdends on the current hidden concept c_t and the seen concept $\boldsymbol{y}(t)$; and where $\mathrm{p}(c_t \,|\, c_{\pi_t})$ plays the role of the transition probabilities. Actually, the requirements of this extension would not significantly affect the proposed estimation algorithms. For instance, the forward recurrence will be slightly modified as follows

$$\alpha_{ctl} = \begin{cases} 1 & t = 0, l = 0 \\ \sum_{t',l',c'} \alpha_{c't'l'} \, \mathrm{p}(c \,|\, c') \, \mathrm{p}(t'{-}t \,|\, c) \, \mathrm{p}(l'{-}l \,|\, t'{-}t, c) \, \mathrm{p}(\boldsymbol{x}_{l'+1}^{l} | \boldsymbol{y}_{t'+1}^{t}, c) & 0 < t \leq I \\ & 0 < l \leq J \\ 0 & \text{otherwise} \end{cases} \tag{7.52}$$

where the sum over t' ranges from $t+1$ to I; likewise the sum over l' ranges from $l+1$ to J; and where c' ranges among all the possible phrase states.

Note that the new unhidden concepts proposed in Eq.(7.51) would capture the syntactic, semantic and grammatical constraints between the source and the target sentence inside the same translation pair. However, we have left this interesting extension out of this thesis, and we intend to develop it in immediate future work.

We have also used the proposed PBHSMM as a feature inside a log-linear model as most of the current state of the art systems do. The results show an improvement over the baseline, both monotone and non-monotone systems, but only if the test probability distribution is similar to the training probability distribution. This improvement would probably be lost as the monotonicity of the language pairs decreases. However, we leave the practical analysis of this model in other language pairs for future work.

The model presented in this chapter, PBHSMM, can have some sparseness issues for some languages. Furthermore, the model extension proposed in (7.51) can aggravate this sparseness problems. In order to alleviate them, we could use word categories or tags, either statistically inspired or syntactically inspired. This approach would give a more reliable estimation of the phrase emission probabilities by reducing the sparsity problems.

Finally, the most undesirable property of the proposed model is its monotonicity. Although the monotonic constraint is a clear disadvantage for this first PBHSMM translation model, it can be extended to non-monotonic processes. For instance, the IBM-like reordering models [Zens et al., 2003] can be included in the proposed model by means of memory states. Furthermore, we can decouple the translation problem in two problems: the reordering of the input and then a monotonic translation. In this way we could define specific input reordering models that do not need to tackle the problem of translating the source sentence but rather reorder it. Afterwards, we could use any monotone translation model to carry out the translation from the reordered source sentence to the target sentence. Nevertheless, these extensions lay far beyond the aim of this thesis.

Bibliography

Jesús Andrés-Ferrer and A. Juan. A phrase-based hidden semi-markov approach to machine translation. In *Procedings of European Association for Machine Translation (EAMT)*, Barcelona, Spain, May 2009. European Association for Machine Translation.

P. F. Brown et al. The Mathematics of Statistical Machine Translation: Parameter Estimation. *Computational Linguistics*, 19(2):263–311, 1993.

J. DeNero, D. Gillick, J. Zhang, and D. Klein. Why generative phrase models underperform surface heuristics. In *Proceedings on the Workshop on Statistical Machine Translation*, pages 31–38, New York City, June 2006. Association for Computational Linguistics.

P. Koehn. Europarl: A parallel corpus for statistical machine translation. In *Proc. of the MT Summit X*, pages 79–86, September 2005.

P. Koehn et al. Moses: Open source toolkit for statistical machine translation. In *Proc. of ACL'07: Demo and Poster Sessions*, pages 177–180, Morristown, NJ, USA, June 2007. Association for Computational Linguistics.

Philipp Koehn. Statistical significance tests for machine translation evaluation, 2004.

Philipp Koehn and Christof Monz. Shared task: Exploiting parallel texts for statistical machine translation. In *The North American Chapter of the Association for Computational Linguistics (NAACL) workshop on Statistical Machine Translation*, 2006. URL `http://www.statmt.org/wmt06/shared-task/`.

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

F.J. Och and H. Ney. The alignment template approach to statistical machine translation. *Computational Linguistics*, 30(4):417–449, 2004.

K. Papineni, S. Roukos, T. Ward, and W. Zhu. BLEU: a Method for Automatic Evaluation of Machine Translation. Technical Report RC22176, Thomas J. Watson Research Center, 2001.

