Word Confidence Estimation and Its Applications in Statistical Machine Translation

Ngoc Quang Luong

UNIVERSITÉ DE GRENOBLE

THÈSE

Pour obtenir le grade de

DOCTEUR DE L'UNIVERSITÉ DE GRENOBLE

Spécialité : **Informatique**

Arrêté ministériel : 7 août 2006

Présentée par

Ngoc Quang LUONG

Thèse dirigée par **Laurent BESACIER** et
codirigée par **Benjamin LECOUTEUX**

préparée au sein du **Laboratoire d'Informatique de Grenoble**
dans **l'École Doctorale Mathématiques, Sciences et Technologies de l'Information, Informatique (MSTII)**

Word Confidence Estimation for Statistical Machine Translation

Thèse soutenue publiquement le **12, Novembre, 2014**
devant le jury composé de :

Mme, Catherine BERRUT
Professeur, Université Joseph Fourrier, Grenoble, Président
M, Kamel SMAÏLI
Professeur, Université de Lorraine, Nancy, Rapporteur
Mme, Lucia SPECIA
Professeur, Université de Sheffield, Angleterre, Rapporteur
M, Guilaume WISNIEWSKI
Maître de conférences, Université Paris-Sud XI, Paris, Examinateur
M, Laurent BESACIER
Professeur, Université Joseph Fourrier, Grenoble, Directeur de thèse
M, Benjamin LECOUTEUX
Maître de conférences, Université Pierre Mendès France, Grenoble, Co-encadrant de thèse

Acknowledgements

First and foremost, I wish to thank sincerely my supervisor, Mr. **Laurent Besacier**, for his enthusiastic guidances, academic supports, precious encouragements and constructive criticisms during the past three years. His novel ideas and in-depth knowledge in the domain have definitely inspired my thinking. On the rough road to finish the thesis, it is unavoidable to face with frustrating moments when the conference papers were rejected, or the experiments' results did not progress as expected, but he was always there with me, and together we analyzed to figure out the issues and find the solutions. Thank you so much Mr. **Laurent Besacier**, your contribution is an indispensable part for the success of my work.

I wish to send many thanks also to my co-advisor, Mr. **Benjamin Lecouteux**. I am really grateful for his support to deal with technical problems. I will never forget the times he spent hours sitting with me to fix bugs in the code or configure the complicated toolkits. Together with my supervisor, he contributed to my thesis with helpful advices and comments.

I would like to thank Mrs. **Catherine Berrut** for accepting to be the President of the jury committee of my defense. I also would like to thank Mr. **Kamel Smaïli** and Ms. **Lucia Specia** to become the reviewers of my thesis. Many thanks to Mr. **Guillaume Wisniewski** to become the member of the jury of my defense. I really appreciate your reports and advices and I strongly believe that they will play a great role to improve my thesis.

I express my sincere gratefulness to the French government for their financial support (via Campus France and French Embassy in Vietnam) dedicated to this thesis. My big and sincere thank also goes to the Laboratory of Informatics of Grenoble and the GETALP team in particular, the institution where I have been studying during three years. The hospitality and enthusiasm of all laboratory's staff leave in my mind the deep and unforgettable impressions.

I will be thankful to all my beloved friends in the laboratory: *Sarah, Frederic, Pat, David, Johann, Andrew, Nadia, Uyanga, Mateusz, Pedro, Marwen, Audrey, Aey, etc.* as well as my Vietnamese friends in Grenoble: *Vinh, Hung, Phuoc, Linh, Diem, Dung, Diep, Hiep, Bang, etc.* Together we shared the beautiful moments and the best memories in the PhD student's life. Thank you all of you guys, you are the warmest gift I get during my stay in France.

My sincere gratefulness for my beloved family. Thank you Dad and Mom: your tireless endeavors made me feel more than proud. Thank you my younger sister and her husband who encouraged me to study abroad and took care of my parents. And, the deep thank to my fiancée for sticking always by my side throughout the PhD journey, I would never reach success without your immense love and sympathy.

My most sincere thanks to all of you!

Abstract

Machine Translation (MT) systems, which generate automatically the translation of a target language for each source sentence, have achieved impressive gains during the recent decades and are now becoming the effective language assistances for the entire community in a globalized world. Nonetheless, due to various factors, MT quality is still not perfect in general, and the end users therefore expect to know how much should they trust a specific translation. Building a method that is capable of pointing out the correct parts, detecting the translation errors and concluding the overall quality of each MT hypothesis is definitely beneficial for not only the end users, but also for the translators, post-editors, and MT systems themselves. Such method is widely known under the name Confidence Estimation (CE) or Quality Estimation (QE). The motivations of building such automatic estimation methods originate from the actual drawbacks of assessing manually the MT quality: this task is time consuming, effort costly, and sometimes impossible in case where the readers have little or no knowledge of the source language.

This thesis mostly focuses on the CE methods at word level (WCE). The WCE classifier tags each word in the MT output a quality label. The WCE working mechanism is straightforward: a classifier trained beforehand by a number of features using ML methods computes the confidence score of each label for each MT output word, then tag this word with highest score label. Nowadays, WCE shows an increasing importance in many aspects of MT. Firstly, it assists the post-editors to quickly identify the translation errors, hence improve their productivity. Secondly, it informs readers of portions of sentence that are not reliable to avoid the misunderstanding about the sentence's content. Thirdly, it selects the best translation among options from multiple MT systems. Last but not least, WCE scores can help to improve the MT quality via some scenarios: N-best list re-ranking, Search Graph Re-decoding, etc.

In this thesis, we aim at building and optimizing our baseline WCE system, then exploiting it to improve MT and Sentence Confidence Estimation (SCE). Compare to the previous approaches, our novel contributions spread of these following main points. Firstly, we integrate various types of prediction indicators: system-based features extracted from the MT system, together with lexical, syntactic and semantic features to build the baseline WCE systems. We also apply multiple Machine Learning (ML) models on the entire feature set and then compare their performances to select the optimal one to optimize. Secondly, the usefulness of all features is deeper investigated using a greedy feature selection algorithm. Thirdly, we propose a solution that exploits Boosting algorithm as a learning method in order to strengthen the contribution

of dominant feature subsets to the system, thus improve of the system's prediction capability. Lastly, we explore the contributions of WCE in improving MT quality via some scenarios. In N-best list re-ranking, we synthesize scores from WCE outputs and integrate them with decoder scores to calculate again the objective function value, then to re-order the N-best list to choose a better candidate. In the decoder's search graph re-decoding, the proposition is to apply WCE score directly to the nodes containing each word to update its cost regarding on the word quality. Furthermore, WCE scores are used to build useful features, which can enhance the performance of the Sentence Confidence Estimation system.

In total, our work brings the insightful and multidimensional picture of word quality prediction and its positive impact on various sectors for Machine Translation. The promising results open up a big avenue where WCE can play its role, such as WCE for Automatic Speech Recognition (ASR) System (when combined with ASR features), WCE for multiple MT selection, and WCE for re-trainable and self-learning MT systems.

Keywords : statistical machine translation, Confidence Estimation, N-best list re-ranking, Boosting, Feature Selection, Quality Estimation.

Table of contents

Chapter 0

Introduction

In 2010, a Dutch news website[1] reported an unimaginable story about a Russian trucker, involved in a bar brawl in the Netherlands was released thanks to an error of Google Translate[2]. In the Russian version of the summons translated from Dutch (using this engine), instead of informing the trucker: *"you are to appear in court on 3 August 2010"*, the content went more like: *"you have to avoid being in court on 3 August 2010"*, which then helped him to go free. In Dutch, the infinitive verb *"voorkomen"* can produce two types of meanings: *"appear, occur"* or *"avoid, prevent"*, and the determination of the right one bases also on the context where it is used. Google Translate accidentally caused a serious situation since it flipped 180 degrees the text's content.

This is one of the serious yet not rare stories to demonstrate that the errors of Machine Translation (MT), in many cases, lead to unanticipated consequences. It is indisputable that MT systems - the computerized softwares that produce translations for text from one language to another (Hutchins and Somers, 1992) - have developed rapidly and fruitfully in recent decades. Google Translate - an online MT engine, hit a record in 2013 with more than one billon translations done a day[3]. With the support of smartphones, handheld devices and web applications for human communication, MT systems are about to bring more instant, more time flexible (e.g. at midnight) yet cheaper responses compared to human translators. Nowadays, the unprecedented flourish of free, online and interactive MT applications (e.g. Google Translate, Bing Translator, SysTran, Babel Fish, Apertium, Asia Online, Prompt, OpenLogos, and much more) gradually enhance human's ability to access a vast store of global information and knowledge, regardless of the language in which it is expressed. In many cases, they convey almost the content expressed in the source text, or at least enable readers to get the gist. In the first example of Figure 1, although the English translation for the French sentence: *"Ce*

[1] http://www.24oranges.nl/2010/08/18/russian-goes-free-thanks-to-google-translation-error/
[2] https://translate.google.fr/?hl=fr
[3] http://www.languageinsight.com/blog/2013/04/08/google-hits-1bn-translations-a-day/

soir, je vais manger avec Pascal Poulet" is imperfect, the English native readers are able to acquire the sentence's main message: Who? (I), do what? (eat), with whom? (Pascal) and when? (tonight). The translation contains only a small confuse biasing the source text content ("chicken" is not mentioned in the source text since "Poulet" is a family name in this context). However, there are also plenty of situations where the translation fails to carry the idea and causes the misunderstandings for users. In the source (English) part of the second example of Figure 1, the fans of Justin Bieber are wondering whether this singer can ever reach puberty. Nevertheless, its French translation gives a negative message: he can never reach it (*"peut jamais atteindre"*).

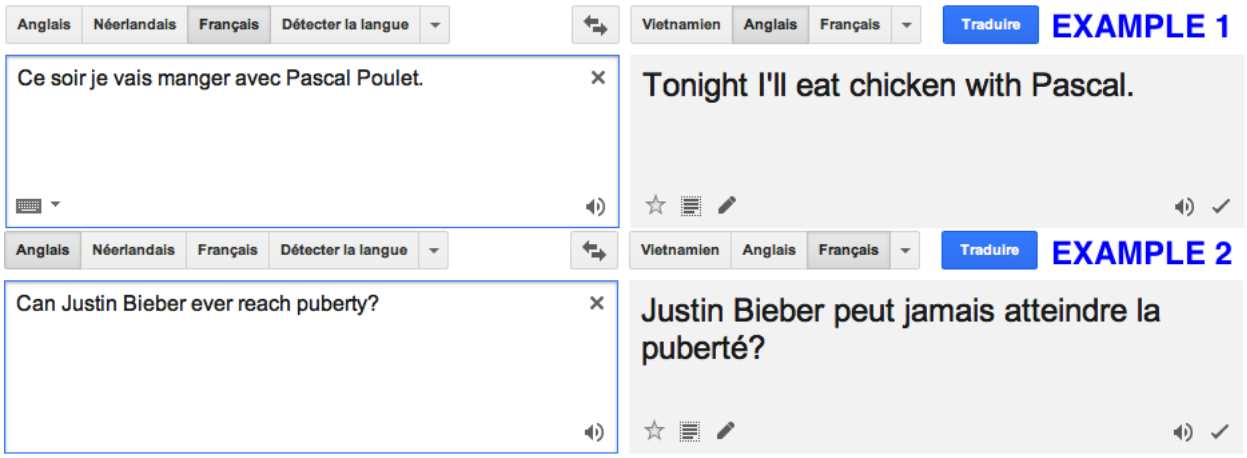

Figure 1 – Examples of Machine Translation errors. They can be minor and the readers are able to get the gist (Example 1), but in some cases so serious that cause misunderstanding (Example 2)

While appreciating MT breakthroughs, users face the fact that its quality still remains a chasm from perfection. Therefore, beside exploiting MT outputs for various purposes, the users need also to be aware of how trusty they are; or even better, which portions are reliable and which are not. Furthermore, these indicators are vital and helpful for other types of actors: the post-editors wish to be sure that correcting a specific translation will not be more time and effort consuming than reproducing an another one from scratch. The multiple MT services consider them pivotal criterion for opting the final translation among discrepant MT systems. MT systems themselves benefit from them to regenerate the better hypothesis. Obviously, using human resources is definitely cumbersome and impractical for performing this task, especially in case where the load of translations to be judged keep increasing and need instant quality responses. Instead, automatic methods which evaluate MT hypotheses at run-time are desirable for all above issues. These methods are commonly known under the name of **"Confidence Estimation"** (CE). Unlike the other evaluation metrics, CE systems are reference-free (e.g.

judging unseen translated text, without knowing anything about the gold-standard translations). Interestingly, they are shown to correlate significantly better with human judgments than the other standard ones such as BLEU[4] or NIST (Specia et al., 2010).

Passing a decade since the first work of Blatz et al. (2003) was published, this domain draws more and more attention and reaches preliminary achievements at different levels of granularity: sentence-level (Blatz et al., 2004; Nguyen et al., 2011; Specia et al., 2009a,b; Xiong et al., 2010), word-level (Nguyen et al., 2011; Ueffing and Ney, 2005, 2007; Ueffing et al., 2003), document-level (Scarton and Specia, 2014) or segment-level (Specia and Giménez, 2010). However, these contributions are just the first steps, and the road ahead is still very challenging (Blatz et al., 2004). In Sentence-level CE, which predicts the sentence's goodness, the major hindrance originates from the fact that an MT output is seldom fully correct at this level, but is the mixture between good and nonsensical portions. Proposing a satisfactory measure to decide whether or not a specific output is correct is therefore extremely hard. On the contrary, the quality prediction for each word (Word-level CE) is more doable, as it is more likely to be correct than the entire sentence. But, this advantage does not mean that this is a straightforward task, since the same words might have different meanings in different contexts. Compare to sentence- and word-level, the prediction for segments and documents are sparsely investigated, and also cope with other nontrivial obstacles (the complexity of discourse relations in document level, and the disambiguation of segment chunking and of the "segment correctness" itself). These harsh challenges can be mitigated if and only if a diversity of helpful characteristics for each translation unit to be judged are extracted from quality training corpora and the Machine Learning (ML) techniques which are capable of learning efficiently from this "knowledge base" are applied. Investigating new features, combining them wisely to avoid redundancy, proposing and applying ML models with tuned parameters, optimizing the predictor's performance, etc. are the main focuses of almost all researches in the domain of CE so far. Their efforts are annually presented in some well-known evaluation campaigns, such as WMT (Workshop on Statistical Machine Translation).

In this thesis, we concentrate mainly on Word-level CE (WCE thereafter), tagging each word in the MT output a quality label. These predicted labels not only inform target language readers about the unreliable words, but also speed up the post-editors' task by channelizing their attention at only erroneous portions, and help to choose best segments from multiple MT systems for system combination. Compare to sentence level CE, this sub-domain is less explored, pioneered by Ueffing et al. (2003). In this work, they used only the information from

[4]see the detailed definition in Chapter 1

SMT system itself to generate binary estimates. Taking cue from this idea, the later researches enriched the feature set by exploiting external resources and experimented with many other classification algorithms.

Our attempts throughout this thesis consist of two major objectives. The first objective is to build efficient WCE classifiers by integrating a variety of features of various types, examining different ML techniques and employing several optimization methods. The second objective is to apply the output given by WCE systems on many sectors of Statistical Machine Translation (SMT), including SMT N-best lists Re-ranking, SMT search graph Re-decoding, or Sentence level CE for SMT. The novelty of this thesis lies in the following contributions:

- Proposing a number of novel word-level features to combine with the conventional ones.
- Investigating the usefulness of features via three different corpora of two language pairs: French - English and English -Spanish.
- Deploying a strategy to train and complement from a number of weak WCE sub-models in order to obtain a composite strong one.
- Proposing sentence-level features based on WCE predicted labels to enhance the prediction performance of SCE systems.
- Establishing re-ranking features from WCE outputs and combining with decoder scores to re-rank the SMT N-best list and select a better candidate.
- Applying WCE scores (labels, probabilities) to update the SMT search graph which then raises up the optimal path to become MT new hypothesis.

The thesis consists of two main parts: State-of-the-art (Chapter 1 and Chapter 2), and Contributions (from Chapter 3 to Chapter 7):

In Chapter 1, we review some basic background of Machine Translation. The brief history with some remarkable points of its development road is summarized. The linguistic and computational architectures of MT along with all approaches for each architecture are listed. We then go further in SMT - the most popular and successful approach so far, starting with its mathematical model, followed by the indispensable components in each SMT system: Language Model (LM), Translation Model and Decoder. We do not forget to talk about some useful toolkits and resources which allow to build a baseline MT system in a fast and convenient way. Manual and automatic metrics to assess the MT quality are the last topic of this chapter.

Chapter 2 brings a complete picture of Word Confidence Estimation (WCE) - main aim of this thesis. The working mechanism of a WCE system, its essential components and major applications are first presented. After citing some most prominent work related to WCE, we lay stress on two issues: *"features need to build the predictor"* and *"ML methods to train them"*. A variety of broadly-used conventional prediction indicators, such as: target (source) side, alignment context, Word Posterior probability, LM Backoff, Part-of-Speech based, and other syntactic and semantic features are consequentially presented. Besides, we introduce ML methods that we will use to train our models, including: Naive Bayes, Logistic Regression, Decision Tree and Conditional Random Fields. The final part of this chapter presents the Workshop on SMT (WMT), where we participate in the WCE task for two years 2013 and 2014.

Our contributions begin from Chapter 3 with propositions to build the baseline WCE systems. We discuss first our (proposed) novel features to combine with the existing ones: Graph Topology, some syntactic-based, LM-based, POS LM-based and pseudo reference features. The following sections depict in detail all the steps for building WCE systems: the baseline SMT systems, the three corpora, the annotated training labels and ML model parameters. In the preliminary experiments, we combine all features in each system and train them using different methods. The performances are then analyzed and compared to other "naive" baselines.

Chapter 4 focuses on two main techniques to optimize the baseline systems built from Chapter 3. The first technique is "Feature Selection" strategy, in which we begin from the whole set and then eliminate the weakest feature in each iteration until it remains only one. The method, one part, helps to sort the set in the descending order of feature's usefulness, another part, returns the best performing feature combination which yields the best performance. In the second technique, which is also known as Boosting method, we divide the feature set into subparts and build weak sub models. The "weak" classifiers are then combined in a reasonable way to take advantage of their complementarity. The final "composite" classifier is expected to be much stronger than each individual thanks to this advantage.

From Chapter 5 to the end of this thesis, we exploit the outputs (labels, probabilities) predicted by WCE systems for some MT sectors. In this chapter, WCE predicted labels are used to synthesize seven sentence-level scores, from which we train a sentence-level quality predictor. We then compare this system to another baseline "pure" SCE system (where all features to train it are directly extracted at sentence level). Interestingly, we propose a method to combine the scores obtained from both of them and use the new "composite" score to classify sentences. The objective of all above tests is to investigate whether the information of word quality can help to improve the effectiveness of SCE predictors.

Chapter 6 concentrates on another contribution of WCE: improving MT quality via N-best list re-ranking. Motivated by the belief that SMT decoder scores are inadequate to build an efficient scoring (objective) function, then result in the decoder's bad decision, we attempt to add more six additional scores, synthesized from WCE predicted labels for each sentence of the list, into this function. In the second pass, our re-ranker takes the new parameter set to calculate again the score for all hypotheses in the list and select the one with new highest score to be the best translation. By doing so, we would like to examine whether the MT system can learn something from the WCE "feedbacks" to perform better. The experiments in this chapter are conducted in both "real" WCE labels and "oracle" (annotated) ones. In another in-depth analysis, we simulate the gradual improvement of WCE system performance and observe its impact on the re-ranking process. Some interesting conclusions are made from the results obtained at the end of the chapter.

The most outstanding contribution of WCE is described in Chapter 7, where its outputs are used to re-decode the SMT's search graph in the second pass of decoding. Relying on the quality labels (or confidence probabilities) provided by WCE system, we suggest some solutions to modify the total cost of all hypotheses in the graph containing these words. Specifically, the reward or penalty scores are determined by WCE outputs and their values are derived from either the current (first pass) best hypothesis's cost, or from the current hypothesis's transition cost. The updates strengthen paths in which many good words are found, whereas weaken those in the other way round. Finally, we seek the optimal path with the highest score over the new updated graph, then backtrack to read off the entire translation. Similar to Chapter 6, we compute the update scores based on both "real" and "oracle" WCE system's outputs. The comparison between two approaches "Re-ranking" and "Re-decoding" with WCE's assistance are then made and some important conclusions are reported to close the chapter.

At the end of the thesis, we resume our approach, emphasize the positive achievements harvested and open up some potential research avenues which can be built up from this work.

Chapter 1

Theory of Machine Translation

1 Introduction

Nowadays, along with the trend of globalization and economic integration, the demand for translations of a variety of documents is growing at a rate far beyond the present supply capacity of the translation profession. Machine Translation (MT) was born as an effective solution to assist human overcoming the language barriers in a faster, cheaper and more interactive way. Since decades, MT research has been spread out on numerous of branches, from training, optimizing, decoding, evaluating and exploiting different types of MT systems, as well as all accompanied resources of each task. No matter what the research topics are, they require the solid fundamental MT background. That is why before moving on the main goal of this thesis, **Word Confidence Estimation for SMT**, we start by reviewing all the key concepts of MT, as they are essential for exploring further our research topic.

The chapter begins with the general definition of MT and some remarkable points on its long history of development. Section 3 presents two major MT architectures: linguistic and computational architectures. Especially, the statistical approach for MT (SMT) - the systems that we focus on to build confidence metrics - is detailed in Section 4, containing all components (Language Model, Translation Model, Decoder). Some useful toolkits for building a SMT system along with free resources are introduced in Section 5. In the next section, we discuss about some popular metrics to assess MT quality. Finally, Section 7 summarizes the chapter.

2 General Definition and Brief History

The terminology **"Machine Translation (MT)"** is now a traditional and standard name for *"computerized systems responsible for the production of translations from one natural language into another, with or without human assistance"* (Hutchins and Somers, 1992). It has several

earlier names such as: *"mechanical translation"* and *"automatic translation"*. Although they are now rarely used in English, but their equivalents in other languages are still common (e.g. "traduction automatique" in French, "avtomatičeskii perevod" in Russian).

It is not a surprise when saying that the automatic mechanization of translation has been one of the oldest dreams of humanity, since long time. The reason is simple: there is too much and gradually-increased information that we would like to disseminate and understand in other languages has been translated by hand. As the world becomes more globalized, this problem becomes more challenging. Since human translators remain limited, expensive and inflexible, MT systems are becoming more and more effective and friendly support with (almost) free, instant and unlimited translations.

Nowadays, it is not an exaggeration when saying that human language interpretation has become a real "industry" with the involvement of top budget firms and government's investments. In this industry, translation activities are very diverse, depending on the riches and the popularity of each language, as well as its speaking community. However, regarding on the level of computer assistance, or in other words the "automation level", translation activities can be categorized into the following sorts:

- **Human translation**: The translation task is entirely conducted by human translators. This is the earliest and most popular translation activity and still important today, especially in case where even minor errors are strictly prohibited (e.g. translations of national constitutions, military treaties, billion-USD-cost purchase contracts, etc.).
- **Machine-aided human translation**: Human is still the main actor in the translation activity, yet their productivity can be improved and accelerated thanks to computers. Supporting toolkits can suggest the next words relying on the already translated words so far typed by humans, give hints about better candidates in the current contexts, correct grammatical mistakes.
- **Human-aided machine translation**: In this type the translation task is mainly performed by computer, yet improved by human. The human intervention can be made at any phase. For instance, before translating, they standardize the source text by removing strange symbols, replacing ambiguous or nonsensical words. During the translation process, if necessary, they are able to remove some source portions that are unknown (or hard to translate) for the decoder. Finally, after having results, they can post-edit it to get the optimal translations.
- **Fully automated machine translation**: Contradictory to the first type, this time humans have to do nothing but typing the source text or storing them under the right

format. Computers take responsibility to pre-process, generate hypotheses and select the best translation for each source sentence. This activity is suitable to translate a huge amount of text instantly (e.g. commercial websites) and users care much more about the gists than details (or they can predict the whole message starting from few key words).

Looking back at the MT history: numerous attempts have made in the past, in the U.S, Europe and some Asian countries, to automate various steps in the translation process. They range from simple online bilingual dictionaries, terminology banks and other translation aids to complete MT systems, as described in (Dorr et al., 1999; Hutchins, 2007).

The years of 1950s witnessed the start-up of several pioneer MT projects and research activities. In the first MT conference, held at Massachusetts Institute of Technology (MIT), USA in June, 1952; almost in-domain researchers agreed that fully automated MT systems with human acceptable quality will be hard to obtain; and emphasized on the human interventions either before or after the translation process. As an activity of showing the feasibility of MT systems, two researchers: Peter Sheridan (IBM) and Paul Garvin (Georgetown) launched their joint MT system on January, 7th, 1954. In their rule-based approach, only six grammar rules are applied over a vocabulary of 250 words to translate 49 Russian sentences into English. Interestingly, such a simple and not significant scientific value attracted a special care of American media, since it demonstrates the dream about a translation engine implemented by machine and handled by human is totally viable. Motivated by this potential research stream, there were many groups have established in the period from 1954 to 1956, e.g. in Cambridge (UK), Milan (Italy), Moscow (Russia), etc.

The early years of 1960s were considered as prosperous for MT researches, while numerous MT research groups had been established in many countries throughout the world, spreading broadly from Europe (Germany, France, Hungary, Bulgaria, Belgium, etc.) to Asia (China, Japan) and America (Mexico, USA), etc. Although human resources were flourished these years, yet remarkable achievements were limited, and one of the most outstanding results came from the work of Benard Vauquois (Grenoble, France). The author developed a system for translating Russian mathematics and physics texts into French, partially based on interlingua approach. In this work, he used a pivot language to represent the logical properties of syntactic relationships and a bilingual transfer mechanism to translate lexical items.

However, in 1964, MT researches faced a "crisis", followed by the funding cut-off as well as the community's neglectfulness afterward. The Automatic Language Processing Advisory Committee (ALPAC) reported a status of MT development, in which they doubted the feasibility of automatic MT system. The report illustrated by giving two following examples:

The pen is in the box. [i.e. the writing instrument is in the container]
The box is in the pen. [i.e. the container is in the playpen or the pigpen]

In these above examples, the word *"pen"* shows two different meanings, and there is no way to predict the right one but relying on the context that might come before or after each sentence. The report therefore raised a question of how the MT system can learn the way to *"remember" a context* and make use of it to interpret the correct meaning of words and phrases?

One of the most significant MT systems, which was born and developed in the early years of 1970s and then installed for many intergovernmental organizations and major corporations, is SYSTRAN. Starting from a Russian - English version (1970), and then English - French (1976), the system was rapidly enlarged to deal with the translation from most of the European languages into English, and the another way round for several of them (e.g. French, Italian, German, Spanish, Portuguese, etc.). Its components were turned more and more flexible to enable the development of a new language pair.

Following the success of SYSTRAN, many other systems appeared in the market (commercial versions). METAL (1982), developed in Munich, Germany, supported initially German - English translations, and then were extended in other languages, such as Dutch, Spanish, French. WEIDNER system (1981) was designed for microcomputers and mainly concentrated on Japanese - English. Besides, personal computer market was also the goal of other commercial softwares, such as PC-TRANSLATOR (from Linguistic Products), GTS (from GlobalLink).

In 1991, the first statistical machine translation (SMT) model was proposed by IBM, and trained by a huge number of source and target aligned sentence pairs. This signaled the flourish of SMT systems in the years afterward, lasting until now. At the end of the 1990s, a project aiming at multinational interlingua MT, was kicked off by the Institute of Advanced Studies of the United Nations University (Tokyo), for initially the six official languages of the United Nations and other widely spoken languages, involving the participation of the groups from 15 different countries. More insightful analysis on the evolution of MT systems and researches can also be referred from (Hutchins, 2007)

3 MT System's Architecture

The MT architecture can be divided into two main classes: *linguistic architecture* and *computational architecture* (Boitet, 2008). The linguistic architecture of an MT system is characterized by the representations (e.g. language transfer rules, interlingua) used during the translation process; meanwhile the computational architecture is characterized by the calculation methods

used during the translation process.

3.1 Linguistic Architectures

All possible linguistic architectures of an MT system are represented in the **"Vauquois triangle"** (Vauquois, 1968), as in Figure 1.1, containing:

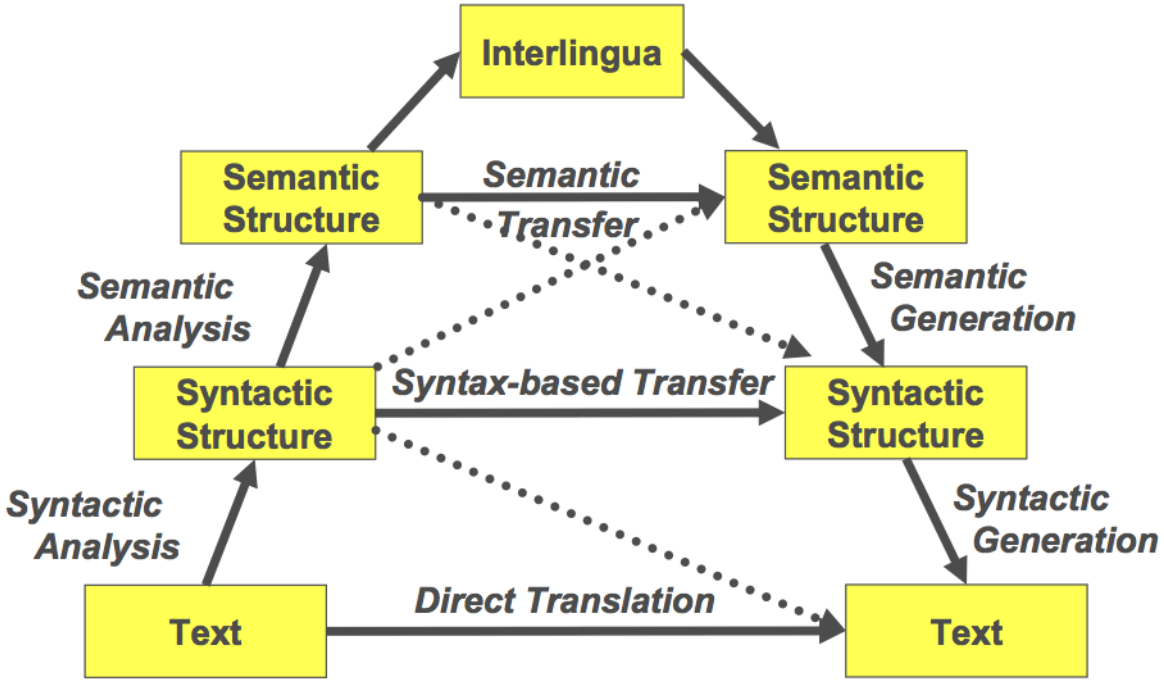

Figure 1.1 – Vauquois's triangle

- **Direct translation systems**: With these systems, the translation from one language to the another can be performed in one unique step, without the need of analyzing the source sentence, as well as generating the target one. The translation is done by using the bilingual dictionary to translate a word or a sequence from source to target language (Hutchins and Somers, 1992). It is not tough to realize that this approach contains plenty of drawbacks and limitations, with the lack of syntactic, lexical and semantic analysis. These shortcomings can be mitigated by the "indirect" translation systems, categorized into two classes: transfer and interlingual systems.
- **Transfer systems** : In these types of translation systems, the translation process contains three main steps: (1) source sentence analysis, (2) transformation, and (3) target sentence generation. They require the linguistic knowledge in both sides, represented by

the dictionary, grammar, etc.; and the transformation rules which connect two languages (e.g. a transformation rule table) (Arnold et al., 1994)

- **Interlingual Systems**: They employ an intermediate language (or pivot language, interlingua) to transform one language into another one. First, the source sentences are analyzed and transferred to the pivot language. Next, the target sentences are generated based on this pivot translation. One advantage of this approach is that it can be easily applied for multiple language translation systems. Indeed, in order to translate between n different languages, we need n analyzers and n generators instead of employing a total of $n(n-1)$ transfer systems. Nevertheless, theoretically, it is complicated to establish these analyzers as well as generators.

