UNIVERSIDAD POLITÉCNICA DE VALENCIA
DEPARTAMENTO DE SISTEMAS INFORMÁTICOS Y COMPUTACIÓN

CONSTRAINED DOMAIN MAXIMUM LIKELIHOOD ESTIMATION AND THE LOSS FUNCTION IN STATISTICAL PATTERN RECOGNITION

Master thesis
presented by Jesús Andrés Ferrer
supervised by Dr. Alfons Juan Císcar

November 13, 2008

NOTATION

Symbol	Meaning
$\mathrm{const}(\boldsymbol{x})$	A constant function on $\boldsymbol{x}$, i.e., a function that *does not* depend on $\boldsymbol{x}$
$p_r(\cdots)$	The actual unknow probability distribution
$\mathrm{p}_{\boldsymbol{\theta}}(\cdots)$	We outline the fact that the probability is not the actual probability but a model that depends upon $\boldsymbol{\theta}$
$\mathrm{p}(\cdots)$	The probabilities depicted in this way are already a model parameter
$A := B$	This symbol is used to stress that A is modelled as B
$\delta(a, b)$	Stands for the Kronecker delta function, i.e., 1 if and only if $a = b$, and 0 otherwise

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

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

we use

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

Since identifying the value ω and the random variable Ω does not has any negative effect, we will henceforth take the latter notation, i.e. Eq. (2).

CONTENTS

CHAPTER 1

PRELIMINARIES

“ *I could prove God, statistically.* ” GEORGE GALLUP

Contents

SEVERAL topics are reviewed in this chapter. Some of them might seem too basic to be reviewed, however, in the following chapters, these basic concepts are extended. For instance, the review of the statistical pattern recognition theory may seem too trivial for the level assumed to the reader, however chapter 3 is mainly focused on the basis of statistical pattern recognition.

The organisation of the current chapter is as follows. Firstly, in section 1.1, we review the basic concepts of statistical pattern recognition. This statistical pattern recognition review requires the analysis of several related topics. We start reviewing the *statistical modelling* in subsection 1.1.1. Afterwards, it is necesary to select the *training criterion* which is reviewed in subsection 1.1.2. One of the most outstanding criterion is the *maximum likelihood (ML)* criterion which is covered in subsection 1.1.3.

Once the fundamental theory is reviewed in section 1.1, the following sections are focussed on statistical pattern recognition tasks. In section 1.2, the *machine translation* problem is analysed from a statistical point of view. This statistical review covers several mainstream statistical models such as *alignment word based models* in subsection 1.3.1, or *phrase-based models* in subsection 1.3.2. Since the automatic evaluation of machine translation systems is a problem by itself, subsection 1.3.3 is devoted to review the main automatic evaluation metrics. In section 1.4, the problem of *text classification* is trackled under the scope of pattern recognition.

Finally, the scientific contributions of the thesis are briefly enumerated in section 1.5.

1.1 Statistical Pattern Recognition

A pattern recognition problem consists in classifying each possible input or object, say $x \in \boldsymbol{X}$, into one class, say ω, from the set of all possible classes, i.e. Ω. Examples of pattern recognition problems include text classification, speech recognition, image classification, face recognition, machine translation, etc.

A classification system is characterised by the *classification function*

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

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, speech recognition.

Several methodologies can be applied to define the classification function, arising the necessity of comparing among them. In order to sort the systems, the *classification error rate (CER)* is defined as the percentage of misclassifications performed by the system.

The performance of a classification function 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 repercussions. 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. 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 of the system 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

$$R(\omega_p|\boldsymbol{x}) = \sum_{\omega_c\in\Omega} \mathrm{l}(\omega_p|\boldsymbol{x},\omega_c)\, p_r(\omega_c|\boldsymbol{x}) \tag{1.2}$$

Usually, we want to compare system risks independently of any specific object $\boldsymbol{x}$. Using the conditional risk, we define the *the global risk* [**?**] as the contribution of all objects to the classifier performance. The global risk is defined as follows

$$R(\mathrm{c}) = \mathrm{E}_{\boldsymbol{x}}[R(\mathrm{c}(\boldsymbol{x})|\boldsymbol{x})] = \int_{\mathcal{X}} R(\mathrm{c}(\boldsymbol{x})|\boldsymbol{x})\, p_r(\boldsymbol{x}) d\boldsymbol{x} \tag{1.3}$$

where $R(\mathrm{c}(\boldsymbol{x})|\boldsymbol{x})$ is the conditional risk given $\boldsymbol{x}$, as defined in 1.3.

In practice, the global risk can rarely been calculated. However, 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

$$\bar{R}_T(\mathrm{c}) = \frac{1}{N}\sum_{n=1}^{N} \mathrm{l}(\mathrm{c}(\boldsymbol{x}_n)|\boldsymbol{x}_n,\omega_n) \tag{1.4}$$

The approximation of the global risk using a test set as in previous Eq. (3.6) is called *empirical risk* on the test set T.

The classification error rate corresponds to the 0–1 loss function defined as follows

$$\mathrm{l}(\omega_p|\boldsymbol{x},\omega_c) = \begin{cases} 0 & \omega_p = \omega_c \\ 1 & \text{otherwise} \end{cases} \tag{1.5}$$

In this case the conditional risk simplifies to

$$R(\omega_p|\boldsymbol{x}) = 1 - p_r(\omega_p|\boldsymbol{x}) \tag{1.6}$$

and the empircial risk is

$$\bar{R}_T(\mathrm{c}) = \frac{1}{N}\sum_{n=1}^{N} \delta(\mathrm{c}(\boldsymbol{x}_n),\omega_n) \tag{1.7}$$

where δ is a function which is equal to 1 if both parameters are equal and 0 otherwise.

Our aspiration it 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 loss of generality, the optimal classification rule, namely *minimum Bayes' risk*, is the one that minimises the conditional risk for each object

$$\hat{\mathrm{c}}(\boldsymbol{x}) = \arg\min_{\omega\in\Omega} R(\omega \mid \boldsymbol{x}) \tag{1.8}$$

Assuming the 0–1 loss function, and using the simplification of the conditional risk in Eq. (1.7)

$$\hat{\mathrm{c}}(\boldsymbol{x}) = \arg\max_{\omega\in\Omega} p_r(\omega_p \mid \boldsymbol{x}) \tag{1.9}$$

This last equation is well-known and usually assumed to be optimal for all the cases because a 0-1 loss function is being assumed.

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 but in simulated experiments. 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 error or Bayes CER*.

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

$$\mathrm{c}(\boldsymbol{x}) = \arg\max_{\omega} \{\mathrm{p}_r(\omega \mid \boldsymbol{x})\} = \arg\max_{\omega} \{\mathrm{p}_r(\omega) p_r(\boldsymbol{x} \mid \omega)\} \tag{1.10}$$

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

Since we are focused, on approximation of actual probabilities, most of the modellisation 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.

1.1.2 Training criterion

In order to train a parametric model, the *optimal* set of parameters, say $\hat{\boldsymbol{\theta}}$ must be found. Although it might seem simple, the word "optimal" in previous definition requires a way to compare different parameter sets $\boldsymbol{\theta}$. Therefore, appropriateness depends upon a criterion which is summarised in the *criterion function ($\mathcal{C}$).* Given a criterion function, the optimal set of parameters, $\hat{\boldsymbol{\theta}}$, is determined by

$$\hat{\boldsymbol{\theta}} = \arg\max_{\boldsymbol{\theta}\in\boldsymbol{\Theta}} \{\mathcal{C}(\boldsymbol{\theta})\} \tag{1.11}$$

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

$$\hat{\boldsymbol{\theta}} = \arg\max_{\boldsymbol{\theta} \in \boldsymbol{\Theta}} \{\mathcal{C}(\boldsymbol{\theta}; D)\} \tag{1.12}$$

Nevertheless expression in Eq. (1.11) 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 current section. 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 parameter set according to our criterion.

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. There is a well-founded motivation for the use of the MLE. In principle, 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 model and the actual distribution, that is

$$\hat{\boldsymbol{\theta}} = \arg\min_{\boldsymbol{\theta} \in \boldsymbol{\Theta}} \{\mathrm{KL}(p_r || p_{\boldsymbol{\theta}})\} \tag{1.13}$$

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

$$\mathrm{KL}(p_r || p_{\boldsymbol{\theta}}) = \int_{\boldsymbol{X}} p_r(\boldsymbol{x}) \log p_r(\boldsymbol{x}) d\boldsymbol{x} - \int_{\boldsymbol{X}} p_r(\boldsymbol{x}) \log p_{\boldsymbol{\theta}}(\boldsymbol{x}) d\boldsymbol{x} \tag{1.14}$$

Plugging previous Eq. (1.14) into Eq. (1.13) yields

$$\hat{\boldsymbol{\theta}} = \arg\max_{\boldsymbol{\theta} \in \boldsymbol{\Theta}} \left\{ \int_{\boldsymbol{X}} p_r(\boldsymbol{x}) \log p_{\boldsymbol{\theta}}(\boldsymbol{x}) d\boldsymbol{x} \right\} \tag{1.15}$$

Since Eq. (1.16) is typically unfeasible to solve, by means of a sample $D = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\}$ and the law of great numbers, it can be approximated by

$$\hat{\boldsymbol{\theta}} = \arg\max_{\boldsymbol{\theta} \in \boldsymbol{\Theta}} \{\mathrm{LL}(\boldsymbol{\theta})\} = \arg\max_{\boldsymbol{\theta} \in \boldsymbol{\Theta}} \left\{ \sum_n \log p_{\boldsymbol{\theta}}(\boldsymbol{x}_n) \right\} \tag{1.16}$$

with the definition of the *log-likelihood function (*LL*)*

$$\mathrm{LL}(\boldsymbol{\theta}) = \sum_n \log p_{\boldsymbol{\theta}}(\boldsymbol{x}_n) \tag{1.17}$$

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.

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

$$\hat{\boldsymbol{\theta}} = \arg\max_{\boldsymbol{\theta}\in\boldsymbol{\Theta}} \{\sum_n \log p_{\boldsymbol{\theta}}(\boldsymbol{x}_n)\} \tag{1.18}$$

Maximum likelihood estimation typically leads to the intuitive solution of the relative frequencies.

The maximum likelihood estimation has been a core technique in pattern recogntion. However, there is a little confusion in the bibliografy around the MLE term. In principle, the MLE is an statistical technique to estimate the optimal set of parameters for a given probability distribution. In pattern recogntion, it is usually refered to estimate the probability $p_r(\boldsymbol{x}, \omega)$ using the following expression

$$\hat{\boldsymbol{\theta}} = \arg\max_{\boldsymbol{\theta}\in\boldsymbol{\Theta}} \{\sum_n \log p_{\boldsymbol{\theta}}(\boldsymbol{x}_n \mid \omega_n) + \log p_{\boldsymbol{\theta}}(\omega_n)\} \tag{1.19}$$

instead of the class posterior probability

$$\hat{\boldsymbol{\theta}} = \arg\max_{\boldsymbol{\theta}\in\boldsymbol{\Theta}} \{\sum_n \log p_{\boldsymbol{\theta}}(\omega_n \mid \boldsymbol{x}_n)\} \tag{1.20}$$

which is usually understood in statistical bibliografy.

The MLE have 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.

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 not be zero, and must be continuous
- The maximum likelihood estimator is consistent

Although, the MLE is asymptitically 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 parameter. In pattern recognition, this problem is very common and it is known as the over-training problem. The over-training problem is restated in pattern recognition as the fact that the learnt set of parameters is very specialised for the training data, and hence, little probability remains to be distributed among the unseen data.

The 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 loose all its theoretical properties.

In chapter 2, we propose a novel smoothing techniques that makes the smoothed vector to retain the desirable properties of the maximum likely estimates, and also to avoid the over-training associated to MLE.

1.2 Machine translation

In this section we review state-of-the-art applications and approaches in the field of *machine translation (MT)*. On the one hand, current MT technology is focused on three main applications:

- Fully-automatic MT in limited domains like weather forecast [LGLL05], hotel reception desk [ABC$^+$00a], appointment scheduling, etc.
- Post-editing for CAT, understanding by 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.

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 consists in transforming the source sentence to a language independent semantic representation, the so-called interlingua, and translating that interlingua expression into the desired target language. The major drawback of this approach is its demanding knowledge resources to represent such language independent information. Further details of this approach can be found in [N$^+$92, NM92, A$^+$93].
- The transfer approach decomposes the translation process into three steps:

  Analysis. The source sentence is syntactically and semantically parsed to some abstract representation.

  Transfer. A transformation from the source representation into the target representation is performed.

  Generation. The final translation is generated from the target representation obtained in the previous step.

  A review of transfer-based systems is presented in [HS92].
- 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:

  Example-based MT. This approximation deals with the translation of new sentences by analysing, using different matching criteria, similar sentences previously translated. See [Som99] for a review of example-based MT[a].

  Statistical MT. A statistical model is inferred from translation examples and the translation process is derived from a statistical decision theory perspective. This thesis is mainly devoted to the statistical approximation to MT that will be further studied in the next section.

[a]Also known as memory-based MT [Bow02, Som03]

1.3 Statistical MT

The goal of MT is the automatic translation of a source sentence x into a target sentence y,

$$\begin{aligned}\boldsymbol{x} &= x_1 \dots x_j \dots x_J \quad x_j \in \boldsymbol{X} \\ \boldsymbol{y} &= y_1 \dots y_i \dots y_I \quad\; y_i \in \boldsymbol{Y}\end{aligned}$$

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

In statistical MT, this translation process is usually presented as a statistical pattern recogntion problem where given a source sentence $\boldsymbol{x}$, the optimal target sentence $\hat{\boldsymbol{y}}$ is searched according to

$$\hat{\boldsymbol{y}} = \arg\max_{y} \; p_r(\boldsymbol{y} \mid \boldsymbol{x}) \tag{1.21}$$

where $p(\boldsymbol{y} \mid \boldsymbol{x})$ is the probability for y to be the actual translation of $\boldsymbol{x}$. Note that Eq. (1.21) 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. (1.21) as

$$\hat{\boldsymbol{y}} = \arg\max_{y} \; p_r(\boldsymbol{x} \mid \boldsymbol{y}) p_r(\boldsymbol{y}) \tag{1.22}$$

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. While the language model $p_r(\boldsymbol{y})$ represents the well-formedness of the candidate translation $\boldsymbol{y}$ [**?**]. It should be noted that the term $p_r(\boldsymbol{x})$ has been intentionally omitted in the denominator of Eq. (1.22), since it is constant for a given $\boldsymbol{x}$ when maximising over $\boldsymbol{y}$.

Note that we are looking the statistical MT as a specific instance of a classification problem where:

- The object to be classified is the sentence $\boldsymbol{x}$ to be translated.
- The set of possible classes are the set of possible sentences in the target language $\boldsymbol{y} \in \boldsymbol{Y}^{\star}$.
- The prior probability distribution is the language model $p_r(\boldsymbol{y})$.
- The conditional probability distribution is the translation model $p_r(\boldsymbol{x} \mid \boldsymbol{y})$.

Therefore, under this point of view the decision rule stated in Eq. (1.21) is optimal under the assumption of a *zero-one* loss function. In statistical MT, the zero-one loss function is better known as *sentence error rate* (SER)[b] and considers that there is an error if the translation given by the system $\hat{\boldsymbol{y}}$ is not identical to the reference translation.

In conclusion, by applying Eq. (1.21) we are minimising the probability of error using SER as a loss function. However, the SER measure provides a rough and superficial evaluation of the translation quality of a translation system and it is rarely used in favour of other more popular evaluation measures like *word error rate* (WER) and *bilingual evaluation understudy* (BLEU) [PRWZ01]. These evaluation measures, further explored in this

[b]SER in statistical MT is equivalent to CER in classification tasks.

thesis, suggest the usage of alternative loss functions, and therefore different decision rules that are closer to actual evaluation measures employed in statistical MT. We will focus on this idea in chapter 3.

The search problem presented in Eq. (1.22) was proveded to be an NP-complete problem [Kni99, UM06]. 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 [BPP96] over greedy [B+94, WW98, G+01] and integer-programming [G+01] approaches to dynamic-programming search [GVC01, TN03].

Nevertheless, most of the current statistical MT systems present an alternative modelisation of the translation process different from that presented in Eq. (1.21). The posterior probability is modelled as a log-linear combination of feature functions [ON04] under the framework of maximum entropy [BPP96]

$$\hat{\boldsymbol{y}} = \arg\max_{y} \sum_{m=1}^{M} \lambda_m h_m(\boldsymbol{x}, \boldsymbol{y}) \tag{1.23}$$

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 $(\boldsymbol{x}, \boldsymbol{y})$. We will get into deeper detail in chapter 3.

Under this framework Eq. (1.22) can be seen as a special case where

$$h_1(\boldsymbol{x}, \boldsymbol{y}) = \log p_r(\boldsymbol{x} \mid \boldsymbol{y}) \tag{1.24}$$

$$h_2(\boldsymbol{x}, \boldsymbol{y}) = \log p_r(\boldsymbol{y}) \tag{1.25}$$

and $\lambda_1 = \lambda_2 = 1$.

Most of state-of-the-art statistical MT systems are based on bilingual phrases [CB+07]. 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 [Wu96, YK01, GK04, Lin04, DP05]. 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 [Chi07], 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 [ABD00, BR95, CV04a, KN04, M+06]. 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.