Lawrence Rabiner. A tutorial on hmm and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, February 1989.

M. Snover et al. A study of translation edit rate with targeted human annotation. In *Proc. of AMTA'06*, pages 223–231, Boston, Massachusetts, USA, August 2006. Association for Machine Translation in the Americas.

A. Stolcke. SRILM – an extensible language modeling toolkit. In *Proc. of ICSLP'02*, pages 901–904, September 2002.

Y. Wang and A. Waibel. Decoding algorithm in statistical translation. In *Proc. of ACL'97*, pages 366–372, Morristown, NJ, USA, July 1997. Morgan Kaufmann / Association for Computational Linguistics.

Y. Wang and A. Waibel. Fast decoding for statistical machine translation. In *Proc. of ICSLP'98*, pages 2775–2778, October 1998.

Translation Richard Zens, Richard Zens, and Hermann Ney. A comparative study on reordering constraints in statistical machine. In *In ACL*, pages 144–151, 2003.

Chapter 8

Conclusions

" *Entities should not be multiplied beyond necessity* " OCCAM'S RAZOR

Contents

8.1 Summary

The work developed in this thesis covers several topics in natural language processing: text classification, language modelling, and statistical machine translation. Moreover, from a statistical point of view, this thesis revisits several statistical techniques used in these natural language processing problems: parameter estimation, loss function design, and statistical probability design.

With regard to parameter estimation, in Chapter 2, we proposed a constrained-domain maximum likelihood estimation technique (CDMLE). Our new proposal avoids the additional heuristic smoothing step that removes the good theoretical properties that the MLE verifies. The proposed technique can be extended to avoid the smoothing stage of any statistical model. In order to introduce this technique, we have applied it to the estimation of the naive Bayes classifier, which follows a multinomial distribution. We tested the novel training algorithm in several text classification tasks: Eutrans-I, 20-Newsgroups, Industry Sectors, and Job Categories. Finally, we observed that the proposed CDMLE technique shows performance that is similar to that obtained by classical smoothing techniques in these text classification tasks.

In Chapter 3, we used the CDMLE idea to smooth the n-gram leaving-one-out smoothing methods. We smoothed the Good-Turing probability estimates by constraining their domain. This novel approach filled the gap between two extremes in LM smoothing, that is to say, between the Good-Turing and the Kneser-Ney smoothing methods. The new proposed smoothing algorithms were compared in practice with respect to the Kneser-Ney baseline in terms of perplexity. Two corpora were used to perform this comparison: the Wall-Street-Journal and the English part of Europarl-v3. The results reported an improvement over the baseline in terms of perplexity for backing-off n-gram language models. The proposed n-gram smoothing techniques are also generalisable to other leaving-one-out estimation problems.

In Chapter 4, we carefully studied the consequences of changing the 0–1 loss function for more complex loss functions. We focused our study on those loss functions that retain a similar decoding complexity when compared with the 0–1 loss function. Several candidate loss functions were presented and tested in several statistical machine translation tasks. Furthermore, two different machine translation models were used to analyse the properties of each loss function: the IBM model 2 and the phrase-based models. We proved that some outstanding translation rules such as the *Direct Translation Rule* or even the log-linear models are, in fact, particular cases of these loss functions.

The remaining three chapters of this thesis, are focused on defining monotone phrase-based models with efficient training algorithms. We started this hard task in Chapter 5 by giving a purely statistical definition and a training algorithm for a phrase-based GIATI extension. However, in this case, the training algorithm had high requirements in terms of both memory and time. This fact made the training algorithm practically unfeasible for many tasks.

After analysing the proposed SGIATI method, we found that a joint model hardens the modelling task unnecessarily since a joint translation model solves a more complex modelling task than what is needed: a conditional translation model. Hence, in Chapter 6, we proposed a monotone phrase-based hidden Markov model (PBHMM) for machine translation. The training algorithms for this new proposal are faster than the previous SGIATI model, which allowed us to obtain results in more complex tasks. However, the time and memory complexity were still demanding. Furthermore, the model did not improve the phrase-based model baseline for complex tasks.