3.2 Computational Architectures

The MT computational architecture can be defined according to the computational methods for performing the translation task, mainly including three categorizations:

- **Rule-based MT (RBMT)**: is a general term that denotes machine translation systems based on linguistic information about source and target languages. Basically, these information can be retrieved from unilingual, bilingual or multilingual dictionaries and grammars covering the main semantic, morphological, and syntactic rules of each language respectively. Having an input source sentence, an RBMT system generates them to output target sentence on the basis of morphological, syntactic, and semantic analysis of both the source and the target languages involved in a concrete translation task. Some popular types of grammars that can be used in RBMT system to model the languages are: Lexical Functional Grammars (LFG) (Kaplan and Bresnan, 1995), Head-driven Phrase Structure Grammar (HPSG) (King, 1999), or Tree Adjoining Grammar (TAG) (Kroch and Joshi, 1985).
- **Example-based MT (EBMT)**: is a method of machine translation often characterized by its use of a bilingual corpus with parallel texts as its main knowledge base, at run-time. These MT systems are trained from bilingual parallel corpora, which contain sentence pairs between a sentence in one language with their translation into another. For doing the translation, the input sentence is firstly decomposed into certain fragmental phrases. Then, these fragmental phrases are translated into the target language phrases by the analogy translation principle with proper examples as its reference. Finally, the target language phrases are composed into one long target sentence (Nagao, 1984).

- **Statistical Machine Translation (SMT)**: is a machine translation paradigm where translations are generated on the basis of statistical models whose parameters are derived from the analysis of bilingual text corpora. These resources allow to estimate the objective functions of the probabilities of target text (words, n-grams, sequences) given a corresponding source text. During the decoding, for each source input, SMT decoder searches for the target hypothesis that maximizes this function value, and consider it as the best translation (Brown et al., 1990, 1993a). The first ideas of statistical machine translation were introduced by Warren Weaver in 1949. The availability of huge bilingual texts as well as the high linguistic independence makes SMT take an increasing importance in the translation industry in recent years with many online engines: Google Translate[1], Bing (Microsoft)[2], etc. This approach will be introduced in detail in the next section.

4 Statistical Machine Translation

4.1 Background

For each source sentence represented by a string $f_1^J = f_1...f_j...f_J$, an SMT system seeks the target sentence $e_1^I = e_1...e_i...e_I$ that maximizes the following posterior probability:

$$\hat{e}_1^{\hat{I}} = \arg\max_{I,e_1^I}\{Pr(e_1^I|f_1^J)\}(1.1)$$

By applying Bayes rule, the above expression can be rewritten as follows:

$$\hat{e}_1^{\hat{I}} = \arg\max_{I,e_1^I}\{\frac{Pr(f_1^J|e_1^I) * Pr(e_1^I)}{Pr(f_1^J)}\}(1.2)$$

We note that the source text remains unchanged during the entire translation process, therefore the probability $Pr(f_1^J)$ does not influence the *argmax* function's calculation. In other words, the task becomes maximizing the product between two components: $Pr(f_1^J|e_1^I)$ and $Pr(e_1^I)$, and both of them can be modeled separately:

- In SMT, the probability $Pr(e_1^I)$ is called the Language Model (LM), which provides the probability of the target words' occurrences in the target language. It captures the well-formedness of the target sentence's syntax. The LM is therefore built on a monolingual

[1]http ://translate.google.com
[2]http ://www.bing.com/translator

corpus of the target language E. Normally, the LM is organized as a multiple-entry file, in which each entry is a sequence of n target words $n = 1, 2, 3, 4, ...$ (n-grams) and its occurrence probability in the target language.

- Meanwhile, the probability $Pr(f_1^J|e_1^I)$ is also known as the Translation Model (TM). The mathematical formula suggests that this model is responsible for searching the word sequence e in the target language which can be considered as the best translation of the source word sequence f (having the highest probability). The translation model is trained on a parallel aligned corpus. To simplify, we can imagine it as a "bilingual dictionary", in which each entry is a relation between a word group of the source language with its target partner (its translation), and each of such relation is assigned with a probability.

Once having these models, the job of SMT decoder is searching among its translation hypotheses the one that gives the highest product of their scores.

4.2 Language Model

One essential and indispensable component of any SMT system is the Language Model (LM), which measures how likely it is that a sequence of words S would be uttered by a target language speaker. In other words, it tells us how likely when we pick randomly a word sequence, it turns out to be S. As stated before, it handles the "syntactic" part of each SMT output, which plays vital role to its quality, since we want a SMT system not only to produce words that are true to the original meaning, but also to connect them properly to form a fluent target language sentence. Apart from MT, the Language Model is also widely used in numerous other NLP applications, such as POS tagging, Information Retrieval (IR) (Manning et al., 2008), Speech Recognition (Rabiner and Juang, 1993), etc.

Among different LM methods, the leading and most well-known one is called **n-gram** language modeling, which is based on statistics of how likely words are to follow each other. Why do we have to break down the entire sentence into **n-gram** for calculating the probability? It is obvious that, in order to calculate the probability of a target sentence $Pr(e_1^I)$ in the target language, one can simply collect a large amount of text and count how often e_1^I occurs in it. Unfortunately, most of the long sequences cannot be found in the corpus at all, even when it is enormous. Hence, this decomposition helps collect sufficient statistic and estimate the probability distributions we need.

According to the **n-gram** model, the probability $Pr(e_1^I)$ in the target language can be

decomposed into:

$$Pr(e_1^I) = Pr(e_1 e_2 ....e_I) = \prod_{i=1}^{I} Pr(e_i|e_1 e_2 ....e_{i-1}) \tag{1.3}$$

Since this model hypothesizes that the word e_i of the sentence is dependent to only *(n-1)* preceding words, the above probability can be ***approximated*** as follows:

$$Pr(e_1^I) = Pr(e_1 e_2 ....e_I) = \prod_{i=1}^{I} Pr(e_i|e_{i-(n-1)} e_{i-(n-1)+1} ....e_{i-1}) \tag{1.4}$$

Specifically, with $n = 2$, we obtain the bigram LM, and with $n = 3$ this is called trigram LM. In trigram LM, for example, the above probability becomes:

$$Pr(e_1^I) = Pr(e_1 e_2 ....e_I) = Pr(e_1) Pr(e_2|e_1) \prod_{i=3}^{I} Pr(e_i|e_{i-2} e_{i-1}) \tag{1.5}$$

All the probabilities in the above product can be estimated thanks to a huge monolingual target corpus. We would like to emphasize that the target corpus must be big enough to be considered as a representative of the general target language; and the bigger it is, the more accurate these probabilities will be (and of course the more sophisticated storing and searching issues will be). In its simplest form, the estimation of trigram word prediction probabilities, for instance: $Pr(e_3|e_1 e_2)$, can be conducted by counting how often in the training corpus the sequence $e_1 e_2$ is followed by the word e_3, as opposed to other words. Based on the maximum likelihood estimation, we compute:

$$Pr(e_3|e_1 e_2) = \frac{count(e_1, e_2, e_3)}{\sum_e count(e_1, e_2, e)} \tag{1.6}$$

where $count(e_1, e_2, e_3)$ is the number of occurrences of the sequence e_1, e_2, e_3 in the training corpus, whereas the sum over all $count(e_1, e_2, e)$ represents the number of occurrences of the sequence e_1, e_2 in the training corpus.

However, the **n-gram** model can encounter a problem when the n-gram contains a word that cannot be found in the corpus. No matter how big the training corpus is, it is impossible to cover all words of the target language (especially for proper names, wrongly spelled words, or those from specific domain, etc). These words are commonly known as OOV (Out-Of-Vocabulary). Even when all words in the n-gram occur in the LM, they might not appear in the right order at the decoder time. These phenomena constitute the zero-value of the n-gram prediction probability, and therefore the entire sentence's probability, which makes

the LM model meaningless. In order to overcome this non-trivial shortcoming, we need some techniques to adjust the empirical counts to the expected counts (to avoid zero counts). They are called *smoothing* algorithms. The add-one smoothing (Chen and Goodman, 1996) adds a fixed number (say, 1) to every n-gram count for eliminating counts of zero. By doing so, we need to consider not only the current n-grams in the corpus, but also all imaginary n-grams that can be made from the corpus's vocabulary. Meanwhile, back-off smoothing (Katz, 1987) raises an idea of stepping back to lower order n-gram with richer statistics, rather than staying at the current n-gram with zero count. It is obvious that we have to define the appropriate weights to weaken the back-off probability below the original one yet still maintain a positive value.

Measuring the LM's quality is also very important. We know that the LM model performance depends on multiple factors, such as the corpus on which it is trained, the value of n (n-gram) used, or the smoothing techniques employed, etc. Ideally, a good LM should assign a high probability to a fluent, meaningful sentence and a low probability to any ill-formed and influent one. In order to measure this capability, **"perplexity"** is proposed (Jelinek et al., 1977) and then becomes very popular in ML evaluation. Perplexity is defined through the entropy (measure of uncertainty) of the LM, which can be written as:

$$H(p_{LM}) = -\frac{1}{n} log p_{LM}(e_1 e_2 .... e_I) = -\frac{1}{n} \sum_{i=1}^{I} log p_{LM}(e_i | e_1 e_2 .... e_{i-1}) \tag{1.7}$$

And the perplexity is a simple transformation from this cross-entropy:

$$PP = 2^{H(p_{LM})} \tag{1.8}$$

According to this definition, a LM L_1 is called better than L_2 if L_1 assigns a lower perplexity (i.e. lower entropy, higher probability) to the identical good-quality test sentence (or corpus) than L_2.

4.3 Translation Model

To train the translation model, the most crucial and indispensable step is word (or phrase) alignment. Given two sentences in which one is the translation of another, word (or phrase) alignment can be considered the connection between these units. According to the type of alignments, translation models can be divided into two main groups: word-based and phrase-based models.

4.3.1 Word-based Model

Let $A = a_1^J = a_1 a_2 ... a_J$ denote the alignment information between the source sentence f_1^J and the target one e_1^I. Each element a_j represents the alignment information for the source word f_j and can take value in the interval $0, 1, ..., I$. For instance, $a_j = i$ means that the word f_j is aligned to e_i. In case of $i = 0$, the word f_i has no actual aligned target word, so we assign it with the "null" word e_0. The probability $Pr(f|e)$ now becomes the sum over all probabilities corresponding to all possible alignments between two sentences:

$$Pr(f|e) = \sum_a Pr(f, a|e) \tag{1.9}$$

The five IBM models define their own way to compute $Pr(f|e)$ (Brown et al., 1993a). In **IBM Model 1** and **IBM Model 2**, in order to generate the sentence f, we need to determine first its length. Then, with each position $j = 1...J$, we select the position a_j of its aligned target word, which depends on the target sentence e_1^I, the length J and the previously aligned words (a_1^{j-1}, f_1^{j-1}). Finally, the word f_j can be produced based on a_1^j, f_1^{j-1}, J and e_1^I. The probability $Pr(f, a|e)$ can be defined as in the equation 1.10:

$$Pr(f, a|e) = Pr(J|e) \prod_{j=1}^{J} Pr(a_j|a_1^{j-1}, f_1^{j-1}, J, e) Pr(f|a_1^j, f_1^{j-1}, J, e) \tag{1.10}$$

The **IBM Model 1** assumes that each word f_j can be aligned to any e_i with the identical probability; and the probability $Pr(J|e)$ is a constant (ϵ). These above assumptions make the translation model to depend solely on the lexical model represented by the conditional probability between f_j and e_{a_j}, denoted by $t(f_j|e_{a_j})$:

$$Pr(f, a|e) = \frac{\epsilon}{(I+1)^J} \prod_{j=1}^{J} t(f_j|e_{a_j}) \tag{1.11}$$

After some mathematical transformations, $Pr(f|e)$ can be obtained by:

$$Pr(f|e) = \frac{\epsilon}{(I+1)^J} \prod_{j=1}^{J} \sum_{i=0}^{I} t(f_j|e_i) \tag{1.12}$$

The **IBM Model 2** adds an explicit model for alignment beside the lexical model $t(f|e)$, which specifies the position of the source word to which the target one is aligned. So, this probability

depends on the position of the target word, as well as the target sentence's length.

$$Pr(f, a|e) = \epsilon \prod_{j=1}^{J} t(f_j|e_{a_j}) a(a_j|j, J, I) \tag{1.13}$$

After some methematical transformations, we obtain:

$$Pr(f|e) = \epsilon \prod_{j=1}^{J} \sum_{i=0}^{I} t(f_j|e_i) a(i|j, J, I) \tag{1.14}$$

where $\sum_{i=0}^{I} a(i|j, J, I) = 1$

The Hidden Markov Model (**HMM**)(Rabiner and Juang, 1986) is similar to **IBM Model 2** when both distortion and lexical laws are taken into account. The only difference is that in **HMM**, the distortion law depends not only on the position of the target words as well as the length of target sentence as **IBM Model 2**, but also on the alignment information of the preceding word.

In **IBM Model 3, 4 and 5**, three steps are applied to transform from the source to the target sentence, including:

- Since the number of words in the source sentence is different in general from that in the target one, each word e_i can select a number of ϕ_i words in the target sentence that it will generate (called its *fertility*). Whenever we meet e_i in the source sentence, we will produce ϕ_i copies of e_i in the target one.
- For ϕ_i copies of e_i, we produce words in the target sentence f_1^J.
- We re-order words in the target sentence to obtain the final sentence.

The **IBM Model 3** uses fertility law $n(\phi_i|e_i)$ beside alignment and lexical ones for each position $i \in [1, I]$ in the source sentence. ϕ_i is the number of target words aligned to e_i. This model also supports the situation where the target word f_j has no actual alignment (consider to be aligned with e_0). We call them the **false words**. The probability of the target word aligned to a real source word is p_0, and the probability to generate a false word is $p_1 = 1 - p_0$. The distortion (alignment) model of this model is identical to that of model 2.

The **IBM Model 4** differs from the model 3 by its distortion model, which defines two distortion laws. The first one locates the first target word generated by the source word e_i, and the second one locates the remaining $\phi_i - 1$ words. The **IBM Model 5** is almost analogous to the model 4, except one difference that it removes the deficiency. The distortion model of IBM-4 does not take into account the target positions that have already been covered. Furthermore,

IBM-4 assigns the probabilities for words that are not really target word sequences. Thanks to the deficiency elimination, IBM-5 yields much more promising results. This complicated model, along with IBM Model 3 and 4 are detailed in (Brown et al., 1993a).

4.3.2 Phrase-based Model

As stated above, in word-based models, each source word is separately translated into a target word, yielding therefore an erroneous translation in almost cases, especially when a sequence of source words can be translated into just one unique target one. Moreover, we all know that the translation of a source word into another language is not simple as a "word-by-word" matching, but depends on the context where it is placed, or more specific, its surrounding words. That is the motivation for exploring phrase-based translation. In this approach, unlike word-based one, the translation's basic unit is a phrase (word sequence), which will be translated into its corresponding target phrase.

Nowadays, phrase-based models become the most widely-used and outperform the other ones thanks to their capability to manage better the word reordering, as well as to translate the idiomatical expressions. With the training conducted on a huge corpus, it is likely that the phrase table covers the entire input sentence, which can result in the perfect translation. There are numerous well-known work on this approach (Koehn et al., 2003; Marcu and Wong, 2002; Och and Ney, 2004). With this model, the translation process contains three main steps:

- Split the source sentence f into a sequence of phrases $\overline{f_1^I}$.
- Translate each source phrase $\overline{f_i}$ in $\overline{f_1^I}$ into the target phrase $\overline{e_i}$, using the translation probability $\phi(\overline{f_i}|\overline{e_i})$.
- Re-order the translated phrases to form a "make-sense" and fluent target sentence. This phrase is modeled by a relative distortion probability distribution $d(a_i - b_{i-1})$, where a_i denotes the start position of the foreign phrase that was translated into the i-th English phrase, and b_{i-1} denotes the end position of the foreign phrase translated into the *(i-1)*th English phrase.

The translation probability of a phrase $\phi(\overline{f_i}|\overline{e_i})$ is computed based on Maximum Likelihood Estimation, and can be written is:

$$\phi(\overline{f_i}|\overline{e_i}) = \frac{count(\overline{f_i}, \overline{e_i})}{\sum_{\overline{f_i'}} count(\overline{f_i'}, \overline{e_i})} \tag{1.15}$$

In summary, the best target output sentence $\hat{e}$ given an input sentence f according to the phrase-based model is:

$$\hat{e} = argmax_e Pr(e|f) = argmax_e Pr(f|e) Pr_{LM}(e) \tag{1.16}$$

where $Pr(f|e)$ is decomposed into:

$$Pr(\overline{f_i}|\overline{e_i}) = \prod_{i=1}^{I} \phi(\overline{f_i}|\overline{e_i}) d(a_i - b_{i-1}) \tag{1.17}$$

4.4 Log-linear Model

The log-linear model underlying SMT makes the combination of multiple components in which each reflects one dimension of the translation's quality. Each component is represented a set of features, which are weighted and multiplied together. In this model, the probability of a translation e of an input sentence f is computed as follows:

$$Pr(e|f) = \prod_i h_i(e|f)^{\lambda_i} \tag{1.18}$$

where h_i are the feature functions and λ_i are the corresponding weights. However, in many case, the representation of this probability is hindered by its too tiny value (≈ 0), and needs more convenient way. One effective proposition is to use logs, which yields the negative value instead. By doing so, what we will compute becomes:

$$logPr(e|f) = \sum_i log(h_i(e, f))\lambda_i \tag{1.19}$$

The tuning stage of the decoder is used to set the weights. They can be optimized by several common algorithms: Minimum Error Rate Training (MERT) (Och, 2003), or MIRA (Margin Infused Relaxed Algorithm) (Crammer et al., 2006). Moreover, in log-linear model, we typically employ the following models for calculating the objective function value: phrase translation model, language model, distance-based reordering model , word penalty, lexicalized reordering model, etc. The model also supports the integration of other additional feature functions in order to enhance the system's performance.

Figure 1.2 – Translation options are created when translating a Spanish sentence into English (source: Callison-Burch and Koehn (2005))

4.5 SMT Decoder

Given a source sentence f_1^J, SMT system seeks the best translation $\hat{e}_1^I$ which maximizes the probability:

$$\hat{e}_1^I = \arg\max_{I,e_1^I} \{Pr(e_1^I|f_1^J)\} \tag{1.20}$$

The searching process over all possible translations T for the optimal one is the decoder's responsibility. This section discusses the phrase-based decoding process with all detailed steps, as implemented in **Pharaoh** (Koehn, 2004) and **Moses** (Koehn et al., 2007) systems. It starts by collecting translation options, then applies them to expand hypotheses and searches for those with cheapest cost. We also present some techniques to avoid the search space explosion, including *"Hypothesis Recombination"* and *"Pruning"*.

4.5.1 Collecting Translation Options

The decoding process starts by collecting all the possible translation options given a source sentence. First, we consider all possible adjacent source word sequences and employ the phrase table to match these sequences with the corresponding target sequences. Relying on the phrase table score of the source - target matching, the future cost of every target sequence (until all source words are covered) is estimated. Among the totality of possible hypotheses, only top scoring sequences (regulated by table pruning parameters) are retained, the remaining are pruned. The retained target sequences are then directly used for beam search process and are called the **translation options**. In SMT, the decoder encapsulates translation option along with its model scores. Ultimately, once all translation options for a particular source sentence are collected, the search phase can proceed.

Figure 1.2 gives an example about a number of translation options created when translating the Spanish sentence: *"Maria no dio una bofetada a la bruja verde"* into English. This example will be used to illustrate other decoding phases throughout this section. It is straightforward

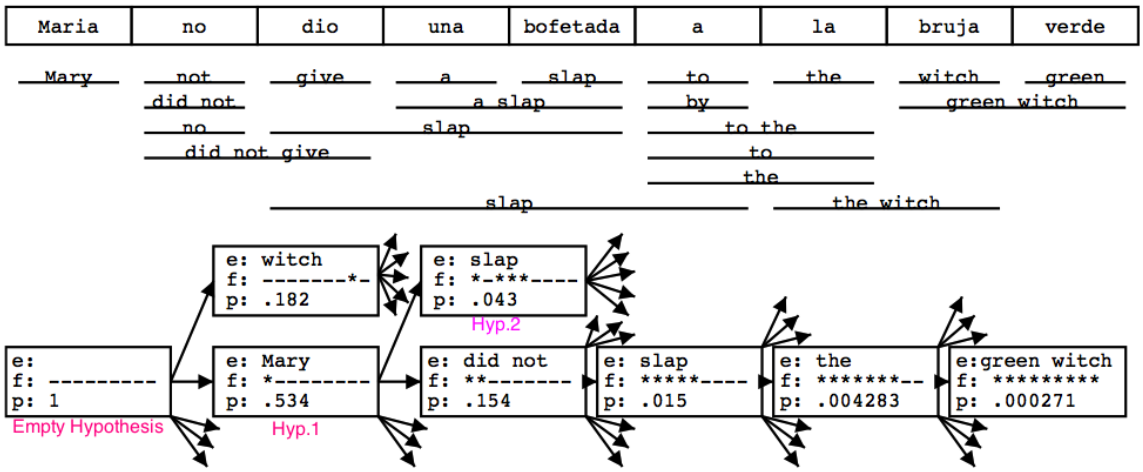

Figure 1.3 – New hypotheses are formed by applying translation options on the existing ones, until all source words are covered (source: Callison-Burch and Koehn (2005))

to see that the translation options vary from short (*"Mary"*) to long (*"did not give"*) and the total number of them can be huge since there are many different ways to segment source words into phrases, and also many different ways to translate each phrase when refering to the phrase table. They cover a single or a number of contiguous source word(s).

4.5.2 Expanding Hypothesis

A **hypothesis** is a partial translation, generated by applying a translation option to translate one (or multiple) source words, in which each source word is translated one and only one time. During the translation process, the hypothesis does not need to be built from the beginning, but by applying a translation option to a previously existing hypothesis. We call this process the **hypothesis expansion**. The expansion starts with an empty hypothesis that has translated no source word and ends with a completed hypothesis that has translated all source words. The highest-scoring completed hypothesis, according to the model score, is returned as most probable translation, $\hat{e}_1^I$. Other incomplete hypotheses are referred to as partial hypotheses. As can be seen on Figure 1.3, an empty hypothesis is first created with no English word, no source word covered, and the probability is 1. By picking the translation option *"Mary"* and apply to this hypothesis, we extend to **Hyp. 1** (see on the figure). **Hyp. 1** covers the Spanish word *"Maria"*, contains one target word of the final translation (*"Mary"*) and has the score value of 0.534. The expansion process continues by picking translation options on that way, resulting in a huge search space for the decoder.

It is noteworthy that although each translation option translates a contiguous sequence

of source words, but successive translation options do not have to be adjacent on the source side (e.g. the **Hyp.1** and **Hyp.2**), depending on the distortion limit. However, the target output is strictly read off from left to right of the target string of successive translation options. Therefore, successive translation options which are not adjacent and monotonic in the source will then be reordered. In order to save the memory and to speed up the searching process, the target translation (MT output) is not stored together with each hypothesis. Instead, the hypothesis contains a reference to translation option and a back-pointer to the best previous hypothesis from which it extended. The target output for any hypothesis can be constructed by simply backtracking from the final hypothesis to the initial empty one and recursively reading off each target phrase from the translation option.

4.5.3 Recombining Hypothesis

Hypothesis Recombination is a risk-free way to lighten the search space as it becomes larger and larger with the increase of translation options. This task is conducted by considering the contextual information used by the language model. We need this step due to the fact that two partial hypotheses which have identical coverage sets and are limited by identical distortion constraints can be expanded by the same set of translation options, causing the redundancy. Commonly, two hypotheses can be recombined without the need of being completely matched, but just only in case they agree in:

- the source words covered so far
- the last **two** target words generated
- the end of the last source phrase covered

In Figure 1.4, both hypotheses **Hyp. 1** and **Hyp. 2** cover the first three spanish words (*"Maria no dio"*) and share the same two last target words generated (*"not give"*), so they can be recombined into one unique hypothesis (Hyp. 2). After the recombination, the weaker path (with lower score) is dropped, and its previous optimal hypothesis will now point at the stronger hypothesis (which is retained). This elimination will not affect the searching quality since the weaker one obviously cannot lead to highest-score complete path, and therefore cannot be part of the best translation.

4.5.4 Beam Searching

Once having the search space built, the next step is to search for the optimal hypothesis, with highest score. The first and most critical challenge we face is the huge size of the search space,

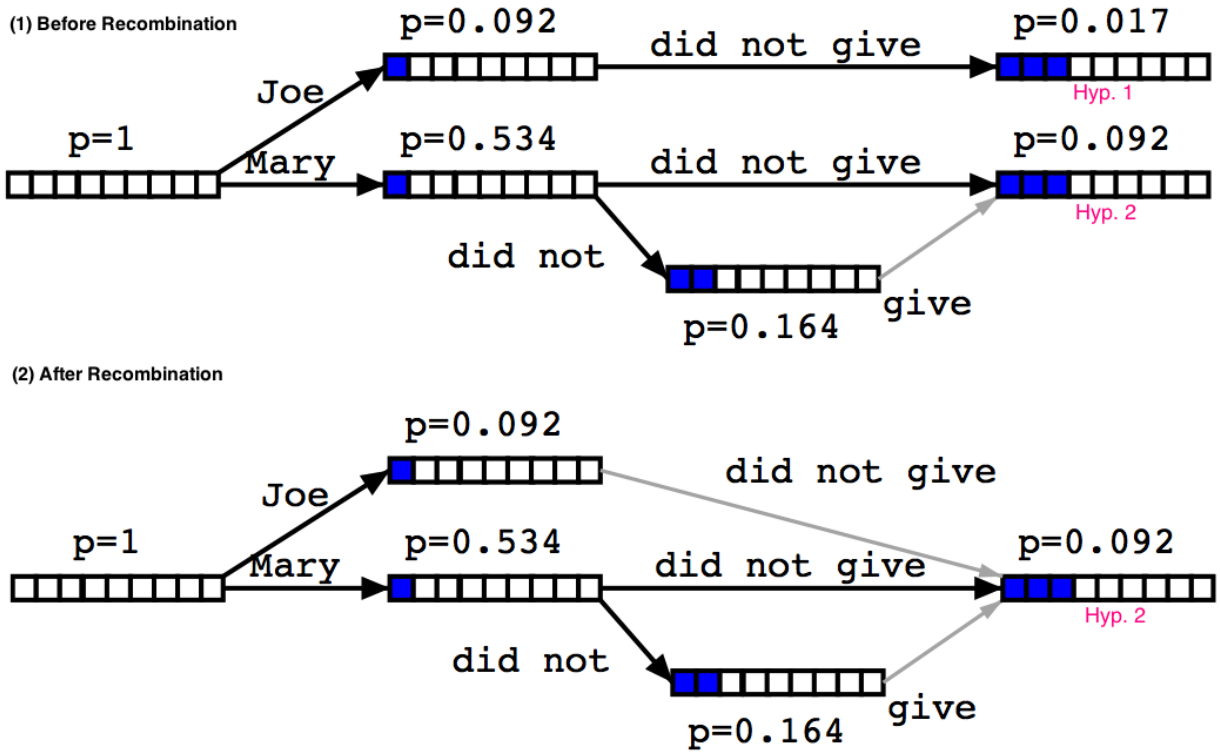

Figure 1.4 – An example of Hypothesis Recombination (source: Callison-Burch and Koehn (2005))

even after implementing the recombination. Specifically, Koehn estimates an upper bound for the number of states generated during a brute force search for translating the source sentence f_1^J as follows:

$$N \approx 2^J * |V_e|^2 * J \tag{1.21}$$

where J is the number of foreign words, and $|V_e|$ the size of the target language vocabulary. This exponentially exploded search space turns the problem of machine translation to be dramatically more challenging than, for instance, speech recognition. Coping with this issue, Koehn et al. (2003) propose the **"Beam Search"** algorithm which compares the hypotheses covering the analogous number of foreign words and discards the inferior hypotheses (which would never lead to the final output). So far, this algorithm is widely used in many SMT decoders and is proven to work fruitfully. Figure 1.5 describes the algorithm used for the beam search . For each number of the source words covered (1...J), a hypothesis stack is created. The initial empty hypothesis is stored in the stack for hypotheses without any source word covered. Starting with this hypothesis, new hypotheses are generated by extending from it using one translation option from the translation options set, and then are pushed to the appropriate stack relying on the number of source words covered so far. This process is identically applied for all other

```
initialize hypothesisStack[0 .. nf];
create initial hypothesis hyp_init;
add to stack hypothesisStack[0];
for i=0 to nf-1:
  for each hyp in hypothesisStack[i]:
    for each new_hyp that can be derived from hyp:
      nf[new_hyp] = number of foreign words covered by new_hyp;
      add new_hyp to hypothesisStack[nf[new_hyp]];
      prune hypothesisStack[nf[new_hyp]];
find best hypothesis best_hyp in hypothesisStack[nf];
output best path that leads to best_hyp;
```

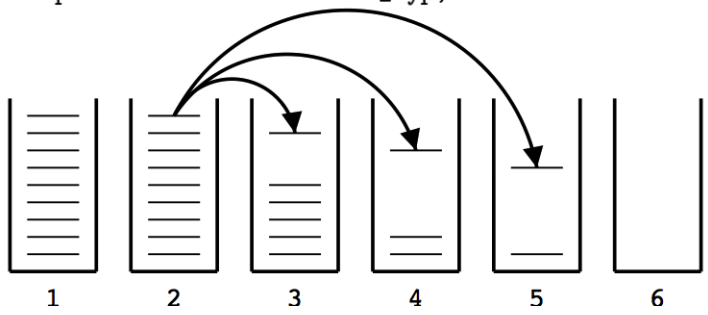

Figure 1.5 – Pseudo-code of Beam Search algorithm along with the stack organization for the search: hypotheses are store in the stacks according to the number of source words translated so far. When a new hypothesis is built from the existing one, it will be placed in a new stack (source: Callison-Burch and Koehn (2005))

stacks. In case where the amount of stack becomes too enormous, they can be pruned by using the threshold or histogram pruning (will be discussed in the next section). At the end, the final stack ($hypothesisStack[J]$) contains completed hypotheses, having all source words translated. The highest scoring hypothesis obtained from this stack is the result of decoding.