1.3.1 Statistical word-based translation systems

A great variety of statistical translation models have been proposed since the IBM article was initially published [B+90, B+93]. In that article, 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 target words $\boldsymbol{a}_i = \{j_1, \ldots, j_l\}$, 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

$$p_r(\boldsymbol{y} \mid \boldsymbol{x}) = \sum_{\boldsymbol{a}_1} \cdots \sum_{\boldsymbol{a}_I} p_r(I \mid \boldsymbol{x}) p_r(\boldsymbol{y}, \boldsymbol{a}_1^I \mid \boldsymbol{x}, I) \tag{1.26}$$

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

$$\boldsymbol{a}_j \subseteq \{1, \ldots, I\} \tag{1.27}$$

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

Some constraints are usually added to the alignment sets $\boldsymbol{a}_1^I$, such as the *coverage constraint* that requires all the source words to be in at least one alignment set. However, these constrains are practically motivated.

The complete probability model in Eq. (1.26), $p_r(\boldsymbol{y}, \boldsymbol{a}_1^I \mid \boldsymbol{x})$, can be decomposed left to right as

$$p_r(\boldsymbol{y}, \boldsymbol{a}_1^I \mid \boldsymbol{x}, I) = \prod_i p_r(\boldsymbol{a}_i \mid \boldsymbol{x}, \boldsymbol{a}_1^{i-1}, \boldsymbol{y}_1^{i-1}, I) p_r(y_i \mid \boldsymbol{x}, \boldsymbol{a}_1^i, \boldsymbol{y}_1^{i-1}, I) \tag{1.28}$$

where two probabilities are used:

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

Different alignment models were proposed in [BPPM93] based on this idea, although only 2 models where directly modelled by parametrising the probabilities in Eq. (1.28). These two models constrained the cardinality of the alignment sets to 1 or 0, that is to say each source word can be aligned to either one word or no word. In order to simplify notation, we redefine the alignment variables since each aligment is composed of one word. Therefore, se 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 void alignment, a NULLword is introduced at the beginning of $\boldsymbol{x}$, i.e. $\boldsymbol{x} = x_0 x_1 \cdots x_J$ where x_0, the so-called NULLword, stands for the non-alignment event.

IBM model 1

The IBM model 1, the first of the IBM models, is basically defined as a statistical bilingual dictionary. The aim of the IBM model 1 typically is to initi the trainning 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 of this function under non-degenerated[c] initialisation.

The IBM model 1 [BPPM93] makes the following assumptions

- The alignment probability is uniform, i.e.

$$p_r(\boldsymbol{a}_i \mid \boldsymbol{x}, \boldsymbol{a}_1^{i-1}, \boldsymbol{y}_1^{i-1}, I) := \mathrm{p}(a_i) = \frac{1}{J+1} \tag{1.29}$$

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

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

$$p_r(y_i \mid \boldsymbol{x}, \boldsymbol{a}_1^i, \boldsymbol{y}_1^{i-1}, I) := p_r(y_i \mid x_{a_i}) \tag{1.30}$$

  where the following normalisation constraint must be verified

$$\sum_b \mathrm{p}(b \mid a) = 1 \qquad \text{forall source word } a \tag{1.31}$$

Taking into account the assumptions in Eqs. (1.29), and (1.30), the model probability is given by

$$p_r(\boldsymbol{y} \mid \boldsymbol{x}) := \sum_{a_1=0}^{J} \cdots \sum_{a_I=0}^{J} \prod_i \frac{1}{J+1}\, \mathrm{p}(y_i \mid x_{a_i}) \tag{1.32}$$

Simply reordering previous equation we obtain

$$p_r(\boldsymbol{y} \mid \boldsymbol{x}) := \prod_i \sum_{j=0}^{J} \frac{1}{J+1}\, \mathrm{p}(y_i \mid x_j) \tag{1.33}$$

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

The IBM model 1 has been widely employed in different applications 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 [Moo02], the alignment of syntactic tree fragments [DGP03], the segmentation of bilingual long sentences for improved word alignment [NCV03], the extraction of parallel sentences from comparable corpora [MFM04], the estimation of word-level confidence measures [UN07] and serves as inspiration for lexicalised phrase scoring in phrase-based systems [**?**, Koe05]. Furthermore, it has also received attention to improve its nonstructural problems [Moo04].

IBM model 2

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

$$p_r(\boldsymbol{a}_i \mid \boldsymbol{x}, \boldsymbol{a}_1^{i-1}, \boldsymbol{y}_1^{i-1}, I) := \mathrm{p}(a_i \mid i, I, J) \tag{1.34}$$

where the following normalisation constraint must be verified

$$\sum_j \mathrm{p}(j \mid i, I, J) = 1 \tag{1.35}$$

Taking into account the assumptions in Eqs. (1.34), and (1.30), the model probability is given by

$$p_r(\boldsymbol{y} \mid \boldsymbol{x}) := \prod_i \sum_j \mathrm{p}(j \mid i, I, J)\, \mathrm{p}(y_i \mid x_j) \tag{1.36}$$

Note that analogously to the model 1 we have exchanged the products and the sums in previous equation.

Since the model is a hidden variable model, the EM algorithm is used to estimate the parameter set, $\{\mathrm{p}(b \mid a), \mathrm{p}(j \mid i, I, J)\}$. In order to train this model, firstly, some iterations of the IBM model 1 are performed obtaining good dictionary estimates. Afterward 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. In this section we review several proposed phrase-based models

Generative phrase-based models

In this section, we outline 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 in the next section.

Let $(\boldsymbol{x}, \boldsymbol{y})$ be a pair of source-target sentences, we introduce the conventional conditional probability $p(\boldsymbol{y} \mid \boldsymbol{x})$ 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 of the target phrase. Let $\boldsymbol{\mu} = \{\mu_0, \mu_1, \ldots, \mu_T\}$ and $\boldsymbol{\gamma} = \{\gamma_0, \gamma_1, \ldots, \gamma_T\}$ be the segment boundary variables for the source and target sentences, respectively. An additional variable $\boldsymbol{a}$ is necessary for mapping each source phrase to one target phrase. 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 alignment between them,*

$$p_r(\boldsymbol{y} \mid \boldsymbol{x}) = \sum_{T=1}^{\min(|\boldsymbol{x}|,|\boldsymbol{y}|)} \sum_{\boldsymbol{\mu}} \sum_{\boldsymbol{a}} \sum_{\boldsymbol{\gamma}} p_r(\boldsymbol{y}, \boldsymbol{a}, \boldsymbol{\mu}, \boldsymbol{\gamma}, T \mid \boldsymbol{x}) \tag{1.37}$$

where

$$\begin{aligned} p_r(\boldsymbol{y}, \boldsymbol{a}, \boldsymbol{\mu}, \boldsymbol{\gamma}, T \mid \boldsymbol{x}) =& p_r(T \mid \boldsymbol{x}) p_r(\boldsymbol{\mu} \mid T, \boldsymbol{x}) p_r(\boldsymbol{\gamma} \mid \boldsymbol{\mu}, T, \boldsymbol{x}) \\ & p_r(\boldsymbol{a} \mid \boldsymbol{\mu}, \boldsymbol{\gamma}, T, \boldsymbol{x}) p_r(\boldsymbol{y} \mid \boldsymbol{a}\boldsymbol{\mu}, \boldsymbol{\gamma}, T, \boldsymbol{x}) \end{aligned} \tag{1.38}$$

The Eq (1.38) can be understood as a generation process where

1. Firstly, we decide on the number of segments T,
2. we split the source sentences by means of the source segmentation $\boldsymbol{\mu}$,
3. afterwards, we define the output segmentation with $\boldsymbol{\gamma}$ given the source phrases, and
4. we align the source phrases with the target segments with the alignment variable $\boldsymbol{a}$,
5. and finally, we generate the output phrases given all the previous information

The last probability in Eq. (1.38), $p_r(\boldsymbol{y} \mid \boldsymbol{a}, \boldsymbol{\mu}, \boldsymbol{\gamma}, T, \boldsymbol{x})$ is usually modelled depending only in the reoredered source phrases $\hat{\boldsymbol{x}}_1^T$, so that the translation process can be decomposed left to right

$$p_r(\boldsymbol{y} \mid \boldsymbol{a}\boldsymbol{\mu}, \boldsymbol{\gamma}, T, \boldsymbol{x}) := \prod_{t=1}^{T} \mathrm{p}(\hat{y}_t \mid \hat{x}_t) \tag{1.39}$$

where $\hat{\boldsymbol{y}}_1^T$ stands for the target phrases.

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. In the model proposed above, this segmentation is explained by the hidden variables T, $\boldsymbol{\mu}$ and $\boldsymbol{\gamma}$, which leads us to a large combinatorial number of possible segmentations to explore. Furthemore, the other main bottleneck that phrase-based models have is to explore all the possible aligments between the source and target phrases. 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 [MW02, BCBMOK06], over the HMM phrase-based models [DB05, AFJC07] to the statistical GIATI model [AFJCC08].

However, the most popular approach to the development of phrase-based systems has been the log-linear combination of heuristically estimated phrase-based models [**?**, ON04], since these systems offer similar or even better performance than those based on generative phrase-based models [DGZK06].

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 $\boldsymbol{\theta}$,

$$\hat{a} = \arg\max_a \mathrm{p}_{\boldsymbol{\theta}}(a \mid x, y) \tag{1.40}$$

can also rewritten

$$\hat{a} = \arg\max_a \mathrm{p}_{\boldsymbol{\theta}}(x, a \mid y). \tag{1.41}$$

The conventional alignments, for instance 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 [**?**, ON03]. 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 [ON03] and the *grow-diag-final* [**?**] 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,

$$\begin{aligned} (\overline{x}, \overline{y}) \text{ consistent with } a \Leftrightarrow & \quad \forall x_j \in \overline{x} : (x_j, y_i) \in a \longrightarrow y_i \in \overline{y} \wedge \\ \wedge & \quad \forall y_i \in \overline{y} : (x_j, y_i) \in a \longrightarrow x_j \in \overline{x} \wedge \\ \wedge & \qquad \exists x_j \in \overline{x},\, y_i \in \overline{y} : (x_j, y_i) \in a \end{aligned} \tag{1.42}$$

basically Eq. (1.42) 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.

[d]This process is also known as symmetrisation.

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 seamlessly 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

$$p_d(\overline{x} \,|\, \overline{y}) = \frac{count(\overline{x}, \overline{y})}{\sum\limits_{\overline{x}} count(\overline{x}, \overline{y})} \qquad p_i(\overline{y} \,|\, \overline{x}) = \frac{count(\overline{x}, \overline{y})}{\sum\limits_{\overline{y}} count(\overline{x}, \overline{y})} \tag{1.43}$$

as well as the direct and inverse lexical translation probability inspired in the M1 model [**?**, CL07]. Other score functions are related to reordering capabilities, such as the distance-based reordering model [ON04] and the lexicalised reordering model [K$^+$05]. Additional score functions are phrase and 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 [Och03] and minimum Bayes risk [KB04]. Furthermore, the most common approach to the decoding process in log-linear models is the well-known multi-stack decoding algorithm [Koe04, ON04]. The Moses toolkit [K$^+$07], 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 employed 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 [CB$^+$07]. 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 [BL05] and translation edit rate (TER) [S$^+$06] are becoming more and more popular.

The WER metric [A$^+$00, C$^+$04] 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 1.0, 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 [PRWZ01] 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

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

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 [CBOK06], it is still a reference error measure for the evaluation of translation quality in MT systems. We take BLEU as a percentage ranging from 0.0 (worst score) to 100.0 (best score).

1.4 Text classification

The problem of text classification is stated as the problem of determining which document category c a given document $\boldsymbol{w}$ belongs to. This problem can be reformulatted as a pattern recognition problem where the documents are the objects to classify and the document categories, $\{1, \ldots, C\}$, are the set of classes. Using the optimal Bayes' rule for the CER loss depicted in Eq.(1.9), the following rule is obtained

$$\hat{\mathrm{c}}(\boldsymbol{w}) = \operatorname*{arg\,max}_{c \in \{1,\ldots,C\}} p_r(c) p_r(L \mid c) p_r(\boldsymbol{w} \mid c, L) \tag{1.44}$$

where we have decomposed the posterior probability $p_r(c \mid \boldsymbol{w})$ into 3 different factors: a *prior class* probability $p_r(c)$, a length model $p_r(L \mid c)$, and a document probability $p_r(\boldsymbol{w} \mid c, L)$.

The legnth model in Eq. (1.44) is usually assumed to be constant and then, the classification rule is simplified to:

$$\hat{\mathrm{c}}(\boldsymbol{w}) = \operatorname*{arg\,max}_{c \in \{1,\ldots,C\}} p_r(c) p_r(\boldsymbol{w} \mid c, L) \tag{1.45}$$

where the prior class probability is used modelled in a non-parametric fashion

$$p_r(c) := \mathrm{p}(c) \tag{1.46}$$

The document probability is more difficult to model, however, the Naive Bayes assumption is often assumed. The Naive Bayes assumption considers that there is no relationship between words, that is to say, that the probability of each word d in a document is only dependent on the document category or class, c,

$$p_r(\boldsymbol{w} \mid c, L) := p_r(\boldsymbol{w} \mid c) := \prod_{l=1}^{L} \mathrm{p}(w_l \mid c) \tag{1.47}$$

The *naive Bayes* text classifier has long been a core technique in information retrieval and, more recently, it has attracted significant interest in pattern recognition and machine learning [Lew98].

The naive Bayes assumption in text classification has the advantage of greatly simplifying maximum likelihood estimation of unknown class-conditional word occurrence probabilities. Although the simplicity of this assumption it yields surprisingly good results [**?**, **?**].

As stated above, the Bayes rule is the optimal decision when we consider CER as evaluation metric. However, this is only the case under the assumption that we know the real probability distributions for $p_r(c)$ and $p_r(\boldsymbol{w} \mid c, L)$. In practise, we can only compute approximations of these probability distributions.

Apart from those classifiers based on the statistical PR approach, different types of classifiers have been used in TC, including regression methods [FP94, IDLA95, LG94, SHP95], decision trees, neural networks [Mit96], incremental or batch methods for learning linear classifiers [SHP95, WPW95, DKR97, NGL97], classifier ensembles, including boosting methods [SS00], and support vector machines [Joa98]. While all these techniques still retain their popularity, it is fair to say that in recent years support vector machines and boosting have been the two dominant learning methods in TC. This fact is mainly due to their superiority on the Reuters task, which is one of the reference task in TC, however their performance is similar to that of other TC techniques in other tasks. The interested reader is referred to [Seb02] for an excellent review in TC.

1.5 Scientific contributions

The objective of this thesis is focused on statisctical pattern recognition. More precisely, the contributions of this thesis imply two basic research lines of statistical pattern recognition: model estimation and loss function.

1. **Constrained domain maximum likelihood estimation.** Constrained domain maximum likelihood estimation (CDMLE), is a modification of the maximum likelihood estimation criterion. This modification yields an already smoothed optimal parameter set, and avoids the need of an additional smoothing step. Furthermore, the CDMLE retains the good theoretical properties that MLE verifies.
2. **Loss function.** We expand the usual CER loss in pattern recogntion by researching other more general loss functions. Specifically, two families of loss functions are analysed. Specially appealing is the family which is a tradeoff between generalisation and computational time.

BIBLIOGRAPHY

[A$^+$93] D.J. Arnold et al. *Machine Translation: an Introductory Guide*. Blackwells-NCC, London, 1993.

[A$^+$00] J. C. Amengual et al. The EuTrans-I speech translation system. *Machine Translation*, 15:75–103, 2000.

[ABC$^+$00a] 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, 2000.

[ABC$^+$00b] 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.

[ABD00] H. Alshawi, S. Bangalore, and S. Douglas. Learning dependency translation models as collections of finite state head transducers. *Computational Linguistics*, 26, 2000.

[ADB00] H. Alshawi, S. Douglas, and S. Bangalore. Learning dependency translation models as collections of finite-state head transducers. *Computational Linguistics*, 26(1):45–60, 2000.

[AFJC07] J. Andrés-Ferrer and A. Juan-Císcar. A phrase-based hidden markov model approach to machine translation. In *Proceedings of New Approaches to Machine Translation*, pages 57–62, January 2007.

[AFJCC08] J. Andrés-Ferrer, A. Juan-Císcar, and F. Casacuberta. Statistical estimation of rational transducers applied to machine translation. *Applied Artificial Intelligence*, page In press, 2008.

[B$^+$90] P. F. Brown et al. A Statistical Approach to Machine Translation. *Computational Linguistics*, 16(2):79–85, 1990.

[B$^+$93] P. F. Brown et al. The Mathematics of Statistical Machine Translation: Parameter Estimation. *Computational Linguistics*, 19(2):263–311, 1993.

[B$^+$94] 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.

[BCBMOK06] 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.

[Ber79] J. Berstel. *Transductions and context-free languages*. B. G. Teubner Stuttgart, 1979.

[BL05] 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.

[Bow02] L. Bowker. *Computer-aided translation technology: A practical introduction*, chapter 5: Translation-memory systems, pages 92–127. Didactics of Translation. University of Ottawa Press, 2002.

[BPP96] A. L. Berger, V. J. Della Pietra, and S. A. Della Pietra. A maximum entropy approach to natural language processing. *Computational Linguistics*, 22(1):39–71, 1996.

[BPPM93] 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.

[BR95] 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.

[BR03] S. Bangalore and G. Riccardi. Stochastic finite-state models for spoken language machine translation. *Machine Translation*, 17(3):165–184, 2003.

[BS05] 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.

[BV04] Stephen Boyd and Lieven Vandenberghe. Convex optimization. March 2004.

[C$^+$04] F. Casacuberta et al. Some approaches to statistical and finite-state speech-to-speech translation. *Computer Speech and Language*, 18:25–47, 2004.