Finally, we improved the PBHMM by using the hidden semi-Markov model formalism. Thus, a phrase-based hidden semi-Markov model was proposed in Chapter 7. This model, while remaining close to the conventional phrase-based models, introduced the hidden semi-Markov formalism in order to define efficient training algorithms. The experimental analysis reported improvements with respect to a phrase-based model when used as a statistical model in a classical SMT system. However, when this model played the role of a feature inside a log-linear loss function (a log-linear model in the SMT

literature), the results were similar to those obtained with the state-of-the-art systems.

In summary, the main contributions of this thesis are the following:

- Constrained-domain maximum likelihood estimation (CDMLE) is proposed as an alternative to the standard maximum likelihood estimation and smoothing post-processing. This novel approach was applied to the naive Bayes text classifier, obtaining good results in practice.
- New n-gram language smoothing models are proposed by applying CDMLE to smooth the leaving-one-out (Good-Turing) probability estimates. Specifically, we have proposed 5 smoothing models: interval-constrained smoothing, quasi-monotonic smoothing, monotonic smoothing, monotonic smoothing with upper constraints, and exact extended Kneser-Ney (eeKN) smoothing.
- A detailed practical analysis of several loss functions in the scope of machine translation is given. We proved that the direct translation rule is a special case of a loss function. Furthermore, we proved that the log-linear models are a special loss function with a parametric vector for characterising the loss. This parametric loss is usually adjusted or trained to resemble an error measure such as the BLEU or the WER.
- Translation models for monotone translation problems are presented as the application of well-known statistical modelling techniques such as HMM or HSMM. The results reported for the the PBHSMM improved the quality of the translations when compared with a phrase-based translation model. Unfortunately, the results obtained when the PBHSMM model plays the role of a feature inside a log-linear loss do not outperform the state-of-the-art translation systems for both monotone and non-monotone systems.
- Exact EM training and Viterbi-like training obtain the same results in practice when phrase-based translation models are used. Specifically, we checked this wide spread intuition with the proposed PBHSMM.

8.2 Ideas and future work

As research is a constantly changing and expanding field when one research line is explored, it is common for several more interesting lines to arise. Since this thesis is not the exception, we have left several interesting and appealing lines for future exploration.

Firstly, we circumvented the problem of smoothing by proposing the CDMLE technique in Chapter 2. The CDMLE technique can be easily expanded to several probability models, and, hence, systems. Specifically, in the case of multinomial distribution, we have left out complex constraints that are similar to complex smoothing techniques.

When applied to n-gram smoothing, the CDMLE yields several novel smoothing techniques. The proposed techniques for smoothing n-gram models reported an improvement in terms of perplexity when a back-off model is used; however, it is not yet clear that this improvement would yield an improvement in terms of WER or BLEU. We think that this is a very interesting research line since, depending on the task, we would get full improvement. For instance, all the improvements in isolated models are not directly transferred to a global improvement in a log-linear loss (log-linear model in SMT literature) for machine translation tasks.

The proposed smoothings followed a backing-off scheme, however, the best practice performance is obtained with linear interpolation models. It would be a very interesting research line to extend the proposed discounting methods to linear interpolation smoothing models. This extension would entail the problem of computing leaving-one-out with *fractional counts*, since an iterative algorithm EM-like would be necessary.

The experimentation carried out in Chapter 3 suggests that the differences between the proposed and the standard smoothing methods lay in the transition from the joint probability domain to the conditional domain. Recall that the techniques based on leaving-one-out, which include the proposed smoothing models, maximise the *joint likelihood function*, $p_r(w, h)$; even though the system goal is to optimise the (conditional) likelihood function, $p_r(w \mid h)$. We observed that the best smoothing technique, exact extended Kneser-Ney, obtains the same *joint perplexities* than the KN technique; however, it is also the technique whose performance is less diminished when it is measured in terms of the standard *(conditional) perplexity*. Therefore, we believe that improvement will be achieved by directly applying the proposed techniques to maximise the (conditional) likelihood function, even if there is no close solution available. Moreover, it would also allow us to have discounting parameters that depend on the n-gram context, h, thereby, flexibilising the smoothing models.

Another interesting observation is that directly applying the optimal smoothing parameters for the backing-off smoothing model to the interpolated smoothing model degrades all the proposed smoothings in Chapter 3. It would be interesting to apply the proposed theory to an interpolated smoothing model, in order to see whether the proposed smoothings improve the interpolation baseline or not.