4.5.5 Pruning

As mentioned before, recombining hypotheses is a safe way to reduce the search space. Nevertheless, in reality, the total number of hypotheses after recombination is still extremely large, causing the beam search algorithm intractable. This challenge leads to another elimination, but this time, it is risky. Pruning is a task of discarding all hypotheses that *seem unlikely* to result in a good translation (but there is no certain guarantee that they are definitely useless). Two types of pruning are employed (Koehn, 2010) to eliminate low scoring hypotheses:

- **histogram pruning**: we keep top n hypotheses in each stack (e.g., n=100)
- **threshold pruning**: we keep all hypotheses whose scores are not less than α times of the score of the current best hypothesis in the stack (e.g., $\alpha = 0.001$).

In order to have a less risky elimination, we compare hypotheses by their *estimated score to completion*, rather than using just the current score. This score can be computed by adding the current score with the estimated score of translating the remaining untranslated source words:

$$h(e, f) \approx h'(e', f, C) + h'(C^*) \tag{1.22}$$

where $C^* = \{1, 2, ..., J\} - C$ is the set of uncovered positions; $h'(e', f, C)$ is the score of a hypothesis that has translated the source words in the coverage C as e'; and $h'(C^*)$ is the future estimated score. Besides, the optimal score for a sequence of source words can be quickly computed with dynamic programming. It is important to emphasize that in case where the source words not covered so far are multiple disconnected sequences (not contiguous), the combined score is simply the product of the score of each sequence.

5 Useful Toolkits and Resources for Statistical Machine Translation

Recently, the task of building a SMT system becomes more and more doable and convenient, thanks to a large number of free corpora and open-source toolkits supporting the different SMT components. For training the translation model, we need a huge bilingual parallel corpus containing source sentences and their translations. These resources can be automatically crawled from bilingual websites (e.g. news, government, or international organizations websites). Besides, several grand corpora, built by translating from the official documents of european parliament (Koehn, 2005) or Canadian parliament (Simard, 1998), etc. can be freely accessed. In this thesis, we will use the Europarl and News parallel corpora that are provided for WMT evaluation campaign in 2010 (total 1,638,440 sentence pairs) to train our French - English translation model. Besides, the language model training is even simpler with the availability of numerous monolingual corpora.

In 1990, the "GIZA" toolkit is developed to train the word alignment and implement the IBM models (from model 1 to model 3) (Al-onaizan et al., 1999). After that, its successor (and also the more advanced version), "GIZA++" is released with more new features (Och and Ney, 2003). This version helps to implement the IBM Model 4, as well as HMM. Also, the toolkit "ISI ReWrite Decoder" (Germann et al., 2001) is designed to deal with IBM Model 4. For the n-gram Language Model training, SRILM toolkit (Stolcke, 2002) is a prominent software.

As far as the SMT decoder is concerned, "Pharaoh" (Koehn, 2004) is the first tool which implements the phrase-based translation model, followed by "MOSES" (Koehn et al., 2007) -

one of the most widely-used SMT systems for research purpose. Both of them encapsulate a set of scripts for building a phrased-based SMT, involving: aligning words (GIZA++), extracting segments, training and estimating system and decoder' parameters, etc. Concerning MOSES, it takes the log-linear model, containing a set of sub-models. Each model provides one or multiple scores, constituting the 14 default parameters of the decoder. They are: 5 scores of translation models, 1 score of distortion model, 1 score of language model, 6 scores of lexical models and 1 score of word penalty model.

6 Machine Translation Evaluation

Generally, in natural language, we do not have only one way to translate from the source to the target sentence. Using different synonyms, morphological or idiomatic styles, or even changing word orders can all lead to the right translation. Therefore, evaluating a SMT output is definitely a nontrivial task and the evaluation metrics need both statistical and linguistic factors. That explains why the first approaches for this issues relate to the human subjective judgements. This approach is good since no one can evaluate MT outputs better than a translator with in-depth knowledge on both source and target languages. However, when dealing with a huge task load, this will become expensive (effort and time-consuming). Furthermore, the evaluation result will not homogenous when being conducted by different individuals (with different language backgrounds, attitudes and emotions, etc). These limitations motivate the automatic (objective) MT evaluation metrics, which assess the MT output by matching it with the (one or multiple) gold-standard reference(s). Both evaluation types will be introduced in this section.

6.1 Human Subjective Judgement

Several methods of evaluation using human subjective judgments are frequently applied in SMT community. In some cases, the quality of system hypothesis is measured directly, such as with human judgments; in other cases, it is measured by performing reading tests or other downstream tasks with the system output, and in still other cases it is measured by calculating the amount of work required to correct the system output (post-edition efforts).

Two common methods are **fluency** and **adequacy** judgments (Callison-Burch et al., 2007; White, 1994). **Fluency** requires a speaker fluent in the target language to judge whether the system output is fluent, regardless of whether content of the output conveys accurately or not what the source sentences want to say. Meanwhile, **Adequacy** disregards the level of fluency

in the system output and, as far as this is possible, measures whether the essential information in the source can be extracted from the system output. The requirements for an annotator of adequacy are stricter than for fluency, as the annotator must be bilingual in both the source and target language in order to judge whether the information is preserved across translation. In practice, an annotator who is fluent only in the target language could also annotate adequacy using a set of high quality human translations of the source sentence.

Apart from them, Post-editing (Parton et al., 2012; Specia and Farzindar, 2010), where the system output is corrected after it is produced, is another common method of measuring translation quality, with more accurate translation requiring less editing and poor translations requiring large amounts of editing. This method suffers as an evaluation metric due to the large amount of work required by human annotators to correct system output, rather than quickly objectively scoring it on an objective scale. **PET** (Post-Editing Tool) (Aziz et al., 2012) is an example of an open-source software that enables post-editors to post-edit and assess machine or human translations while gathering detailed statistics about post-editing time amongst other effort indicators. In another interesting toolkit: **SECTra_w.1** (Huynh et al., 2008), humans are enabled to post-edit the MT results using a web translation editor and then measure the post-editing time in an intuitive way. The post-edited sentences can be added to the set of reference translations, or become the first one in case where there were not any reference yet.

6.2 Automatic Measures

6.2.1 Word Error Rate (WER)

One of the first automatic metrics used to evaluate SMT systems was Word Error Rate (WER), which is the standard evaluation metric for Automatic Speech Recognition (ASR). WER is computed as the Levenshtein distance (Levenshtein, 1966) between the words of the MT hypothesis and the words of the reference translation divided by the length of the reference translation. The Levenshtein distance is computed using dynamic programming to find the optimal alignment between the MT output and the reference translation, with each word in the MT output aligning to either 1 or 0 words in the reference translation, and vice versa. Those cases where a reference word is aligned to nothing are labeled as deletions, whereas the alignment of a word from the MT output to nothing is an insertion. If a reference word matches (or is identical) the MT output word it is aligned to, this is marked as a match, and otherwise is a substitution. The WER is then the sums of the number of substitutions (S), insertions (I), and deletions (D)

divided by the number of words in the reference translation (N) as shown in Equation 1.23

$$WER = \frac{S + I + D}{N} \tag{1.23}$$

6.2.2 BLEU

BLEU (**B**ilingual **E**valuation **U**nderstudy) (Papineni et al., 2002) is the current standard for automatic machine translation evaluation. The key characteristic of BLEU is its direct exploitation of multiple references. The BLEU score of a system output is calculated by counting the number of n-grams (or word sequences), in the system output that occur in the set of reference translations. BLEU is a precision-oriented metric as it measures how much of the system output is correct, rather than measuring whether the references are fully reproduced in the system output. The score is finally defined as follows:

$$p_n = \frac{\sum_{C \in Candidates} \sum_{n-gram \in C} Count_{clip}(n - gram)}{\sum_{C \in Candidates} \sum_{n-gram \in C} Count(n - gram)} \tag{1.24}$$

$$BP = \begin{cases} 1 & \text{if } c > r \\ e^{1-\frac{r}{c}} & \text{if } otherwise \end{cases} \tag{1.25}$$

$$BLEU = BP * exp(\sum_{n=1}^{N} w_n log_{p_n}) \tag{1.26}$$

Equation 1.24 shows the calculation of the BLEU precision scores for $n-gram$ of length n, where $Candidates$ represents all sentences in the test corpus, $Count(n-gram)$ is the number of times that an n-gram occurs in a hypothesis, and $Count_{clip}(n-gram) = min(Count, Max\_ref\_count)$ is the minimum value between *the n-gram's count in the hypothesis* and *the maximum number of times it occurs in any reference translation* (we clip to ensure that n-gram's count will not exceed its highest count observed in any reference). Equation 1.25 computes the BLEU brevity penalty, where c and r are the lengths of the hypothesis and the reference translation, respectively. These terms are combined, as shown in Equation 1.26, to calculate the total BLEU score, where N is typically 4, and the weight w_n is usually set to $\frac{1}{N}$. In addition, BLEU score can also be represented in the log domain, by transforming Equation 1.26:

$$logBLEU = min(1 - \frac{r}{c}, 0) + \sum_{n=1}^{N} w_n log_{p_n} \tag{1.27}$$

Since its introduction, BLEU has become widespread in the machine translation community and is the most commonly reported evaluation metric. However, it contains still several shortcomings (Callison-burch and Osborne, 2006; Lavie et al., 2004; Turian et al., 2003), such as: the lack of recall in its formula, the compatibility only over large test corpora and not reliable for individual sentence, the absence of synonym matching, and the inability to detect multiple proper word orders, etc.

6.2.3 METEOR

METEOR (**M**etric for **E**valuation of **T**ranslation with **E**xplicit **OR**dering) (Banerjee and Lavie, 2005) is an evaluation specifically designed to address several observed drawbacks in BLEU metric. METEOR is a recall-oriented metric, whereas BLEU is generally precision-oriented. It firstly calculates both precision and recall[3], and combines the two, to form the first element called F_{mean} as shown in Equation 1.28 (with a large bias towards recall):

$$F_{mean} = \frac{P * R}{\alpha * P + (1 - \alpha) * R} \tag{1.28}$$

In addition to the F_{mean}, METEOR also uses a fragmentation penalty to bias the score against hypotheses that have many short sequences of consecutive matches (with the references), called chunks. Fragmentation is calculated as the number of chunks divided by the number of unigram matches. The fragmentation penalty is calculated as shown in Equation 1.29, with default parameters of $\beta = 3.0$ and $\gamma = 0.5$:

$$Pen = \gamma * frag^{\beta} \tag{1.29}$$

It is noteworthy that unlike BLEU and NIST, METEOR uses flexible criterion to match the unigrams between the hypothesis and the references. In the *"exact"* mode (as used in BLEU and NIST), two unigrams are matched if and only if they are exactly the same (e.g. *"houses"* maps to *"houses"* but not *"house"*). Besides, the *"stemming"* mode allows two unigrams derived from the same root form to be mapped (e.g. *"houses"* with *"house"*, *"gave"* with *"given"*). Furthermore, the *"synonymy"* mode maps two unigrams if they are the synonyms of each other (e.g. *"choose"* with *"select"*, *"truck"* with *"lorry"*). Finally, the METEOR score is calculated by Equation 1.30:

$$METEOR = (1 - Pen) * F_{mean} \tag{1.30}$$

[3]Detailed definitions of Precision and Recall can be referred in Chapter 2

Although having a dominant advantage of covering both precision and recall into its value, METEOR lacks one of the BLEU's key features: multiple reference exploitation, so that it is unable to combine knowledge from multiple reference to judge the MT output.

6.2.4 Translation Edit Rate (TER)

Translation **E**rror **R**ate (TER) (Snover et al., 2006) addresses the phrase reordering shortcoming of WER by allowing block movement of words, also called shifts, within the hypothesis as a low cost edit, a cost of 1, the same as the cost for inserting, deleting or substituting a word. The shifting constraints used by TER aim at reducing the computational complexity of the model and better modeling the quality of translation.

When TER is used in the case of multiple references, it does not combine the references, but instead, scores the hypothesis against each reference individually. The reference with which the hypothesis has the fewest number of edits is deemed the closest reference, and that number of edits is used as the numerator for calculating the TER score, as is done in Multi-reference WER (MWER). Rather than using the number of the words in the closest reference as the denominator, TER uses the average number of words across all of the references. Thus, the equation for the TER score, where SUB, INS, DEL and $SHIFT$ are the number of substitutions, insertions, deletions and shifts, respectively, and N_{avg} is the average number of reference words, is shown in the following equation.

$$TER = \frac{SUB + INS + DEL + SHIFT}{N_{avg}} \tag{1.31}$$

Nonetheless, TER cannot exploit multiple references as is done in BLEU nor does it incorporate external linguistic knowledge, e.g., synonyms, as is done in METEOR. These disadvantages can be overcome through the use of targeted references - references created by humans specifically for a particular machine translation output - changing it from an automatic metric (TER) to a semi-automatic metric: HTER.

6.3 Semi-automatic Measure: HTER

HTER (**H**uman-targeted **T**ranslation **E**rror **R**ate) (Snover et al., 2006) is a human-in-the-loop variant of TER that has also been used to evaluate SMT. HTER requires the use of *mono-lingual human annotators* who create references that are targeted to a particular system output. A targeted reference is crafted by modifying the MT hypothesis with a minimal number of edits so that this reference is fluent while still preserves the meaning of the other reference translations.

The use of targeted references has been shown to increase the correlation of automatic metrics with subjective human judgments, and can be seen as a method of addressing the sparsity of reference translations. However, since the targeted references cannot be reused and due to the requirement for human annotators to create targeted references, HTER is unsuited as a purely automatic machine translation evaluation metric.

7 Conclusions

This chapter presented the basic concepts of Machine Translation. It is a task where translation is automated partially or entirely by the aid of computer. MT has a long history of development from the early years of the last century, and is now still attracting researchers and companies worldwide. MT systems are divided into multiple types based on linguistic or computational architectures. Among MT approaches, statistical methods are more prominent than the others with numerous fruitful achievements in both research and commercial systems; thanks to the availability of bilingual training corpus and the language-independent working mechanism.

The chapter also mentioned some useful toolkits for building various components of a SMT system (GIZA++, SRILM, MOSES, etc.). From them, researches are able to build their own experimental systems rapidly. The MT evaluation metrics to access the MT systems performance were also presented.

Chapter 2

Theory of Word Confidence Estimation

1 Introduction

1.1 Confidence Estimation

Statistical Machine Translation (SMT) systems in recent years have marked impressive breakthroughs with numerous commendable achievements, as they produced more and more user-acceptable outputs. Nevertheless the users still face with some open questions: are these translations ready to be published as they are? Are they worth to be corrected or do they require retranslation? It is undoubtedly that building a method which is capable of pointing out the correct parts, detecting the translation errors, and concluding the overall quality of each MT hypothesis (output) is crucial to tackle these above issues. Such method is widely known under the name Confidence Estimation (CE) or Quality Estimation (QE). Technically, ***Confidence Estimation can be defined as a task of predicting the quality of the MT output for a given input***. Depending on the use cases, contexts, user types and applications, the concept *"MT output to be judged"* can vary from an entire document, a sentence, a segment or only a word. In terms of granularity, CE problem can be categorized into the following levels:

- *Document-level Confidence Estimation (DCE)*: provides the quality prediction for a new, unseen translated document. This type of CE is important in scenarios where the entire text needs to be published without post-edition.
- *Sentence-level Confidence Estimation (SCE)*: measures the entire sentence goodness.
- *Segment-level Confidence Estimation (Seg-CE)*: estimate the quality of a specific segment. The segment can be an n-gram, a phrase (noun phrase, verb phrase, etc) or the entire sentence (in this case, it becomes the second type).
- *Word-level Confidence Estimation (WCE)*: points out the confidence score (and quality label) for each word in the MT hypothesis.

This thesis mainly focuses on the latter estimate.

Formally, let t_i denote the i-th word, e_j denote the j-th segment in the target sentence t generated by an SMT system from a given source sentence s. The source and target sentences s and t belong to the source and target documents S and T. The job of CE systems is to calculate the probability of MT output to be correct, called confidence score:
Word level:

$$word\_score = P(Good|i, t_i, s) \tag{2.1}$$

Segment level:

$$segment\_score = P(Good|j, e_j, s) \tag{2.2}$$

Sentence level:

$$sentence\_score = P(Good|t, s) \tag{2.3}$$

Document level:

$$document\_score = P(Good|T, S) \tag{2.4}$$

What are the motivations behind the idea of building such automatic estimation methods? The answers are expected to originate from the actual shortcomings of assessing manually the translation quality. Firstly, ***this task is time and effort consuming***. Reading a long text for assessment is definitely a nontrivial work, as it takes time to understand, and even take more if it is badly expressed. Moreover, the readers would be disappointed after spending a lot of time scanning text and then realize that it cannot be used for anything and it is better to re-translate from scratch. Secondly, ***this task is sometimes impossible***, e.g. in case where the readers have little or no knowledge in the source language, so they cannot say anything about the adequacy of a given translation.

Due to the fact that an increasing number of MT systems appear and compete in the translation industry nowadays, many automatic evaluation metrics are proposed to evaluate their quality, such as BLEU (Papineni et al., 2002), TER (Snover et al., 2008), NIST (Doddington, 2002), METEOR (Lavie and Denkowski, 2009), etc. Nevertheless, ***unlike*** these above metrics, QE methods measure MT quality when the translation references are not available. In other words, QE gives quality predictions for MT outputs without having any idea about the expected quality (e.g. post-editions). One another discrepancy is that, QE methods principally aim at telling how good the translated text is for a given MT system in use when translating an unseen source text, rather than investigating the evolution of a specific MT system, or comparing the performance of multiple ones.

1.2 Mechanism and Components of a WCE System

In the general case, the CE's objective is to judge each MT output (e.g. word, segment, sentence, document) quality by tagging it with an appropriate quality label. The quality label is "Good" in case the output is perfect and does not need any editing operator. On the contrary, if the output contains errors, depending on the judgement's scope, the label can be just only "Bad" (need to be edited), or be further divided into more specific error types (e.g. accuracy, fluency, coherence, consistency, etc.).

In this thesis, we focus mostly on the binary variant (where quality label set encompasses "Good" and "Bad") of the WCE. Figure 2.1 demonstrates the operating mechanism of a binary classifier: given an MT output word, a classifier which has been trained beforehand calculates its confidence score, and then compares with a pre-defined threshold α. All words with scores that exceed this threshold are categorized in the *Good* label set; the rest belongs to the *Bad* label set.

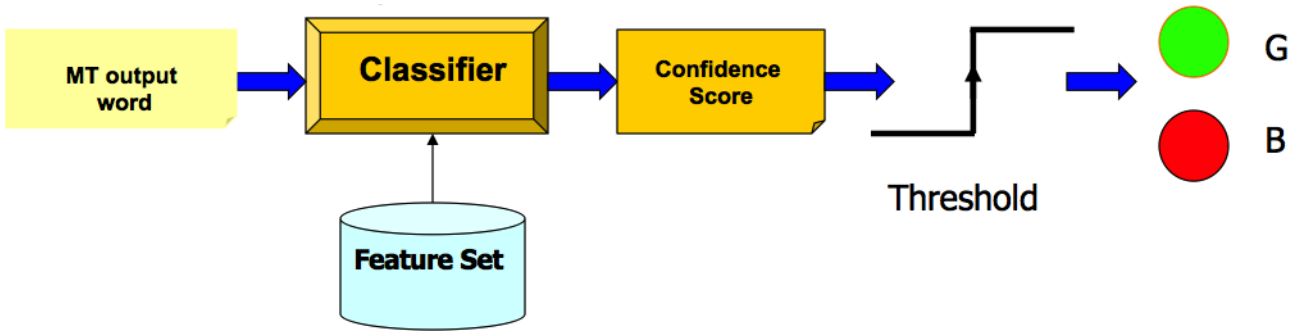

Figure 2.1 – The working mechanism of a binary classifier

Formally, the binary classifier is defined as:

$$label(t_i) = \begin{cases} Good & \text{if } p(t_i|s) \geq \alpha \\ Bad & \text{if } otherwise \end{cases} \tag{2.5}$$

In general, when **incorrect** words are further decomposed into multiple error types, the classifier computes the score for every class, and select the one with the highest score to tag the word. Regardless of binary or multi-class type, WCE system requires two crucial elements to operate, involving **Feature Set** and **Machine Learning (ML) method**.

Features (also commonly known as *Prediction Indicators*) are the indispensable factor in any WCE system as they constitute the knowledge base on which the classifier relies to judge words. Essentially, they are the characteristics (under the numerical or non-numerical form),

which capture every aspect of target word (e.g. lexical, semantic, syntactic, statistical etc.). Word posterior probability, number of senses, or number of occurrences in sentence, etc are examples of numerical features. Meanwhile, the Part-of-Speech tag, constituent label, aligned source word(s), etc are some representatives for the other type. Depending on the method employed, the non-numerical features might be required to discretize before using. Word features are collected from plenty of resources, related to or independent from SMT components. From SMT N-best list, alignment information, Language Model to outside syntactic and semantic parsers, all elements can be exploited to obtain useful indicators for word. Once having all features extracted, each word in the training and test data is represented by a vector of features, which will be used to train and test the classifier.

In the training phase, **ML algorithm** takes the responsibility of learning the probability distribution function for each possible label of word. We assume that the word t_i is represented by the feature vector $f_{t_i}^K = \{f_1, f_2, ..., f_K\}$. Then, the probability distribution function learned by ML algorithm is written as:

$$p(label(t_i); t_i) = p(label(t_i)|f_{t_i}^K) \tag{2.6}$$

where $label(t_i) \in \{Good, Bad\}$ in case of binary classifier. These distributions are then used to perform the classification. The different approaches in building these functions yield a variety of ML methods, such as: *Naive Bayes*, *Logistic Regression*, *Decision Tree*, *Conditional Random Fields*, etc. Among them, those which are employed in our experiments will be reviewed in Section 4.

1.3 WCE Applications

WCE shows an increasing importance in many aspects of MT. Firstly, it assists the post-editors to quickly identify the translation errors, determine whether to correct the sentence or retranslate it from scratch, hence improve their productivity (Nguyen et al., 2011). Secondly, it informs readers of portions of sentence that are not reliable (Specia et al., 2011). Such information is vital to avoid the misunderstanding about the sentence's content. Thirdly, it selects the best translation among options from multiple MT and/or translation memory (TM) (He et al., 2010). Therefore, by using WCE, we can recommend SMT output to a TM user when it is predicted more suitable for post-editing than the hits provided by TM. Next, WCE can also be used by the translators in an interactive scenario (Gandrabur and Foster, 2003). More specifically, the system facilitates translators after every character they type, by displaying all

possible words will come next, based on the source text and part of the word that has been typed so far. Last but not least, WCE scores can help to improve the MT quality via some scenarios: N-best list re-ranking, Search Graph Re-decoding, etc. In re-ranking (Duh and Kirchhoff, 2008; Nguyen et al., 2011; Zhang et al., 2006), more features are integrated with the existing multiple model scores for re-selecting the best candidate among N-best list. Meanwhile, the re-decoding process (Venugopal et al., 2007) intervenes directly into the decoder's search graph (SG) using WCE score, driving it to the optimal path (cheapest hypothesis), or the optimal hypothesis.

2 Measuring the Performance of a CE System

We present some common metrics which are widely used to measure the performance of Confidence Estimation system at word and sentence level.

2.1 Precision, Recall and F-score

Generally, the performance of WCE classifiers are measured by using common evaluation metrics: Precision (Pr), Recall (Rc) and F-score (F). Suppose that we would like to quantify these values for label Bad ("B"). Let X be the number of words whose true label is B and have been tagged with this label by the classifier, Y is the total number of words classified as B, and Z is the total number of words which true label is B. From these concepts, Pr, Rc and F can be defined as follows:

$$Pr = \frac{X}{Y}; Rc = \frac{X}{Z}; F = \frac{2 \times Pr \times Rc}{Pr + Rc} \tag{2.7}$$

These calculations can be applied in the same way for other labels. The higher the Precision is, the better our classification result will be. Precision of a specific label characterizes the ability of system to predict correctly (for it) over all classified words. Meanwhile, the Recall reflects how efficient the system is in retrieving the accurate labels from database. F-score is the harmonic mean of Precision and Recall.

In addition, for multiple-label classification, the system's overall F score can be computed as the average of all labels' F scores, weighted by their frequencies in the test data (Bojar et al., 2013).

2.2 Classification Error Rate

Another metric, called "Classification Error Rate" (CER) is also widely used in a number of researches (Blatz et al., 2004; Sanchis et al., 2007; Ueffing and Ney, 2005; Xiong et al., 2010).

It is the ratio of number of translation errors (i.e. words that are wrongly classified) over total number of classified words.

$$CER = \frac{\#wrongly\_classified\_words}{\#classified\_words} \tag{2.8}$$

2.3 Mean Absolute Error (MAE) and Root Mean Square Error (RMSE)

Two conventional metrics are used to measure the CE system's performance at sentence level include **"Mean Absolute Error"** (MAE) and **"Root Mean Square Error"** (RMSE)[1]. The MAE is a quantity used to measure how close predicted scores are to the reference scores. Expressed in words, it is the average over the verification sample of the absolute values of the differences between the predicted and the reference scores.

Meanwhile, the RMSE measures the average of the errors (difference between predicted and reference scores). Expressing the formula in words, the differences between reference and predicted score are each squared and then averaged over the sample. Finally, the square root of the average is taken. Since the errors are squared before they are averaged, the RMSE gives a relatively high weight to large errors. This means the RMSE is especially useful when large errors are particularly undesirable.

Formally, given a test set $S = s_1, s_2, ..., s_N$, let $R(s_i)$ and $H(s_i)$ $(i = 1, ..., N)$ be the reference score and hypothesis (predicted) score for sentence s_i, respectively. Then, MAE and RMSE can be formally defined by:

$$MAE = \frac{\sum_{i=1}^{N} |R(s_i) - H(s_i)|}{N} \tag{2.9}$$

$$RMSE = \sqrt{\frac{\sum_{i=1}^{N} (|R(s_i) - H(s_i)|)^2}{N}} \tag{2.10}$$

3 Previous Work

Confidence measure has been widely exploited in automatic speech recognition (e.g. in dialogue system or unsupervised training). In the years of 2003 and 2004, the researchers started to broaden it in MT domain (Blatz et al., 2003, 2004; Gandrabur and Foster, 2003; Ueffing et al., 2003) and carried out more and more in-depth related work until present. Coping with

[1] http://www.52nlp.com/mean-absolute-error-mae-and-mean-square-error-mse/

WCE, various approaches have been proposed, aiming at two major issues: features and Machine Learning (ML) methods to build the classifier. In this review, we refer mainly to two general types of features: internal and external features. "*Internal features*" (or "*system-based features*") are extracted from the components of MT system itself, generated before or during translation process (N-best lists, word graph, alignment table, language model, etc.). "*External features*" are constructed thanks to external linguistic knowledge sources and tools, such as Part-Of-Speech (POS) Tagger, syntactic parser, WordNet, stop word list, etc.

The authors in (Blatz et al., 2003) combined a considerable number of features by applying neural network and naive Bayes learning algorithms. In their study, both SCE and WCE were experimented. Among their features, Word Posterior Probability (henceforth WPP) proposed by Ueffing et al. (2003) was shown to be the most effective system-based feature. According to (Ueffing et al., 2003), WPP of a word can be determined by summing up the posterior probability of all sentences which contain it in a specific position, and then applying a normalization term on this summation. WPP can be computed on both word graph (constructed by SMT system) or on the N-best list (SMT's output). They based on these feature to suggest two alternative measures on N-best list, involving: "Relative frequency" and "Rank sum".

In the work of Blatz et al. (2004), the combination of WPP *"any"* (regarding all sentences containing the word at any position), WPP *"source"* (considering all sentences where the word occurs as the translation of the same source word), and IBM-Model 1 feature (Brown et al., 1993a) was also shown to outperform all the other single ones, including heuristic and semantic features. This top 3 helped to improve the baseline of 7% absolute in classification error rate (CER) (reduced from 36.2% to 29.2%). Another noticeable observation in this work is that there was almost no progression in system's performance when more and more features were added from top 3 (CER 29.2%) to all 17 features (CER 29.6%).

Using solely N-best list as resource, Sanchis et al. (2007) suggested 9 different features and then adopt a smoothed naive Bayes classification model to train the classifier. Testing results on bilingual English - Spanish technical manuals translation were consistent with those obtained by Blatz et al. (2004): IBM Model 1 based features are helpful in detecting wrongly translated words (CER 17.1%). Besides, features computed by WPP perform more efficiently (CER 18.4%) than those based on relative frequencies (CER 19.1%) or rank weights (CER 18.8%). Also, among the variants of WPP, the one computed over all N-best sentences that contain the target word at the ***"exact"*** position as it appears in the 1-best was shown to perform more fruitfully than the ***"Levenshtein position"*** or ***"any position"*** variants.