[Cas00a] 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, 2000.

[Cas00b] 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, 2000. 5th International Colloquium Grammatical Inference -ICGI2000-. Lisboa. Portugal.

[CB$^+$07] 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.

[CBOK06] 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.

[CG96] 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.

[CG98] 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.

[Chi07] D. Chiang. Hierarchical phrase-based translation. *Computational Linguistics*, 33(2):201–228, 2007.

[CL07] 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.

[CV04a] F. Casacuberta and E. Vidal. Machine translation with inferred stochastic finite-state transducers. *Computational Linguistics*, 30(2):205–225, 2004.

[CV04b] F. Casacuberta and E. Vidal. Machine translation with inferred stochastic finite-state transducers. *Computational Linguistics*, 30(2):205–225, 2004.

[CVP05] F. Casacuberta, E. Vidal, and D. Picó. Inference of finite-state transducers from regular languages. *Pattern Recognition*, 38:1431–1443, 2005.

[DB05] 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.

[DGP03] 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.

[DGZK06] 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.

[DH73] R. O. Duda and P. E. Hart. *Pattern Classification and Scene Analysis*. Wiley, 1973.

[DKR97] I. Dagan, Y. Karov, and D. Roth. Mistakedriven learning in text categorization. In *Proc. of EMNLP'97*, pages 55–63, Cambridge, MA, USA, 1997.

[DLR77] 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, 1977.

[DP05] 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.

[EVS89] 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.

[FP94] N. Fuhr and U. Pfeifer. Probabilistic information retrieval as combination of abstraction, inductive learning and probabilistic assumptions. *ACM Transactions on Information Systems*, 12(1):92–115, 1994.

[G$^+$01] 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.

[GK04] 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.

[Goo01] Joshua Goodman. A bit of progress in language modeling. *CoRR*, cs.CL/0108005, 2001.

[GVC01] 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.

[HS92] J. Hutchins and H. L. Somers. *An introduction to machine translation*. Academic Press, 1992.

[IDLA95] D. J. Ittner, D. D. D. Lewis, and D. D. Ahn. Text categorization of low quality images. In *Proc. of SDAIR'95*, pages 301–315, April 1995.

[IO99] R. M. Iyer and M. Ostendorf. Modelling long distance dependence in language: Topic mixtures versus dynamic cache models. *IEEE Transactions on Speech & Audio Processing*, 7(1):30–39, 1999.

[Jel97] F. Jelinek. *Statistical Methods for Speech Recognition*. MIT Press, 1997.

[Joa98] T. Joachims. Text Categorization with Support Vector Machines: Learning with Many Relevant Features. In *Proc. of ECML'98*, pages 137–142, April 1998.

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

[K$^+$07] 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.

[KAO98] 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.

[Kat87] 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.

[KB03] 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.

[KB04] 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.

[KDB06] 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.

[KN95] 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.

[KN04] 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.

[Kni99] K. Knight. Decoding complexity in word-replacement translation models. *Computional Linguistics*, 25(4):607–615, 1999.

[Koe04] 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.

[Koe05] P. Koehn. Europarl: A parallel corpus for statistical machine translation. In *Proc. of the MT Summit X*, pages 79–86, September 2005.

[KS93] R. Kneser and V. Steinbiss. On the dynamic adaptation of stochastic language models. In *Proc. of ICASSP'93*, volume II, pages 586–589, April 1993.

[Lew98] 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.

[LG94] D. D. Lewis and W. A. Gale. A sequential algorithm for training text classifiers. In W. Bruce Croft and Cornelis J. Van Rijsbergen, editors, *Proc. of SIGIR'94*, pages 3–12, Dublin, Ireland, 1994. Springer Verlag, Heidelberg, Germany.

[LGLL05] 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.

[Lin04] 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.

[LVC02] 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.

[M$^+$06] J. B. Mariño et al. N-gram-based machine translation. *Computational Linguistics*, 32(4):527–549, 2006.

[Mäk99] E. Mäkinen. Inferring finite transducers. Technical Report A-1999-3, University of Tampere, 1999.

[MFM04] 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.

[Mit96] T. M. Mitchell. *Machine learning*. McGraw Hill, New York, 1996.

[Moo02] R.C. Moore. Fast and accurate sentence alignment of bilingual corpora. In *Proc. of AMTA'02*, pages 135–244, October 2002.

[Moo04] R.C. Moore. Improving IBM Word-Alignment Model 1. In *Proc. of ACL'04*, pages 519–524, July 2004.

[MW02] 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.

[N$^+$92] S. Nirenburg et al. *Machine Translation: A Knowledge-based Approach*. Morgan Kaufmann, 1992.

[NCV03] F. Nevado, F. Casacuberta, and E. Vidal. Parallel corpora segmentation using anchor words. In *Proc. of EAMT/CLAW'03*, pages 33–40, May 2003.

[NGL97] H. T. Ng, W. B. Goh, and K. L. Low. Feature selection, perceptron learning, and a usability case study for text categorization. In Nicholas J. Belkin, A. Desai Narasimhalu, and Peter Willett, editors, *Proc. of SIGIR'97*, pages 67–73, Philadelphia, USA, 1997. ACM Press, New York, USA.

[NH98] 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.

[NM92] 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.

[Och03] 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.

[OGV93] 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.

[ON03] F. J. Och and H. Ney. A systematic comparison of various statistical alignment models. *Computational Linguistics*, 29(1):19–51, 2003.

[ON04] F. J. Och and H. Ney. The alignment template approach to statistical machine translation. *Computational Linguistics*, 30(4):417–449, 2004.

[PC01] David Picó and Francisco Casacuberta. Some statistical-estimation methods for stochastic finite-state transducers. *Machine Learning*, 44:121–142, July-August 2001.

[PJR07] D. Pinto, A. Juan, and P. Rosso. Using query-relevant documents pairs for cross-lingual information retrieval. In *Proc. of TSD'07*, pages 630–637, September 2007.

[PRWZ01] 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.

[Rab89] Lawrence Rabiner. A tutorial on hmm and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, February 1989.

[Rob56] Herbert Robbins. An empirical bayes approach to statistics. *Proceeding of the Third Berkeley Symposium on Mathematical Statistics*, 1:157–163, 1956.

[S$^+$06] 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.

[Seb02] F. Sebastiani. Machine learning in automated text categorisation. *ACM Computing Surveys*, 34(1):1–47, 2002.

[SHP95] Hinrich Schütze, David A. Hull, and Jan O. Pedersen. A comparison of classifiers and document representations for the routing problem. In Edward A. Fox, Peter Ingwersen, and Raya Fidel, editors, *Proc. of SIGIR'95*, pages 229–237, Seattle, US, July 1995. ACM Press, New York, US.

[SM06] 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.

[Som99] H. L. Somers. Review article: Example-based machine translation. *Machine Translation*, 14(2):113–157, 1999.

[Som03] H. L. Sommers. *Computers and translation: a translator's guide*, chapter 3: Translation memory systems, pages 31–48. John Benjamins, 2003.

[SS00] R. E. Schapire and Y. Singer. Boostexter: A boosting-based systemfor text categorization. *Machine Learning*, 39(2-3):135–168, 2000.

[TN03] 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.

[TN04] 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.

[UM06] R. Udupa and H. K.r Maji. Computational complexity of statistical machine translation. In *Proc. of EACL'06*, April 2006.

[UN07] N. Ueffing and H. Ney. Word-level confidence estimation for machine translation. *Computational Linguistics*, 33(1):9–40, 2007.

[Vil00] 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.

[VN00] 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.

[VTCdlHC05a] 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, 2005.

[VTCdlHC05b] 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, 2005.

[WPW95] E. D. Wiener, J. O. Pedersen, and A. S. Weigend. A neural network approach to topic spotting. In *Proc. of SDAIR'95*, pages 317–332, April 1995.

[Wu83] C. F. Jeff Wu. On the convergence properties of the EM algorithm. *The Annals of Statistics*, 11(1):95–103, 1983.

[Wu95] 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.

[Wu96] 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.

[WW98] Y. Wang and A. Waibel. Fast decoding for statistical machine translation. In *Proc. of ICSLP'98*, pages 2775–2778, October 1998.

[YK01] 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.

CHAPTER 2

CONSTRAINED DOMAIN MAXIMUM LIKELIHOOD ESTIMATION

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

Contents

ALMOST all statistical pattern recognition systems are built using probability distributions despite actual distribution functions are never known in real tasks. In order to approximate such distributions a statistical model is often defined. Each statistical model is characterised by a probability function parametrised with a set of parameters $\boldsymbol{\theta} \in \boldsymbol{\Theta}$.

In this framework, the question of which the actual distribution is can be restated as finding the optimal parameter set. Since the training data is assumed to have been generated by the probability model, this data should be very likely. One of the most spreaded criterium to select the set of parameters is maximum likelihood estimation (MLE). This criterum finds the most likely set of parameters, that is to say, the parameter set that gives the most probability to the training data.

One of the most important problems of this approach is that by selecting the parameters that give maximum likelihood (ML) to the training data, the model is over-trained, that is to say that the model memorises the data without enough level of generalisation. Note that this results follows from the fact that the actual set of parameters will not give the highest probability to the trainning data. The over-training shortcoming is often alleviated by "smoothing the maximum likelihood parameters", i.e., by wisely disturbing the ML set of parameters. This smoothing is a heuristic process which could be even more complex than finding the optimal set of parameters.

Smoothing the MLE makes the disturbed optimal parameter set not to hold most of the desirable theoretical properties of the MLE. In this chapter, instead of "smoothing" the ML set of parameters, we "smooth" the domain itself, i.e. we apply some constraints into the parameter domain avoiding the over-trained parameters. Specifically, we constraint the multinomial distribution. The selection of this distribution is motivated because its simplicity and success when applied to real text classification problems.

2.1 Introduction

Most of the pattern recognition systems are based into the optimal Bayes' rule (see section 1.1 and section 3.2). This rule makes use of 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 set of parameters, $\boldsymbol{\theta} \in \boldsymbol{\Theta}$.

The selection of the optimal $\boldsymbol{\theta}$ depends on the function criterium. As reviewed in section 1.1.3, maximum likelihood estimation (MLE) is one of the most widespread techniques. This criterium finds the set of parameters $\hat{\boldsymbol{\theta}}$ that maximises the likelihood function which is defined in section 1.1.3. One of the most important flaws of this criterium is that it tends to over-fit the parameters to the training data at the expense of reserving small probabilities or even zero probability to the remaining data. This over-training problem is often due to 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 over-fitting 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 over-training. Instead of smoothing the optimal solution obtained by MLE, we support the idea of constraining the domain of the parameters, $\boldsymbol{\Theta}$, before maximising them. In this way, the optimal set of parameters is smoothed in the optimisation of the parameters itself provided that there is no possible over-trained set of parameters in the domain. Even more, the optimal set of parameters obtained from the constrained domain retains the properties of the MLE whilst the classical smoothed set of parameters does not.

We apply the idea of *constrained domain maximum likelihood estimation* (CDMLE) to the multinomial probability distribution. This probability is used in several pattern recognition problems, such as text classification [MN98, JN02, VNJV04] or image classification.Specifically, we study how to avoid over-training in the text classification context.

The chapter is organised as follows. Firstly in the following section 2.2, some basic convex optimisation theory is revised. In section 2.3, the multinomial distribution is introduced reviewing the classical MLE of such probability distribution. The proposed constrained maximum likelihood estimation is analysed for the multinomial distribution in section 2.4. Section 2.5 is devoted to the application of our constrained approach to the text classification problem. In section 2.6, the experimental behaviour of our approximation is evaluated in both synthetic and real data. Concluding remarks are gathered in section 2.7.

2.2 Karush-Kuhn-Tucker Conditions

In section 1.1.2 we have analysed that in order to obtain the optimal set of parameters according to our criterion, it is needed to optimise Eq. (1.12). Almost all the optimisation problems derive from Eq. (1.12), are subject at least to some normalisation constraint. In order to solve those optimisation problems with constraints it is used the *convex optimisation* theory [BV04].

A typical example of a complex optimisation is the following. We want to solve the following equation

$$\hat{\boldsymbol{\theta}} = \operatorname*{arg\,max}_{\boldsymbol{\theta}\in\boldsymbol{\Theta}} \{\mathcal{C}(\boldsymbol{\theta}; D)\} \tag{2.1}$$

subject to

$$\begin{aligned} \mathrm{P}_1(\boldsymbol{\theta}) &= 0 \\ &\dots \\ \mathrm{P}_N(\boldsymbol{\theta}) &= 0 \end{aligned} \tag{2.2}$$

In order to solve the previous optimisation the *Lagrangian function* must be defined

$$\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\lambda}) = \mathcal{C}(\boldsymbol{\theta}; D) - \sum_n \lambda_n \, \mathrm{P}_n(\boldsymbol{\theta}) \tag{2.3}$$

where a *Lagrangian multiplier* (λ_n) is defined for each constraint P_n.

Theory states that, solving Eq. (1.12), subject to Eq. (2.2) is equivalent to solving the following problem

$$\hat{\boldsymbol{\theta}} = \operatorname*{arg\,max}_{\boldsymbol{\theta}\in\boldsymbol{\Theta}} \{\max_{\boldsymbol{\lambda}} \mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\lambda})\} \tag{2.4}$$

Therefore, the optimal point must verify the following property

$$\nabla \mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\lambda})|_{\hat{\boldsymbol{\theta}}, \hat{\boldsymbol{\lambda}}} = 0 \tag{2.5}$$

rising up a linear system from which the value of $\hat{\boldsymbol{\theta}}$ can be worked out.

The previous optimisation example is typically known as an *equality constrained program*. Through this chapter, however, we solve some optimisation problems which also include inequality constraints. In order to solve problems with inequality constraints the *Karush-Kuhn-Tucker (KKT)* conditions are needed. The problem is now stated as the equality constraints but with the additional inequality constraints, i.e. solving Eq. (2.1) subject to Eq. (2.2) and to

$$\begin{aligned} \mathrm{Q}_1(\boldsymbol{\theta}) &\leq 0 \\ &\dots \\ \mathrm{Q}_M(\boldsymbol{\theta}) &\leq 0 \end{aligned} \tag{2.6}$$

In this case, the Lagrangian function is defined as

$$\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\lambda}, \boldsymbol{\mu}) = -\,\mathcal{C}(\boldsymbol{\theta}; D) + \sum_n \lambda_n \, \mathrm{P}_n(\boldsymbol{\theta}) + \sum_m \mu_m \, \mathrm{Q}_m(\boldsymbol{\theta}) \tag{2.7}$$

Solving Eq. (1.12) subject to Eq. (2.2) and to Eq. (2.6) is the same than solving

$$\hat{\boldsymbol{\theta}} = \arg\min_{\boldsymbol{\theta} \in \boldsymbol{\Theta}} \min_{\boldsymbol{\lambda}, \boldsymbol{\mu}} \mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\lambda}, \boldsymbol{\mu}) \tag{2.8}$$

The KKT necessary conditions for a point $(\boldsymbol{\theta}, \boldsymbol{\lambda}, \boldsymbol{\mu})$ to be a maximum point are

$$\nabla_{\boldsymbol{\theta}} \, \mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\lambda}, \boldsymbol{\mu})|_{\hat{\boldsymbol{\theta}}, \hat{\boldsymbol{\lambda}}, \hat{\boldsymbol{\mu}}} = 0 \tag{2.9}$$

$$\begin{aligned} \mathrm{P}_1(\boldsymbol{\theta}) &= 0 \\ &\dots \\ \mathrm{P}_N(\boldsymbol{\theta}) &= 0 \end{aligned} \tag{2.10}$$

$$\begin{aligned} \mu_1 \, \mathrm{Q}_1(\boldsymbol{\theta}) &= 0 \\ &\dots \\ \mu_M \, \mathrm{Q}_M(\boldsymbol{\theta}) &= 0 \end{aligned} \tag{2.11}$$

$$\begin{aligned} \mu_1 &\geq 0 \\ &\dots \\ \mu_M &\geq 0 \end{aligned} \tag{2.12}$$

$$\begin{aligned} \mathrm{Q}_1(\boldsymbol{\theta}) &\leq 0 \\ &\dots \\ \mathrm{Q}_M(\boldsymbol{\theta}) &\leq 0 \end{aligned} \tag{2.13}$$

It is worth noting that the KKT conditions are *necessary* conditions but not sufficient conditions. That is to say that a maximum point must verify them, but not all points that verify them are a maximum point. 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 [BV04]. Once the possible optimal points are given then checking if the latter sufficient and necessary condition is verified is a simple mathematical exercise. However, in most of the cases if the characterisation or form 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. Anyway, once the form of the solution is known, it is often possible to define an efficient algorithm that obtains such a solution.

2.3 The Multinomial distribution

In this section we review the maximum likelihood estimation (MLE) for the multinomial distribution as well as some generic and simple smoothing techniques for this distribution. Afterwards, we introduce the constrained maximum likelihood estimation algorithm.