In Chapter 4, we have explored the loss functions that lay in between the 0–1 loss function and the general error loss functions, such as the WER or the BLEU, which have already been studied [R. Schlüter and Ney, 2005, Ueffing and Ney, 2004]. We found that the *log-linear models* are really a *log-linear loss function*. The results showed that none of the proposed loss functions can beat these log-linear loss functions. Although some outstanding loss functions were studied in Chapter 4, there are some appealing loss functions that have been left out, such as the remaining information. If these loss functions were introduced inside a log-linear loss function that approximates the error criterion, then we think that the system performance would be improved.

Finally, in the remaining three chapters we proposed several monotone translation models. Specifically, the last model improved (in some circumstances) the baseline, while having a very clear statistical foundation. Note that we have decided to adhere to the monotonic constraint since for mainly monotone (at phrase level) language pairs such as Spanish and English, the translation task is still an open problem. Furthermore, adding complex reordering models only incurs in slight improvements.

We think that a more detailed experimentation with other language pairs is necessary in order to see to what extent monotone formulation is good for those language pairs. Additionally, the PBHSMM proposed in Chapter 7 can be greatly extended, as proposed in Section 7.7. The PBHSMM extensions range from substituting the phrase emission probabilities by a word-level model, to expanding the model to be non-monotone at the phrase-level. However, we think that the most appealing extension is the extension of the model with a "hidden concept" or state. This will generate a mixture of phrase dictionaries to be used at different positions in the source sentence while performing the translation. This extension would take into account not only semantic relationships but also syntactic or grammatical dependencies.

The reader should keep in mind that this extension would incur in slight modifications of the PBHSMM training algorithms. Obviously, such an extension would require a huge parameter set. However, this set can be reduced by modelling the emission probabilities at each state by IBM models 1 and 2 [Brown et al., 1993]. This would eventually lead to a PBHSMM where the phrase tables are dynamically built whenever a source phrase is needed.

Surprisingly, we found that simplified translation tasks based on real data such as the Europarl-20 corpus report similar or worse results than those obtained with the full corpora. How can long sentences be properly translated if short sentences cannot be correctly translated with similar training data? We consider this to be a problem in the current state-of-art, phrase-based log-linear models and/or error measures. Note that the length of the translated sentences does not seem to simplify the task when the training data is also restricted. Therefore, it seems to us that the current models lack enough generalisation capacity. Although the phrase table is a good aid in the translation process, we think that

it is necessary to use a fined-grain unity in the translation process such as word-based phrase-tables, as we proposed in the above extension to the PBHSMM.

One of the major criticisms to the proposed PBHSMM model lies in its monotonicity at phrase level. This problem would be solved by one of the proposed extensions in Section 7.7. We are referring to the implementation of IBM-like reorderings [Zens et al., 2003] by means of memory states. Also, the translation process can be divided in two steps (a reordering of the input and a monotone translation from the reordered input to the output [Kanthak et al., 2005]), in order to make the model non-monotone.

8.3 Scientific publications

Most of the work in this thesis has directly yielded international articles in workshops, conferences and journals. In this section, we enumerate these contributions to the scientific community, highlighting the relationship with this thesis.

The theory and experimental results in Chapter 2 have yielded one publication in an international conference:

- **J.Andrés-Ferrer** and Alfons Juan. Máxima versoimilitud con dominio restringido applicada a clasificación de textos. In *Proceedings of "Campus Multidisciplinar en Percepción e Inteligencia"*, CMPI-06, pages: 791–803, Albacete, Spain July 10-14, 2006.

It has also yielded a publication in an international journal:

- **J.Andrés-Ferrer** and Alfons Juan. Constrained domain maximum likelihood estimation for naive Bayes text classification. *Pattern Analysis and Applications (PAA)*. Published online. 2009.

Some of the smoothing techniques proposed in Chapter 3 have produced a participation in an international conference:

- **J. Andrés-Ferrer** and H. Ney. Extensions of absolute discounting (Kneser-Ney method). In *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing*, ICASSP 2009, Taipei, Taiwan, 2009. Association for Computational Linguistics.

This contribution was awarded with the *IEEE Spoken Language Processing Student Travel Grant*[a], which honours the student of an outstanding paper in the spoken language processing area accepted for publication in the ICASSP conference or a ASRU workshop sponsored by the IEEE Signal Processing Society.