Another study (Ueffing and Ney, 2005) introduces a novel approach that explicitly explores the phrase-based translation model for detecting word errors. A phrase is simply considered as a

contiguous sequence of words and is extracted from the word-aligned bilingual training corpus. Its translation probability is computed as a log-linear interpolation of the relative frequency and the IBM-1 probability. The confidence value of each target word is then computed by summing over all the probabilities of the phrase pairs in which the target part contains this word. The authors tested on three different language pairs: French → English, Spanish → English and German → English, and compared the performances to other state-of-the-art methods, including word graph, N-best list, and IBM-1 based approaches. Results in all cases indicated that the proposed method yielded impressive relative reductions of the CER (up to 7.8%) compared to the best existing method on the same language pairs.

In Xiong et al. (2010), the classifier was built by integrating the target word itself, its POS, its WPP with another lexical feature named "Null Dependency Link" (or *"null link"*, to verify whether it has grammatical relations with the surrounding words). Moreover, in order to capture the contextual environment, in the word's features vector, they looked as well at the combinations with features of two words before and two words after the current one. After that, they trained these indicators by Maximum Entropy model (Berger et al., 1996) and optimized their weights on a separated development set. Interestingly, linguistic features (target word, POS and null link) sharply outperformed WPP feature in terms of F-score and CER by a relative improvement of 15.58% and 8.64%, respectively. Moreover, the syntactic feature "Null Dependency Link" showed its contribution in detecting translation errors when boosting the *"Recall"* score of other classifiers whenever it is used.

Unlike most of previous work, the authors in Soricut and Echihabi (2010) applied solely external features to rank the MT outputs from best to worse, with the hope that their ranker can deal with various MT approaches, from statistical-based to rule-based. They focused on predicting the quality at the document-level. The feature pool includes those relied on (target and source) text, LM, pseudo reference, examples on development dataset, and the corresponding MT system's training data. The training label used was the BLEU score. Then, given an MT output, its label (BLEU score) is predicted by their regression model. Another new point in this work is that the metric used to measure their solution's performance targeted directly at translation quality gains. In their large scale experiments conducted on 10 different language pairs in three different domains, the rankers achieved impressive improvement in volume-weighted BLEU gain (i.e. the average BLEU gain obtained when trading off volume for accuracy on a predefined scale, denoted by vBLEUΔ) varying among domains, from +6.4 (Travel) to +13.5 (HiTech). Also, results show consistent performances across various language pairs by a high averaged vBLEUΔ of +9.9 for five language pairs considered.

A method to compute the confidence score for both words and sentences relied on a feature-

rich classifier is proposed by Nguyen et al. (2011). The novel features employed include source side information, alignment context, and dependency structure. Among them, the two formers took into account the ***surface*** structures of both source and target sentences, whereas the latter utilized ***deep linguistic*** structures. Interestingly, the "dependency structure" feature investigates whether the target and source word have a *"Father-Child"* agreement (when they are aligned and so are their fathers) and/or *"Children"* agreement (their children are also aligned). In their approach, the sentence's goodness was synthesized from word scores. The training process was conducted on a 72000-sentence dataset using Conditional Random Fields (CRF) (Lafferty et al., 2001) model, followed by the feature selection and parameter tuning on 2707-sentence development one. On the unseen ($\approx$ 3000 sentences) test set, these first-time-proposed indicators overwhelmed WPP and POS for absolute F score by best gaps of 2.8 and 2.4 in both binary (Good, Bad) and 4-class (Good, Insertion, Substitution, Shift) systems. Moreover, their CE scores assisted MT system to re-rank the N-best list which improves considerably translation quality (measured by TER score).

In the submitted system to the WMT12 shared task on Quality Estimation, the authors in Langlois et al. (2012) added a total of 49 new features to the baseline provided by the organizers, including averaged, intra-lingual, basic parser and out-of-vocabulary features. They are then trained by SVM model, then filtered by forward-backward feature selection algorithm. This algorithm waives features which linearly correlated with others while keeping those relevant for prediction. It increases slightly the performance of all-feature system in terms of Root Mean Square Error (RMSE, 0.01 point gained). The optimization of the radial basis function parameters was not able to get more improvement compared to the system with default parameters and feature selection (0.77 RMSE). Finally, further analysis on kernel selection, parameter optimization suggest these techniques cannot yield significant effect on the classifier performance if the features employed are not strong enough.

WMT 2013 (Bojar et al., 2013) also witnessed several attempts dealing with WCE in its first launch as shared task. Aiming at an MT system-independent quality assessment, "referential translation machines" (RTM) method proposed in Bicici (2013) shows its encouraging prediction performance, without accessing any SMT system specific resource and prior knowledge used to train data or model. RTM takes into account the acts of translation when translating between two data sets with respect to a reference corpus in the same domain. In addition, an RTM model relied on the selection of the common training data which is relevant and close to both the training set and the test one. Bicici (2013) also extended the global learning model by dynamic training with adaptive weight updates in the perceptron training algorithm. Meanwhile, Han et al. (2013) employed the CRF model as their Machine Learning method to

address the problem as a sequence labeling task. As far as prediction indicators are concerned, Bicici (2013) proposed seven word feature types and found among them the "CCL links" the most outstanding. Han et al. (2013) focused only on various n-gram combinations for target words. Interestingly, this was also the first time we participated in this share task. We (Luong et al., 2013b) integrated a number of new prediction indicators to build the classifier, as well as a number of optimization attempts to enhance the baseline (classification threshold tuning, feature selection and boosting technique). They will be consequently mentioned in the next chapters of this thesis.

4 Features (Prediction Indicators)

In this section, we review and depict in details a number of state-of-the-art features which have been widely used by previous work, and will be inherited by us to train our WCE classifiers. Our proposed features will be described in Chapter 3. Generally, the prediction indicators represent various statistical characteristics or linguistic functions of words and can be extracted from numerous SMT dependent or independent resources. We categorize them into one of the following principal classes:

4.1 System-based Features

They are the features extracted directly from the SMT system, without the participation of any additional external component. Based on the resources where features are found, they can be sub-categorized as following:

4.1.1 Target Side Features

The information of every target word (at position i in the MT output) that can be taken into account includes:

- The word itself (Xiong et al., 2010)
- The sequences formed between it and a word before $(i-1/i)$ or after it $(i/i+1)$
- The trigram sequences formed by it and two previous and two following words (including: $i-2/i-1/i$; $i-1/i/i+1$; $i/i+1/i+2$)
- The number of occurrences in the sentence

For example, with the target sentence: *"This decision, crucial to the future of an unstable region, will test the resolve and western unity."*, these above features for the word *"future"* are:

- The word itself: *"future"*
- The bigram sequences: *"the/future"* and *"future/of"*
- The trigram sequences: *"to/the/future"*, *"the/future/of"* and *"future/of/an"*
- The number of occurrences: 1

4.1.2 Source Side Features

Using the alignment information, we can track the source words which the target word is aligned to. To facilitate the alignment representation,the BIO[2] format can be applied: in case of multiple target words are aligned with one source word, the first word's alignment information will be prefixed with symbol "B-" (means "Begin"); and "I-" (means "Inside") will be added at the beginning of alignment information for each of the remaining ones. The target words which are not aligned with any source word will be represented as "O" (means "Outside"). Table 2.1

Target words (MT output)	**Source aligned words**	**Target words (MT output)**	**Source aligned words**
The	B-le	to	B-de
public	B-public	look	B-tourner
will	B-aura	again	B-à|nouveau
soon	B-bientôt	at	B-son
have	I-aura	its	I-son
the	B-l'	attention	B-attention
opportunity	B-occasion	.	B-.

Table 2.1 – Example of using BIO format to represent the alignment information

shows an example for this representation, in case of the english hypothesis is *"The public will soon have the opportunity to look again at its attention."*, given its french source: *"Le public aura bientôt l'occasion de tourner à nouveau son attention."*. Since two target words *"will"* and *"have"* are aligned to *"aura"* in the source sentence, the alignment information for them will be "B-aura" and "I-aura" respectively. In case a target word has multiple aligned source words (such as *"again"*), we separate these words by the symbol "|" after putting the prefix "B-" at the beginning.

4.1.3 Alignment Context Features

These features are proposed by Nguyen et al. (2011) in regard with the intuition that collocation is a believable indicator for judging if a target word is generated by a particular source word.

[2]http://www-tsujii.is.s.u-tokyo.ac.jp/GENIA/tagger/

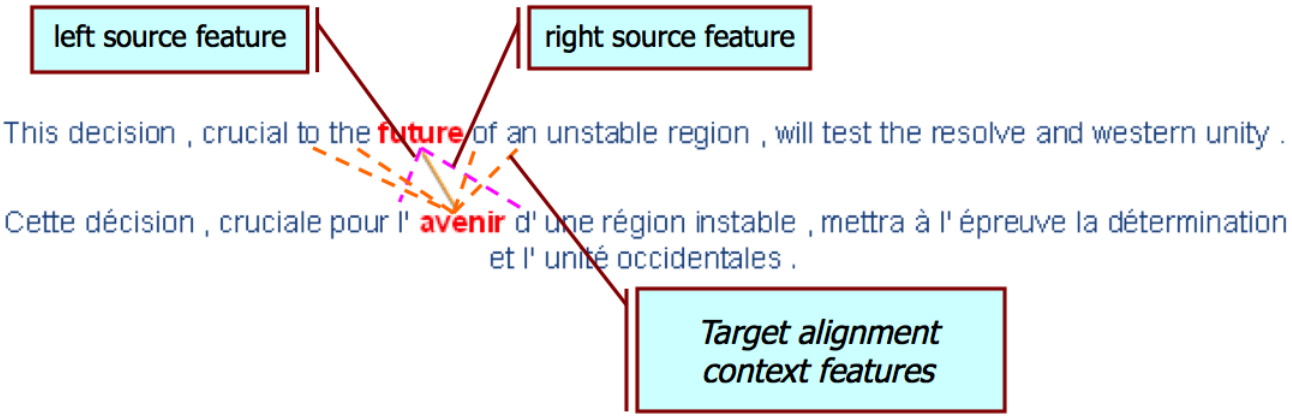

Figure 2.2 – Example of alignment context features

Among various representatives for this type, the most widely-employed surrounding alignment context information involves:

- Source alignment context features: the combinations of the target word and one word before (left source context) or after (right source context) the source word aligned to it.
- Target alignment context features: the combinations of the source word and each word in the window ±2 (two before, two after) of the target word.

Figure 2.2 shows an example of these features. For instance, in case of the target (english) word *"future"*, the aligned source (french) word is *"avenir"*, and the source alignment context features are: *"future/l' "* and *"future/d' "*. We also get its target alignment context features, encompassing: *"avenir/to"*, *"avenir/the"*, *"avenir/of"*, and *"avenir/an"*.

4.1.4 Word Posterior Probability

Word posterior probability (WPP) (Ueffing et al., 2003) is the likelihood of the word occurring in the target sentence, given the source sentence. Numerous knowledge sources have been proposed to calculate it, such as word graphs, N-best lists, etc. To quantify it, the key point is to determine sentences in N-best lists that contain the word e under consideration in a fixed position i . Let $p(f_1^J, e_1^I)$ be the joint probability of source sentence f_1^J and target sentence e_1^I. The WPP of e occurring in position i is computed by aggregating probabilities of all sentences containing e in this position:

$$p_i(e|f_1^J) = \frac{p_i(e, f_1^J)}{\sum_{e'} p_i(e', f_1^J)} \tag{2.11}$$

where

$$p_i(e, f_1^J) = \sum_{I, e_1^I} \Theta(e_i, e) \cdot p(f_1^J, e_1^I) \tag{2.12}$$

Here $\Theta(.,.)$ is the Kronecker function. The normalization in Equation 2.11 is:

$$\sum_{e'} p_i(e', f_1^J) = \sum_{I, e_1^I} p(f_1^J, e_1^I) = p(f_1^J) \tag{2.13}$$

In this thesis, we exploit the graph that represents MT hypotheses (Ueffing et al., 2002). From this, the WPP of word e in position i (denoted by WPP *exact*) can be calculated by summing up the probabilities of all paths containing an edge annotated with e in position i of the target sentence. Another form is "WPP *any*" in case we ignore the position i, or in other words, we sum up the probabilities of all paths containing an edge annotated with e in any position of the target sentence. Here, both forms are used and the above summation is performed by applying the forward-backward algorithm (Ueffing and Ney, 2007).

4.1.5 Language Model Backoff Feature

Applying SRILM toolkit (Stolcke, 2002) with the bilingual corpus, we build 4-gram language models for the target language. Then, we employ a feature named the *backoff behavior*, proposed in Raybaud et al. (2011) of the backward target language model which investigates deeply the role of two previous words by considering various cases of their occurrences. In other word, we consider the LM's backoff behavior to find the target word's backward sequences. Backoff score is assigned to each word w_i, as following:

$$B(w_i) = \begin{cases} 7 & \text{if } w_{i-2}, w_{i-1}, w_i \text{ exists} \\ 6 & \text{if } w_{i-2}, w_{i-1} \text{ and } w_{i-1}, w_i \text{ both exist} \\ 5 & \text{if only } w_{i-1}, w_i \text{ exists} \\ 4 & \text{if } w_{i-2}, w_{i-1} \text{ and } w_i \text{ exist separately} \\ 3 & \text{if } w_{i-1} \text{ and } w_i \text{ both exist} \\ 2 & \text{if only } w_i \text{ exists} \\ 1 & \text{if } w_i \text{ is out of vocabulary} \end{cases} \tag{2.14}$$

(The concept "exist" here means "appear in the language model").

4.2 Lexical Features

A prominent lexical feature that has been widely explored in WCE researches is word's Part-Of-Speech (POS) (Blatz et al., 2004; Luong, 2012; Nguyen et al., 2011; Xiong et al., 2010). This tag is assigned to each word due to its syntactic and morphological behaviors to indicate its

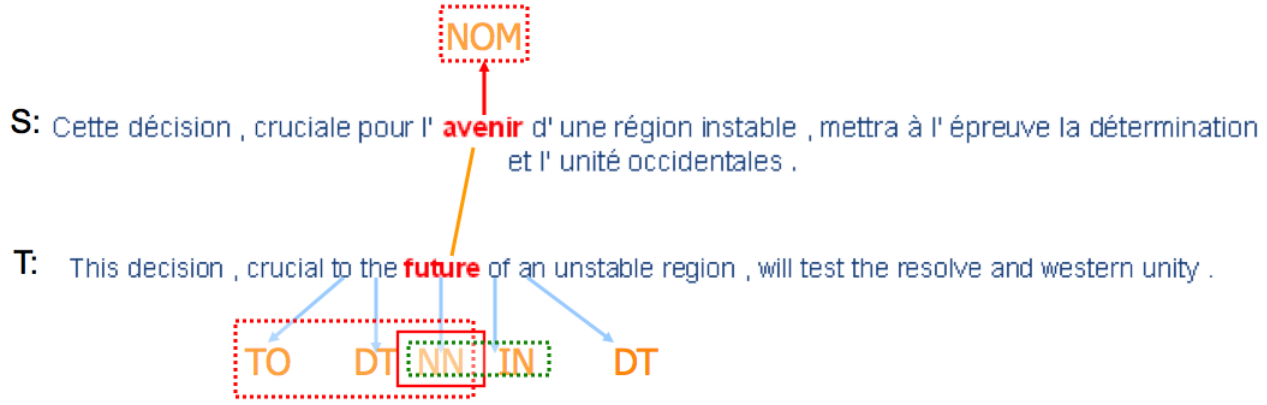

Figure 2.3 – Example of lexical features

lexical category. We use TreeTagger[3] toolkit for POS annotation task and obtain the following features for each target word:

- Its POS
- Sequence of POS of all source words aligned to it (in BIO format)
- Bigram and trigram sequences between its POS and the POS of previous and following words. Bigram sequences are POS_{i-1}, POS_i and POS_i, POS_{i+1} and trigram sequences are: $POS_{i-2}, POS_{i-1}, POS_i$; $POS_{i-1}, POS_i, POS_{i+1}$ and $POS_i, POS_{i+1}, POS_{i+2}$

In addition, we also build four other binary features that indicate whether the word is a:

- *stop word* (based on the stop word list for target language): commonly used words in the language (e.g. *the*, *an*) which bring few or almost no content information for the sentence.
- *punctuation* symbol: spacing, conventional signs, or certain typographical devices aiding to the understanding and correct reading, both silently and aloud, of handwritten and printed texts (e.g. apostrophe, brackets, colon, comma, dash, hyphen, etc.).
- *proper name*: words that refer to an unique entity, such as: *Paris*, *Henry*, *Jupiter*, *Microsoft*, etc.
- *numerical*: the decimal number.

In Figure 2.3, with the target word *"future"*: the POS is *"NN"*. The POS of aligned source word(s) is *"NOM"* (french noun). Bigram POS sequences are: *"DT/NN"*, *"NN/IN"*. Trigram POS sequences are: *"TO/DT/NN"*, *"DT/NN/IN"* and *"NN/IN/DT"*. This word is neither a stop word nor punctuation, proper name and number.

[3]http://www.ims.uni-stuttgart.de/projekte/corplex/TreeTagger/

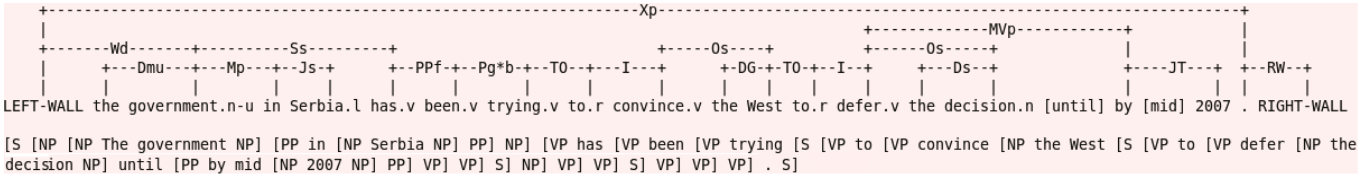

Figure 2.4 – Example of parsing result generated by Link Grammar

4.3 Syntactic Features

Besides lexical features, the syntactic information about a word is also a potential hint for predicting its correctness. If a word has grammatical relations with the others, it will be more likely to be correct than those which have no relation. In order to obtain the links between words, the Link Grammar Parser[4] is an effective solution, affording us to build a syntactic structure for each sentence in which each pair of grammar-related words is connected by a labeled link. In case of Link Grammar fails to find the full linkage for the whole sentence, it will skip each word one time until the sub-linkage for the remaining words has been successfully built. Based on this structure, we get the *"Null Link"* (Xiong et al., 2010) characteristic of the word. This feature is binary: 0 in case of word has at least one link with the others, and 1 if otherwise. Figure 2.4 represents the syntactic structure of an MT output: *"The government in Serbia has been trying to convince the West to defer the decision until by mid 2007."*. It is intuitive to observe in this structure that the words in brackets (including "*until*" and "*mid*") have no link with the others, meanwhile the remaining ones have. For instance, the word "*trying*" is connected with "*to*" by the link "TO" and with "*been*" by the link "Pg*b". Hence, the value of "Null Link" feature for "*mid*" is 1 and for "*trying*" is 0.

4.4 Semantic Features

The word semantic characteristic that we study is its polysemy count (Blatz et al., 2003, 2004). We hope that the *number of senses* of each target word given its POS (or that information of aligned source word) can be a reliable indicator for judging if it is the translation of a particular source word. For **English** as the target language, the features *"Polysemy count"* is built by applying a Perl extension named Lingua::WordNet[5], which provides functions for manipulating the WordNet [6] database. In case where the target language is **Spanish**, we employ BabelNet[7] - a multilingual semantic network that works similarly to WordNet but covers more European

[4]http://www.link.cs.cmu.edu/link/
[5]http://search.cpan.org/dist/Lingua-Wordnet/Wordnet.pm
[6]http://wordnet.princeton.edu/
[7]http://babelnet.org

languages, including this language.

5 Machine Learning Techniques

As stated in Section 1.2, a specific ML algorithm will be needed to learn the probability distribution functions for each label in the training set. These functions play the role of a knowledge base to perform classification on the test set. This section reviews in details several ML techniques which will build our classifiers. For the sake of consistency, throughout this section, the following notations are used:

- $X = (x_1, x_2, \ldots, x_N)$: the data sequence (in our case, they are the words of a training sentence)
- $Y = (y_1, y_2, \ldots, y_N)$: the output sequence obtained after the labeling task for X. In binary classification, y_i $(i = 1, 2, ..., N)$ can take the value in $\{Good, Bad\}$ (or $\{1, 0\}$).
- c: the class variable. In binary classification, $c \in \{0, 1\}$ (or $\{Good, Bad\}$).
- $f = \{f_1, f_2, ..., f_K\}$: the feature vector for each word (we assume to have K features, therefore the word is characterized by a K-dimensional feature vector).

5.1 Naive Bayes

Naive Bayes model (Lowd, 2005) has been broadly used for clustering and classification. It is proven to be a very attractive alternative to Bayesian networks for general probability estimation, especially in large or real-time domain. In this model, the class posteriors of the training instance x_i (represented by the vector f) can be calculated via the **"Bayes rule"** as follows:

$$P(c|f) = \frac{P(c)P(f|c)}{\sum_{c'} P(c')P(f|c')} \tag{2.15}$$

The key hypothesis of Naive Bayes model (and also the main reason why it is so named) is that the features are statistically independent, given a class variable c, so that the conditional distribution $P(f|c)$ can be written as:

$$P(f|c) = \prod_{k=1}^{K} P(f_k|c) \tag{2.16}$$

Given a sample of N training instances, the maximum likelihood estimate of the unknown probabilities are the conventional frequencies:

$$P(c) = \frac{N(c)}{N} \tag{2.17}$$

$$P(f_k|c) = \frac{N(f_k, c)}{N(c)}, k = 1, ..., K \tag{2.18}$$

where the $N(.)$ notions represent the event counts. Specifically: $N(c)$ is the number of training instances in class c, and $N(f_k, c)$ is the number of instances with feature value f_k in class c. The maximum likelihood estimates assign the zero value (0) to the conditional probability of all features that can not be found in the training set for class c. However, this behavior is harmful in case where a new instance that has this feature will automatically be given a probability $P(f|c) = 0$, neglecting the other features it may have.
In order to avoid the null estimates, we take into account an absolute discounting smoothing model borrowed from the statistical language modeling. The idea is to discount a small constant $b \in (0, 1)$ to every positive count and then distribute the gained probability mass (with a uniform back-off) among the null counts (unseen events). Therefore, when considering a class c, if $N(f_k, c) = 0$ for one or more possible values of f_k, we denote by N_+ the number of features f_k with $N(f_k, c) > 0$ and N_0 the number of features f_k with $N(f_k, c) = 0$. This model changes the probability estimate in Equation 2.18 into:

$$P(f_k|c) = \begin{cases} \frac{N(f_k,c)-b}{N(c)} & if\ N(f_k, c) > 0 \\ \frac{b}{N(c)} * \frac{N_+}{N_0} & if\ N(f_k, c) = 0 \end{cases} \tag{2.19}$$

Once the model has been trained by this manner ($P(f_k|c)$ takes the non-zero value), the conditional probabilities of correctness are computed using Equation 2.15.
The above calculation runs smoothly over all features whose value set is limited and discrete. Nevertheless, in reality, some features may have continuous rather than discrete values. In that case, we can discretize its value range into a fixed number (normally around 20 to 30) of intervals . The discretization is conducted in a semi-automatic way by assigning a lower bound and upper bound for each feature. The value range is then split in a number of equal intervals of fixed size. The lower bound, upper bound and number of intervals can be set based on the intuitive observation of the feature's histograms of the instances from the correct and incorrect classes. Given this information, our naive Bayes implementation encompasses a func-

tion to maps the continuous feature value f_k to the corresponding discrete interval number (or index). Then, the probability estimation procedure is used identically as depicted above on the discretized features.

5.2 Logistic Regression

Logistic Regression (LR) (Friedman et al., 2000) is an ML approach, which assumes a parametric form for the distribution $P(Y|X)$, then estimates directly its parameters from the training data. In case where Y takes binary values (i.g. "Good" or "Bad"), the assumed parametric model is:

$$P(Y = Good|X) = \frac{1}{1 + exp(w_0 + \sum_{i=1}^{n} w_i x_i)} \tag{2.20}$$

and

$$P(Y = Bad|X) = 1 - P(Y = Good|X) = \frac{exp(w_0 + \sum_{i=1}^{n} w_i x_i)}{1 + exp(w_0 + \sum_{i=1}^{n} w_i x_i)} \tag{2.21}$$

In general, where $Y = (y_1, y_2, \ldots, y_N)$, then the form of $P(Y = y_k|X)$ for $Y = y_1, Y = y_2, ..., Y = y_{N-1}$ can be written as:

$$P(Y = y_k|X) = \frac{exp(w_{k0} + \sum_{i=1}^{n} w_{ki} x_i)}{1 + \sum_{j=1}^{N-1} exp(w_{j0} + \sum_{i=1}^{n} w_{ji} x_i)} \tag{2.22}$$

When $Y = y_N$, it becomes:

$$P(Y = y_N|X) = \frac{1}{1 + \sum_{j=1}^{N-1} exp(w_{j0} + \sum_{i=1}^{n} w_{ji} x_i)} \tag{2.23}$$

Where w_{ji} denotes the weight associated with the j-th class $Y = y_j$ and with the input x_i. The two earlier expressions are actually a special case of the two latter ones. The major difference between the case where Y takes boolean value and that of multi-values is that: when Y takes N values, we build $N-1$ different linear expressions to capture the distributions for $N-1$ first values. The final distribution is computed based on the first $N-1$ ones, on regarding that their sum is equal to 1.

In a nutshell, LR is a function approximation algorithm that uses training data to directly estimate $P(Y|X)$, in contrast to NB, which estimate parameters for $P(Y)$ and $P(X|Y)$. In this sense, LR is often called as disciminative classifier since we are able to see the distribution $P(Y|X)$ as directly discriminating the value of the target value Y for any given instance X.

5.3 Decision Tree

Decision Tree learning (Quinlan, 1986) is one of the most well-known and widely used ML methods for inductive inference. It uses the Decision Tree (DT) to represent the learned functions during training and then approximates the discrete-valued target class labels. For the sake of human readability, the learned trees are converted into the if-then-else rules. This method is not only simple, but also robust to noisy or incomplete data, capable of learning disjunctive expressions, handling both numerical and categorical data, and training rapidly from large amount of data.

In a DT, each node represents an attribute of training instances, and each branch descending from it corresponds to one of its possible values. The classification process for each test instance can be summarized as follows: we start from the root node, and move downward to the branch whose value fits its attribute value. The searching continues until it reaches the leaf node (most-bottom node). Finally, the value of this node is used to tag the test instance.

At each node of the tree, we are able to determine the optimal attribute to be tested among the entire set by using a statistical property named *Information Gain* (IG). Essentially, this measure computes the expected reduction in entropy caused by partitioning the instances using this attribute. Formally, the IG of an attribute a_i with respect to a collection of N training instances can be written as:

$$\mathbf{IG}(N, x_i) = Entropy(N) - \sum_{v \in Values(x_i)} \frac{N_v}{N} Entropy(N_v) \tag{2.24}$$

Where $Values(x_i)$ represents the set of possible values for x_i and N_v denotes the subset of N whose x_i's value is v. Beside many advantages, DT learning faces several practical issues, including: overfitting the training data, managing unexpected errors in pruning, etc. Hence, various pre- and post- pruning, as well as attribute handling methods are proposed to tackle them.

5.4 Conditional Random Fields

Conditional Random Fields (CRF) (Lafferty et al., 2001) is a framework for building probabilistic models for segmenting and labeling sequence data. The core idea of CRF can be summarized as follows: let $X = (x_1, x_2, \ldots, x_N)$ be the random variable over data sequence to be labeled, $Y = (y_1, y_2, \ldots, y_N)$ be the output sequence obtained after the labeling task. In our case, X ranges over words in the MT output, and Y represents the labels tagged for words. Each element y_i $(i = \overline{1..N})$ is assigned one value in the binary set $Y^N = \{Good, Bad\}$. Then,

the probability of sequence Y given X is:

$$P_\theta(Y|X) = \frac{1}{Z_\theta(X)} \exp\{\sum_{k=1}^{K} \theta_k F_k(X,Y)\} \tag{2.25}$$

where

$$F_k(X,Y) = \sum_{t=1}^{N} f_k(y_{t-1}, y_t, x_t) \tag{2.26}$$

$\{f_k\}(k = \overline{1..K})$ is a set of feature functions, $\{\theta_k\}(k = \overline{1..K})$ are the associated parameter values, and $Z_\theta(x)$ is a normalization factor, in which the value is calculated by:

$$Z_\theta(X) = \sum_{y \in Y^N} \exp \sum_{k=1}^{K} \theta_k F_k(X,Y) \tag{2.27}$$

In order to estimate the conditional maximum likelihood given T independent sequences $\{X^i, Y^i\}, i = \overline{1..T}$ where X^i and Y^i contains N^i symbols, we have to minimize the negated conditional log-likelihood of the observations, with respect to θ:

$$l(\theta) = -\sum_{i=1}^{T} log p_\theta(Y_i|X_i) = \sum_{i=1}^{T} \{log Z_\theta(X_i) - \sum_{k=1}^{K} \theta_k F_k(X^i, Y^i)\} \tag{2.28}$$

The standard solution for this minimization is to apply an additional l^2 penalty term, determined by $\frac{\rho_2}{2}||\theta||_2^2$, where ρ_2 is a regularization parameter. The objective function is then a smooth convex function to be minimized over an unconstrained parameter space. Beside l^2, l^1 penalty calculated by $\rho_1||\theta||_1$ can also be exploited to perform the feature selection. It plays the role of controlling the amount of regularization as well as the number of extracted features. Their combination helps to decrease the number of nonzero coefficients and to avoid the numerical problems which can appear in a huge dimensional parameter environment. The objective function corresponding to this combination is $l(\theta) + \rho_1||\theta||_1 + \frac{\rho_2}{2}||\theta||_2^2$.

Several optimization and regularization methods have been proposed to alleviate the parameter estimation issue: quasi-Newton (L-BFGS and OWL-QN), resilient propagation (R-PROP), stochastic gradient descent (SGD-L1), block-wise coordinate descent (BCD) (Lavergne et al., 2010). Among them, we apply the BCD method to optimize our feature weights.