We say that a D-dimensional random vector of natural numbers $\boldsymbol{x} \in \mathbb{N}^D$ follows a multinomial distribution of parameter vector $\boldsymbol{\theta}_1^D$ and length L if its probability mass function is expressed as follows,

$$p_{\boldsymbol{\theta}}(\boldsymbol{x}|L) = \left(\begin{array}{c} L \\ \boldsymbol{x} \end{array} \right) \prod_{d=1}^{D} \theta_d^{x_d} \tag{2.14}$$

where

$$\sum_{d=1}^{D} \theta_d = 1 \tag{2.15}$$

and with

$$\left(\begin{array}{c} L \\ \boldsymbol{x} \end{array} \right) = \frac{L!}{\prod_{d=1}^{D} x_d!} \tag{2.16}$$

The basis of the multinomial distribution is the existence of D (classes of) events, $d = 1, \cdots, D$, and that x_d is the count of the events belonging to the class of events d. The events are assumed to be independent one of the other. For instance, the classical example of an experiment that follows a multinomial distribution is the following. There are D bags each full of differently coloured balls. If we extract L balls with replacement, then the vector that counts extracted balls of each colour, follows a multinomial distribution. Another outstanding example is the probability of a text with naive Bayes assumption. Assuming that the words appearing in a document are not related one to another, then if the vocabulary has D words and the length of the documents is L, the word count vector follows a multinomial distribution (see section 2.5 for mathematical details).

The remaining of this section is focused on the multinomial training. The multinomial training refers to the problem of deciding a criterion and a method to compute the $\hat{\boldsymbol{\theta}}$ that has generated a i.i.d. sample, $D = \{\boldsymbol{x}_n\}_{n=1}^N$. As stated in section 2.1, we focus the study on the MLE criterion.

2.3.1 Maximum likelihood training

The likelihood criterion consist in maximising the so-called *likelihood function* for a given sample, $D = \{\boldsymbol{x}_n\}_{n=1}^N$. Since the logarithm is a increasing function, maximising the likelihood function, is equivalent that maximising its logarithmic version, the so-called *log-likelihood (LL)* (see 1.1.3). Specifically, in the multinomial case the LL simplifies to

$$\begin{aligned} \mathrm{LL}(\boldsymbol{\theta}; D) &= \textstyle\sum_n \log \left(\begin{array}{c} L \\ \boldsymbol{x} \end{array} \right) + \sum_n \sum_d x_{nd} \log \theta_d && (2.17) \\ &= \mathrm{const}(\boldsymbol{\theta}) + \textstyle\sum_n \sum_d x_{nd} \log \theta_d && (2.18) \end{aligned}$$

Therefore, the maximum likelihood estimate is expressed as

$$\hat{\boldsymbol{\theta}} = \arg\max_{\boldsymbol{\theta}} \sum_n \sum_d x_{nd} \log \theta_d \tag{2.19}$$

subject to the normalisation constraint in Eq. (2.15).

In order solve the previous optimisation is necessary to make use of convex optimisation techniques. Specifically, the Lagrange multipliers are specially useful for this case (see section 2.2). In this case the Lagrangian function is

$$\mathcal{L}(\boldsymbol{\theta}, \lambda) = \mathrm{LL}(\boldsymbol{\theta}) + \boldsymbol{\Lambda}(\boldsymbol{\theta}, \lambda) \tag{2.20}$$

with

$$\boldsymbol{\Lambda}(\boldsymbol{\theta}, \lambda) = -\lambda \left(\sum_d \theta_d - 1 \right) \tag{2.21}$$

The maximum vector of parameters must verify that the partial derivative of the Lagrangian function $\mathcal{L}(\cdots)$ depicted in Eq. (2.20) is equal to 0 in that point. The partial derivatives are the following

$$\frac{\partial \mathcal{L}(\boldsymbol{\theta}, \lambda)}{\partial \theta_d} = \sum_n \frac{x_{nd}}{\theta_d} - \lambda \tag{2.22}$$

$$\frac{\partial \mathcal{L}(\boldsymbol{\theta}, \lambda)}{\partial \lambda} = \sum_d \theta_d - 1 \tag{2.23}$$

Constraining Eqs. (2.22), and (2.23) to 0 and working out the value of θ_d yields the following solution,

$$\hat{\theta}_d = \frac{N_d}{N} \tag{2.24}$$

with N_d being the occurrences in the training data of events of class d, i.e.

$$N_d = \sum_n x_{nd} \tag{2.25}$$

and N the total number of outcomes

$$N = \sum_d N_d \tag{2.26}$$

Despite of the optimality of the solution in Eq. (2.24), it is usually *smoothed* as commented in section 2.1. In this specific distribution, the motivation of the smoothing can be seen in an example. For instance, if in our data, one of the counts N_d is equal to 0, i.e. in the training no ball from the d bag is extracted (or the d-th word in the vocabulary is not observed in the data); then the MLE parameter is $\theta_d = 0$. This implies that if we see another dataset, then the simple fact of extracting just one ball of this bag (or an occurrence of d-th word of the vocabulary) makes the model to give 0 probability to this dataset. This undesirable property can be avoided by means of the smoothing.

A popular smoothing method for (2.24) consists of simply adding a “pseudo-count” $\delta > 0$ to every N_d count leading to the following smoothed vector

$$\tilde{\theta}_d = \frac{N_d + \delta}{\sum_{d'} (N_{d'} + \delta)} \tag{2.27}$$

with $\delta = 1$ as the default value. This method is sometimes referred to as *Laplace smoothing* [MN98].

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 [JN02, VNJV04]. 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,

$$\beta_d = \frac{1}{D} \tag{2.28}$$

or whatever distribution depending on d. 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:

$$\tilde{\theta}_d = \begin{cases} \dfrac{N_d - b}{\sum_{d'} N_{d'}} & \text{if } N_d > 0 \\ \\ M \dfrac{\beta_d}{\sum_{d':N_{d'}=0} \beta_{d'}} & \text{if } N_d = 0 \end{cases} \tag{2.29}$$

where the probability mass gained in class c is:

$$M = \frac{b\,|\{d' : N_{d'} > 0\}|}{\sum_{d' \geq 1} N_{d'}} \tag{2.30}$$

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

$$\tilde{\theta}_d = \max\left\{0, \frac{N_d - b}{\sum_{d'} N_{d'}}\right\} + M\,\beta_d \tag{2.31}$$

where $0 < b \leq 1$.

2.4 Constrained domain maximum likelihood

As discussed in the introduction (see 2.1), 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 work, 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 MLE naive Bayes training constrained to probability estimates not smaller than a predefined non-negative constant ϵ. That is, we are interested in the maximisation of (2.17) subject to the normalisation constraint in Eq. (2.15), and also subject to

$$\theta_d \geq \epsilon \qquad (d = 1, \ldots, D) \tag{2.32}$$

where ϵ is the minimum probability of extracting a ball of any bag ($0 \leq \epsilon \leq \frac{1}{D}$), or alternatively, the minimum occurrence of any word of the vocabulary in any document. 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 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 (2.17) subject to constraints (2.15) and (2.32), 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 section 2.2). The Lagrangian function is

$$\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\lambda}, \boldsymbol{\mu}) = -L(\boldsymbol{\theta}) + \boldsymbol{\Lambda}(\boldsymbol{\theta}, \lambda) + \boldsymbol{\Phi}(\boldsymbol{\theta}, \boldsymbol{\mu}) \tag{2.33}$$

with

$$\boldsymbol{\Lambda}(\boldsymbol{\theta}, \lambda) = \lambda \left(\sum_d \theta_d - 1 \right) \tag{2.34}$$

and with

$$\boldsymbol{\Phi}(\boldsymbol{\theta}, \boldsymbol{\mu}) = \sum_d \mu_d \, (\epsilon - \theta_d) \tag{2.35}$$

and where λ and μ_d are Lagrange multipliers associated with constraints (2.15) and (2.32), respectively $(d = 1, \ldots, D)$. The KKT conditions for a point $\hat{\boldsymbol{\theta}}$, $\hat{\lambda}$, $\hat{\boldsymbol{\mu}}$ to be a global maximum are

$$\nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}, \lambda, \boldsymbol{\mu}) \big|_{\hat{\boldsymbol{\theta}}, \hat{\lambda}, \hat{\boldsymbol{\mu}}} = \mathbf{0} \tag{2.36}$$

$$\sum_d \hat{\theta}_d = 1 \tag{2.37}$$

$$\hat{\theta}_d \geq \epsilon \qquad (d = 1, \ldots, D) \tag{2.38}$$

$$\hat{\mu}_d (\epsilon - \hat{\theta}_d) = 0 \qquad (d = 1, \ldots, D) \tag{2.39}$$

$$\hat{\mu}_d \geq 0 \qquad (d = 1, \ldots, D) \tag{2.40}$$

From Eqs. (2.36), (2.37), (2.38), (2.39), and (2.40) immediately follows that, the optimal vector of parameters cannot be computed in closed-form. From (2.36), we have

$$\hat{\theta}_d = \frac{1}{\hat{\lambda} + \hat{\mu}_d} N_d \qquad (d = 1, \ldots, D) \tag{2.41}$$

but now we cannot rewrite $\hat{\lambda} + \hat{\mu}_d$ in terms of word counts to arrive at a closed-form solution like (2.24). Instead, by some straightforward manipulations, we arrive at the following characterisation

$$\hat{\theta}_d = \begin{cases} \epsilon & \text{if } \vartheta_d \leq \epsilon \\ \vartheta_d & \text{if } \vartheta_d > \epsilon \end{cases} \qquad (d = 1, \ldots, D) \tag{2.42}$$

where

$$\vartheta_d = \frac{N_d}{\sum\limits_{d' : \vartheta_{d'} > \epsilon} N_{d'}} (1 - M) \qquad (d = 1, \ldots, D) \tag{2.43}$$

with

$$M = |\{d' : \vartheta_{d'} \leq \epsilon\}| \, \epsilon \tag{2.44}$$

The idea behind this characterisation is as follows. First note that we distinguish between “rare” events, in the sense that we assign a probability of exactly ϵ to them ($d : \vartheta_d \leq \epsilon$), and “frequent” events, which have probability greater than ϵ ($d : \vartheta_d > \epsilon$). The probability mass allotted to rare events is simply their number times ϵ and is denoted by M in (2.44). The remaining probability mass, $1 - M$, is distributed among frequent events in accordance with (2.43), which is simply a normalisation of event counts as in the conventional case (2.24). Thus, generally speaking, we proceed as in the conventional case, but using only the probability mass not assigned to events that are below the threshold of ϵ.

2.4.2 The algorithm

The above characterisation does not tell us how to partition events 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. The basic algorithm consists in first assuming that the set of rare events is empty, $R^{(0)} = \emptyset$; and then, in iteration k ($k = 1, 2, \ldots$), the new set of rare events, $R^{(k)}$, is obtained from $R^{(k-1)}$ by addition of each event d,

$$R^{(k)} = R^{(k-1)} \cup \{d\} \tag{2.45}$$

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

$$\vartheta_d^{(k-1)} \leq \epsilon \tag{2.46}$$

where

$$\vartheta_d^{(k-1)} = \frac{N_d}{\sum\limits_{d' \notin R^{(k-1)}} N_{d'}} (1 - M^{(k-1)}) \tag{2.47}$$

with

$$M^{(k-1)} = |R^{(k-1)}|\, \epsilon \tag{2.48}$$

At the end of iteration k, the algorithm assures that condition (2.46) is satisfied for all words in $R^{(k)}$. This condition may be also satisfied by words not in $R^{(k)}$ though, in general, it will not be satisfied by most of them.

As R^0 is empty, $M^{(0)}$ is zero and the initial probability estimates, $\vartheta_d^{(0)}$, are exactly those obtained in the conventional MLE case (2.24). Therefore, in the first iteration, we use conventional probability estimates to distinguish between rare and frequent events. Part of the probability mass assigned to frequent events is transferred to rare events for them to arrive at ϵ. The remaining probability mass is redistributed according to (2.47) and, as it is smaller than that distributed before the transference, it may well happen that a frequent event becomes a new rare event. If it happens, a new iteration is carried out; otherwise, the algorithm stops and returns the desired $\hat{\theta}_d$, as characterised by (2.42).

A detailed description of the basic algorithm described above is given in algorithm 2.1. Given D, the training data and ϵ, it returns $\hat{\theta}_d$ and d, as characterised by Eqs.(2.42)-(2.44). After computation of event counts (lines 14–17), the optimal CDMLE solution is obtained iteratively (lines 18–34). Initially, no events are considered rare ($R := \emptyset$) and $\hat{\theta}_d$ is computed for all words as in the conventional case (during the first iteration of the loop in lines 20–34). If an event d is found such that $\hat{\theta}_d \leq \epsilon$ (line 25), 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 d be a non-rare event in iteration $k-1$ ($d \notin R^{(k-1)}$) for which (2.46) holds. Then, it follows that

$$1 - \frac{\epsilon}{1 - M^{(k-1)}} \leq 1 - \frac{N_d}{\sum_{d' \notin R^{(k-1)}} N_{d'}} \tag{2.49}$$

and, rearranging terms,

$$\frac{1 - M^{(k-1)} - \epsilon}{\sum_{d' \notin R^{(k-1)}} N_{d'} - N_d} \leq \frac{1 - M^{(k-1)}}{\sum_{d' \notin R^{(k-1)}} N_{d'}} \tag{2.50}$$

Algorithm 2.1 The CDMLE algorithm.

```
Declarations:
Input:
    $D$                                                   // number of events
    $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N$                              // $N$ training samples
    $\epsilon : 0 \leq \epsilon \leq \frac{1}{D}$                // minimum event occurrence probability
Output:
    $\{\hat{\theta}_d\}$                   // solution as characterised by Eqs.(2.42)-(2.44)
Variables:
    $\{N_d\}$ // event counts
    $R', R$ // previous and current set of rare events
    $S', S$ // previous and current sum of non-rare event counts
    $M', M$// previous and current rare event probability mass
Method:
for $d := 1$ to $D$ do $N_d := 0$ endfor
for $n := 1$ to $N$ do
        for $d := 1$ to $D$ do $N_d := N_d + x_{nd}$ endfor
endfor                                              // event counts computed
$R := \emptyset$; $S := 0$; $M := 0$
for $d := 1$ to $D$ do $S := S + N_d$ endfor
repeat                                                        // main loop
    $R' := R$; $S' := S$; $M' := M$; $transfers :=$ false
    for $d := 1$ to $D$ do
        if $d \notin R'$ then
            $\hat{\theta}_d := \frac{N_d}{S'} \cdot (1 - M')$
            if $\hat{\theta}_d \leq \epsilon$ then
                $\hat{\theta}_d := \epsilon$                       // $d$ has minimum probability
                $R := R \cup \{d\}$                       // $d$ is a new rare event
                $S := S - N_d$
                $M := M + \epsilon$
                $transfers :=$ true
            endif
        endif
    endfor
until not $transfers$
```

As $d \notin R^{(k-1)}$ but satisfies Eq. (2.46), the algorithm adds d to the set of rare events in iteration k, $R^{(k)} = R^{(k-1)} \cup \{d\}$. Using this updated set of rare events, Eq. (2.50) can be rewritten as

$$\frac{1 - M^{(k)}}{\sum_{d' \notin R^{(k)}} N_{d'}} \leq \frac{1 - M^{(k-1)}}{\sum_{d' \notin R^{(k-1)}} N_{d'}} \tag{2.51}$$

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

$$\vartheta_{d''}^{(k)} \leq \vartheta_{d''}^{(k-1)} \tag{2.52}$$

by multiplying each side of Eq. (2.51) by $N_{d''}$. From Eq. (2.52) and the fact that $\vartheta_{d''}^{(k-1)} \leq \epsilon$ for all $d'' \in R^{(k)}$, it follows that $\vartheta_{d''}^{(k)} \leq \epsilon$ for all $d'' \in R^{(k)}$. This means that, in iteration k, event 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 event 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 $\Omega(ND)$. On the other hand, the repeat-until loop is executed D times in the worst case, and thus the algorithm has $O(ND + D^2)$ 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 algorithm 2.1. However, it is straightforward to derive a refined implementation of $O(ND + D \log D)$ time complexity. The idea behind this refinement is to apply Eq. (**??**) in non-decreasing order of occurrence probability, as estimated in the conventional case. That is, in iteration k, the next event d to be considered in Eq. (2.45) must have minimum occurrence probability, as given in Eq. (2.24), among all non-rare events. It can be easily checked that, if condition (2.46) does not hold for d, then it will not hold for any other non-rare event and, therefore, the optimal CMLE solution will have been found.

2.5 Text Classification

In previous section we have introduced a novel smoothing technique for the multinomial distribution, although we have not applied it in practice. In this section we review the application of the multinomial distribution to the context of text classification. Recall from section 1.4, that given the document class c and length L, the *naive Bayes* assumption states that the probability of occurrence of a word w does not depend on its position or other words in the document. In spite of being completely unrealistic, the naive Bayes assumption in text classification has the advantage of greatly simplifying maximum likelihood estimation of unknown class-conditional word occurrence probabilities.

We denote the class variable by $c = 1, \ldots, C$, 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 $\boldsymbol{w}$ may be written as:

$$p_r(c, L, \boldsymbol{w}) = p_r(c), p_r(L)\, p_r(\boldsymbol{w} \mid c, L) \tag{2.53}$$

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

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 $w_{l'}$, $l' \neq l$,

$$p_r(\boldsymbol{w}_1^L \mid c, L) := \prod_{i=1}^{L} \mathrm{p}(w_i \mid c) \tag{2.54}$$

Since word order is not important we can redefine the conditional probability in Eq. (2.54), in terms of the number of occurrence of each vocabulary word d,

$$p_r(\boldsymbol{x}(\boldsymbol{w}_1^L) \mid c, L) = p_r(\boldsymbol{x}_1^D \mid c, L) := \left(\begin{array}{c} L \\ \boldsymbol{x} \end{array} \right) \prod_{d=1}^{D} \theta_{cd}^{x_d} \tag{2.55}$$

where θ_{cd} is the parameter that corresponds to the probability of occurrence of the the d-th word of the lexicon in a document of the class c, and where x_d is the number of times the d-th word, v_d, occurs into the document $\boldsymbol{w}_1^L$

$$x_d = \sum_{i=1}^{L} \delta(v_d, w_i) \tag{2.56}$$

It is worth noting that the conditional text occurrence probability $p(\boldsymbol{x}_1^D \mid c, L)$ in Eq. (2.55) follows a multinomial distribution.