The results obtained with IBM model 2 in Chapter 4 were published in two international conferences:

- **J.Andrés-Ferrer**, I. García-Varea, F. Casacuberta. Análisis teórico sobre las reglas de traducción directa e inversa en traducción automática estadística. In *Proceedings of "Campus Multidisciplinar en Percepción e Inteligencia"*, CMPI-06, pages: 855–867, Albacete, Spain July 10-14, 2006.
- **J.Andrés-Ferrer**, I. García-Varea, F. Casacuberta. Combining translation models in statistical machine translation. In *Proceedings of the 11th International Conference on Theoretical and Methodological Issues in Machine Translation*, TMI-07, pages: 11–20, Skovde, Sweden September 7-9, 2007.

The phrase-based results and a summary of the theory were published in an international journal:

- **J.Andrés-Ferrer**, D. Ortiz-Martínez, I. García-Varea, F. Casacuberta. On the use of different loss functions in statistical pattern recognition applied to machine translation. *Pattern Recognition Letters*. Volume 29, pages: 1072–1081, 2008.

[a] More information at `http://research.microsoft.com/en-us/people/alexac/award.aspx`

The statistical extension to GIATI, the SGIATI model, which is presented in Chapter 5, was published in the following journal:

- **J.Andrés-Ferrer**, A. Juan and F. Casacuberta. Statistical estimation of rational transducers applied to machine translation. *Applied Artificial Intelligence* 22(1-2):4–22, 2008.

The hidden Markov model approach to machine translation discussed in Chapter 6 was published in the following international workshop:

- **J.Andrés-Ferrer** and A. Juan. A phrase-based hidden Markov model approach to machine translation. In *Proceedings of New Approaches to Machine Translation*, pages 57–62, January 2007.

Finally, the proposed phrased-based hidden semi-Markov model approach and some of the results in Chapter 7 were published in the following international conference:

- **J.Andrés-Ferrer** and A. Juan. A phrase-based hidden semi-Markov approach to machine translation. In *Proceedings of European Association for Machine Translation (EAMT)*, pages 168–175, May 2009, Barcelona (Spain). European Association for Machine Translation.

Bibliography

P. F. Brown et al. The Mathematics of Statistical Machine Translation: Parameter Estimation. *Computational Linguistics*, 19(2):263–311, 1993.

Stephan Kanthak, David Vilar, Evgeny Matusov, Richard Zens, and Hermann Ney. Novel reordering approaches in phrase-based statistical machine translation. In *Proceedings of the ACL Workshop on Building and Using Parallel Texts: Data-Driven Machine Translation and Beyond*, pages 167–174, 2005.

V. Steinbiss R. Schlüter, T. Scharrenbach and H. Ney. Bayes risk minimization using metric loss functions. In *Proceedings of the European Conference on Speech Communication and Technology, Interspeech*, pages 1449–1452, Lisbon, Portugal, September 2005.

N. Ueffing and H. Ney. Bayes decision rules and confidence measures for statistical machine translation. In *EsTAL - Espa for Natural Language Processing*, pages 70–81, Alicante, Spain, October 2004. Springer Verlag, LNCS.

Translation Richard Zens, Richard Zens, and Hermann Ney. A comparative study on reordering constraints in statistical machine. In *In ACL*, pages 144–151, 2003.

Appendix $\mathcal{A}$

Karush-Kuhn-Tucker Conditions

In Chapters 2 and 3, we have used advanced convex optimisation techniques that deserve a superficial survey. In Section 1.1.2 Chapter 1, we have analysed that in order to obtain the optimal parameter set, it is needed to find the maximum parameter set according to a given criterion $\mathcal{C}(\boldsymbol{\theta}; D)$. Almost all the optimisation problems that derive form this formulation, are subject at least to some normalisation constraint. In order to solve these constrained optimisation problems the *convex optimisation* theory [Boyd and Vandenberghe, 2004] is usually applied.

First, we review a typical convex optimisation example. We wish to solve the following equation

$$\hat{\boldsymbol{\theta}} = \arg\max_{\boldsymbol{\theta} \in \boldsymbol{\Theta}} \{\mathcal{C}(\boldsymbol{\theta}; D)\} \quad , \tag{A.1}$$

subject to

$$\mathrm{P}_n(\boldsymbol{\theta}) = 0, \qquad (n = 1, \ldots, N) \quad . \tag{A.2}$$

In order to solve the previous optimisation the *Lagrangian function* must be defined

$$\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\lambda}) = \mathcal{C}(\boldsymbol{\theta}; D) - \sum_n \lambda_n \, \mathrm{P}_n(\boldsymbol{\theta}) \quad , \tag{A.3}$$

where a *Lagrangian multiplier* (λ_n) is defined for each equality constraint P_n.