6 WCE at WMT Campaigns

The Workshop on Statistical Machine Translation (WMT) was first organized in 2006 and featured a number of SMT related shared tasks: Translation, Evaluation, Quality Estima-

tion (QE), System Combination, and several in-depth domain translations (featured, medical). Among them, QE shared task was first launched in 2012, with sentence-level prediction (officially called Task 1), and then was extended with word-level variant (Task 2) one year later. We participated in Task 2 from its first edition to present (2013, 2014).

In Task 2, the WMT organizers provide a data set consisting of the source text, its translation into a target language. The participants are required to assign each translated word an appropriate quality label. The label can belong to one of the following settings:

- WMT 2013 (Bojar et al., 2013)
  - Binary variant: *"Keep"* (K- no translation error) or *"Change"* (C- need for editing)
  - Multi-class variant: *"Keep"* (K- no translation error), or *"Delete"* (need to be removed from sentence), or *"Substitute"* (need to be replaced by a better word).
- WMT 2014 (Bojar et al., 2014)
  - Binary variant: *"OK"* (no translation error) or *"BAD"* (need for editing).
  - Level 1 variant: *"OK"* (no translation error), or *Accuracy* issue (the word does not accurately reflect the source text), or *Fluency* issue (the word does not relate to the form or content of the target text).
  - Multi-class variant: Beside three labels of Level 1 variant, translation errors are further decomposed into 15 labels based on MQM metric: *Terminology, Mistranslation, Omission, Addition, Untranslated, Style/register, Capitalization, Spelling, Punctuation, Typography, Morphology (word form), Part-of-speech, Agreement, Word order, Function words, Tense/aspect/mood, Grammar* and *Unintelligible*.

In order to support the participants training their classifiers, the training data including source text and target translations, along with labels annotated for target tokens are provided.

- In WMT 2013, data set for English → Spanish language pair is used. The target sentences are SMT outputs. The annotated labels are derived automatically by computing TER between the MT outputs and their post-editions. In addition, participants are encouraged to apply a variety of additional resources for extracting their features: source and target language models for words and POS tags, SMT N(1000)-best list, parallel data to build the SMT system, and the code to re-run the entire Moses (Koehn et al., 2007) system (in case where they need other system-dependent information).

- In WMT 2014, more language pair options are proposed: English $\longleftrightarrow$ Spanish, English $\longleftrightarrow$ German (we participate in only English $\rightarrow$ Spanish one). However, the principal discrepancy from WMT 2013 is a novel setting towards a SMT-independent evaluation. More specifically, target sentences s are collected from multiple translation means (machine and human), therefore all SMT specific settings (and the associated features that could have been extracted from it) become unavailable. The annotations are manually obtained by professional translators. This initiative puts more challenges on participants, yet motivates number of SMT-unconventional approaches and inspires the endeavors aiming at an "Evaluation For All".

The submissions of all teams are evaluated and ranked in term of their classification performance (Precision, Recall, F-score). Besides, in WMT 2014, in order to appreciate the translation error detection capability of systems, the main evaluation metric is the average F score for all but the *"OK"* class. However, *in this thesis, we will report not only this above score, but also the average F score for all classes, since we believe that the latter one can fully judge systems.* For the non-binary variant, the average is weighted by the frequency of the class in the test data. All our preparations and official results in both WMT 2013 and WMT 2014 are reported in Chapter 3.

7 Summary

This chapter introduced the basic background of Word Confidence Estimation (WCE) problem. WCE systems take a job of identifying the correctly translated words and detecting erroneous ones. They are trained over a set of training instances (words) in which each one is represented by a feature vector along with an annotated label. Several ML algorithms can be employed to learn the probability distribution functions from these instances, which are then used for performing the classification. WCE shows an increasing importance in many MT sectors: assisting post-editors in identifying MT errors, informing readers of unreliable portions of translation output, facilitating translators in interactive scenarios, enhancing MT quality (via re-ranking N-best list or re-decoding the search graph).

Next, we reviewed various WCE approaches. They aim at proposing, integrating and filtering prediction indicators (features), experimenting with different conventional and/or proposed ML methods, and optimizing models, parameters to enhance the prediction performance. The chapter then described a set of state-of-the-art features which are broadly employed by previous work, followed by a number of well-known ML techniques. Evaluation metrics for WCE were also presented.

Finally, we discussed about WMT, an annual workshop which proposes many interesting shared tasks in the field of SMT. We focused on WCE tasks (first launched in 2013) with the data provided, missions required and assessment methods.

After presenting the basic background of SMT, CE and WCE, the remaining part of this thesis will report in detail all our contributions for the domain. Our work differs from other previous researches at these main points: firstly, we integrate various types of prediction indicators: system-based features extracted from the MT system (N-best lists with the score of the log-linear model, source and target language model etc.), together with lexical, syntactic and semantic features to see if this combination improves the baseline's performance (Luong, 2012). Extended from our first work (Luong, 2012), we apply multiple ML models to train this feature set and then compare the performance to select the optimal one among them. Secondly, the usefulness of all features is deeper investigated in detail using a greedy feature selection algorithm. Thirdly, we propose a solution which exploits Boosting algorithm as a learning method in order to strengthen the contribution of dominant feature subsets to the system, thus improve of the system's prediction capability. Lastly, we explore the contributions of WCE in improving MT quality via re-ranking the N-best list and re-decoding the decoder's search graph, as well as enhancing the quality estimation at sentence level. All these initiatives will be consequentially introduced, starting by the WCE baseline system building in the next chapter.

Chapter 3

WCE Baseline System Building And Preliminary Experiments

1 Introduction

After reviewing the basic background WCE, in this chapter, we aim at preparing all components to build our baseline WCE systems and experiment with them. First, in Section 2, we propose a set of novel features to be incorporated with the existing ones, as presented in Chapter 2. They are extracted from various resources and expected to add new and complementary information to the classification model.

Next, all the indispensable experimental settings for system building and experiments are discussed in Section 3, including: the baseline SMT systems (**fr-en**, **en-es (WMT 2013)**, **en-es (WMT 2014)**), training and test data, feature set for each system, Machine Learning (ML) model, annotated labels setting. While describing each step, we present also corresponding toolkits or resources needed.

Section 4 details the preliminary results (in terms of Precision, Recall and Fscore) obtained for each system on the **dev** (if possible) and **test** set. With systems where **dev** set is available, we use it to tune the classification threshold and then apply the optimal value to classify the **test** set. After that, the systems' performances are analyzed and compared to the naive baselines (which classify words into random or unique label). In this section, we also compare the performance of different ML approaches on the **fr-en** test set.

2 Our Proposed Features

During the use of existing resources for extracting state-of-the art features, we discover a number of other brand new features to further depict word's characteristic. These discoveries

motivate us to extract and formulate the novel features. In this chapter, we integrate all of them with the existing ones (presented in Chapter 2), with the hope that they can complement each other. The usefulness of each type of features, as well as the best-performing subset will be investigated in Chapter 4.

2.1 Graph Topology Features

The graph topology features are extracted relying on the N-best list graph merged into a confusion network. A Confusion Network (CN) (Mangu, 2000), also known as a **"sausage"**, is a weighted directed graph, with a peculiarity that each path from the start node to the end node goes through all the other nodes. On this graph, each word in the hypothesis is labelled with its WPP, and belongs to one *confusion set*. The total probability of all edges (arcs) between two consecutive nodes sums up to 1. When using CN to visualize the N-best list, every completed path passing through all nodes in the network represents one sentence in the N-best, and must contain exactly one link from each confusion set. Figure 3.1 gives a simple CN built from a 12-best list, containing 12 possible paths from the initial node (node 1) to the final one (node 5). Looking into a confusion set (which the hypothesis word belongs to),

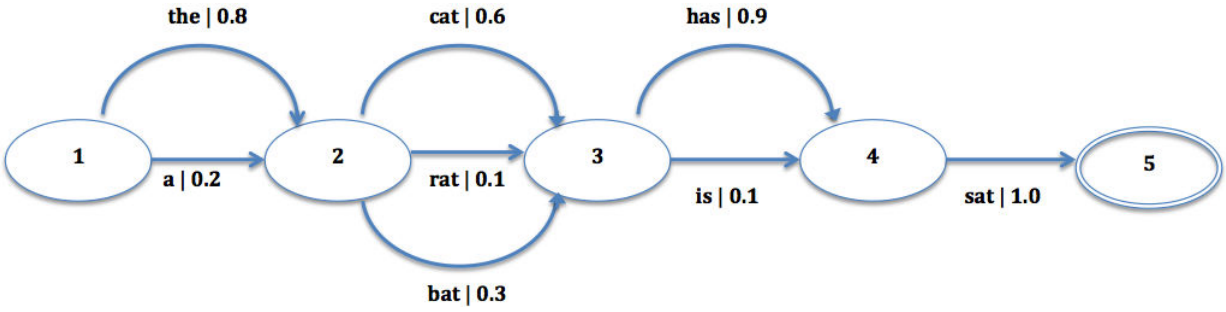

Figure 3.1 – Example of Confusion Network

we find some information that can be useful indicators, including: the *number of alternative paths* it contains (called *Nodes*), and the distribution of posterior probabilities tracked over all its words (most interesting are *maximum and minimum probabilities*, called *Max* and *Min*). We assign three above numbers as features for the hypothesis word. For instance, in case of the word *"bat"* which belongs to the confusion set connecting node 2 and node 3: its posterior probability is 0.3; the number of alternative paths is 3 (i.e. *"cat", "rat", "bat"*); the minimum and maximum values of the posterior probability distribution are 0.1 and 0.6, respectively.

2.2 Syntactic Features

As stated in Chapter 2, the syntactic characteristics of a word can convey predictions about its correctness. If a word has grammatical relations with the others, it will be more likely to be correct than those which has no relation. In case of **the target language is English**, we select the Link Grammar Parser[1] as our syntactic parser (only for English), allowing us to extract the *"Null Link"* feature (which is used in previous work). Beside this, another benefit yielded by this parser is the "constituent" tree, representing the sentence's grammatical structure (showing noun phrases, verb phrases, etc.). This tree helps to produce more word syntactic features, including *its constituent label* and *its depth in the tree* (or the distance between it and the tree root). While the POS tag reflects word's lexical function, the constituent label says about its grammatical role in the sentence. Moreover, the distance to the root node is expected to be an useful indicator, since the wrong translation tends to request further parsing, hence remains in a deeper position in the constituent tree (compare with the correct one).

Figure 3.2 represents the syntactic structure as well as the constituent tree for an MT output: *"The government in Serbia has been trying to convince the West to defer the decision until by mid 2007."*. It is intuitive to observe that the words in brackets (including "*until*" and

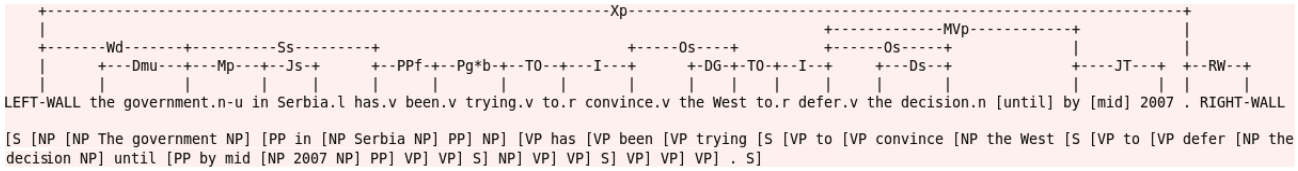

Figure 3.2 – Example of constituent tree generated by Link Grammar Parser

"*mid*") have no link with the others, meanwhile the remaining ones have. For instance, the word "*trying*" is connected with "*to*" by the link "TO" and with "*been*" by the link "Pg*b". Hence, the value of "Null Link" feature for "*mid*" is 1 and for "*trying*" is 0. The figure also brings us the constituent label and the distance to the root of each word. In case of the word "*government*", these values are "NP" and "2", respectively.

2.3 Pseudo References

Although WCE is a quality prediction without gold standard references, the comparison of MT output to another reference (coming from external popular MT engine, such as Google Translate[2] in this work) can become a helpful assistance. It is obvious that this reference

[1] http://www.link.cs.cmu.edu/link/
[2] https://translate.google.com

cannot replace the human translation to judge the MT output, but the better external MT system is, the closer distance between them will be, and therefore the more reliable this feature becomes.

Given a source sentence, we use Google Translate to generate the pseudo reference. Using this translation, we can assign a binary value $1, 0$ for each word in the MT output: in case where this word occurs in the Google Translate's translation, the value is 1; and otherwise, 0. It is important to note that in this feature, the position of occurrence does not matter, or in other manner, the word can appear anywhere in the pseudo-reference. Figure 3.3 illustrates

Source (en)	Norway 's rakfisk : Is this the world 's smelliest fish ?
Target (es)	Rakfisk de Noruega : ¿ Es esto el pescado del mundo más maloliente ?
Pseudo-Reference	Rakfisk de Noruega : ¿ Este pescado apestosa del mundo ?

Figure 3.3 – Using pseudo-reference (from Google Translate) to extract feature

this feature: in the MT hypothesis *"Rakfisk de Noruega : Es esto el pescado del mundo más maloliente ?"*, given the source sentence *"Norway 's rakfisk : Is this the world 's smelliest fish ?"*, the words *"es"*, *"esto"*, *"el"*, *"más"* and *"maloliente"* take value 0 since they can not be found in the corresponding Google Translate translation *"Rakfisk de Noruega : Este pescado apestosa del mundo ?"*. The remaining words take value 1.

2.4 LM Based Features

In Chapter 2, the LM is exploited by some previous work to take a look at various cases of occurrences of the current token and its two precedences (back-off behavior). Beside of this, we also want to investigate the longest possible n-grams that it can combine with the previous one. We hypothesize that the longer sequence between the current token and the precedences can be found in the LM, the more likely this one is to be a good translation. Applying SRILM toolkit (Stolcke, 2002) on the bilingual corpus, we build 4-gram language models for both target and source side, which permit to compute two features: the *"longest target n-gram length"* and *"longest source n-gram length"* (length of the longest sequence created by the current word and its previous ones in the target and source LM). Formally, with the target word w_i: if the sequence $w_{i-2}w_{i-1}w_i$ appears in the target language model while the sequence $w_{i-3}w_{i-2}w_{i-1}w_i$ does not, the n-gram value for w_i will be 3. The feature value set to each word hence ranges from 0 to 4.

Similarly, we compute the same value for the word aligned to w_i in the source language model. Nevertheless, it is vital to note that in case where the target word is aligned to multiple source ones, that longest n-gram found among those formed by each of them will be chosen.

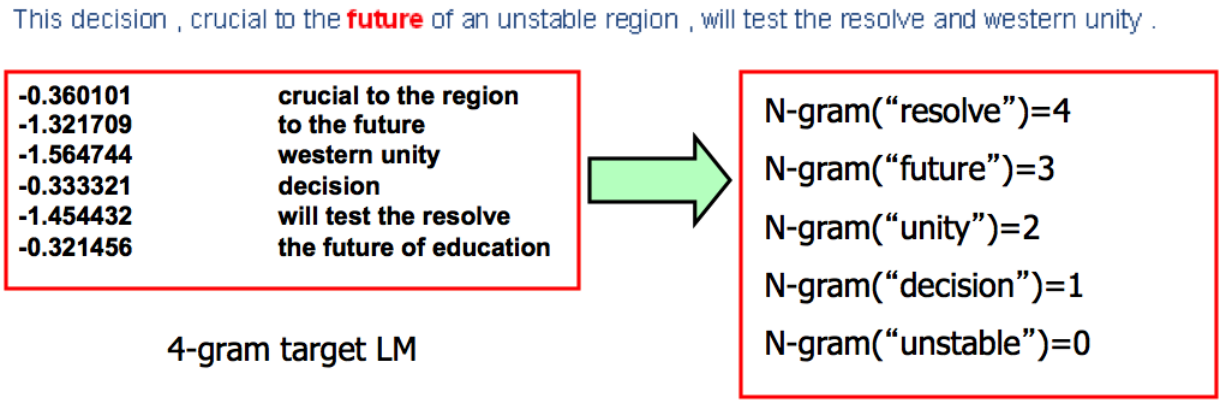

Figure 3.4 – Example of Language Model based Features

Given a 4-gram LM and the target sentence as in Figure 3.4, the longest n-gram value for the word *"future"* is 3, since the sequence *"to the future"* can be found in the LM, whereas *"crucial to the future"* can not. Similarly, in case of *"resolve"*, this value will be 4, thanks to the presence of *"will test the resolve"* in the LM.

2.5 POS LM Based Features

We exploit the Spanish and English LMs of POS tag (provided as additional resources for WMT2014 Quality Estimation tasks[3]) for quantifying the maximum length of the sequences created by the current target token's POS and those of previous ones. The same score for POS of aligned source word(s) is also computed. If the target word has multiple source alignments, only the longest POS sequence's length will be selected. Actually, we calculate the similar feature as listed in Section 2.4, but for **POS tags** instead of **words**.

Similarly, the **back-off behavior** of the word's POS tag is also taken into consideration. It takes a look at the co-ocurrence of the POS sequence of the current words and its two precedences. Let p_i denotes the POS of the word w_i. Then, POS's back-off score of w_i can be

[3]http://www.statmt.org/wmt14/quality-estimation-task.html

determined by:

$$POS\_backoff(w_i) = \begin{cases} 7 & \text{if } p_{i-2}, p_{i-1}, p_i \text{ exists} \\ 6 & \text{if } p_{i-2}, p_{i-1} \text{ and } p_{i-1}, p_i \text{ both exist} \\ 5 & \text{if only } p_{i-1}, p_i \text{ exists} \\ 4 & \text{if } p_{i-2}, p_{i-1} \text{ and } p_i \text{ exist separately} \\ 3 & \text{if } p_{i-1} \text{ and } p_i \text{ both exist} \\ 2 & \text{if only } p_i \text{ exists} \\ 1 & \text{if } p_i \text{ does not exist.} \end{cases} \tag{3.1}$$

(The concept "exist" here means "appear in the POS's language model").

In summary, three POS LM's new features for **en-es (WMT14)** system are built, including: *"longest target POS's n-gram length"*, *"longest POS's source n-gram length"* and *"POS's back-off score"*. This is the first time we experiment with these new features (actually, in our **fr-en** system which was built prior, we did not think about them).

2.6 Occurrence in Multiple Reference Systems

This feature is designed solely for **en-es (WMT2014)** system, based on the novel point of this campaign compared to the previous ones: the target outputs come from multiple translation means (from statistical MT, rule-based MT systems or even from humans) for the same source sentences. Given a translation, all the remaining ones of the same source sentence can be considered as its references. Naturally, one would have an intuition that: the occurrence of a word in all (or almost) systems implies a higher likelihood of being a correct translation. Relying on this observation, we add a new binary-value feature, telling whether the current token can be found in more than **N%** (in our experiments, we choose $N = 50$) out of all remaining translations generated for the same source sentence. Here, in order to make the judgments more accurate, we propose several additional references besides those provided in the corpora, coming from: (1) Google Translate system, (2) The baseline SMT engine provided for WMT2013 English - Spanish QE task. These two MT outputs are added to the already available (maximum) four MT outputs of a given source sentence, before calculating the (above described) binary feature. In Figure 3.5, we have five references to match with the current MT hypothesis: *"pero , de hecho , todo es caro . una caña de cerveza o un sándwich te cuestan la friolera de 9 ($ 14) cada uno ."*. The word *"todo"* takes value 1 since it appears in

Source (en)	but then everything is expensive - a small glass of beer or a sandwich knock you back £ 9 ($ 14) each .
Target (es)	pero , de hecho , **todo** es caro . una caña de cerveza o un sándwich te **cuestan** la **friolera** de 9 £ ($ 14) cada uno .
Pseudo Ref. 1	pero entonces **todo** es caro , un pequeño vaso de cerveza o un bocadillo llamar a 9 libras esterlinas (14 dólares) cada uno .
Pseudo Ref. 2	pero entonces **todo** es caro - un pequeño vaso de la cerveza o un bocadillo le golpea atrás 9 £ (14 $) cada uno .
Pseudo Ref. 3	pero entonces **todo** es caro , un pequeño vaso de cerveza o un bocadillo pegarle atrás 9 libras (14 dólares) cada uno .
Pseudo Ref. 4	pero luego **todo** es caro - un pequeño vaso de cerveza o un bocadillo que te golpean la espalda £ 9 ($ 14) cada uno .
Pseudo Ref. 5	pero entonces **todo** es caro - un vasito de cerveza o un sándwich de hacerte volver £ 9 (14 dólares) cada uno .

Figure 3.5 – Example of multiple-system occurrence feature

all references (100%), whereas *"cuestan"* and *"friolera"* take value 0 for their absence in all translations (0%).

3 Experimental Settings

3.1 Baseline SMT System

3.1.1 French - English System

Our baseline French - English SMT system (or **"fr-en"** thereafter for short) is a phrase-based system, constructed using the Moses toolkit (Koehn et al., 2007). This open-source toolkit contains all of the necessary components to train the translation model. We keep the Moses's default setting: log-linear model with 14 weighted feature functions, including: 7 reordering, 1 language model, 5 translation model and 1 word penalty features (Potet et al., 2010). To train the **translation model**, we use the Europarl and News parallel corpora that are used for WMT evaluation campaign in 2010 (total 1,638,440 sentences). Our target **language model** is a standard n-gram language model trained using the SRI language modeling toolkit (Stolcke, 2002) on the news monolingual corpus (48,653,884 sentences). Some pre-processing steps are invoked before decoding, encompassing normalize texts (to remove all strange characters, symbols...), tokenize and lowercase them (using the scripts: *normalize-punctuation.perl*, *tokenizer.perl* and *lowercase.perl* available in Moses toolkit, respectively). Then, the translation model is filtered according to the input text in order to avoid loading the entire phrase table

into the internal memory (using the script *filter-model-given-input.pl* available in Moses toolkit), thus speeding up the decoding process. After that, in the decoder phase, we also called some following extended options of Moses for tracking both source and target sides information which is mandatory to build our system-based features, as well as prepare resources for further WCE exploitations in MT improvement (discussed in Chapter 6 and Chapter 7):

- *-print-alignment-info-in-n-best*: Display source-to-target and target-to-source word-to-word alignments into the N-best list
- *-n-best-list FILE SIZE [distinct]*: Generate an n-best file of up to SIZE distinct sentences into file FILE
- *"-output-search-graph"*, *"-search-algorithm 1"* and *"-cube-pruning-pop-limit 5000"*: generates the search graph which contains all possible hypotheses, uses cube pruning and adds 5000 hypotheses to each stack.

3.1.2 English - Spanish System

Similarly, the English - Spanish (**"en-es"**) SMT system used in WMT 2013 is also a Moses phrase-based system, trained on Europarl and News Commentaries corpora provided by WMT (Bojar et al., 2013). The n-gram (n=3) LMs of source and target languages are generated using the SMT training corpora and SRILM toolkit (Stolcke, 2002). The unigram, bigram and trigram are also provided. The IBM Model 1 lexical labels are generated by using GIZA++ toolkit (Och and Ney, 2003). The configuration file used for decoding, as well as the code to re-run the entire Moses system can be downloaded from: `http://www.quest.dcs.shef.ac.uk`. For feature extraction, we use the entire system and all related resources provided. In WMT14, since the MT output are collected from multiple translation means (statistical, rule based systems, even human), there is no specific SMT setting released. We had to use new strategies to maintain part of system-based features.

3.2 Corpora

3.2.1 French - English Corpus (fr-en)

We use our above SMT system to generate the translation hypothesis for 10,881 source sentences taken from several news corpora of the WMT evaluation campaign (from 2006 to 2010). A post-edition task was implemented by using a crowd sourcing platform: Amazon's Mechanical Turk (MTurk)[4], which allows a "requester" to propose a paid or unpaid work and a "worker" to

[4]https://www.mturk.com

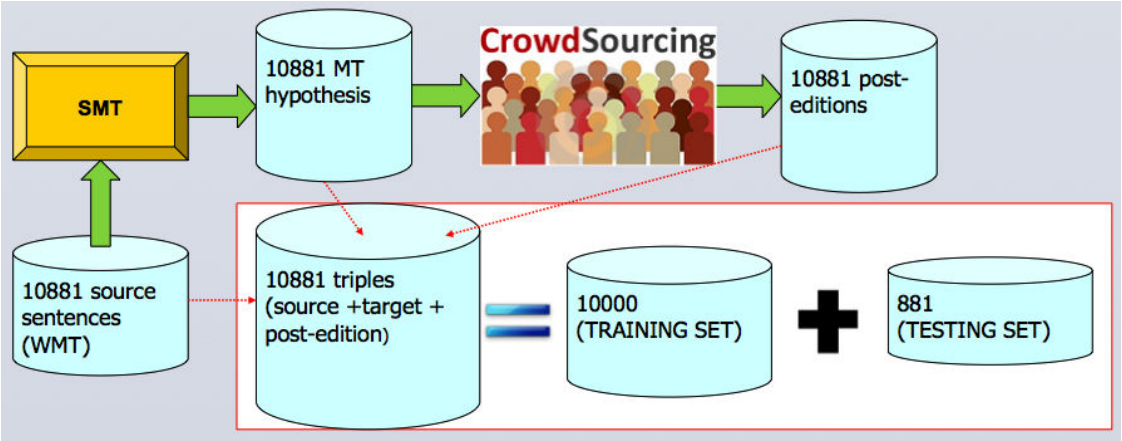

Figure 3.6 – French - English corpus preparation: training set = 10,000 triples, test set = 881 triples

perform the proposed tasks. To avoid the huge gaps between the hypothesis and its post-edition since the correctors can paraphrase or reorder words to form the smoother translation, we highly recommended them to keep the number of edit operations as low as possible, but still ensure the accuracy and fluency of this translation with the French sentence. A sub-set (containing 311 sentences) of these collected post-editions was evaluated by a former professional post-editor. Testing result showed that 87.1% of post-editions improve the hypothesis, while the rest of 12.5% remains equivalent, and only the rest of 0.04% degrades. The corpus construction process is visualized in Figure 3.6, as well as its detailed description can be found in Potet et al. (2012). Finally we extracted randomly 10,000 sentences triples (including source sentence, translation hypothesis and post-edited hypothesis) to form the training set, and keep the remaining 881 sentence triples for the test set.

3.2.2 English - Spanish Corpus of WMT 2013 (en-es_13)

In WMT13, the organizers provided two bilingual data sets, from English to Spanish: the training and the test ones. The training set consists of 803 MT outputs, in which each token is annotated with one appropriate label. In the binary variant, the words are classified into *"K"* (Keep: no translation error) or *"C"* (Change: edit operator needed) label, meanwhile in the multi-class variant, they can belong to *"K"* (Keep), *"S"* (Substitution) or *"D"* (Deletion). However, in this thesis, we report only the binary task, and the multiple class experiments can be found in detail in (Luong et al., 2013c) . The test set contains 284 sentences where all the labels accompanying words are hidden. For optimizing parameters of the classifier, we extract 50 sentences from the training set to form a development set. Since a number of repetitive sentences are observed in the original training set, the dev set was carefully chosen to ensure that there is no overlap with the new training set (753 sentences), keeping the tuning process

accurate. Some statistics about each set can be found in Table 3.1.

Data set	Train	Dev	Test
#segments	753	50	284
#distinct segments	400	50	163
#words	18435	1306	7827
%K : %C	70: 30	77: 23	-

Table 3.1 – Statistics of training, dev and test sets - WMT 2013 **en-es** corpus.

3.2.3 English - Spanish Corpus of WMT 2014 (en-es_14)

Similarly to WMT 2013, the WMT 2014 organizers released two bilingual data sets for **en-es** Quality Estimation (QE) task : the training and the test ones. The major difference is that the target sentences are collected from multiple translation means (e.g. SMT, rule-based MT, and even human), with the aim towards a SMT system-independent and broadly-applied estimation. The training set contains 1.957 MT outputs, in which each token is annotated with one appropriate label. In the binary variant, the words are classified into *"OK"* (no translation error) or *"BAD"* (edit operators needed) label. Meanwhile, in the level 1 variant, they belong to *"OK"*, *"Accuracy"* or *"Fluency"* (two latter ones are divided from *"BAD"* label of the first subtask). In the last variant, multi-class, beside *"Accuracy"* and *"Fluency"* we have further 15 labels based on MQM metric: *Terminology, Mistranslation, Omission, Addition, Untranslated, Style/register, Capitalization, Spelling, Punctuation, Typography, Morphology_(word_form), Part_of_speech, Agreement, Word_order, Function_words, Tense/aspect/mood, Grammar* and *Unintelligible*. The test set consists of 382 sentences where all the labels accompanying words are hidden. For optimizing parameters of the classifier, we extract last 200 sentences from the training set to form a development (dev) set. Besides, the Spanish - English corpus provided in WMT 2013 (total of 1087 tuples) is also exploited to enrich our WMT 2014 system. Unfortunately, WMT2013 data can only help us in the binary variant, due to the discrepancy in training labels. This thesis reports the experiments with the binary system, and those of the other tasks can be referred in (Luong et al., 2014). Some statistics about each set can be found in Table 3.2.

Data set	Train	Dev	Test
#segments	1757	200	382
#distinct segments	1757	200	382
#words	40975	6436	9613
%G (OK) : %B (BAD)	67 : 33	58 : 42	-

Table 3.2 – Statistics of training, dev and test sets - WMT 2014 **en-es** corpus.

3.3 Annotated (Oracle) Labels Setting

The quality labels for words are used along with features to train the classifier. They can be set manually by human (which is accurate yet expensive and unfeasible for huge data set) or automatically by various toolkits. For instance, Xiong et al. (2010) exploit the Levenshtein alignment between the hypothesis and its reference. In another method, the Translation Error Rate (TER) alignment is performed by Nguyen et al. (2011).

We inherit the annotations for **en-es(WMT2013)** corpus which are derived automatically by computing Word Error Rate (WER) between the MT hypothesis and its post-edited version. In case of **en-es(WMT2014)**, the annotations are performed manually by professional translators. As depicted above, WMT13 supports two types of labels: multi-class and binary, meanwhile WMT14 provides binary, level-1 (three labels) and multi-class (16 labels). In the training set of WMT 2013, 70 % are labelled as "good" and the remaining 30% are the "bad" ones. Meanwhile, in WMT 2014 training set, this percentage is 67% and 33%.