Finally, the prior probability of a class is modeled by a parameter,

$$p_r(c) := \pi_c \tag{2.57}$$

the document length distribution is assumed uniform, i.e.

$$p_r(L) := \frac{1}{\mathcal{L}} \tag{2.58}$$

with $\mathcal{L}$ being the maximum document length.

Plugging everything into the classification Bayes rules in Eq. (1.10) yields

$$\mathrm{c}(\boldsymbol{w}_1^L) = \mathrm{c}(\boldsymbol{x}_1^D) \quad = \arg\max_c \mathrm{p}_{\boldsymbol{\pi}}(c)\, \mathrm{p}_{\boldsymbol{\theta}}(\boldsymbol{x} \mid c) \tag{2.59}$$

$$= \arg\max_c \pi_c \prod_{d=1}^{D} \theta_{cd}^{x_d} \tag{2.60}$$

Note that the constant terms on c have been removed in the last step.

In order to train the naive Bayes classifier for a given sample, $D = \{(\boldsymbol{x}_n, c_n)\}_{n=1}^N$, with the maximum likelihood criterion the following optimisation problem must be solved

$$(\hat{\boldsymbol{\theta}}, \hat{\boldsymbol{\pi}}) = \arg\max_{\boldsymbol{\theta},\boldsymbol{\pi}} \{\mathrm{LL}(\boldsymbol{\theta}, \boldsymbol{\pi})\} = \arg\max_{\boldsymbol{\theta}} \{\mathrm{LL}(\boldsymbol{\theta}) + \arg\max_{\boldsymbol{\pi}} \mathrm{LL}(\boldsymbol{\pi})\} \tag{2.61}$$

subject to the following normalisation constraints

$$\sum_d \theta_{cd} = 1 \quad c = 1, \ldots, C \tag{2.62}$$

$$\sum_c \pi_c = 1 \tag{2.63}$$

and with the definitions

$$\mathrm{LL}(\boldsymbol{\theta}, \boldsymbol{\pi}) = \sum_n \log \mathrm{p}_{\boldsymbol{\pi}}(c_n) + \log \mathrm{p}_{\boldsymbol{\theta}}(\boldsymbol{x}_n \mid c_n) \tag{2.64}$$

$$\mathrm{LL}(\boldsymbol{\pi}) = \sum_n \log \mathrm{p}_{\boldsymbol{\pi}}(c_n) \tag{2.65}$$

$$\mathrm{LL}(\boldsymbol{\theta}) = \sum_n \log \mathrm{p}_{\boldsymbol{\theta}}(\boldsymbol{x}_n \mid c_n) \tag{2.66}$$

$$\tag{2.67}$$

As suggested in Eq. (2.61) the optimisation can be slitted into two different optimisation problems. The first problem optimises $\boldsymbol{\pi}$ subject to Eq. (2.63), i.e.

$$\boldsymbol{\pi} = \arg\max_{\boldsymbol{\pi}} \mathrm{LL}(\boldsymbol{\pi}) \tag{2.68}$$

The second optimisation maximises $\boldsymbol{\theta}$ subject to Eq. (2.62), i.e.

$$\hat{\boldsymbol{\theta}} = \arg\max_{\boldsymbol{\theta}} \mathrm{LL}(\boldsymbol{\theta}) \tag{2.69}$$

Recall that this last optimisation problem can be decomposed into C multinomial independent trainings by

$$\begin{aligned} \mathrm{LL}(\boldsymbol{\theta}) &= \sum_c \sum_{n:c_n=c} \log \mathrm{p}_{\boldsymbol{\theta}_c}(\boldsymbol{x}_n \mid c) \\ &= \sum_c \mathrm{LL}_c(\boldsymbol{\theta}_c) \end{aligned} \tag{2.70}$$

where $\boldsymbol{\theta}_c = (\theta_{c1}, \cdots, \theta_{cD})$ and with $\mathrm{LL}_c(\boldsymbol{\theta}_c)$ defined as

$$\mathrm{LL}_c(\boldsymbol{\theta}_c) = \sum_{n:c_n=c} \log \mathrm{p}_{\boldsymbol{\theta}_c}(\boldsymbol{x}_n \mid c) \tag{2.71}$$

Finally, this leads to C optimisation problems as follows

$$\hat{\boldsymbol{\theta}}_c = \arg\max_{\boldsymbol{\theta}_c} \mathrm{LL}_c(\boldsymbol{\theta}_c), \qquad c = 1, \ldots, C \tag{2.72}$$

Conventional convex optimisation techniques as discussed in section 2.2 yields the solution to the first problem

$$\hat{\pi}_c = \frac{N_c}{N} \tag{2.73}$$

where N_c is the number of samples with the class tag c_n equal to c and N is the total amount of samples.

The latter optimisation problem can be decomposed into C, multinomial trainings. If conventional MLE is applied then the solution is

$$\hat{\theta}_{cd} = \frac{N_{cd}}{N_c} \tag{2.74}$$

where N_{cd} stands for occurrences of the d-th word in documents of the class c. Note that Eq. (2.74) is the MLE for a multinomial distribution depicted in Eq. (2.24) with the training restricted to those documents in class c.

Conventional MLE also implies the smoothing of the multinomial vector of parameters for each class c and each word d, θ_{cd}. The very same techniques discussed in section 2.3 are usually applied [JN02, VNJV04]. Specifically, interpolation and back-off are often used with a generalised smoothing distribution β_d , that can be uniform as in Eq.(2.28), *unigram*

$$\beta_d = \frac{\sum_c N_{cd}}{\sum_{c'} \sum_{d'} N_{c'd'}} \tag{2.75}$$

In order to apply CDMLE, we only need to apply algorithm 2.1 for each class c separately, in the same way that the conventional training can be decomposed into C, multinomial trainings each with a different training extracted from the full training.

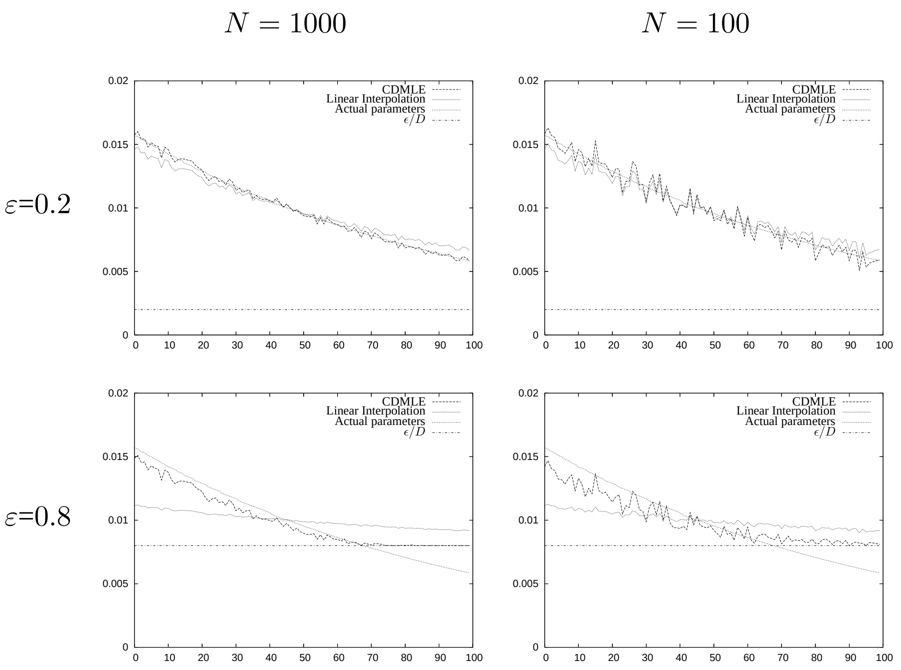

Figure 2.1: Results on the exponential simulated data. In Column 1, a sample of 1000 elements is plotted for values $\varepsilon = 0.2$ and $\varepsilon = 0.8$ top-down, respectively. The x-axis plots the words indexes d whilst the y-axis plots its respective parameter, θ_d.

2.6 Experiments

In order to deeply study the properties of our proposed method, two sets of experiments were carried out. The former set of experiments consists in simulated data that allows us to draw some interesting properties regarding our proposal. The latter experiment set were carried out on real text classification data.

2.6.1 Simulated experiments

One of the most interesting things of the constrained domain maximum likelihood estimation algorithm 2.1 is that it resembles the uniform linear interpolation smoothing. In this section we analyse the resemblances and differences between CDMLE and linear interpolation.

In the simulated data we have fix the multinomial parameter vector $\boldsymbol{\theta}$, and then we know the actual distribution. The performed simulation were three:

- A exponential parameter vector:

$$\theta_d \propto \exp(-\frac{1}{D}) \tag{2.76}$$

- A parameter vector following the Zipf's law for word occurrence:

$$\theta_d \propto \frac{1}{D^2} \tag{2.77}$$

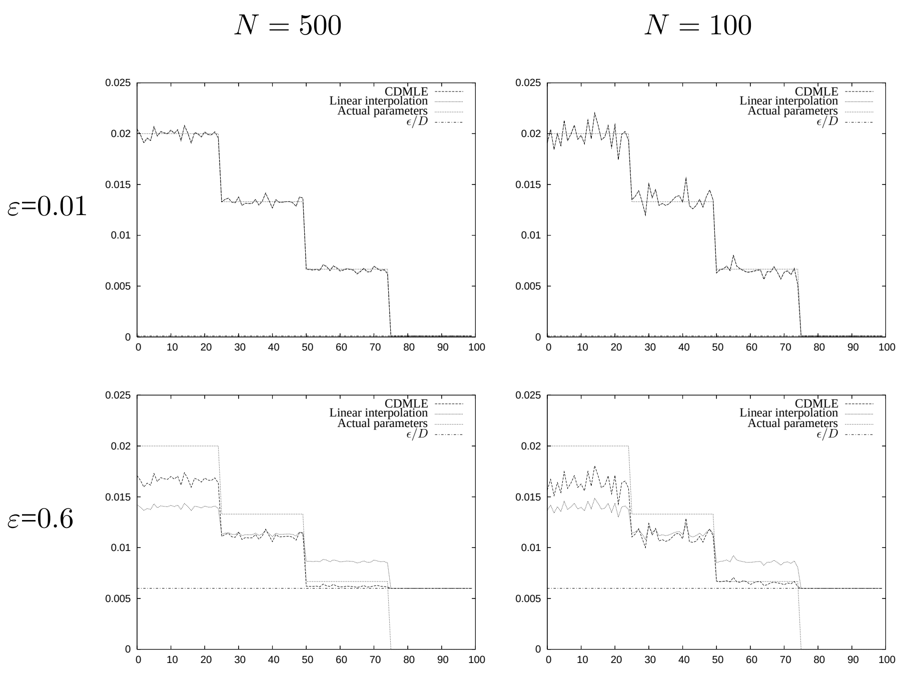

Figure 2.2: Results on the 3-steps simulated data. Columns represent different sample sizes N, and rows different values of ε. The x-axis plots the words indexes d whilst the y-axis plots its respective parameter, θ_d.

- A parameter vector made up of three steps:

$$\theta_d = \begin{cases} 2 & d \leq \frac{D}{3} \\ 1 & \frac{D}{3} < d \leq \frac{2D}{3} \\ 0 & \frac{2D}{3} < d \leq D \end{cases} \tag{2.78}$$

We compared the behaviour of two techniques: the conventional linear interpolation smoothing and the CDMLE. In the former case the generalised smoothing distribution is

$$\beta_d = \frac{\varepsilon}{D} \tag{2.79}$$

In the latter case, the ϵ value defined in the constraints in Eq.(2.32), has been fixed to

$$\epsilon = \frac{\varepsilon}{D} \tag{2.80}$$

In all the plots the experiments were repeated 100 times, in order to get an small confidence interval. The confidence intervals were not plotted because they were very small and disturb the plots.

In figure 2.1, the results on the exponential simulated data are plotted for several sample sizes and ε values. The CDMLE solution is closer to the actual values than that of linear interpolation, although all the d for which the constrain is active, are fixed to ϵ. On the other hand, the linear interpolation yields poor approximations to the actual parameters in comparison to the CDMLE. Nevertheless, the linear interpolation keeps the discriminative power among different words d though differences are smaller in magnitude.

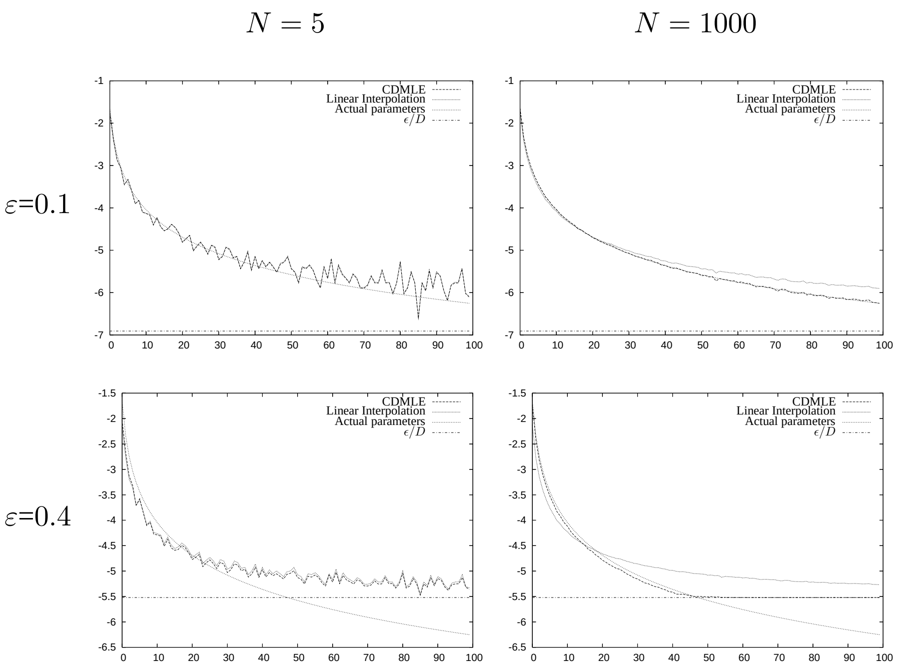

Figure 2.3: Results on the Zipf's law simulated data. Columns represent different sample sizes N, and rows different values of ε. The x-axis plots the words indexes d whilst the y-axis plots its respective parameter in logarithmic scale, $\log \theta_d$.

Summarising, the CDMLE provides better approximations to the actual parameters at the expense of making equal the "rare" words, while the linear interpolation obtains worse approximations to the actual parameters but retains the qualitative difference among "rare" words. This last stament, could make the difference in some scenarios. The best technique depends on whether the "rare" words are nosy or are properly estimated by conventional naive Bayes. This result is also supported by figures 2.2, and 2.3, which are the analogous plot for 3-steps data and Zipf's law data, respectively.

2.6.2 Real data experiments

The proposed CDMLE approach was empirically compared to the usual practice of simply smoothing relative counts, as described in Section 2.3.1. This comparison was carried on four text classification data sets (tasks): *Traveller, 20 Newsgroups*, *Industry Sector* and *Job Category*.

2.6.3 Datasets

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 sub-domains. The *20 Newsgroups* corpus is a collection of ap-

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
Type of documents	job titles	web pages
Number of documents	131 643	9 629
Running words	11 221K	1 834K
Avg. document length	85	191
Vocabulary size	84 212	64 551
Singletons (Vocab %)	34.9	41.4
Classes	65	105
Class 1-tons (Vocab %)	49.2	58.7
Class 2-tons (Vocab %)	14.0	11.6

	20 Newsgroups	Traveller (English)
Type of documents	newsgroups	sentences
Number of documents	19 974	8 000
Running words	2 549K	79K
Avg. document length	128	10
Vocabulary size	102 752	391
Singletons (Vocab %)	36.0	4
Classes	20	23.0
Class 1-tons (Vocab %)	61.1	74.9
Class 2-tons (Vocab %)	12.9	18.3

proximately $20,000$ newsgroup documents, partitioned (nearly) evenly across 20 different newsgroups. We used the original version of this data set as provided by [Ren01], 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 [ABC$^+$00, Ren01, McC02, VNJV04].

The *rainbow* toolkit [McC98] 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 stoplist removal, stemming or vocabulary pruning by occurrence count.

2.6.4 Results

Figure 2.4 shows the results obtained in each data set. The proposed CDMLE algorithm is compared to:

Table 2.2: Summary of the best results obtained for the 4 corpora and for all the smoothing techniques.

	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	$\mathbf{37.8}$	$\mathbf{14.8}$	3.3
CDMLE	$\mathbf{33.0}$	38.6	15.3	$\mathbf{3.1}$

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. (2.27)
2. *AD+1gBO or 1gI:* b has its usual meaning, as defined in Eq. (**??**), 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 $[E\% \pm 1\%]$ bur for Job Category of which interval is $[E\% \pm 0.4\%]$.