Theory concludes that solving Eq. (A.1) subject to Eq. (A.2) is equivalent to solve the following problem

$$\hat{\boldsymbol{\theta}} = \arg\max_{\boldsymbol{\theta} \in \boldsymbol{\Theta}} \{\max_{\boldsymbol{\lambda}} \mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\lambda})\} \quad . \tag{A.4}$$

Therefore, an optimal point must verify the following property

$$\nabla \mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\lambda})|_{\hat{\boldsymbol{\theta}}, \hat{\boldsymbol{\lambda}}} = 0 \quad , \tag{A.5}$$

rising up a linear system from which the value of $\hat{\boldsymbol{\theta}}$ is hopefully worked out.

The above optimisation example is typically known as an *equality constrained program*. However, in this thesis, we solve some optimisation problems that also include inequality constraints. In order to solve problems with inequality constraints, the *Karush-Kuhn-Tucker (KKT)* conditions are needed.

The new problem is similar to the previous constrained problem but with some additional inequality constraints, that is to say, we wish to solve Eq. (A.1) subject to Eq. (A.2) and subject to the following constraints

$$\mathrm{Q}_m(\boldsymbol{\theta}) \leq 0, \quad (m = 1, \ldots, M) \quad . \tag{A.6}$$

In this case, the Lagrangian function is defined as follows

$$\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\lambda}, \boldsymbol{\mu}) = -\mathcal{C}(\boldsymbol{\theta}; D) + \sum_n \lambda_n \, \mathrm{P}_n(\boldsymbol{\theta}) + \sum_m \mu_m \, \mathrm{Q}_m(\boldsymbol{\theta}) \quad . \tag{A.7}$$

Solving Eq. (1.15) subject to Eq. (A.2) and to Eq. (A.6) is the equivalent to solve

$$\hat{\boldsymbol{\theta}} = \arg\min_{\boldsymbol{\theta} \in \boldsymbol{\Theta}} \min_{\boldsymbol{\lambda}, \boldsymbol{\mu}} \mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\lambda}, \boldsymbol{\mu}) \quad . \tag{A.8}$$

The KKT necessary conditions for a point $(\boldsymbol{\theta}, \boldsymbol{\lambda}, \boldsymbol{\mu})$ to be a maximum point of Eq. (A.8) are the following

$$\nabla_{\boldsymbol{\theta}} \, \mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\lambda}, \boldsymbol{\mu})|_{\hat{\boldsymbol{\theta}}, \hat{\boldsymbol{\lambda}}, \hat{\boldsymbol{\mu}}} = 0 \tag{A.9}$$

$$\mathrm{P}_n(\boldsymbol{\theta}) = 0, \quad (n = 1, \ldots, N) \tag{A.10}$$

$$\mu_m \, \mathrm{Q}_m(\boldsymbol{\theta}) = 0, \quad (m = 1, \ldots, M) \tag{A.11}$$

$$\mu_m \geq 0, \quad m = 1, \ldots, M \tag{A.12}$$

$$\mathrm{Q}_m(\boldsymbol{\theta}) \leq 0, \quad (m = 1, \ldots, M) \tag{A.13}$$

Even though KKT conditions are *necessary* conditions, they are not sufficient conditions. That is to say that a maximum point must verify them, but not all points that verify them are maximum points. An additional condition must be verified in order to check whether a point that verifies the KKT conditions is optimal or not. This condition states that the Hessian of the Lagrangian function must be positive at a maximum point [Boyd and Vandenberghe, 2004]. After the possible optimal points are given, checking whether this sufficient and necessary condition is verified or not, is a simple mathematical exercise. If the characterisation of the solution is unique, then the solution is necessarily the maximum (if it exists).

The KKT conditions often provide just a characterisation of the solution, but not a procedure to obtain it. Hopefully, once the form of the solution is known, it is often possible to define an efficient algorithm to obtain this characterised solution.

Bibliography

Stephen Boyd and Lieven Vandenberghe. Convex optimization. pages 244–254, March 2004.

List of Figures

List of Tables