For **fr-en** corpus, this task is performed by TERp-A toolkit (Snover et al., 2008). As an extension of TER, TERp-A takes into account the linguistic edit operations, such as *Stem matches*, *Synonyms matches* and *Phrase Substitutions* besides the TER's conventional ones (*Exact match, Insertion, Deletion, Substitution* and *Shift*). These additions allow us to avoid categorizing the hypothesis word as *Insertion* or *Substitution* in case it shares same stem, or belongs to the same synonym set on WordNet, or is the phrasal substitution of word(s) in the reference. Also in TERp-A, each above-mentioned edit cost has been tuned to maximize the correlation with human judgment of **Adequacy** at the segment level. Table 3.3 illustrates the labels generated by TERp-A for one MT hypothesis (*"The result of the hard-line trend is also important ."*) and its reference (*"The consequence of the fundamentalist movement also has its importance ."*). Each word or phrase in the hypothesis is aligned to a word or phrase in the reference with different types of edit: "I" (insertions), "S" (substitutions), "T" (stem matches), "Y" (synonym matches), and "P" (phrasal substitutions). The lack of a symbol indicates an exact match and will be replaced by "E" thereafter. We do not consider the words marked with "D" (deletions) since they appear only in the reference (post-edition). Then, to train a binary classifier, we re-categorize the obtained 6-label set into binary set: The E, T and Y belong to the *Good* (G), whereas the S, P and I belong to the *Bad* (B) category. Finally, we observed that out of total words (train and test sets) are 85% labeled "G", 15% labeled "B".

Reference	The	consequence	of	the	fundamentalist	movement		also	has	its importance	.
		S			S	Y	I		D	P	
Hyp After Shift	The	result	of	the	hard-line	trend	is	also		important	.

Table 3.3 – Example of training label obtained using TERp-A for **fr-en** corpus.

3.4 Feature Set for Systems

We build the feature set for each experimental baseline WCE system by integrating our proposed features with the state-of-the-art ones.

No	fr-en	en-es (WMT2013)	en-es (WMT2014)
1	Target word	Target word	Target word
2	Source word	Source word	Source Word
3	Target POS	Target POS	Target POS
4	Source POS	Source POS	Source POS
5	Right target context	Right target context	Right target context
6	Left target context	Left target context	Left target context
7	Right source context	Right source context	Right source context
8	Left source context	Left source context	Left source context
9	Constituent label	Constituent label	Constituent label
10	WPP *any*	WPP *any*	Occur in multiple systems
11	Max	Max	Longest target POS gram length
12	Min	Min	Longest source POS gram length
13	Nodes	Nodes	Occur in Google Translate
14	Longest target gram length	Longest target gram length	Longest target gram length
15	Longest source gram length	Longest source gram length	Longest source gram length
16	Backoff behaviour	Backoff behaviour	Backoff behaviour
17	Number of occurrences	Number of occurrences	Number of occurrences
18	Punctuation	Punctuation	Punctuation
19	Stop word	Stop word	Stop word
20	Proper name	Proper Name	Proper name
21	Numeric	Numeric	Numeric
22	Polysemy count (target)	Polysemy count (target)	Polysemy count (target)
23	Distance to Root	Distance to Root	Distance to Root
24	WPP *exact*	Polysemy count (source)	Polysemy count (source)
25	Null link	Occur in Google Translate	

Table 3.4 – The entire features used to build **fr-en**, **en-es(WMT14)** and **en-es (WMT13)** baseline WCE classifiers. Please note that feature sets are ***not exactly the same*** for three systems.

Since the feature extraction depends on various conditions (e.g. the language pair, the availability of resources or toolkits, etc.), the set used for **fr-en**, **en-es (WMT13)** and **en-es (WMT14)** system are not identical (although they share the major part). Among our proposed features, several ones can not be applied to all three WCE systems. The detailed

feature set used for each system is listed as seen in Table 3.4. For preliminary experiments, all of them are used to train the classifier. To wrap up, we employ 25 feature types in **fr-en** and **en-es(WMT2013)** systems, and 24 feature types in **en-es(WMT2014)** system.

3.5 Classifiers and Toolkits

Motivated by the idea of treating WCE as a sequence labeling task, we employ the *Conditional Random Fields* (CRF) model (Lafferty et al., 2001) as our principal model, used to experiment with all systems. Beside this, in order to judge objectively the CRF model's performance, we apply also several other conventional models on **fr-en** system, including: *Decision Tree*, *Logistic Regression* and *Naive Bayes* using KNIME platform[5], and then compare their results. KNIME is a user-friendly graphical workbench for the entire system building, testing and performance analysis/report processes. Each functional module (e.g. the learner, predictor, scorer, data reader and writer, etc.) is visualized by a "node" which can be easily dragged from the toolbox and dropped into the working window. They can be customized by users and are connected together to form the entire data flow. The intuitive design of **Naive Bayes, Logistic Regression** and **Decision Tree** classifiers using this toolkit is shown in Figure 3.7.

Among CRF based toolkits, we selected a console-based one named WAPITI (Lavergne et al., 2010) to train our classifier. This toolkit is developed by the LIMSI laboratory (France) with the main advantage of segmenting and labeling rapidly large numbers of sequences with discriminative models. Besides, it proposes various optimization and regularization methods to improve both the computational complexity and the prediction performance of these standard models, such as: Quasi-Newton L-BFGS and OWL-QN, Resilient propagation (R-PROP), Stochastic gradient descent (SGD-L1), Block-wise coordinate descent (BCD), etc. All parameters invoked by each method in the training phase are listed in Table 3.5. We also compare our classifier with two naive baselines, including:

- **Baseline 1** (all-good system): all words in each MT hypothesis are classified into G label.
- **Baseline 2**: we assigned them randomly yet with respect to the corresponding percentage of these labels in the corpus (e.g. in case of **fr-en** corpus, these percentages are: 85% G, 15% B).

[5]http://www.knime.org/knime-desktop

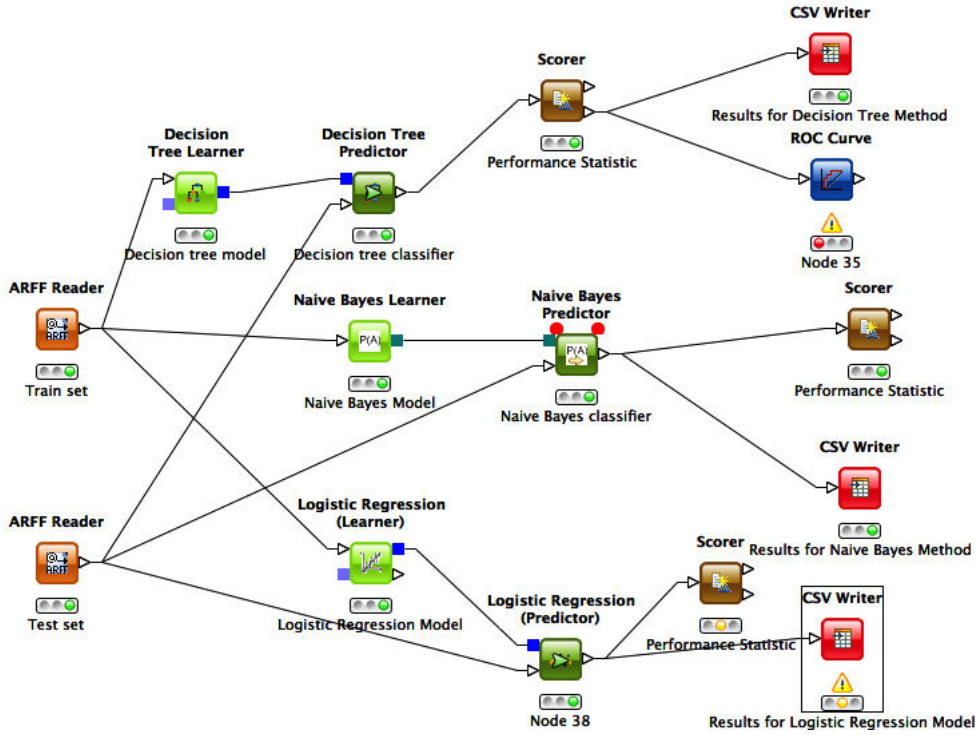

Figure 3.7 – **Naive Bayes, Logistic Regression** and **Decision Tree** classifiers building and testing using KNIME toolkit

4 Preliminary Results and Analysis

4.1 Results of CRF Model

We evaluate the performance of our classifiers by using three common evaluation metrics: Precision (Pr), Recall (Rc) and F-score (F). We perform the preliminary experiments by training first CRF classifiers with the combination of all features. The classification task is then conducted multiple times, corresponding to a threshold increase from 0.300 to 0.975 (step = 0.025). When threshold = α, all words in the test set which the probability of G class exceeds α will be labelled as "G", and otherwise, "B". The values of Pr and Rc of *"G"* and *"B"* label are tracked along this threshold variation, and then are averaged for "all-feature" and two naive baseline systems. It is important to note that, with **fr-en** system, the optimal classification threshold is measured on the test set, meanwhile for **"en-es (WMT14)"** and **"en-es (WMT13)"** ones, it is tuned on the dev set (since the annotated labels of the test set are released only at the end of the WMT campaign).

Before moving on analyzing the performance of each classifier, we take a look at an example on Figure 3.8, illustrating the classification results and the prediction accuracy given by our

ML Method	Parameters	Function	Value
CRF	a	algorithm	sgd-l1
	maxiter	maximum number of iterations	200
	stopeps	stop epsilon value	0.00005
	stopwin	stop window size	6
Decision Tree	Pruning method	Method to reduce tree size to avoid over-fitting	Minimal Description Length
	Quality measure	Select the quality measure according to which the split is calculated	Gain Ratio
	Min number records	the minimum number of records required in each node to ensure the tree's further growth.	3
	Skip out-of-domain column	If checked, nominal columns containing no domain value information are skipped	Unchecked
Logistic Regression	values	specify the independent columns that should be included in the regression model	all
Naive Bayes	Skip missing values	Ignoring missing values in the model (if this option is activated) or consider them during the class probability calculation.	Yes (1)
	Maximum number of unique nominal values per attribute	All nominal columns with more unique values than the defined number will be skipped during learning.	100.000

Table 3.5 – Summary of parameters used in different ML methods for training the classifiers

fr-en classifier in a hypothesis given a source sentence. Each word in the MT hypothesis is assigned by a label *"G"* or *"B"* (displayed in the last row). The oracle labels are in the fourth row. Comparing these labels, we can easily identify all good words that are correctly judged by the system, bounded by pink rectangles, such as *"operation", "therefore", "added", ...* Besides, some translation errors are correctly detected, as shown in green dashed rectangles (*have*, *is*, *a-t-il*,...). On the contrary, the grey dotted rectangles contains our system's prediction errors: some nonsensical words are recognized as good one (*"not*$_2$*"*, *combat*,...) and vice versa (*not*$_1$, *a*,...).

The classification results of three systems and their "naive" baselines can be seen on Table 3.6. Now, we analyze each system.

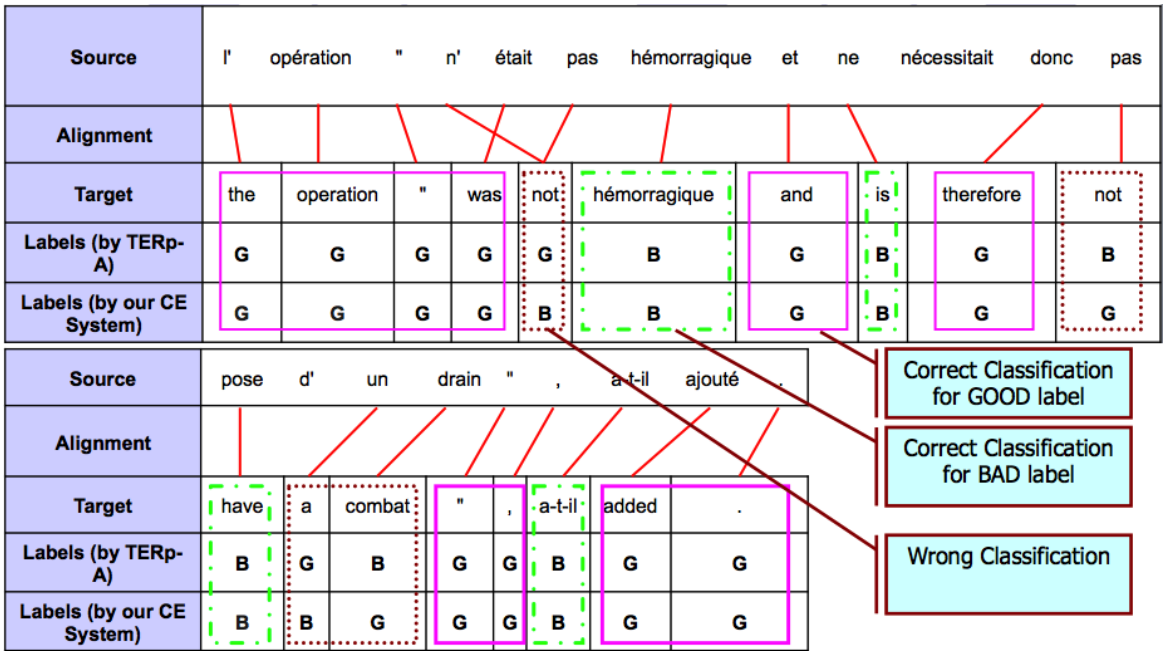

Figure 3.8 – Example of our WCE classification results for one MT hypothesis

4.1.1 "fr-en" System

The results of **"fr-en"** system is reported in the green zone on Table 3.6. They are the scores obtained on the test set (since we do not have the dev set) and averaged over all scores during threshold variation process (as depicted above). The results suggest that the **"all feature"** classifier reaches an very impressive F score for "G" label (87.07%), meanwhile an acceptable performance for "B" one (37.76%). We claim it as "acceptable" due to the fact that the imbalance occurrence between "B" and "G" label (15% vs 85%) in the corpus would affect the learning functions. Moreover, the **"all feature"** classifier outperforms dramatically both two baselines. Compare to **Baseline 1** (always predicts "G"), it is slightly worse in the F score of "G" labels (2.92 points under), yet exceeds far above in that of "B" one (37.76% vs 0%). With **Baseline 2** (randomly classify), it overwhelms in both labels (3.62 % for "G" and 21.5% for "B" -relative F score).

We break down the analysis into the fluctuations of F score for **"G"** and **"B"** labels in *"all feature"* classifier, as shown in Figure 3.9. It can be seen that with the increase of threshold, the F score ("G" label) gradually and slightly drops from 0.885 to 0.835. On the contrary, the score ("B" label) moves on the opposite direction from 0.341 to 0.415. The "optimal point" where the average value between two scores $((F(\text{“}G\text{”}) + F(\text{“}B\text{”}))/2)$ reaches maximum is 0.70.

In a nutshell, we find two conclusions can be made from observations are: (1) Since the

System		Label	dev			test		
			Pr(%)	Rc(%)	F(%)	Pr(%)	Rc(%)	F(%)
fr-en	All features	Good	-	-	-	85.99	88.18	87.07
		Bad	-	-	-	40.48	35.39	37.76
	Baseline 1	Good	-	-	-	81.78	100.00	89.98
		Bad	-	-	-	-	0	-
	Baseline 2	Good	-	-	-	81.77	85.20	83.45
		Bad	-	-	-	18.14	14.73	16.26
en-es (WMT13)	All features	Good	85.79	84.68	85.23	78.83	83.54	81.12
		Bad	50.96	53.16	52.04	52.14	44.45	48.01
	Baseline 1	Good	74.24	100.00	85.21	70.21	100	82.49
		Bad	-	0	-	0	-	0
	Baseline 2	Good	76.21	73.97	75.07	72.71	69.87	71.26
		Bad	26.35	27.50	26.91	22.12	23.69	22.87
en-es (WMT14)	All features	Good	68.62	82.69	75.01	67.88	82.92	74.65
		Bad	64.38	45.73	53.47	54.22	37.18	44.10
	Baseline 1	Good	66.31	100.00	79.74	63.01	100	77.31
		Bad	-	0	-	0	-	-
	Baseline 2	Good	70.61	66.65	68.57	68.01	63.33	65.58
		Bad	30.76	32.50	31.60	28.98	30.65	29.79

Table 3.6 – Average Pr, Rc and F for labels of each all-feature system and then two naive baselines.

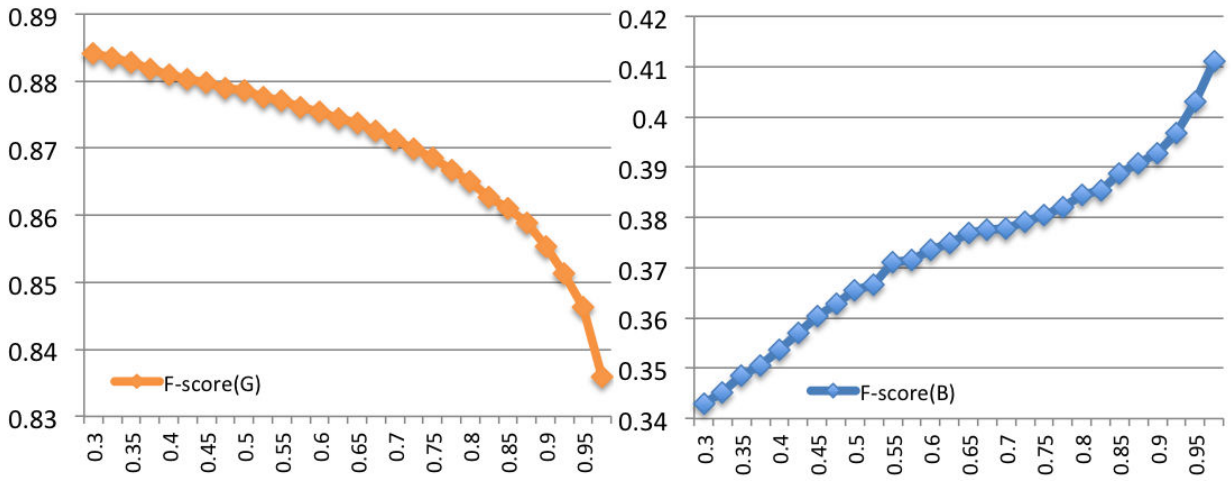

Figure 3.9 – The fluctuation of F score of "G" and "B" label during threshold variation (**"fr-en"** system)

classifier tends to memorize "G" labels much better than "B" one, the system optimization should focus on endeavors to boost the translation error detection capability; and (2) The use of various types of features is indispensable and helps to enhance sharply WCE system far above the naive baselines.

4.1.2 "en-es-WMT13" System

The scores of **"en-es-WMT13"** system on both **dev** and **test** set are reported in the red zone of Table 3.6. On **dev** set, we observe that the **"all feature"** classifier outperforms both baselines in both labels. More specifically, it wins marginally **Baseline 1** (0.02%) yet significantly **Baseline 2** (10.16%) in terms of F score (relative) of "G" class. Especially in "B" class, its score leaves big gaps to both baselines (52.04% vs 0% and 26.91%). During

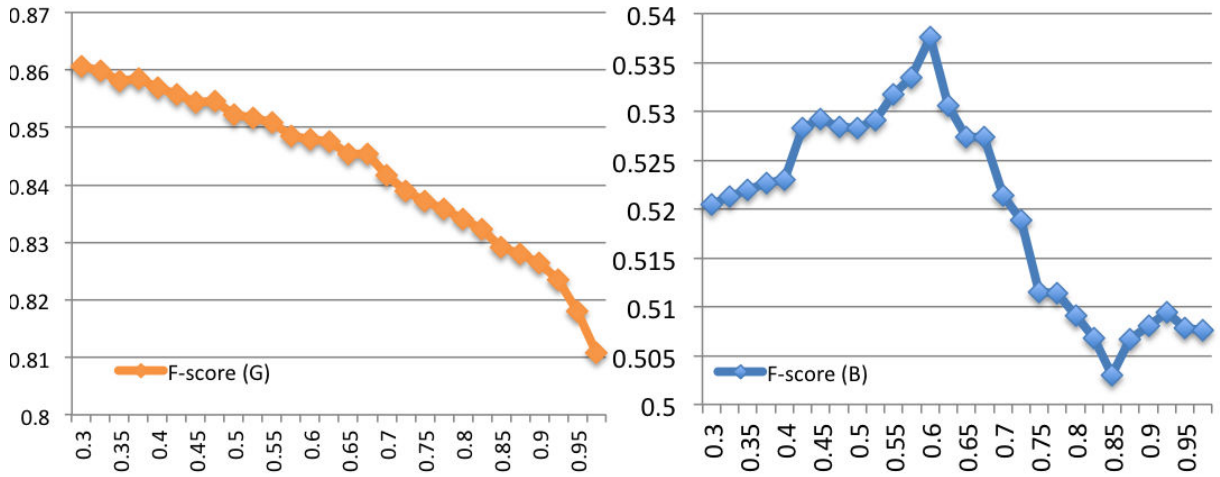

Figure 3.10 – The fluctuation of F score of "G" and "B" label during threshold variation (**"en-es-wmt2013"** system)

the classification threshold tuning (Figure 3.10), the F score of "G" class goes down (almost linearly) from 86.07% to 80.06%. Meanwhile, the fluctuation of that in "B" one is chaotic and the peak corresponds to the threshold value of 0.60. In order to determine the optimal threshold which will be used to classify the **test** set, we calculate the average F score of both labels $(F(G) + F(B))/2$ and plot them as on Figure 3.11. The curve suggests that we can obtain the highest overall performance at the classification threshold value of 0.625.

Applying this optimal threshold on the blind **test set** provided by WMT 2013 organizers, we get the official results of 81.12% and 48.01% F score for "G" and "B" labels, respectively. As a coherence with previous systems, this one beats its two naive baselines in terms of $(F(G) + F(B))/2$ criterion. It is very commendable that more than a half of translation errors is correctly detected, pointed by the Pr score of 52.14% ("B" class). More interestingly, our

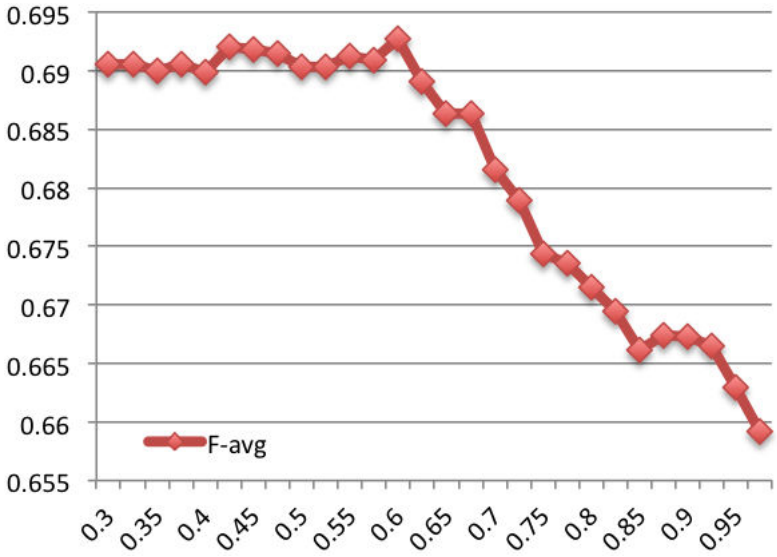

Figure 3.11 – The optimal threshold decision via threshold variation (**"en-es-wmt2013"** system)

submitted WCE systems achieved the **first rank** for Task 2 of WMT 2013 (Bojar et al., 2013), for both *binary* and *multi-class* variants. Nevertheless, it is important to note that we did not submit directly the above **"all feature"** system; yet optimize it by a number of techniques and then select two best performing ones as submission. These optimization solutions will be discussed in the next chapter (Chapter 4).

4.1.3 "en-es-WMT14" system

The results of **"en-es-WMT14" system** can be viewed in the white zone on Table 3.6. Actually, it is the system that we use to participate in WMT 2014 (along with another one obtained after the feature selection process) (Luong et al., 2014). During experiments, we observe that the system trained on both WMT2014 + WMT2013 data (which is already available from WMT 2013) slightly outperforms the one trained solely on WMT2014 dataset. Therefore, we select the former one to report in this thesis, and for the sake of simplicity, we call it the **"en-es-WMT14"** system.

Dealing with WMT 2014 **dev** set, the **"all system"** classifier reaches 75.01% F score for "G" and 53.47% for "B" class. Compared to that of WMT 2013, this system is outperformed by the score of "G", yet wins in that of "B" label. The two baselines are totally below it in average score $(F(G) + F(B))/2$.

During threshold variation (Figure 3.12), F score ("G") ranges from 73.61% to 74.81% (it tends to raise up at the beginning, then goes parallel with the x-axis until the threshold reaches

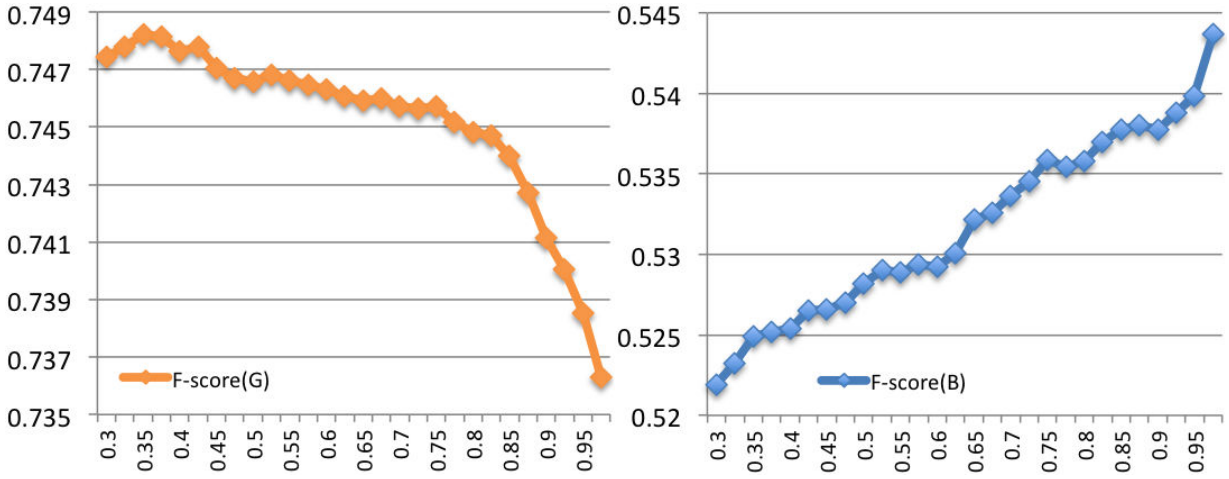

Figure 3.12 – The fluctuation of F score of "G" and "B" label during threshold variation (**"en-es-wmt2014"** system)

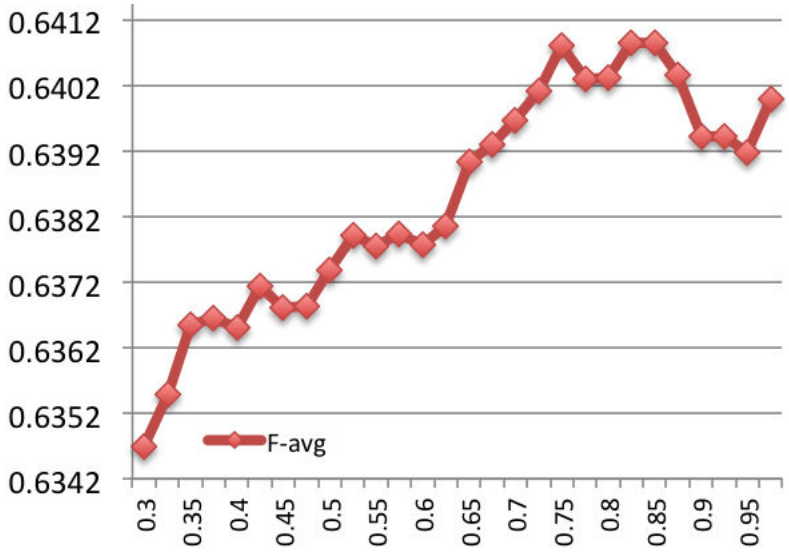

Figure 3.13 – The optimal threshold decision via threshold variation (**"en-es-wmt2014"** system)

0.775 and finally drops quickly to the bottom). Meanwhile, in case of "B", the score increases almost gradually from 52.20% up to 54.47%. Since the growth speed of "B" is faster than the degradation speed of "G" as seen in the two curves, the average score $(F(G) + F(B))/2$ tends to augment from the first bottom point up to the peak at the threshold value of 0.75 and then fluctuates until the final point (Figure 3.13). We then select this value to classify the WMT 2014 unseen **test** set.

The official results on **test set** show promising performance for "G" (74.65%) and honorable for "B"; and confirm once again that using new features helps to boost the classifier dramatically above the baselines. Similarly to **"en-es-WMT13"** system, we attempt to optimize the **"all feature"** first (discussed in the next chapter) before deciding the official versions to participate

in the campaign.