From the results in Fig. 2.4, 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 (2.75) to obtain reliable class-independent word probability estimates. Clearly, this estimates can be used to replace (2.32) 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

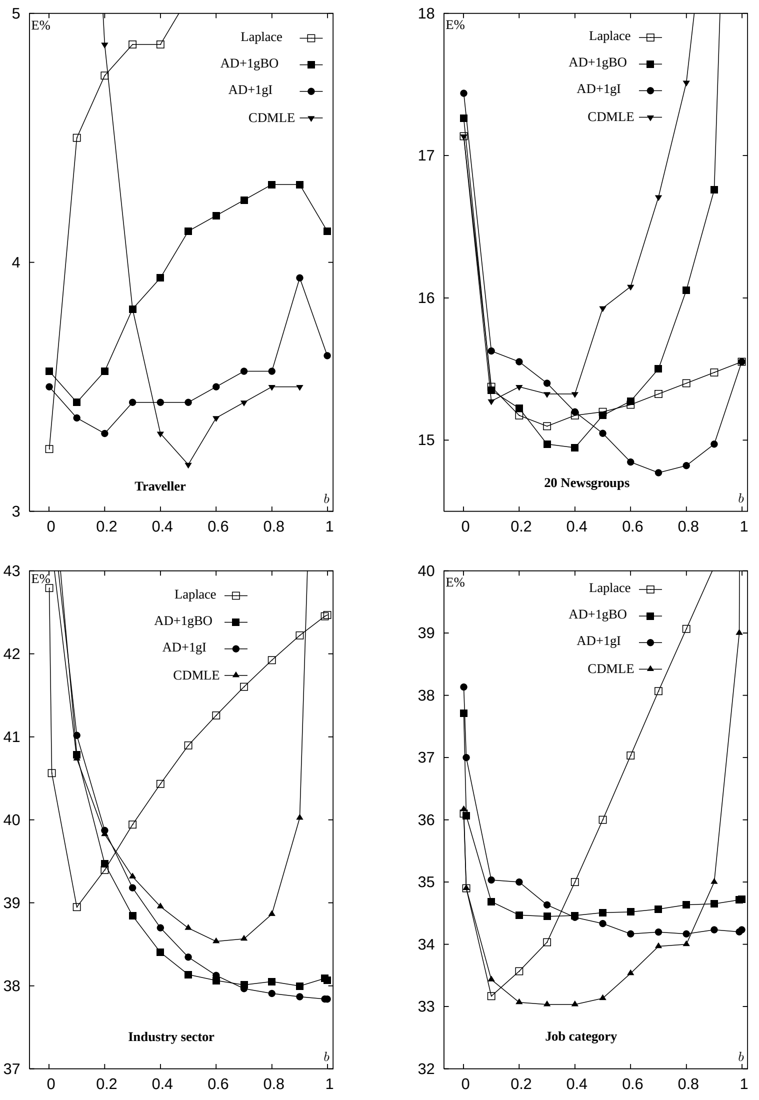

Figure 2.4: Results obtained in the *Traveller*, *20 Newsgroups*, *Industry sector* and *Job category* data sets. Each plot shows the classification error rate as a function of the discount parameter b, for the four classification techniques considered (*Laplace*, *AD+1gBO*, *AD+1gI* and *CDMLE*).

between these two cases arises from the number of repeat-until iterations executed (lines 20–34 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.

2.7 Conclusions

In this work, conventional naive Bayes training with parameter smoothing has been restated as a constrained 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.(2.32) and making it dependent on both the class c and the word d.

We think that the proposed approach and technique are 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 maximum likelihood estimation can be used without any side effect. Also, this constrained approach might be useful in the case of training criterium other than maximum likelihood such as discriminative training [JVN07].

The theory and experimental results in this chapter yield 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.

and a publication in a journal is pending:

- **J.Andrés-Ferrer** and Alfons Juan. Constrained domain maximum likelihood estimation for naive Bayes text classification. *Pattern Analysis and Applications (PAA)*. Pending

BIBLIOGRAPHY

[ABC$^+$00] 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, 2000.

[BV04] Stephen Boyd and Lieven Vandenberghe. Convex optimization. March 2004.

[JN02] Alfons Juan and Hermann Ney. Reversing and Smoothing the Multinomial Naive Bayes Text Classifier. In *Proc. of PRIS 2002*, pages 200–212, 2002.

[JVN07] 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.

[McC98] A. McCallum. Bow: A toolkit for statistical language modeling, text retrieval, classification and clustering, 1998. `www.cs.umass.edu/~mccallum/bow/rainbow`.

[McC02] A. McCallum. Industry Sector data set, 2002. `www.cs.umass.edu/~mccallum/code-data.html`.

[MN98] 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.

[Ren01] J. Rennie. Original 20 Newsgroups data set, 2001. `people.csail.mit.edu/jrennie/20Newsgroups`.

[VNJV04] David Vilar, Hermann Ney, Alfons Juan, and Enrique Vidal. Effect of Feature Smoothing Methods in Text Classification Tasks. In *Proc. of PRIS 2004*, pages 108–117, 2004.

CHAPTER 3

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

BAYES' DECISION RULE is the foundation of statistical pattern recognition. This well-known rule is obtained by the minimisation of the classification risk. It is not possible to know the exact loss in which a classification system will incur. Therefore, we compute the risk in which a system is "expected" to incur based upon our "loss" criteria. This criteria is formalised by the *loss function*. By means of the the loss function we measure the penalty of incorrectly classifying an object, depending upon some a priori preferences or costs. In practice, however, neither costs nor preferences are given but for the correct/incorrect criteria. So the system only incurs in a constant loss if an object is misclassified. Thereby, the minimisation of the risk yields the intuitive concept of minimising the classification error rate (CER).

The loss associated to each error is a very important matter since the system decisions highly depends on it. The classification error rate is not the best risk to minimise in all the classification problems, although it may seem to be. One simple example is when system misclassification imply economic consequences. So, if classifying an object into a wrong class has a different economic loss depending on the object, the correct class or/and the proposed wrong class, so must the loss function have it. Otherwise, we can end with a waste system.

During this chapter, we assume that we have a good approximation to the actual probability distributions or even the actual distributions themselves. We focus our analysis on the way to build the optimal classification system with the best possible estimation or approximation to the probability distributions. Hence, we are not dealing with the training criteria but with the decoding criteria.

Along this chapter we cover all the outlined aspects of the loss function from a statistical point of view. We explore the theoretical advantages and drawbacks of the most general loss functions. Afterwards by constraining this general loss, we analyse simpler and faster loss functions. Finally, we apply the developed theory to statistical machine translation in order to study the proposed theory in a real task.

3.1 Introduction

Statistical pattern recognition is a well-founded discipline that 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 (*$\mathrm{c} : \mathcal{O} \rightarrow \Omega$*)* [DHS00].

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 misclassifications do not have the same repercussions. Therefore, a criteria 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 [DHS00]. It is well known that, if a 0–1 loss function is provided, then the optimal system minimises the classification error rate [DHS00].

This chapter is mainly dedicated to design loss functions that should improve system performance while keeping the simplicity of 0–1 optimal classification system. In [RSN05] complex classification rules were analysed using a *metric loss function*. There are other works that analyse the most general loss functions, for instance [UN04]. 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 the chapter is organised as follows. In section 3.2 pattern recognition problems are analysed from a decision theory view. In section 3.3, we introduce statistical machine translation as a case of study. Finally, concluding remarks are summarised in section 3.4.

3.2 Bayes Decision Theory

A classification problem is an instance of a decision problem (DP). From this point of view, a classification problem is composed of three different items:

1. A set of *Objects* ($\mathcal{O}$) 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 $\boldsymbol{x} \in \mathcal{O}$.
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 [DHS00]

$$\mathrm{c} : \mathcal{O} \rightarrow \Omega \tag{3.1}$$

Therefore, when an object $\boldsymbol{x} \in \mathcal{O}$ 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 the system when classifying an object $\boldsymbol{x}$, the so-called *conditional risk given* $\boldsymbol{x}$, as

$$R(\omega_p|\boldsymbol{x}) = \sum_{\omega_c \in \Omega} \mathrm{l}(\omega_p|\boldsymbol{x}, \omega_c) \cdot p_r(\omega_c|\boldsymbol{x}) \tag{3.2}$$

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 probability distribution, $p_r(\omega|\boldsymbol{x})$.

Using the conditional risk, we define the *the global risk* [DHS00] as the contribution of all objects to the performance of classifiers, i.e.

$$R(\mathrm{c}) = \mathrm{E}_{\boldsymbol{x}}[R(\mathrm{c}(\boldsymbol{x})|\boldsymbol{x})] = \int_{\mathcal{X}} R(\mathrm{c}(\boldsymbol{x})|\boldsymbol{x}) \cdot p_r(\boldsymbol{x}) d\boldsymbol{x} \tag{3.3}$$

where $R(\mathrm{c}(\boldsymbol{x})|\boldsymbol{x})$ is the conditional risk given $\boldsymbol{x}$, as defined in Eq. (3.3).

Our aspiration it 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 loss of generality, the optimal classification rule, namely *minimum Bayes' risk*, is the one that minimises the conditional risk, i.e.

$$\hat{\mathrm{c}}(\boldsymbol{x}) = \arg\min_{\omega \in \Omega} R(\omega|\boldsymbol{x}) \tag{3.4}$$

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. (3.4), are often called *decoding algorithms* or *search algorithms*. Analogously, 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 criteria, method or algorithm is better for improving the system performance. Instead, we deal with the following stage of the system design. 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 between systems. In order to do so, we need to compare the global risk of those systems. The global risk in Eq. (3.3), can be understood as the expected loss with respect to the object-class joint probability distribution

$$R(\mathrm{c}) = \int_{\mathcal{X}} \sum_{\omega \in \Omega} \mathrm{l}(\mathrm{c}(\boldsymbol{x})|\boldsymbol{x},\omega) p_r(\omega,\boldsymbol{x}) d\boldsymbol{x} \tag{3.5}$$

with $p_r(\omega,\boldsymbol{x}) = p_r(\omega|\boldsymbol{x})p(\boldsymbol{x})$. Thefore, 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

$$\bar{R}_T(\mathrm{c}) = \frac{1}{N} \sum_{n=1}^{N} \mathrm{l}(\mathrm{c}(\boldsymbol{x}_n)|\boldsymbol{x}_n,\omega_n) \tag{3.6}$$

We call this approximation the *empirical risk* on the test set T.

The question of which the best loss function is, does not have a unique and general answer. 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 classification (loss of 1) and correct classification (zero loss), i.e.,

$$\mathrm{l}(\omega_p|\boldsymbol{x},\omega_c) = \begin{cases} 0 & \omega_p = \omega_c \\ 1 & \text{otherwise} \end{cases} \tag{3.7}$$

This function is called 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. (3.7) is used, the minimum Bayes' risk in Equation (3.4) can be simplified yielding the well-known optimal Bayes' classification rule [DHS00]:

$$\mathrm{c}(\boldsymbol{x}) = \arg\max_{\omega \in \Omega} p(\omega \mid \boldsymbol{x}) \tag{3.8}$$

where $\boldsymbol{x}$ is the object to be classified, and ω denotes one class from Ω.

However, while the 0–1 loss function is adequate for many problems with 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. 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. 3.1

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

$$\mathrm{l}(\omega_p|\boldsymbol{x}, \omega_c) = \begin{cases} 0 & \omega_p = \omega_c \\ \epsilon(\boldsymbol{x}, \omega_p, \omega_c) & \text{otherwise} \end{cases} \tag{3.9}$$

The error function must verify the following finiteness property,

$$\sum_{\omega_c \in \Omega} p_r(\omega_c|\boldsymbol{x})\, \epsilon(\boldsymbol{x}, \omega_p, \omega_c) < \infty \tag{3.10}$$

The optimal Bayes' classification rule corresponding to the previous loss function in Eq. (3.9) is

$$\mathrm{c}(\boldsymbol{x}) = \operatorname*{arg\,min}_{\omega_p \in \Omega} \sum_{\omega_c \neq \omega_p} p_r(\omega_c|\boldsymbol{x})\, \epsilon(\boldsymbol{x}, \omega_p, \omega_c) \tag{3.11}$$

The previous classification rule in Eq. (3.11), 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. (3.11), and the rule in Eq. (3.8), 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. (3.9) and in Eq. (3.7) 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. (3.11), 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,

$$\mathrm{l}(\omega_p|\boldsymbol{x}, \omega_c) = \begin{cases} 0 & \omega_p = \omega_c \\ \epsilon(\boldsymbol{x}, \omega_c) & \text{otherwise} \end{cases} \tag{3.12}$$

then the optimal classification rule is

$$\mathrm{c}(\boldsymbol{x}) = \operatorname*{arg\,min}_{\omega_p \in \Omega} \sum_{\omega_c \neq \omega_p} p_r(\omega_c|\boldsymbol{x})\, \epsilon(\boldsymbol{x}, \omega_c) \tag{3.13}$$

Applying some basic arithmetic operations to the classification rule in previous Eq. (3.13), the classification rule is significantly simplified, i.e.,

$$\mathrm{c}(\boldsymbol{x}) = \operatorname{arg\,min}_{\omega_p \in \Omega} \textstyle\sum_{\omega_c \neq \omega_p} p_r(\omega_c|\boldsymbol{x})\, \epsilon(\boldsymbol{x}, \omega_c) \tag{3.14}$$

$$\mathrm{c}(\boldsymbol{x}) = \operatorname{arg\,min}_{\omega_p \in \Omega} \{-p_r(\omega_p|\boldsymbol{x})\, \epsilon(\boldsymbol{x}, \omega_p) + S(\boldsymbol{x})\} \tag{3.15}$$

$$\mathrm{c}(\boldsymbol{x}) = \operatorname{arg\,min}_{\omega_p \in \Omega} \{-p_r(\omega_p|\boldsymbol{x})\, \epsilon(\boldsymbol{x}, \omega_p)\} \tag{3.16}$$

$$\mathrm{c}(\boldsymbol{x}) = \operatorname{arg\,max}_{\omega_p \in \Omega} p_r(\omega_p|\boldsymbol{x})\, \epsilon(\boldsymbol{x}, \omega_p) \tag{3.17}$$

with $S(\boldsymbol{x}) = \sum_{\omega \in \Omega} \epsilon(\omega, \boldsymbol{x}) p_r(\omega|\boldsymbol{x})$.

By comparing Eqs. (3.17) and (3.8), we conclude that the cost is almost the same, but 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. (3.8). Therefore, the 0–1 loss function is the simplest error function of this category. If we compare Eqs. (3.17) and (3.11) it can be seen that the former is fastest that the latter. Assuming that the cost of computing the error functions, $\epsilon(\cdot)$ and $p_r(\omega|\boldsymbol{x})$ is constant we classify loss functions into two categories:

- The most general loss function depicted in Eq. (3.9) which has an asymptotic cost of $O(|\Omega|^2)$.
- The loss functions that drop the dependence on the proposed class defined as in Eq. (3.12), which has an asymptotic cost of $O(|\Omega|)$.

Analysing the Eq. (3.12), the question of which the best error function is, raises immediately. The answer is not too clear, 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 3.1 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. If the loss of classifying an object $\boldsymbol{x}$ into the class ω_l although it belongs to ω_h, and vice-verse, is the same, then the system could always fail in classifying objects that belong to the class ω_h. Since the class ω_h is more probable, the system will fail more times than if the loss of misclassifying object of the class ω_h were the highest. This idea is analysed into detail for the statistical machine translation problem in section 3.3.

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 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 in order to agree with the frequency of occurrence.

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 criteria, 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 can be used to approximate a generic error function $\epsilon(\boldsymbol{x}, \omega_c, \omega_p)$ with a fastest 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 3.3.3 under the view of statistical machine translation.

3.3 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) [BPPM93].*

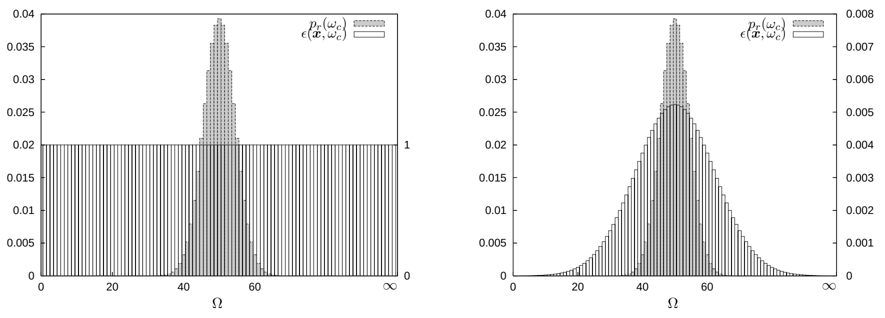

Figure 3.1: Difference of using a 0–1 loss function (on the left) and an approximation to the true class probability as the loss function (on the right) when using a loss function of the sort of Eq. (3.12). The left-scale of the y axis shows a possible actual probability over the target sentences. The 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 infinite enumerable set of possible target sentences (or classes).

Statistical machine translation consists in finding the translation $\mathbf{y}$ of a 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.

Typically, the SMT systems are based on the Bayes' classification rule for the 0–1 loss function depicted in Eq. (3.8). Usually, the class posterior probability is decomposed using Bayes' theorem into two probabilities,

$$\hat{\mathbf{y}} = \hat{\mathrm{c}}(\mathbf{x}) = \arg\max_{\boldsymbol{y}_p \in \boldsymbol{Y}^*} \left\{ p_r(\mathbf{x}|\boldsymbol{y}_p) p_r(\boldsymbol{y}_p) \right\} \tag{3.18}$$

Previous Eq. (3.18) 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 $\boldsymbol{y}$ conditionally to the given information $\boldsymbol{x}$.

Equation (3.18) 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 [Kni99, UM06]. However, several search algorithms

[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 $\mathbf{x}$.

have been proposed in the literature to solve this problem efficiently [B+90, WW97, AO+99, G+01, Jel69, GVC01, TN03].

Another drawback of the ITR, is that it is obtained using the 0–1 loss function. As stated in Sec. 3.2, this loss function is not particularly appropriate when the number of classes is huge as occurs 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. (3.7)) 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.

3.3.1 General error functions

As stated above, the most generic loss functions depicted in Eq (3.9), produce minimisations which have a quadratic cost depending on the size of 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. There are some works that have already explored this kind of loss functions [UN04, RSN05].

The more appealing application of this loss functions is the use of a metric loss function [RSN05]. For instance, in machine translation one widespread metric is the WER (see Section 1.3 for a definition), since the loss function in Equation (3.12) depends on both, the proposed translation and the reference translation, the WER can be used as loss function [UN04]. Nevertheless, due to the high complexity, the use of these quadratic loss functions, is only feasible in constrained situations like n-best lists [KB04].

3.3.2 Simplified error functions

The search algorithms generated by the classification rule in Eq. (3.12) 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 3.2, a more suitable loss function than the 0–1 loss, is obtained using as the error function the target sentence probability, $\epsilon(\mathbf{x}, \mathbf{y}_j) = p_r(\mathbf{y}_j)$,

$$\mathrm{l}(\mathbf{y}_p|\mathbf{x}, \mathbf{y}_c) = \begin{cases} 0 & \mathbf{y}_p = \mathbf{y}_c \\ p_r(\mathbf{y}_c) & \text{otherwise} \end{cases} \tag{3.19}$$

This loss function seems to be more appropriate than the 0-1. This is due to the fact that if the system makes an error translating a set of source sentences, this loss function tries to force the system to fail in the source sentence $\mathbf{x}$ whose 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. (3.19) is

$$\hat{\boldsymbol{y}}(\mathbf{x}) = \underset{\boldsymbol{y}_p \in \boldsymbol{Y}^*}{\arg\max} \left\{ p_r(\boldsymbol{y}_p|\mathbf{x}) p_r(\boldsymbol{y}_p) \right\} \tag{3.20}$$

Previous Eq. (3.20) 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 [OTN99, ON04a, KOM03, Zen08]. Note that

[b]Here lies the importance of distinguishing between the translation proposed by the system $\mathbf{y}_p$ and the correct translation $\mathbf{y}_c$ of the source sentence $\mathbf{x}$.

the DTR was introduced as an heuristic version of the ITR in Eq. (3.18), 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 obtained as the Bayes' optimal classification rule *if the loss function in Eq.* (3.19) *is used*.

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 employed for approximate the translation probabilities may not be good enough. Thus, the model error, 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 manner so that the translation system will be the same and then these asymmetries will be reduced. By simply applying the Bayes' theorem to Eq. (3.20), we obtain the equivalent rule:

$$\hat{\mathbf{y}} = \underset{\mathbf{y}\in\boldsymbol{Y}^*}{\arg\max} \left\{p_r(\mathbf{y})^2 p(\mathbf{x}|\mathbf{y})\right\} \tag{3.21}$$

The difference between the Eq (3.20) and Eq (3.21) measure the asymmetries of the translation models as well as the error in the modelling.

Nevertheless, 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 and the best estimation that can be achieved for the selected model with the finite sample that we have for training.

As stated in section 3.2 a refined loss function is designed using the joint probability as the error function, $\epsilon(\mathbf{x}, \mathbf{y}_j) = p_r(\mathbf{x}, \mathbf{y}_j)$,

$$\mathrm{l}(\mathbf{y}_p|\mathbf{x}, \mathbf{y}_c) = \begin{cases} 0 & \mathbf{y}_p = \mathbf{y}_c \\ p_r(\mathbf{x}, \mathbf{y}_c) & \text{otherwise} \end{cases} \tag{3.22}$$

which leads to:

$$\hat{\boldsymbol{y}} = \underset{\mathbf{y}\in\boldsymbol{Y}^*}{\arg\max} \left\{p_r(\mathbf{x}, \mathbf{y}) p_r(\boldsymbol{y}|\boldsymbol{x})\right\} \tag{3.23}$$

Depending on how we model probabilities in Eq. (3.23), 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

$$\hat{\mathbf{y}} = \underset{\mathbf{y}\in\boldsymbol{Y}^*}{\arg\max} \left\{p_r(\mathbf{y}) p_r(\mathbf{x} \mid \mathbf{y}) p_r(\mathbf{y} \mid \mathbf{x})\right\} \tag{3.24}$$

The interpretation of this rule is a refinement of the direct translation rule. In this case, if the system makes a mistake, then it is done in the least probable pairs $(\mathbf{x}, \mathbf{y})$ in terms of $p(\mathbf{y}, \mathbf{x})$.

3.3.3 Approximation to general error functions

As stated in section 3.2, the loss functions of the kind in Eq. (3.12), are faster than the general loss functions depicted in Eq. (3.9). The former loss function sacrifices the use of the proposed translation in order to speed the search process. Unfortunately, the automatic evaluation metrics used to rank 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.

One way to define 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 translation. Then we define the error function

$$\epsilon_{\boldsymbol{\lambda}}(\boldsymbol{x}, \boldsymbol{y}_c) = \prod_{k=1}^{K} f_k(\boldsymbol{x}, \boldsymbol{y}_c)^{\lambda_k} \tag{3.25}$$

If our actual evaluation error function is

$$\epsilon(\boldsymbol{x}, \boldsymbol{y}_p, \boldsymbol{y}_c) = 1 - \mathrm{BLEU}(\boldsymbol{y}_p, \boldsymbol{y}_c) \tag{3.26}$$

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. (3.26). For instance, the maximum entropy algorithm [BPP96] is typically applied to find the optimal parameter vector $\boldsymbol{\lambda}$,

The error function defined in Eq. (3.25) leads to the following classification rule

$$\hat{\boldsymbol{y}}_{\boldsymbol{\lambda}}(\boldsymbol{x}) = \operatorname*{arg\,max}_{\boldsymbol{y}_c \in \boldsymbol{Y}^*} p_r(\boldsymbol{y}_c|\boldsymbol{x}) \prod_{k=1}^{K} f_k(\boldsymbol{x}, \boldsymbol{y}_c)^{\lambda_k} \tag{3.27}$$

Note that we can always extend the vector of parameters lambda, $\boldsymbol{\lambda}$, and the feature vector, $\boldsymbol{f}$, by adding the conditional probability, $p_r(\boldsymbol{y}_c|\boldsymbol{x})$ as a new feature [c]. Therefore, the classification rule expressed in terms of the extended feature vector, $\bar{\boldsymbol{f}}$ and the extended parametric vector $\bar{\boldsymbol{\lambda}}$ is

$$\hat{\boldsymbol{y}}_{\bar{\boldsymbol{\lambda}}}(\boldsymbol{x}) = \operatorname*{arg\,max}_{\boldsymbol{y}_c \in \boldsymbol{Y}^*} \prod_{k=1}^{K} \bar{f}_k(\boldsymbol{x}, \boldsymbol{y}_c)^{\bar{\lambda}_k} \tag{3.28}$$

If we apply the logarithm to the previous Eq. (3.28) we obtain the equivalent expression

$$\hat{\boldsymbol{y}}_{\bar{\boldsymbol{\lambda}}}(\boldsymbol{x}) = \operatorname*{arg\,max}_{\boldsymbol{y}_c \in \boldsymbol{Y}^*} \sum_{k=1}^{K} \bar{\lambda}_k \log \bar{f}_k(\boldsymbol{x}, \boldsymbol{y}_c) \tag{3.29}$$

Inspired by Eq. (3.29), a more general error function can be defined to approximate the actual error function in Eq. (3.26)

$$\epsilon(\boldsymbol{x}, \boldsymbol{y}_c) = \sum_{k=1}^{K} \lambda_k h_k(\boldsymbol{x}, \boldsymbol{y}_c) \tag{3.30}$$

with $\boldsymbol{h}(\boldsymbol{x}, \boldsymbol{y}_c)$ being the feature vector analogous to $\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{y}_c)$. Note that if we define $\boldsymbol{h}(\boldsymbol{x}, \boldsymbol{y}_c) = \log \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{y}_c)$, then Eq. (3.30) is logarithmically equivalent to Eq. (3.25).

[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. $\bar{\lambda}_l = \lambda_l + 1$.

Again following an analogous process we can define the extended feature vector $\bar{\boldsymbol{f}}$ and the extended parameter vector $\bar{\boldsymbol{\lambda}}$ obtaining the classification rule

$$\hat{\boldsymbol{y}}_{\bar{\lambda}}(\boldsymbol{x}) = \operatorname*{arg\,max}_{\boldsymbol{y}_c \in \boldsymbol{Y}^*} \sum_{k=1}^{K} \bar{\lambda}_k \bar{h}_k(\boldsymbol{x}, \boldsymbol{y}_c) \tag{3.31}$$

Although we have introduced above the feature vector, we have not yet covered which are the typical features used in the state-of-the-art systems. Typical features range among [ON04a, M$^+$06] the followings:

- *Direct translation models:* a typical feature is to use a direct translation model

$$h_k(\boldsymbol{x}, \boldsymbol{y}) = p_{\boldsymbol{\theta}}(\boldsymbol{y}|\boldsymbol{x}) \tag{3.32}$$

  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

$$h_k(\boldsymbol{x}, \boldsymbol{y}) = p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{y}) \tag{3.33}$$

  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,

$$h_k(\boldsymbol{x}, \boldsymbol{y}) = p_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{y}) \tag{3.34}$$

- *An n-gram language model:* that is to say

$$h_k(\boldsymbol{x}, \boldsymbol{y}) = p_{\boldsymbol{\theta}}(\boldsymbol{y}) \tag{3.35}$$

- *Word bonus:* it is a well know problem of the n-gram language models that they give more probability so short sentences. Additionally, the translation models tend to distribute the probability among ill-formed sentences as the length of the sentence increases [BPPM93]. Therefore, in order to keep the translation systems from always producing poor translations because of trying to shorten them, the following feature is used

$$h_k(\boldsymbol{x}, \boldsymbol{y}) = \exp(|\boldsymbol{y}|) \tag{3.36}$$

Most of the state of the art systems use this idea, although they present it as if it were a log linear model [ON04a, M$^+$06]. Specifically, if in Eq. (3.8) we model the direct probability as a log linear model

$$p_{\boldsymbol{\lambda}}(\boldsymbol{y}|\boldsymbol{x}) = \frac{1}{p_{\boldsymbol{\lambda}}(\boldsymbol{x})} \exp(\sum_{k=1}^{K} \lambda_k h_k(\boldsymbol{x}, \boldsymbol{y})) \tag{3.37}$$

with

$$p_{\boldsymbol{\lambda}}(\boldsymbol{x}) = \sum_{\boldsymbol{y} \in \boldsymbol{Y}^*} \exp(\sum_{k=1}^{K} \lambda_k h_k(\boldsymbol{x}, \boldsymbol{y})) \tag{3.38}$$

then using the model in Eq. (3.37) in the rule in Eq. (3.8) we obtain the following rule

$$\hat{\boldsymbol{y}}_{\boldsymbol{\lambda}}(\boldsymbol{x}) = \operatorname*{arg\,max}_{\boldsymbol{y} \in \boldsymbol{Y}^*} \frac{1}{p_{\boldsymbol{\lambda}}(\boldsymbol{x})} \exp(\sum_{k=1}^{K} \lambda_k h_k(\boldsymbol{x}, \boldsymbol{y})) \tag{3.39}$$

$$\hat{\boldsymbol{y}}_{\boldsymbol{\lambda}}(\boldsymbol{x}) = \operatorname*{arg\,max}_{\boldsymbol{y} \in \boldsymbol{Y}^*} \exp(\sum_{k=1}^{K} \lambda_k h_k(\boldsymbol{x}, \boldsymbol{y})) \tag{3.40}$$

$$\hat{\boldsymbol{y}}_{\boldsymbol{\lambda}}(\boldsymbol{x}) = \operatorname*{arg\,max}_{\boldsymbol{y} \in \boldsymbol{Y}^*} \sum_{k=1}^{K} \lambda_k h_k(\boldsymbol{x}, \boldsymbol{y}) \tag{3.41}$$

Note that if you compare Eqs. (3.41) and (3.31) 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. (3.37) in never trained in its full form, i.e., it is only trained in the form of classification rule (3.41) to minimise the general loss function in Eq. (3.26). Additionally, this view presents the log-linear model in Eq. (3.37) as an hyper-model because most of the used features are probability models by themselves.

3.3.4 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 use 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 state-of-the-art 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 were trained using a 5-gram model obtained with the SRILM toolkit [Sto02].

Similarly to [G$^+$01], 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)*.

3.3.5 Corpora

Three different corpora were used for the experiments that were carried out in this chapter: Eutrans-I (Tourist), Europarl and Xerox.

	Test Set		Train Set	
	Spa	Eng	Spa	Eng
sentences	1K		170K	
avg. length	12.7	12.6	12.9	13.0
vocabulary	518	393	688	514
singletons	107	90	12	7
running words	12.7K	12.6K	2193K	2206K
perplexity	3.62	2.95	3.50	2.89

Table 3.1: Basic statistics of the Spanish-English TOURIST task.

Table 3.1 summarises some of the statistics of the Tourist corpus [ABC$^+$96]. 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 3.2 shows some statistics of the Europarl corpus [Koe05]. Specifically, this is the version that was used in the shared task of the NAACL 2006 Workshop on SMT [NAA06]. 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 3.2, 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 3.2: Statistics of the Europarl corpus

		Spanish	**English**
Training	Sentences	730 740	
	Running Words	15 725 136	15 222 505
	Vocabulary	102 885	64 122
	Avg. sentence length	21.5	20.8
Test	Sentences	3 064	
	Running Words	91 730	85 232
	Perplexity	102	120

Table 3.3 some statistics of the Xerox corpus [Ato01]. 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 3.3, the Xerox corpus contains a considerable number of sentences and medium-size vocabularies.

Word Based Translation experiments

In this section, the IBM Model 2 [BPPM93] is used to approximate the translation probability distributions. Together with the IBM Model 2 [BPPM93], its corresponding search

Table 3.3: Statistics of the Xerox corpus

		Spanish	**English**
Training	Sentences	55761	
	Running Words	752607	665400
	Vocabulary	11051	7957
	Avg. sentence length	13.5	11.9
Test	Sentences	1125	
	Running Words	10106	8370
	Perplexity	35	47

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. Finally, the search problem can be solved exactly using dynamic programming for the case of the direct translation rule depicted in Eq. (3.20).

In order to train the IBM Model 2 we used the standard tool *GIZA*++ [Och00]. We re-implemented the algorithm presented in [GVC01] to perform the search process in translation 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 3.3.5). 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 3.2 shows the differences in terms of the WER among all the mentioned forms of the DTR: "IFDTR" (Eq. 3.21), and "DTR" (Eq. 3.20). 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 and the length distribution usually allows the model to ommit the translation of a few words. Anyway, the "DTR" and "DTR-N" performed worse than the ITR (Table 3.4).

Figure 3.3 shows the results achieved with search algorithms base on the most important 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

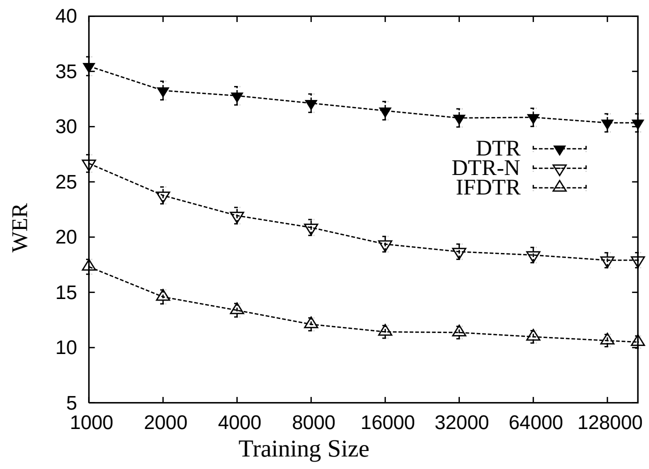

Figure 3.2: Asymmetry of the IBM Model 2 measured with the respect to the WER for the TOURIST test set for different training sizes.

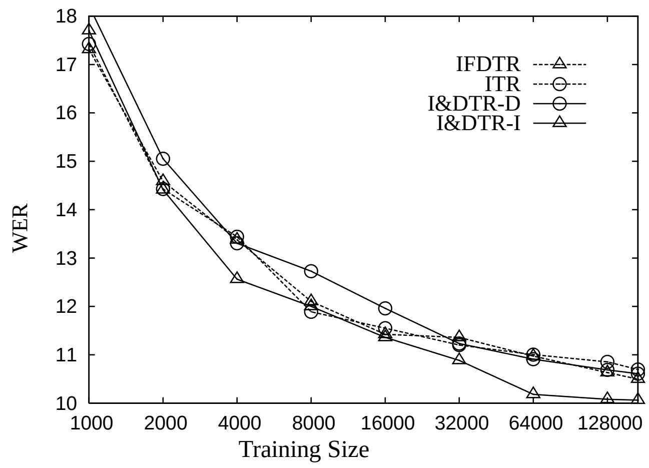

Figure 3.3: WER results for the TOURIST test set for different training sizes and different classification rules.

DTR (“IFDTR”) behaved similarly 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 3.4 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 in WER in Figure 3.3. 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.