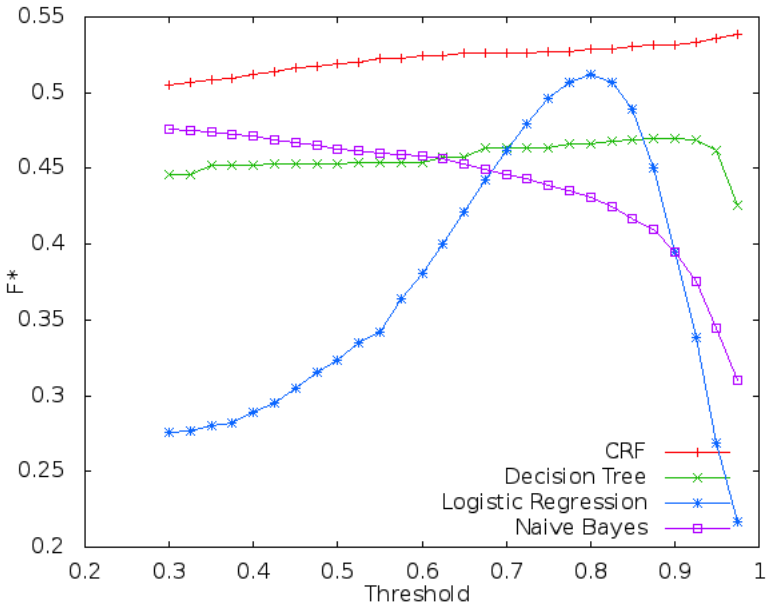

Figure 3.14 – Performance (F^*) of different "all feature" classifiers (by different models)

4.2 Results of Other ML Methods And a Comparison to CRF Model

In an attempt of investigating the performance of CRF model, we compare the **"fr-en all feature"** system with those built by several other models, including: *Decision Tree*, *Logistic Regression* and *Naive Bayes*. These three classifiers are trained in the same conditions (feature set, training data set) as we use for CRF model, and then are used to deal with the usual **fr-en test** set. The pivotal problem is how to define an appropriate metric to compare them efficiently? Due to the fact that in our **fr-en** training corpus, the number of G words sharply outperforms the B ones, so it is fair to say that with our classifiers, detecting a translation error should be more appreciated than identifying a good translated word. In other words, we put a priority in bad translation detection productivity when judging system's performance. Therefore, we propose a "composite" score called F^* putting more weight on the system capability of detecting B words: $F^* = 0.70 * Fscore(B) + 0.30 * Fscore(G)$, where $Fscore(G)$ and $Fscore(B)$ are the F scores at each value of threshold. These weights $(70\%, 30\%)$ are chosen since they emphasize the role of $Fscore(B)$ while their values are not extremely polarized. Next, we track all scores amid the threshold variation and then plot them in Figure 3.14. In *Logistic Regression* classifier, F^* raises up monotonously from 28.14% to the peak of 51.03% and then drops rapidly down to the bottom (21.68%). The performance of *Decision Tree* classifier has

a narrower change, from 44.93% to 46.27% and then down to 43.01%. Meanwhile, in *Naive Bayes* classifier, this score moves in the opposite direction, degrading gradually from 43.01% to 31.23%. Finally, the **"CRF"** classifier's score increases from the starting point (50.96%) up to the final point (54.14%). Another notable observation is that the "optimal" threshold (which gives the best F^*) for each classifier is different from the others: 0.975 for *CRF*, 0.925 for *Decision Tree*, 0.800 for *Logistic Regression* and 0.300 for *Naive Bayes* classifier. Comparing these curves, the topmost position of *CRF* one shown in the figure reveals that the *CRF* model performs better than all the remaining ones, and it is more suitable to deal with our features and corpus. We observe different fluctuations of this score.

Hence, in the next sections which propose ideas to improve the prediction capability, we work only with the CRF classifier.

5 Summary and Conclusion

This chapter reported our preliminary contributions to the domain, starting with gathering prominent previous features along with suggesting novel ones to combine with them. The proposed features came from different internal or external resources, and their extractions required a number of toolkits and libraries. The graph topology (Nodes, Min, Max) features rely on the N-best list graph merged into a confusion network. The syntactic features (constituent label, depth in constituent tree) were extracted using specific parsers. The pseudo references were exploited to build word's occurrence features. Language Models (in both source and target sides) for word as well as for the word's POS tag were also used to obtain additional useful information (LM-based, POS LM-based, Back-off behavior features).

The next contribution was the exploitation of several datasets to train the baseline WCE classifiers, followed by the setting of all other crucial elements for experiments. We extracted 25 feature types for **fr-en** corpus, 25 feature types for **"en-es (WMT2013)"** and 24 feature types for**"en-es (WMT2014)"** ones; then divided into **train**, **dev** and **test** sets. The word annotated labels were automatically obtained by matching the translation and its reference. Several Machine Learning models were tested and compared to the main approach - CRF model.

The performances (F scores) obtained on three **"all feature"** systems reflected the similar and coherent phenomena: the classifiers identified good translation better than detected translation errors, and the feature sets helped to boost efficiently the prediction capability of all systems above their naive baselines (all-good and random predictions). However, the acuteness of each feature has not been considered yet and the question of whether or not redundant

features exist in the set and undermine the rest remains open. These issues will be directly addressed in Chapter 4 as an endeavor to improve the "all feature" classifiers' performances.

In addition, among exploited ML algorithms, the CRF approach overwhelmed all remaining ones on **fr-en** data in terms of average score between F score of "G" and "B" classes; revealing that treating WCE as a sequence labeling task is a wise try. The CRF model will continue be applied in the up-coming tests.

Chapter 4

WCE System Optimization Techniques

1 Introduction

The integration of all features boosts dramatically the WCE classifiers (**fr-en**, **en-es (WMT 2013)** and **en-es (WMT 2014)**) performance over their naive baselines, as shown in Chapter 3. Nonetheless, the question concerning whether these "all-feature" systems are the most effective ones that we can achieve given existing resources and settings remains unanswered. In this chapter, we propose several solutions to improve the prediction's accuracy, starting from "all-feature" systems.

Putting all features altogether might not be the wise try, since some among them are poor predictors and harm the others when combined. In section 2, we apply the feature selection strategy to investigate the usefulness of each one, which then helps to retain only the informative candidates as well as to waive redundant ones. It also yields the best performing subset along with the optimal system's performance.

Next, another issue that we are concerned is the system's learning capability when dealing with a large amount of features. This leads to another attempt of forming multiple feature subsets from the original, building a number of sub-models from them, and finally combining "weak" classifiers in a reasonable way to take advantage of their complementarity. The final "composite" classifier is expected to be much stronger than each individual thanks to this fact. The entire process, which is normally known as **Boosting technique** is detailed in section 3.

2 Feature Selection

As stated before, the all-feature **fr-en**, **en-es (WMT2013)** and **en-es (WMT2014)** systems yielded appealing F scores for G label, but not very convincing F scores for B label. That can be originated from the risk that not all of features are really useful, or in other words,

some are poor predictors and might be the obstacles weakening the other ones. In order to mitigate this drawback, we propose a method to filter the best features relied on the "Sequential Backward Selection" (SBS) algorithm[1]. We start from the full set of N features, and in each step sequentially remove the most useless one. To do that, all subsets of (N-1) features are considered and the subset that leads to the best performance gives us the weakest feature (not included in the considered set). It is important to note that the discarded feature is not considered in the following steps. We iterate the process until there is only one remaining feature in the set, and use the following score for comparing systems: $F_{avg}(all) = \beta \cdot F_{avg}(G) + \gamma \cdot F_{avg}(B)$, where $F_{avg}(G)$ and $F_{avg}(B)$ are the averaged F scores for G and B label, respectively, when threshold varies from 0.300 to 0.975. The coefficients β and γ are determined with respect to the balance between G and B labels in the corpus: the minor class will be put more weight than the major one, since it is harder to predict. In **fr-en** system, the value (β, γ) is $(0.3, 0.7)$, and those in **en-es(WMT2013)** and **en-es(WMT2014)** system are analogous with the value of $(0.5, 0.5)$.

Formally, the SBS algorithm can be explained as follows. Let Y=$\{y_1, y_2, ..., y_N\}$ denote the set of all features, $X_k = \{x_j | j = 1, 2, ..., k\}; x_j \in Y; k = (0, 1, 2, ..., N)$ denote the subset of feature space of a specified size k. The pseudo-code of this algorithm can be written as:

Algorithm 1 Sequential Backward Selection

Input: $Y = \{y_1, y_2, ..., y_N\}$
Output: $X_k = \{x_j | j = 1, 2, ..., k\}; x_j \in Y; k = (0, 1, 2, ..., N)$

{**Initialize the feature set**}
$X_0 \leftarrow Y$
$k \leftarrow N$
while $k < N$ **do**
  {**Identify the worst feature**}
  $x^- = \arg\max_{x \in X_k} [F(X_k - x)]$
  {**Update feature set**}
  $X_{k+1} \leftarrow X_k - x^-$
  {**Continue the selection process**}
  $k \leftarrow k + 1$
end while
Output X_k.

Initially, the feature set X_0 involves all members, yet at each iteration rejects the weakest one. The removed feature is one whose absence helps to maximize the objective function F ($F_{avg}(all)$ in our case). The set will then be gradually reduced until its size reaches a specified

[1]http://research.cs.tamu.edu/prism/lectures/pr/pr_l11.pdf

threshold k. In our work, we investigate until the set remains one unique feature. This strategy enables us to sort the features in descending order of importance, as displayed in Table 4.1. Figure 4.1 shows the evolution of the WCE performance as more and more features are removed, and the details of best feature subsets yielding the highest objective function $F_{avg}(all)$. Next, we will analyze the usefulness of features for each system

No	fr-en	en-es (WMT2013)	en-es (WMT2014)
1	**Source POS**	**Source POS**	**Target POS**
2	Source word	**Occur in Google Translate (*)**	Longest target gram length (*)
3	**Target word**	**Nodes (*)**	Occur in mult. systems (*)
4	Backoff behaviour	**Target POS**	Target word
5	**WPP** any	**WPP** *any*	**Occur in Google Translate (*)**
6	**Target POS**	**Left source context**	**Source POS**
7	Constituent label (*)	**Right target context**	Numeric
8	**Left source context**	Numeric	Polysemy count (target) (*)
9	Null link	Polysemy count (target) (*)	Left source context
10	**Stop word**	Punctuation	**Right Target context**
11	Max (*)	**Stop word**	Constituent label (*)
12	**Right target context**	Right source context	Longest target POS gram length (*)
13	**Nodes (*)**	**Target word**	Punctuation
14	Punctuation	Distance to root (*)	Stop word
15	Polysemy count (*)	Backoff behaviour	Number of occurrences (*)
16	Longest source gram length (*)	Constituent label (*)	Left target context
17	Number of occurrences (*)	Proper name	Backoff behaviour
18	Numeric	Number of occurrences (*)	Polysemy count (source) (*)
19	Proper name	Min (*)	Source Word
20	Left target context	Max (*)	Proper Name
21	Min (*)	Left target context	Distance to root (*)
22	Longest target gram length (*)	Polysemy count (source)	Longest source gram length (*)
23	Right source context	Longest target gram length *	Right source context
24	Distance to root (*)	Longest source gram length (*)	Longest source POS gram length (*)
25	WPP *exact*	Source Word	-

Table 4.1 – The ordered feature lists in **fr-en**, **en-es(WMT14)** and **en-es (WMT13)** classifiers after the Feature Selection. It is worth noticing that feature sets are not exactly the same for three systems. All bold ones work efficiently in at least two systems. All ones with " $*$ " symbol are proposed by us

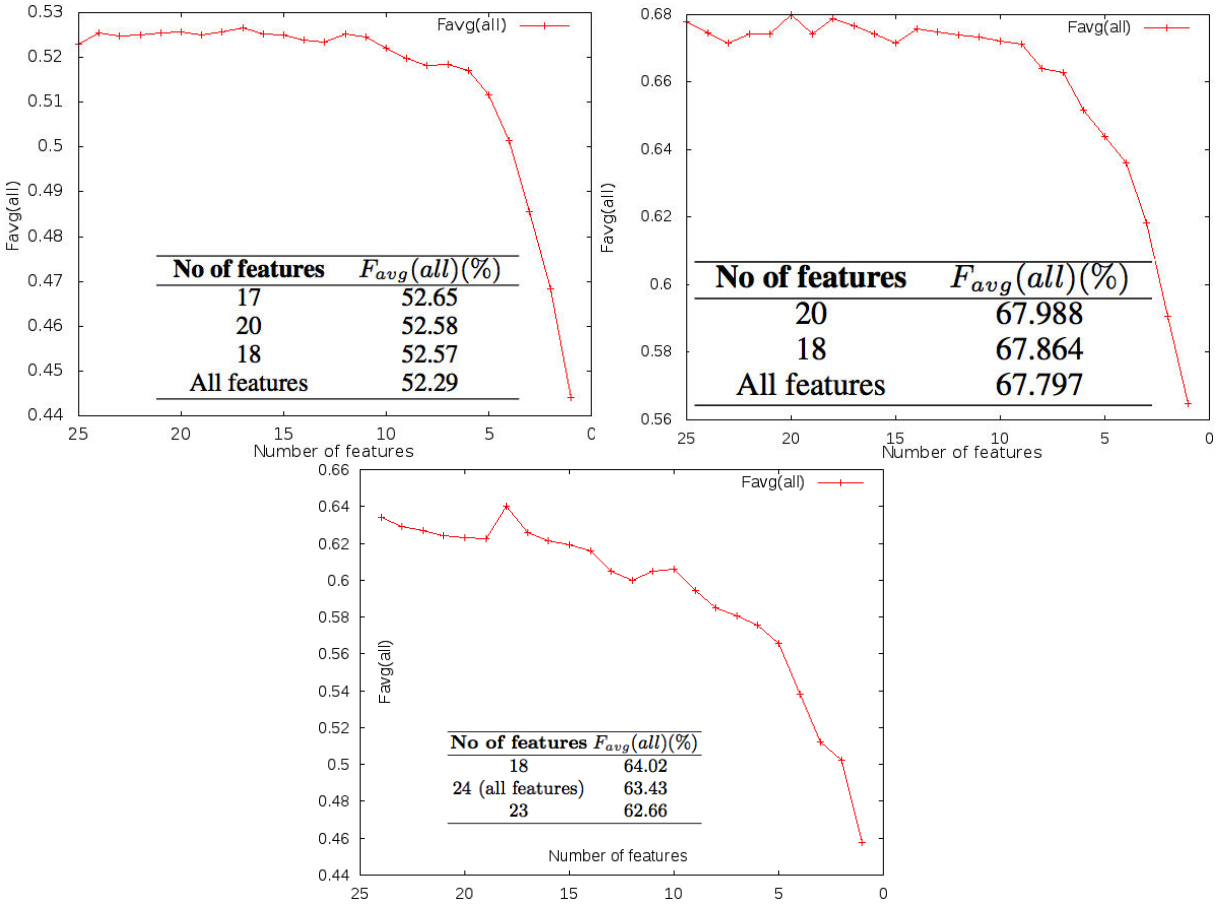

Figure 4.1 – Evolution of system performance ($F_{avg}(all)$) during Feature Selection process on **fr-en** (left, above) and **en-es (WMT2013)** (right, above) and **en-es(WMT2014)** (below) system

2.1 fr-en system

Table 4.1 hints that the system-based and lexical features seemingly outperform the other types in terms of usefulness, since in top 10, they contribute 8 (5 system-based + 3 lexical). However, 2 out of 3 syntactic features appear in the top 10 (*"Constituent label"* and *"Null Link"*), indicating that their role cannot be disdained. It is hard to conclude about the contribution of semantic feature because so far we have exploited only one representative of this type and it ranks 15 out of 25. Concerning our proposed features, only *Constituent Label*, *Max* and *Nodes* emerge in the upper part of the list, the remaining ones function ineffectively.

Furthermore, the observations in 10-best and 10-worst performing features suggest that features belonging to the word origin (source and target words, source and target POS, Stop word) perform effectively, meanwhile those from word statistical knowledge sources (target and source language models) are less beneficial. As shown in Figure 4.1 (**fr-en** part), the best-

performing subset of feature is top 17 with the $F_{avg}(all)$ value of 52.65%, which slightly exceeds the top 20 (52.58%) and top 18 (52.57%). The system trained on top 17 is therefore considered the optimal one obtained thanks to feature selection.

In addition, when the size of the feature set is small (from 1 to 7), we can observe sharply the growth of the system performance ($F_{avg}(all)$). Nevertheless the scores seem to saturate as the feature set increases from 8 up to 25, with almost no progression.

2.2 en-es (WMT 2013) system

In this system, the lexical information of word seems very useful for word quality prediction as they hold high positions in the list, peculiarly those in top 10: POS of both target and source sides, numerical and punctuation characteristics. Besides, surrounding words of the target token or of the source one also help when having 2 out of 4 representatives in top 10 as well. The outstanding system-based features of this system are *WPP* (at arbitrary position) and *Nodes*. Semantic type contributes one: the polysemy count for target side. On the contrary, three features encompassing *Left target context*, *Min* and *Longest target n-gram length* are proven weak with their bottom-most positions in the list. It is disappointing when the Language Models (in both sides), which is expected to bring useful information to the model, aggravate it in reality instead.

One commendable point is that the first-time-experimented feature "Occur in Google Translate" is the most prominent (rank 2) among our proposed features for this system, implying that such an online MT system can be a reliable reference channel for predicting word quality. In general, the proposed feature set yields a positive contribution to the entire set in this case: 3 of them work efficiently (in top 10), the other 5 individuals play the honorable role from rank 11 to rank 20, and only two remaining ones perform really poorly (rank 21 to 25).

Discarding 5 last features in the list brings the best system with $F_{avg}(all)$ of 67.988%, followed by top 18 ($F_{avg}(all)$= 67.864%)

2.3 en-es (WMT2014) system

The **en-es** (WMT2014) system confirms again the acuteness of information from referential translation means (references): both representatives of this type (*"Occurrence in multiple systems"* and *Occur in Google Translate*) can be found in high position of top 10 (rank 3 and 5). This suggests to exploit them in the future systems. Apart from these two above features, we have two more ones appear in top 10, including *"Longest target gram length"* and *"Pol-*

ysemy count" (target side). The role of new feature in this system is also very clear with 4 representatives in top 10, 4 others in the middle and only 3 last ones at the bottom.

The best possible classifier is trained over 18 first features in the list ($F_{avg}(all) = 64.02$), followed by top 24 ($F_{avg}(all) = 63.43$) and top 23 ($F_{avg}(all) = 62.66$).

2.4 Common Observations in All Systems

Along with appreciating the features working well in each specific system, it is very appealing to highlight also those that perform well in all (or multiple) systems, since they are the truly valuable prediction indicators for WCE. It can be seen on three lists (upper parts) of Table 4.1 that *"Source POS"*, *"Target POS"*, *"Target word"* and *"Right target context"* play clearly their positive roles in all lists as they emerge in the leading sets. From this fact, we are encouraged to further extract lexical and context alignment information of word in the future. Beside of this, we observe that *"Polysemy count"*, *"WPP (any)"* and *"Left source context"* should also be appreciated since they function fruitfully in two systems. On the opposite direction, we do not find any "extremely" weak feature appearing in the bottom of all lists. However, those extracted from LM are shown to be less beneficial in two systems (*"Longest target gram length"* and *"Longest source gram length"*).

Another observation is the progression of performance seems to saturate in all cases when the number of features increases (from 7 or 8 up to 24 or 25 individuals). In other words, trivial progressions are obtained when adding more features from these sizes. This phenomenon raises a hypothesis about our classifier's learning capability when coping with a large number of features, hence drives us to an idea of splitting our feature set into smaller subsets, constructing subsystems and applying Boosting for improving the classification scores, which is detailed in the next section.

3 Classifier Performance Improvement Using Boosting

If we build a number of "weak" (or "basic") classifiers by using subsets of our features and combining them in a reasonable way, should we get a final "composite" classifier with non-negligible performance improvement over each one? When deploying this idea, our hope is that multiple models can complement each other as one feature set might be specialized in a part of the data where the others do not perform very well. This is also the main idea of *Boosting* technique.

3.1 Overview of Boosting Method

Boosting is a machine learning meta-algorithm for reducing bias in supervised learning. It refers to a general and effective method of producing a very accurate "composite" predictors by combining basic and moderate individuals. Boosting algorithms work by learning iteratively "weak" classifiers respect to a distribution and then adding them to a final "strong" classifier. When they are added, they are typically weighted relied on their accuracies. Once having added a "weak" learner, the training instances will be also re-weighted: we put more weights for miss-classified data and less for those are correctly predicted. This act forces the base learner to pay more attention on the "hardest" examples. In other words, after each iteration, a new weak classifier will be generated and the current data will be weighted again with more focus on hard examples, so that the next classifier will handle better on these kind of data. Finally, the algorithm combines all weak classifiers (by simply taking a **weighted majority vote** of their predictions) to form the final composite one, which is believed to be much more accurate than each single classifier.

Boosting methods were addressed by a number of authors. Schapire (1990) proposes a polynomial-time method for converting a weak learning algorithm into one that achieves arbitrarily high accuracy. Later, Freund (1995) relies on this idea and improve it by combining a huge number of hypotheses, each of which is generated by training the given algorithm on a different set of example. Although they are proven optimal in a certain sense, the most prominent one is AdaBoost (Freund and Schapire, 1995) since it solves plenty of the practical difficulties of the previous ones. It is also the one we employ in this thesis. The mechanism of this algorithm is simply illustrated in Figure 4.2. The AdaBoost algorithm takes as input a training set $(x_1, y_1), (x_2, y_2), ..., (x_m, y_m)$ where each x_i belongs to the instance space X, and each label y_i is on the label set Y. We assume that $Y = \{-1, +1\}$ (-1 for "Bad" and +1 for "G"). AdaBoost then calls a underlying learning algorithm (such as Decision Tree, Neural Network, etc.) repeatedly in a series of iterations $t = 1, 2, ..., T$. One of the main ideas of this algorithm is to maintain a distribution or set of weights over the training set. The weight of this distribution on training example i on loop t is denoted as $D_t(i)$. Initially, all weights are set equal, but on each round, the weights of incorrectly classified examples are augmented so that the base learner is encouraged to pay more attention on the hard examples in the training set. The underlying learner's job is to find a *base classifier* $h_t : X \to R$ appropriate for the distribution. In the simplest case, the range of each h_t is binary, i.e., restricted to $\{-1, +1\}$; the base learner's job then is to minimize the *error*:

$$\epsilon_t = Pr_{i \sim D_t}[h_t(x_i) \neq y_i] \tag{4.1}$$

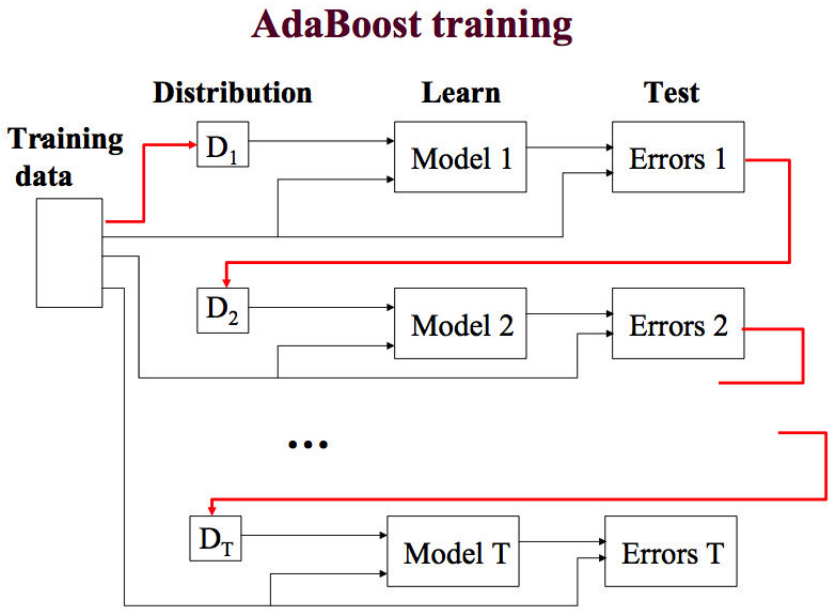

Figure 4.2 – AdatBoost algorithm

Once the base classifier h_t has been received, AdaBoost chooses a parameter $\alpha_t \in R$ that intuitively measures the importance that it assigns to h_t. For binary classifier , we typically set the value for α_t as follows:

$$\alpha_t = \frac{1}{2} ln \frac{1-\epsilon_t}{\epsilon_t} \tag{4.2}$$

as in the original description of AdaBoost given by Freund and Schapire (1995). The distribution D_t is then updated based on the errors made by the precedent classifier to make the up-coming generated one handles them better:

$$D_{t+1} = \frac{D_t(i)}{Z_t} c(x) \tag{4.3}$$

where

$$c(x) = \begin{cases} e^{-\alpha_t} & if\ y_i = h_t(x_i) \\ e^{\alpha_t} & if\ y_i \neq h_t(x_i) \end{cases} \tag{4.4}$$

therefore

$$D_{t+1} = \frac{D_t(i)}{Z_t} e^{-\alpha_t y_i h_t(x_i)} \tag{4.5}$$

The final or composite classifier H is a weighted majority vote of the underlying classifiers where α_t is the weight assigned to h_t:

$$H(x) = sign(\sum_{t=1}^{T} \alpha_t h_t(x)) \tag{4.6}$$

The pseudo-code of AdaBoost algorithm can be written as followings:

Algorithm 2 The boosting algorithm AdaBoost

Input: $(x_1, y_1), (x_2, y_2), ..., (x_m, y_m)$ where $x_i \in X, y_i \in Y = \{-1, +1\}$
Output: The composite classifier $H(x)$

{**Initialize the distributions**}
for $i = 1$ to m **do**
  $D_1(i) \leftarrow \frac{1}{m}$;
end for
for $t = 1$ to T **do**
  Train base classifier h_t using distribution D_t;
  Apply h_t over the test set to obtain the score (label) $h_t(x_j)$ for each word x_j;
  Choose $\alpha_t \in R$;
  {**Update the distributions**}
  for $i = 1$ to m **do**
    $D_{t+1}(i) \leftarrow \frac{D_t(i)}{Z_t} e^{-\alpha_t y_i h_t(x_i)}$;
    {**where Z_t is a normalization factor (chosen so that D_{t+1} will be a distribution)**}
  end for
end for
{**output the final classifier**}
$H(x) = sign(\sum_{t=1}^{T} \alpha_t h_t(x))$;

3.2 Boosting Training Data Preparation

The training data for Boosting technique consists of outputs (confidence score) from numerous base classifiers trained on subsets of all features. Figure 4.3 depicts how we build the training data to boost **fr-en**, **en-es (WMT2013)** and **en-es (WMT2014)** systems. First, we prepare 23 feature subsets $(F_1, F_2, ..., F_{23})$ to train 23 basic classifiers, in which:

- F_1 contains all features
- F_2 is the best performing set after selection (top 17 in **fr-en** and and top 20 in **en-es** systems)
- F_i $(i = \overline{3..23})$ contains 9 randomly chosen features.

Next, the N-fold ($N = 10$) cross validation is applied on the training set. We divide it into 10 equal subparts (S_1, S_2, ..., S_{10}). In the loop i $(i = \overline{1..10})$, S_i is used as the test set and the remaining data is trained with 23 feature subsets. After each loop, we obtain the results from 23 classifiers for each word in S_i. In the obtained data, the "probability of word to be

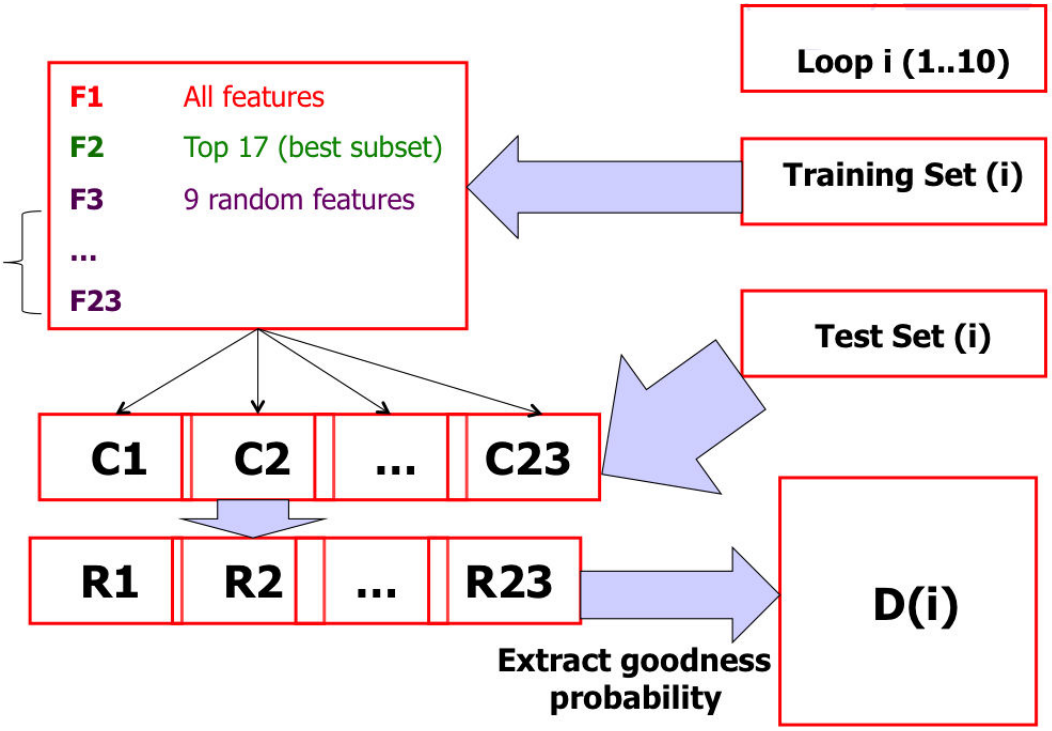

Figure 4.3 – Boosting data training algorithm

G label" (logged in each result file) is filtered to form part of the training data D_i. Once having 10 loops completed, the concatenation of all D_i gives us the total training set for our Boosting system. Therefore, the Boosting training file has 23 columns, each represents the output of one basic classifier for our training set. The detail of this algorithm is described as below:

Algorithm 3 Algorithm to build Boosting training data

Input: Training data $\{S_k\}$, Subparts of features $\{F_j\}$
Output: Boosting training data $\{D_i\}$

1: **for** $i = 1$ to 10 **do**
2: TrainSet(i) $\leftarrow \cup S_k$ $(k = \overline{1..10}, k \neq i)$
3: TestSet(i) $\leftarrow S_i$
4: **for** $j = 1$ to 23 **do**
5: Classifier $C_j \leftarrow$ Train *TrainSet(i)* with F_j
6: Result $R_j \leftarrow$ Use C_j to test S_i
7: Column $P_j \leftarrow$ Extract the *"probability of word to be G label"* in R_j
8: **end for**
9: Subpart D_i (23 columns) $\leftarrow \{P_j\}$ $(j = \overline{1..23})$
10: **end for**
11: Boosting training set $D \leftarrow \cup D_i$ $(i = \overline{1..10})$

After that, the Bonzaiboost toolkit[2] is used for building Boosting model. This is a general purpose machine learning program based on Decision Tree for building a classifier from text and/or attribute-value data. In the training command, we invoke AdaBoost algorithm and keep the number of iterations as 300. All other options are set by default.

In all systems, the Boosting test set is prepared from the usual test data. Firstly, we train 23 feature sets on the training set to obtain 23 classifiers. Then, we use them to test the usual test set, finally extract the 23 probability columns (like in the above pseudo code) to form the Boosting test set.