Table 3.4: 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**	**0.847**	1.3	34
I&DTR D	10.6	51.6	0.844	9.7	2
IFDTR	10.5	60.0	0.837	2.7	35
ITR	10.7	58.1	0.843	1.9	43
DTR N	17.9	74.1	0.750	0.0	2
DTR	30.3	92.4	0.535	0.0	2

In order to check this hypothesis, Figure 3.4 shows the analogous version of Figure 3.3 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 becomes better. 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.

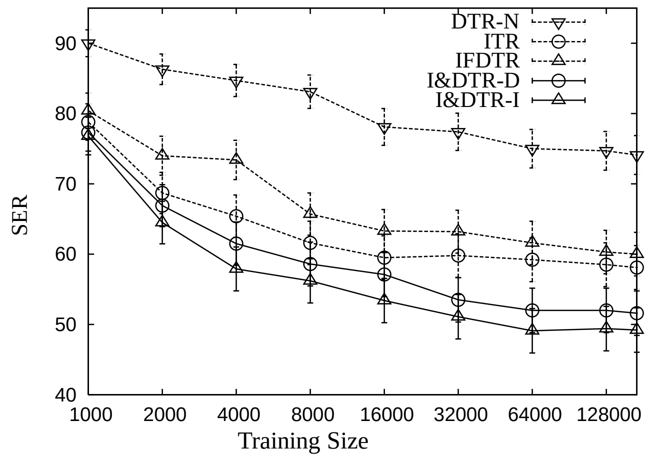

Figure 3.4: SER results for the TOURIST test set for different training sizes and different classification rules.

In conclusion, there are two sets of rules: the first set is made up of IFDTR and ITR, and the second 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 system it is expected that the behaviour of the rules correspond to the result obtained with

Table 3.5: 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	$p_r(\mathbf{x}\vert\mathbf{y})p_r(\mathbf{y})$	0.2681	61.138	45.24
DTR	$p_r(\mathbf{y}\vert\mathbf{x})p_r(\mathbf{y})$	0.2060	61.057	48.89
I&DTR	$p_r(\mathbf{y}\vert\mathbf{x})p_r(\mathbf{x}\vert\mathbf{y})p_r(\mathbf{y})$	**0.2813**	**58.988**	**43.34**
IFDTR	$p_r(\mathbf{x}\vert\mathbf{y})[p_r(\mathbf{y})]^2$	0.2223	62.460	48.30

English → Spanish				
Rule	Formula	BLEU	WER	PER
ITR	$p_r(\mathbf{x}\vert\mathbf{y})p_r(\mathbf{y})$	0.2567	60.734	45.83
DTR	$p_r(\mathbf{y}\vert\mathbf{x})p_r(\mathbf{y})$	0.1988	62.043	51.31
I&DTR	$p_r(\mathbf{y}\vert\mathbf{x})p_r(\mathbf{x}\vert\mathbf{y})p_r(\mathbf{y})$	**0.2606**	**59.441**	**45.12**
IFDTR	$p_r(\mathbf{x}\vert\mathbf{y})[p_r(\mathbf{y})]^2$	0.2148	62.685	49.37

the smallest corpus size, where no significant difference exists among the systems in terms of SER.

Phrase-based translation experiments

In order to perform translation experiments, different PBT 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++ [Och00] 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 [OGVC05]. Specifically, the refined symmetrisation method was used [ON04b].
- Finally, a phrase-based model was estimated, using the THOT toolkit [OGVC05].

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 [G$^+$01, OGVC03] 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 3.5 and 3.6 show the translation quality measures for the Europarl and Xerox tasks, respectively, for the different loss functions proposed in section 3.2. 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 with the ITR, it can be

Table 3.6: 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	$p_r(\mathbf{x}\|\mathbf{y})p_r(\mathbf{y})$	0.617	25.9	17.6
DTR	$p_r(\mathbf{y}\|\mathbf{x})p_r(\mathbf{y})$	**0.590**	**27.0**	**18.9**
I&DTR	$p_r(\mathbf{y}\|\mathbf{x})p_r(\mathbf{x}\|\mathbf{y})p_r(\mathbf{y})$	0.616	25.9	17.5
IFDTR	$p_r(\mathbf{x}\|\mathbf{y})[p_r(\mathbf{y})]^2$	0.606	26.2	18.0
English → Spanish				
Rule	Rule (Search Alg.)	BLEU	WER	PER
ITR	$p_r(\mathbf{x}\|\mathbf{y})p_r(\mathbf{y})$	0.636	25.6	18.5
DTR	$p_r(\mathbf{y}\|\mathbf{x})p_r(\mathbf{y})$	**0.628**	**26.0**	**19.1**
I&DTR	$p_r(\mathbf{y}\|\mathbf{x})p_r(\mathbf{x}\|\mathbf{y})p_r(\mathbf{y})$	0.646	25.1	18.1
IFDTR	$p_r(\mathbf{x}\|\mathbf{y})[p_r(\mathbf{y})]^2$	0.628	26.2	19.0

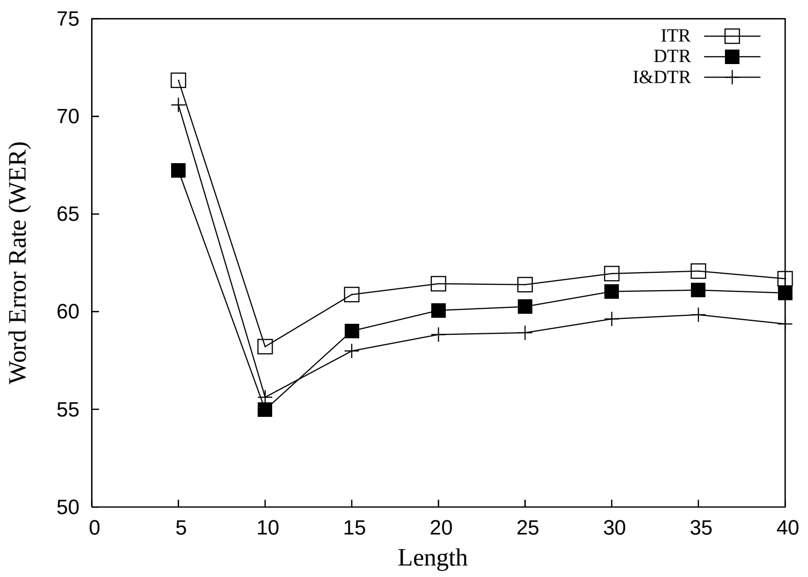

Figure 3.5: The WER results obtained for the Europarl test set (Spanish to English) with the length of the reference sentences restricted to be less than the value of the x-axis.

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.

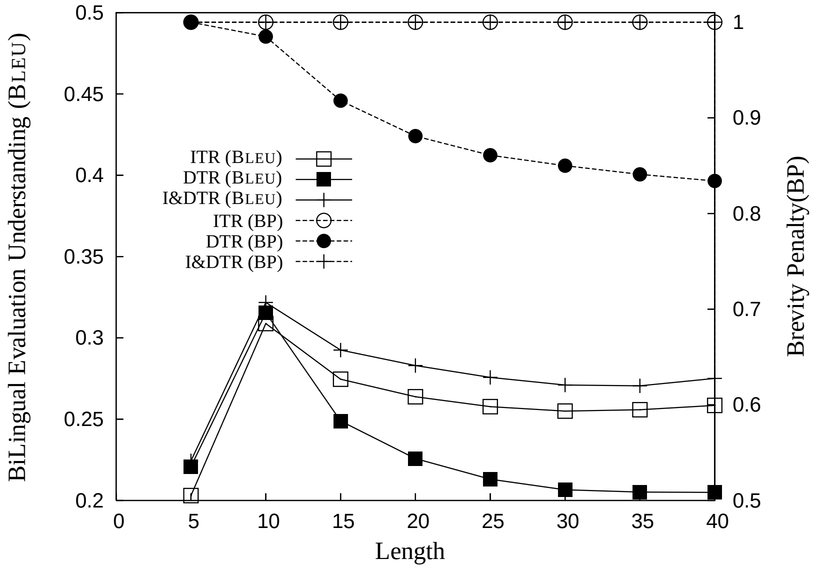

Figure 3.6: The BLEU results (on the left y-scale) and the brevity penalty (BP) of BLEU score (on the right y-scale) obtained for the Europarl test set (Spanish to English) with the length of the reference sentences restricted to be less than the value of the x-axis.

Tables 3.5 and 3.6 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 3.5 and 3.6 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. 3.6). Note that in Fig. 3.5, 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 3.7 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* using 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

Table 3.7: 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.

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 .
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 .
REF:	*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 .

by introducing a length normalisation in the loss function or in the models.

3.4 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 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. (3.9), 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 over 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. (3.12) 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 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. (3.9) 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 not lineal with the probability. In this sense more interesting loss functions could be obtained using information theory. For instance, we can penalise the system by the *remaining information*. That is, if we knew $p_r(\boldsymbol{x}, \boldsymbol{y})$, then the information associated with a target sentence $\mathbf{y}_c$ would be $-\log(p_r(\boldsymbol{x}, \boldsymbol{y}_c))$. The remaining information, or the information that the system has learnt when it fails is given by $-\log(1 - p_r(\boldsymbol{x}, \mathbf{y}_c))$, leading to the the error function

$$\epsilon(\boldsymbol{x}, \boldsymbol{y}_c) = -\log(1 - p(\boldsymbol{x}, \boldsymbol{y}_c)) \tag{3.42}$$

Figure 3.7, 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 $\mathrm{p}_r(\boldsymbol{y})$ or $\mathrm{p}_r(\boldsymbol{x}, \boldsymbol{y})$.

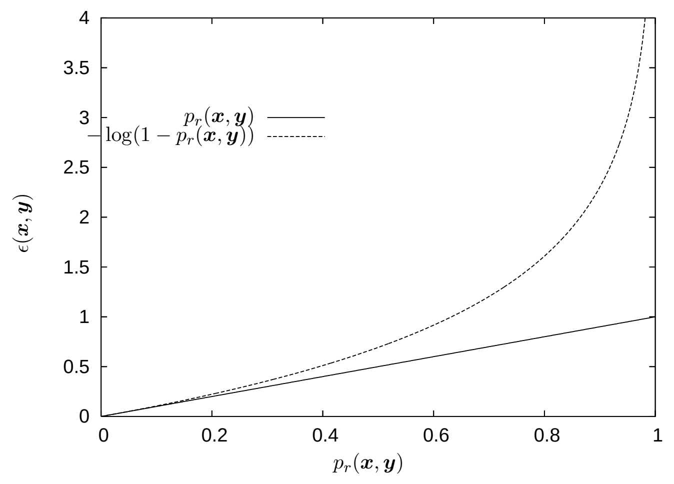

Figure 3.7: Difference between the remaining information and the probability as error functions.

Another very interesting research line is derived from approximating complex loss functions in Eq. (3.9) with simple loss functions in Eq. (3.12). Although, many of the state-of-art SMT systems indirectly make use of this idea, as analysed in section 3.2 (page 58), this idea can be exploited significantly in order improve the systems.

The part of the theory and the word-based results obtained in this chapter 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 obtained in this chapter were published in the following 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*. Volumen 29, pages: 1072–1081, 2008.

BIBLIOGRAPHY

[ABC$^+$96] 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.

[AO$^+$99] 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.

[Ato01] 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.

[B$^+$90] P. F. Brown et al. A Statistical Approach to Machine Translation. *Computational Linguistics*, 16(2):79–85, 1990.

[BPP96] A. L. Berger, V. J. Della Pietra, and S. A. Della Pietra. A maximum entropy approach to natural language processing. *Computational Linguistics*, 22(1):39–71, 1996.

[BPPM93] 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.

[DHS00] Richard O. Duda, Peter E. Hart, and David G. Stork. *Pattern Classification*. John Wiley and Sons, New York, NY, 2nd edition, 2000.

[G$^+$01] 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.

[GVC01] 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.

[Jel69] F. Jelinek. A fast sequential decoding algorithm using a stack. *IBM Journal of Research and Development*, 13:675–685, 1969.

[KB04] S. Kumar and W. Byrne. Minimum bayes-risk decoding for statistical machine translation, 2004.

[Kni99] Kevin Knight. Decoding complexity in word-replacement translation models. *Computational Linguistics*, 25(4):607–615, 1999.

[Koe05] P. Koehn. Europarl: A parallel corpus for statistical machine translation. In *Proc. of the MT Summit X*, pages 79–86, September 2005.

[KOM03] 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.

[M$^+$06] J. B. Mariño et al. N-gram-based machine translation. *Computational Linguistics*, 32(4):527–549, 2006.

[NAA06] `http:///nlp.cs.nyu.edu/hlt-naacl06/`, 2006.

[Och00] F. J. Och. GIZA++: Training of statistical translation models, 2000. `http://www-i6.informatik.rwth-aachen.de/\~och/software/GIZA++.html`.

[OGVC03] 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.

[OGVC05] 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.

[ON04a] F. J. Och and H. Ney. The alignment template approach to statistical machine translation. *Computational Linguistics*, 30(4):417–449, 2004.

[ON04b] F.J. Och and H. Ney. The alignment template approach to statistical machine translation. *Computational Linguistics*, 30(4):417–449, 2004.

[OTN99] 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.

[RSN05] 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.

[Sto02] A. Stolcke. SRILM – an extensible language modeling toolkit. In *Proc. of ICSLP'02*, pages 901–904, September 2002.

[TN03] 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.

[UM06] 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.

[UN04] 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.

[WW97] 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.

[Zen08] R. Zens. *Phrase-based Statistical Machine Translation: Models, Search, Training*. PhD thesis, RWTH Aachen University, Aachen, Germany, February 2008.

CHAPTER 4

CONCLUSIONS

“ *Entities should not be multiplied beyond necessity* ” OCCAM’S RAZOR

Contents

4.1 Summary

This thesis is mainly focused on two pattern recongnition aspects: the MLE estimation and the loss function. On both research lines, we have proposed new methods that have shown a similar or even better practical performance than classical methods. In theory, the advantages of the proposed methods are very appealing.

In chapter 2, conventional naive Bayes training with parameter smoothing has been restated as a constrained domain maximum likelihood estimation (CDMLE) problem for which an optimal, iterative algorithm has been proposed. The general idea behind our contribution is to avoid parameter estimates that may 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.

The work developed over chapter 3 is focused on the study of linear loss functions 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 widely used rules. For instance, this theoretical frame predicts an improvement (in terms of SER), 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 analysis of the loss function is an appealing research line. The results of analysing different loss functions range from allowing to use metric loss functions such as BLEU, or WER; to proving the properties of some outstanding classification rules such as the direct translation rule, the inverse translation rule or even the log-linear classification rule. For each different error function $\epsilon(\mathbf{x}, \mathbf{y}_j, \mathbf{y}_k)$, there is a different optimal Bayes' rule. The point of using one specific rule is an heuristic and practical issue.

The proposed modifications to the 0–1 loss function discussed in chapter 3 handle the intuitive idea of penalising a wrong action based on the repercussions of the correct action. For instance, if the correct translation, $\boldsymbol{y}_c$, for a given source sentence, $\boldsymbol{x}$, is a very unlikely sentence; a failure in the translation of such a sentence is not important. Oppositely, a failure in the translation of a likely sentence is an important mistake. It is important to note the fact that the proposed linear 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 an incorrect translation due to semantic errors. In order to take into account such cases, it is necessary to work with general loss functions despite its cost. However, the idea of penalising the mistakes proportionally to the correct translation probability can also be used in case of dealing with more complicated decision rules and, with more complicated search algorithms.

4.2 Scientific publications

The constrained maximum likelihood estimation technique analysed in chapter 2, yield 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.

and a publication in a journal is pending for acceptance:

- **J.Andrés-Ferrer** and Alfons Juan. Constrained domain maximum likelihood estimation for naive Bayes text classification. *Pattern Analysis and Applications (PAA)*. Pending

The theory and ideas developed in chapter 3 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.

and in the following 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*. Volumen 29, pages: 1072–1081, 2008.

4.3 Future work

We think that the CDMLE approach proposed in chapter 3 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 maximum likelihood estimation can be used without any side effect. Also, this constrained approach might be useful in the case of training criterium other than maximum likelihood such as discriminative training [**?**].

On the other hand, the extensions to the loss function that have been covered in chapter 3 are very promising as well. 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 be used to define error functions. Maybe the growing of the loss function should better be not lineal with respect to the probability. In this sense more interesting loss functions could be obtained using information theory. For instance, we can penalise the system by the *remaining information*. That is, if we knew $p_r(\boldsymbol{x}, \boldsymbol{y})$, then the information associated with a target sentence $\mathbf{y}_c$ would be $-\log(p_r(\boldsymbol{x}, \boldsymbol{y}_c))$. The remaining information, or the information that the system has learnt when it fails is given by $-\log(1 - p_r(\boldsymbol{x}, \mathbf{y}_c))$, leading to the the error function

$$\epsilon(\boldsymbol{x}, \boldsymbol{y}_c) = -\log(1 - p(\boldsymbol{x}, \boldsymbol{y}_c)) \tag{4.1}$$

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 $\mathrm{p}_r(\boldsymbol{y})$ or $\mathrm{p}_r(\boldsymbol{x}, \boldsymbol{y})$.

Another very interesting research line is derived from approximating complex loss functions in Eq. (3.9) with simple loss functions in Eq. (3.12). Although, many of the state-of-art SMT systems indirectly make use of this idea, as analysed in section 3.2 (page 58), this idea can be exploited significantly in order improve the systems.

Finally, 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, currently it is only feasible on constrained situations like n-best lists.

LIST OF FIGURES

LIST OF TABLES