3.3 Results And Analysis

In the testing phase, we use Boosting models to classify the usual test sets and compare the F scores of "G" and "B" labels to those of corresponding "all feature" estimators. In case of **fr-en**, where the **dev** set is not available, $F(G)$ and $F(B)$ of Boosting classifier are averaged along the threshold variation on the test set, as the protocol applied on "all-feature" version, which ensures the fair comparison between them. With **en-es (WMT2013)** and **en-es (WMT2014)** systems, we tune the classification threshold on the **dev** set and then use the optimal value (that yields the best performance on this set) to determine the results on the **test** set. In other words, the results for them are all computed on these optimal thresholds. The entire scores for all predictors are combined and displayed in Table 4.2.

System			F(G) (%)	F(B) (%)
fr-en	test	Boosting	**87.02**	**40.65**
		all-feature	**87.07**	**37.76**
en-es (WMT2013)	dev	Boosting	**85.54**	**54.15**
		all-feature	**85.23**	**52.04**
	test	Boosting	**82.77**	**50.16**
		all-feature	**81.12**	**48.01**
en-es (WMT2014)	dev	Boosting	**75.09**	**54.11**
		all-feature	**75.01**	**53.47**
	test	Boosting	**74.73**	**44.42**
		all-feature	**74.65**	**44.10**

Table 4.2 – Comparison of the average F-score between CRF and Boosting systems, obtained on dev and test set

The scores suggest that using Boosting algorithm on our (CRF) classifiers' output is an efficient way to make them predict better. With **fr-en** system, on the one side, we maintain

[2]http://bonzaiboost.gforge.inria.fr/x1-20001

the already good achievement on G class (only 0.05% lost); on the other side we augment 2.89% the performance in B class. Since the error detection should be more appreciated than good translation prediction, this improvement in B class is very meaningful. More remarkably, on **en-es (WMT2013)** system, the improvements are obtained in both labels when dealing with both **dev** and **test** set, using the optimal classification threshold of 0.575. On the **dev** set, the gains are 0.31% and 2.11% for G and B, respectively; and they are even more commendable on the test set (1.65% and 2.15% improved). It is worth emphasizing this system since it is our official submission to the WMT13 Shared Task (Quality Estimation) and ***ranks first among participants*** for this task (We are also ***the winner in the multiple-label task***, in which the predicted labels are *"Good"*, *"Insertion"*, *"Substitution"* instead of binary labels). Consistently, this tendency repeats once again with **en-es (WMT2014)** system with the corresponding threshold of 0.700. When dealing with the **dev** set, slight improvement is observed for "G" (0.08%) yet a bit more significant for "B" (0.64%). And also, Boosting method helps to boost marginally the baseline system with the unseen **test** data, of about 0.08% and 0.32% for two labels. If we use the $F^*(all)$ (as used for Feature Selection) as a judgement criteria, it is straightforward that Boosting technique brings gains for all systems (all variants).

It is likely that Boosting enables different models to better complement one another, in terms of the later model becomes experts for instances handled wrongly by the previous ones. Another advantage is that Boosting algorithm weights each model by its performance (rather than treating them equally), so the strong models (come from all features, best performing set after selection, etc.) can make more dominant impacts than the others.

4 Summary

Optimizing WCE system performance can be conducted by various ways to strengthen separately or in parallel its components: features and ML model. With the first proposition, Feature Selection strategy, we aimed solely at the interoperability of features. The first achievement of this method is to reveal the best compatible indicators which yield the highest scores for each system, optimal than the "all-feature" candidates. Moreover, it helped to understand about truly valuable indicators for WCE by displaying those that worked effectively in all three systems, or in more than one system. On the contrary, by taking a look at the bottom of three lists, we could also observe the really useless and redundant ones that need serious considerations when being combined with others. Next, the ranking of each feature contributed to an analysis of the usefulness of each major category (system-based, lexical, syntactic and seman-

tic). For instance, in **fr-en** system, system-based and lexical features overwhelmed the others in best-performing top 10 or top 15. This therefore suggested more exploitations of features in the leading category. However, in this thesis, we stopped only at pointing out informative or useless features, and the best possible way they can be combined, but still left unanswered the explanation about why they performed fruitfully or poorly. This question might need in-depth statistical and linguistic analysis and would be a future work.

In **Boosting approach**, we would like to intervene in both features and ML model to improve the performance. From the entire set, multiple subparts were formed to build the weak models, which then generated the data to train the Boosting model. This training was performed by applying the base ML algorithm (Decision Tree) in order to get the "weak" classifier, and then using the prediction errors from the current classifier to build the next one. In other words, the future classifier was constructed taking into the consideration all the errors (wrongly classified words) of the current one, so it was expected to handle better these kind of data. Finally all weak classifiers were combined into a final one which was hopefully much stronger than each of them. In all **fr-en**, **en-es(WMT2013)** and **en-es(WMT2014)**, the method increased the error detection capability while maintained the already good achievement in predicting good translation. These results confirmed that if we take advantage of the portion handled successfully by each weak system and make them complement each other, we can constitute a composite "stronger" predictor.

So far, we proposed ideas to build and optimize the WCE systems. Once having them, the next appealing task is to exploit their application in many sectors of SMT. How far can WCE go to contribute to SMT, and by which manner? The first contribution will be investigated in the next chapter (Chapter 5).

Chapter 5

WCE for Improving Sentence Quality Estimation

1 Introduction

Although the quality of SMT output, in general, is getting better, the translation errors are still unavoidable, ranging from trivial to serious. Therefore, informing the target language's reader whether or not they can trust the specific translation is vital (to avoid the misunderstanding of the information conveyed in the source text). This chapter will not conduct an in-depth investigation about the sentence quality assessment techniques. Instead, we aim at exploring whether the word confidence score can play a positive role to help evaluating better the target sentence generated by MT system.

Firstly, we briefly summarize the basic background of SCE in Section 2, concerning the sentence level feature types, ML method can be used to train the predictor, as well as the way to annotate sentence with a quality label based on post-edition efforts. Next, we propose an appealing idea to examine the WCE's effectiveness in enhancing SCE. We build two SCE systems using (almost) identical common settings: the datasets (both **fr-en** and **en-es** language pairs), the ML technique and the methods of annotation, etc. The unique discrepancy between them is the feature set: one is trained by truly sentence level features while the other is trained by sentence level features which are synthesized from WCE's labels. Furthermore, we incorporate these two above systems by considering their decisions before determining the final results (for sentence's quality). This idea is reported in detail in Section 3 and Section 4. The comparison of performance of three systems (including the combined one) tells us more about the WCE usefulness (Section 5). The last section summarizes and draws some conclusions about our findings.

2 Sentence-level Confidence Estimation

In SCE, as mentioned in Chapter 2, the task is to judge the quality of a target sentence generated by SMT given its source sentence. This assessment yields various useful applications, including informing readers of the target language whether or not they trust the specific translation, selecting all eligible translations for human post-editing or information gathering; combining output from different MT systems; re-ranking the MT N-best list, etc.

It is well known that the features for SCE are gathered from various knowledge sources and toolkits. Most of them are continuous values, since they represent for the entire sentence and therefore (in many cases) are synthesized from the value of each word by averaging. Regarding on the resources used for extraction, they can be divided into two main types: *black-box features* (which can be built from only input sentence and MT hypothesis, and probably monolingual and bilingual corpora) and *glass-box features* (need additional components and resources from the translation process)

- **black-box features**: generally, this type exploits the simple and "shallow" characteristics of both source and target sides, such as: source and target sentence lengths and their ratio; source and target n-gram frequency in the corpus, etc. These features are essential in case where the in-depth ones are inaccessible (e.g. in some commercial MT systems). However, it is not always simple to obtain an efficient and accurate prediction model relying solely on such simple and basic features.
- **glass-box features**: this type includes all features that need more SMT resources during the translation process: N-best list, decoder's search graph, LM, phrase table, etc. Some examples of this type can be: SMT model score, phrase and word probabilities, alternative translations per source word, the degree to which phrases are translated in the same way throughout the N-best list. It is important to note that, beside the direct components of the translation process, there are plenty of other lexical, syntactic and semantic resources that can also be used for extracting sentence-level features, such as: parsers, semantic networks (WordNet, BabelNet, etc), dependency tree, etc.

Similar to WCE, SCE classifier needs a ML method to learn from these above features. Various techniques are proposed and experimented. Blatz et al. (2004) use *Multi-layer perceptrons* and *Naive Bayes* to train their 91 features. Meanwhile, Gamon et al. (2005) build a classifier based on *Support Vector Machine* (SVM) using linguistic features. *Linear Regression* is also exploited and proven to reach promising performance (Quirk, 2004).

If in WCE, annotations are made for each word; this task extends to conclude the entire

quality of the sentence in SCE. Although this is not a trivial and straightforward task, it can be quantified based on the post editing efforts (e.g. how many percent of words need to be corrected), combining with the subjective judgement of the editor on several criteria (understandability, consistency, coherence, fluency of the translation). WMT 2012 and WMT 2013 invite professional translators to perform this manual annotation, regarding to the following scoring scheme (guidelines) (Callison-Burch et al., 2012):

- [1] The MT hypothesis totally fails to convey the information (content) of source sentence and is incomprehensible. The editing efforts are therefore much more costly than re-translating from scratch.
- [2] The major part (about 50% to 70%) of the MT hypothesis needs to be corrected. Little information is accurately transferred. In order to be qualified for publishing, it takes significant editing efforts.
- [3] Readers can understand the gist of the hypothesis, but a portion (from 25% - near 50%) of text requires editing
- [4] Generally the hypothesis is clear and understandable. Only a small percentage (10% - near 25%) will be post-edited. Post-editing is cheaper than re-translating from scratch in this case.
- [5] The hypothesis is flawless and accurate. It conveys exactly what the source sentences says and no (or very little) mistake need to be fixed. It might not be the ideal translation, but reaches the publishable level.

This thesis does not focus on in-depth research about SCE (building and optimizing system), but investigate whether WCE information can help to improve its performance. In order to do that, firstly, we build a baseline SCE system using basic sentence-level features.

3 SCE Baseline System (System 1)

The baseline estimator is built and tested on the analogous datasets of WCE system, described in Chapter 3 for both **fr-en** and **en-es** language pairs.

- **fr-en**: 10000 training samples + 881 testing samples.
- **en-es (WMT2013)**: 753 training samples + 284 testing samples.

The ML method is CRF (WAPITI toolkit). Instead, for feature set, we take advantage of the feature extraction software provided by WMT 2012. The source code is free and available for download at: https://github.com/lspecia/QualityEstimation/blob/master/baseline_system. It analyses the source and the target files as well as the SMT training corpus to extract a total of 17 system-independent features for training the predictor. These features are:

- Number of tokens (words) in the source and target sentences
- Average source token length
- Average number of occurrences of the target word within the target sentence
- Number of punctuation marks in source and target sentences
- Language Model probability of source and target sentences using their language models.
- Average number of translations per source word in the sentence: as given by IBM 1 model that is thresholded so that $P(t|s) > 0.2$, and so that $P(t|s) > 0.01$ weighted by the inverse frequency of each word in the source side of the SMT training corpus
- percentage of unigrams, bigrams and trigrams in frequency quartiles (lower frequency words) and 4 (higher frequency words) in the source side of the SMT training corpus
- percentage of unigrams in the source sentence seen in the source side of the SMT training corpus

For training the baseline system, the annotated label for each sentence is a score ranging from 1 to 5, as used in WMT 2012. The difference is that we do not have the judgment sentence score from post-editors, yet rely on the TER score generated by TERp-A toolkit to determine it. In other words, we match each hypothesis against its post-edition by TERp-A and its TER score, which will then be used to annotate it. The mapping between this annotated score and TER score is following:

$$score(s) = \begin{cases} 5 & \text{if TER(s)} < 0.1 \\ 4 & \text{if } 0.1 < \text{TER(s)} <= 0.3 \\ 3 & \text{if } 0.3 < \text{TER(s)} <= 0.5 \\ 2 & \text{if } 0.5 < \text{TER(s)} <= 0.7 \\ 1 & \text{if TER(s)} > 0.7 \end{cases} \tag{5.1}$$

4 Proposed WCE-based SCE System (System 2)

In this section, we build another SCE classifier, but unlike the previous one, this classifier is trained on a set of features based totally on information of WCE outputs. In other words, we start from all quality labels for words predicted by WCE system, then synthesize them (by averaging) to form an unique score of the entire sentence. Specifically, we propose seven SCE features based on the WCE system as follows:

- The ratio of number of good words to total number of words. (1 feature)
- The ratio of number of good nouns to total number of nouns. The similar ones are also computed for other POS: verb, adjective and adverb. (4 features)
- The ratio of number of n consecutive good word sequences to total number of consecutive word sequences. Here, n=2 and n=3 are applied. (2 features)

All above features are extracted on both **fr-en** and **en-es** corpus, using the identical ML method and toolkit, and annotated labels. This means that both systems (SYS1 and SYS2) are comparable: they are build on almost analogous settings, except the features.

5 Experiments and Results

Once having the baseline SCE classifier (SYS1) and the SCE classifier based on WCE output (SYS2) built for both **fr-en** and **en-es (WMT 2013)** training set, we apply them to deal with the corresponding test sets. Furthermore, to observe the positive impact and effectiveness of the WCE output on SCE system, we design a third system (called **SYS1+SYS2**), which takes the results yielded by **SYS1** and **SYS2**, post-processes them and makes the final decision. Normally, for each sentence of the test set, **SYS1** and **SYS2** generate five probabilities for five integer labels it can be assigned, we then select the label with highest probability as the official result. Meanwhile, **SYS1+SYS2** collects probabilities coming from both systems for each label, then computes the average between them as its official probability of that label. Finally, the label with highest likelihood is assigned to this sentence.

Table 5.1 illustrates how **SYS1+SYS2** processes the results of **SYS1** and **SYS2** for one sentence. In case of the label "2", **SYS1** predicts the probability of 0.34632 and that value by **SYS2** is 0.12775. Therefore, the ultimate value that **SYS1+SYS2** computes for this sentence is: $(0.34632 + 0.12775)/2 = 0.23704$. Finally, each system concludes the official label based on the one with the highest value ("2" by **SYS1**, "3" by **SYS2** and "3" by **SYS1+SYS2**).

Labels	SYS1	SYS2	SYS1+SYS2
1	0.12135	0.21437	0.16786
2	0.34632	0.12775	0.23704
3	0.17829	0.45362	0.31596
4	0.29075	0.06754	0.17915
5	0.06329	0.13672	0.09999
Predicted label:	2	3	3

Table 5.1 – An example about the predicted label generated by **SYS1+SYS2** based on the output of **SYS1** and **SYS2**

The performances measured by MAE and RMSE for three systems of each dataset (**fr-en** and **en-es**) can be viewed on Table 5.2. The scores suggest that the WCE-based SCE features seem to be slightly beneficial compared to "pure" SCE features. In **fr-en** corpus, **SYS2** reduces 0.0386 MAE and 0.0358 RMSE points of **SYS1**. In case of **en-es(WMT2013)**, these improvements are 0.1021 and 0.1218 respectively. Notably, the most remarkable improvement occurs when the information of both systems is incorporated together. Two **SYS1+SYS2** systems in both metrics are the best-performing and boost significantly the pure SCE systems (the gains obtained amount to 0.1428 MAE in **en-es** and 0.1463 RMSE in the same language pair). We find some interesting conclusions that can be drawn from these results: firstly, the quality of separate words can hint something about the quality of the entire sentence. Secondly, when the information about word quality and sentence quality are embedded, they can complement each other and perform more satisfactory than when they stand alone.

System		MAE	RMSE
fr-en	SYS1	0.5584	0.9065
	SYS2	0.5198	0.8707
	SYS1+SYS2	**0.4835**	**0.8415**
en-es (WMT 2013)	SYS1	0.7056	1.0339
	SYS2	0.6035	0.9121
	SYS1+SYS2	**0.5628**	**0.8876**

Table 5.2 – Scores of 3 different SCE systems.

6 Conclusions

This chapter emphasized one important role of WCE in SMT: enhancing the performance of Sentence level Confidence Estimation. Starting from the intuition that the predicted word quality labels can tell something about the entire sentence's goodness, we proposed to build

and compare three different systems. The first one (**SYS1**) was a pure SCE assessor, trained by a set of sentence-level features (17 in total, provided by WMT 2012). The second one (**SYS2**) was built on the same datasets, identical language pairs and ML method, yet all features were synthesized from the output of WCE system. The last system (**SYS1+SYS2**) was actually a combination of the results yielded by two previous ones: using their average score as the score to determine the official label. These above predictors were trained on both **fr-en** and **en-es (WMT2013)** datasets . Dealing with the test sets, the WCE-based features were shown to slightly overwhelm pure SCE-based ones. However, the combination between them dramatically boosted the performance (by MAE and RMSE) above each individual. The results confirmed that WCE shows a positive impact in predicting the sentence's quality; especially when it can be properly and wisely integrated into the SCE system.

Chapter 6

WCE for SMT N-best List Re-ranking

1 Introduction

Despite tireless endeavors to improve, SMT models are still imperfect and SMT outputs still remain a big gap with human translations and are not yet able to be published as they are. As a consequence, a number of methods to improve MT hypotheses after decoding have been proposed in the past, such as: post-editing, re-ranking or re-decoding. Post-editing (Parton et al., 2012) is a human-inspired task where the machine post edits translations in a second automatic pass. In re-ranking (Duh and Kirchhoff, 2008; Nguyen et al., 2011; Zhang et al., 2006), more features are used along with the multiple model scores for re-determining the 1-best among the N-best candidates. Meanwhile, re-decoding process (Venugopal et al., 2007) intervenes directly into the decoder's search graph (e.g. adds more reward or penalty scores), driving it to a better path.

In this chapter, we investigate the first application of WCE in improving MT quality: re-ranking the N-best list. The method is conducted on **fr-en** SMT system. Generally, during the translation task, the decoder traverses through paths in its search space, computes the objective function values for them and outputs the one with the highest score as the best hypothesis. Besides, those with lower scores can also be generated in a so-called N-best list. The decoder's function consists of parameters from different models, such as translation, distortion, word penalties, reordering, language models, etc. In the N-best list, although the current 1-best beats the other ones in terms of model score, it might not be exactly the closest to the human reference. Therefore, adding more decoder independent features would be expected to raise up a better candidate (which is located currently somewhere else in the list). In this chapter, we build six additional features based on the labels predicted by our WCE system, then integrate them with the existing decoder scores for re-ranking hypotheses in the N-best list. More precisely, ***in the second pass***, our re-ranker aggregates over decoder and WCE-based weighted scores

and utilizes the obtained sum to select the best candidate. The novelty of our method lies in the following contributions: the correlation between WCE-based sentence-level scores and conventional evaluation scores (BLEU, TER, TERp-A) is first investigated. Then, we conduct the N-best list re-ranking over different WCE system performance levels: starting by a real WCE, passing through several gradually improved (simulated) systems and finally the "oracle" one. From these in-depth experiments, the role of WCE in improving MT quality via re-ranking N-best list is confirmed and reinforced.

In section 2, we summarize some outstanding approaches in N-best list re-ranking problem. Section 3 describes the correlation between the word confidence score and some other conventional metrics, followed by the WCE system construction and the proposed *re-ranking* features. The experiments along with the results and in-depth analysis of WCE scores' contribution (as WCE system gets better) are presented in Section 4 and Section 5.

2 *N*-best List Re-ranking: General Concept and Related Work

Conventionally, in SMT, for each source sentence f, the decoder applies all model scores (translation, LM, penalty etc.) to build the objective function, which helps to seek the optimal path (highest score) on its huge search space to be the output e^1 of the translation process. Beside this output, for many different goals, the decoder generates also $N-1$ other alternative hypotheses $\{e^2, e^3, ..., e^N\}$, having lower score than e^1 and being ranked according to this score. This list is known as the N-best list.

Since the SMT models are not ideal, the fact of having a sub-optimal sentence on the highest position of the N-best list occurs as a challenge for SMT improvement's endeavors. Table 6.1 illustrates for this phenomena. Compared to the reference (post-edition) given the source sentence, it is straightforward to realize that the third ranked hypothesis is better than the current 1-best, with only one "Shift" operator (*"association udf"* $\rightarrow$ *"udf association"*) is required to become the reference, although its model score is lower. This hints that the model scores might not be the reliable criteria for the output quality in some cases. This shortcoming motivates the idea of building a better N-best list by re-ranking the original one generated by the decoder. This task can be fulfilled by proposing brand new features which are not used during decoding and then combining them with model scores so that the better candidates in the old list have the opportunity to emerge as "optimal" translations.

Walking through various related work concerning this issue, we observe some prominent

Source (fr)	l'association udf hausse le ton et somme le nouveau centre de ne plus utiliser son sigle.
Post-edition (en)	the udf association increases the tone and commands the new centre to stop using its acronym.
Hypothesis e^1	the ***association udf increase*** the tone and ***after*** the new centre to stop using its acronym.
Hypothesis e^2	the association udf rise the tone and warn the new centre to stop using the acronym.
Hypothesis e^3	the ***association udf*** increases the tone and commands the new centre to stop using its acronym.
Hypothesis e^4	the association udf tone and increase the amount the new centre to use its acronym.
Hypothesis e^5	the association udf increase the tone and warn the new centre not to use its abbreviation.

Table 6.1 – Example of N-best list

ideas. The first attempt focuses on proposing additional Language Models. Kirchhoff and Yang (2005) train one word-based 4-gram model (with modified Kneser-Ney smoothing) and one factored trigram one, then combine them in a log-linear fashion with seven decoder scores for re-ranking N-best lists ($N = 2000$) of several SMT systems (translation from Finnish, Spanish, French into English). Their proposed LMs increase the translation quality of the baselines (measured by BLEU score) from 21.6 to 22.0 (Finnish - English), or from 30.5 to 31.0 (Spanish - English). Nevertheless, they observe that these significant improvements represent a small portion of the possible increase in BLEU score as indicated by the oracle results (29.8 and 37.4 for two above systems in that order). This fact suggests that better language models do not have a significant effect on the overall system performance unless the translation model is improved as well.

Meanwhile, Zhang et al. (2006) experiment a distributed LM where each server, among the total of 150, hosts a portion of the data and responses its client, allowing them to exploit an extremely large corpus (2.7 billion word English Gigaword) for estimating N-gram probability. The quality of their Chinese - English hypotheses after the re-scoring process by using this LM is improved 4.8% (from BLEU 31.44 to 32.64, oracle score = 37.48).

In one other direction, several authors propose to replace the current linear scoring function used by the decoder by more efficient functions. Sokolov et al. (2012a) learn their non-linear scoring function in a learning-to-rank paradigm, applying Boosting algorithm. Their gains on the WMT'{10, 11, 12} are shown modest yet consistent and higher than those based on linear scoring functions. More important, their experiments also show that under the tested conditions

with a tight integration with decoder, a pruning that bases on linear scoring function and a few standard features, we are not able to conclude that non-linear approach boost dramatically MT quality towards the oracle performance. Actually, non-linear approach should be employed earlier in the search space construction with an increased number of features.

Duh and Kirchhoff (2008) use Minimum Error Rate Training (MERT) (Och, 2003) as a weak learner and build their own solution, BoostedMERT, a highly-expressive re-ranker created by voting among multiple MERT ones. Their proposed model dramatically beats the decoder's log-linear model (43.7 vs. 42.0 BLEU) in IWSLT 2007 Arabic - English task. Nevertheless, this achievement is still far away from the oracle (BLEU 56.00). Furthermore, the authors also observe that BoostedMERT is somewhat sensitive to the size of the training set. For small training data, it improves the training score quite drastically, but the improvement per iteration fades out as data size increases.

Applying solely *goodness* (the sentence confidence) scores, Nguyen et al. (2011) obtain very consistent TER reductions (0.007 and 0.006) as well as BLEU improvements (0.4 and 0.2) on the dev and test set, respectively) after a 5-list re-ranking for their Arabic - English SMT hypotheses. The *"goodness"* score of a sentence is computed by simply averaging the confidence probabilities over all words. In fact, in order to obtain the reliable *"goodness"* values, the word confidence probabilities must be reliable as well, so the authors concentrate on building their WCE systems based on linguistic and context features. They reach 77.5% and 63.8% as F scores for *"G'"* and *"B"* labels.

This latter work is the one that is the most related to our contributions. However, the major differences are: (1) our proposed sentence scores *are computed based on word confidence labels*; and (2) we perform an in-depth study of the use of WCE for N-best reranking and assess its usefulness in a simulated interactive scenario.

3 Our Approach

Our approach can be expressed in three steps: investigate the potential of using word-level score in N-best list re-ranking, build the WCE system and extract additional features to integrate with the existing log-linear model.

3.1 Principal Idea

The principle idea is visualized in Figure 6.1. Inheriting the N-best list generated by the conventional 1-pass decoder, we first apply our WCE system to assign quality labels for all

words. After that, a *"Sentence-level score Builder"* will be invoked to synthesize from these labels to scores at sentence level (the way to compute them is further mentioned in Section 3.4), resulting in a total of six features per hypothesis. The next task is to combine them with 14 existing model scores (of 1-pass decoder); and the new objective function therefore contains 20 parameters. The weights of these parameters are initialized by default (e.g. Moses's default settings for 14 model scores and "all 1s" for six re-ranking scores) and then optimized using MERT and MIRA methods. After that, the *"N-best List Re-ranker"* takes the responsibility of calculating new scores for all candidates, updating weights after each iteration, and finally rank again the list in the descending order of their final scores (once having the optimization process finished). The effectiveness of our proposed method is measured by three metrics: BLEU, TER and TERp-A when matching the new 1-best with the post-edition. Before moving on the implementation, we verify in the next section whether the *"Word Quality Score"* can be a reliable indicator for the sentence quality assessment.

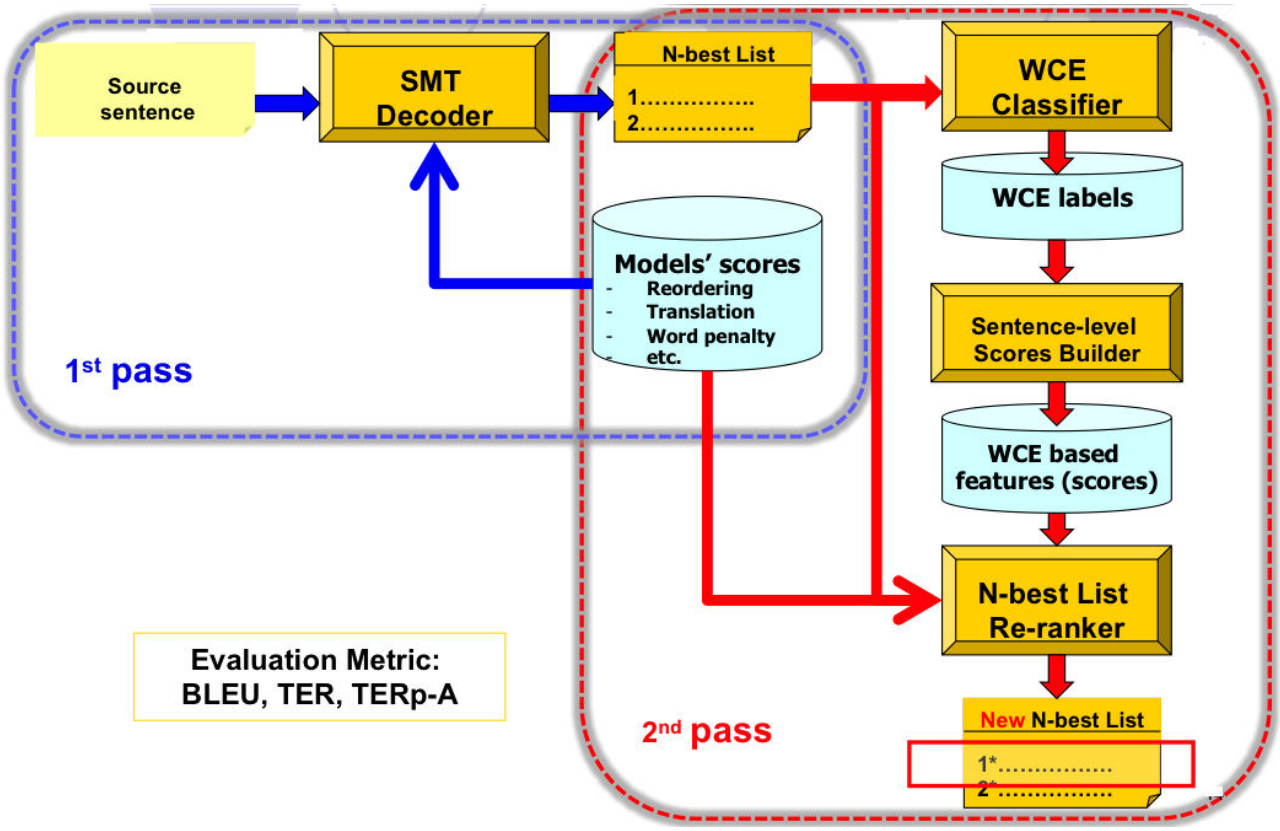

Figure 6.1 – Our approach in applying WCE in re-ranking the SMT N-best list

3.2 Investigating the correlation between "word quality" scores and other metrics

Firstly, we investigate the correlation between sentence-level scores (obtained from WCE labels) and conventional evaluation scores (BLEU (Papineni et al., 2002), TER and TERp-A (Snover et al., 2008)). For each sentence, a word quality score (WQS) is simply calculated by the ratio between the number of correctly translated words over the total number of words:

$$WQS = \frac{\#''G''(good)\ words}{\#words} \tag{6.1}$$

In other words, we are trying to answer the following question: can the high percentage of *"G"* (good) words (predicted by WCE system) in an MT output ensure its possibility of having a better BLEU and lower TER (TERp-A) value ? This investigation is a strong prerequisite for further experiments in order to check that WCE scores do not bring additional "noise" to the re-ranking process.

In this experiment, we compute WQS over our entire French - English data set (total of 10,881 1-best translations) for which WCE **oracle labels** are available (see Section 3.2 to see how they were obtained). For each hypothesis, the BLEU score is calculated by applying the sentence-level BLEU+1 (Nakov et al., 2012) between it and the post-edition. The TER and TERp-A score are done by TERp-A toolkit (Snover et al., 2008) along with the labels. The results are plotted in Figure 6.2, where the y axis shows the *"G"* (good) word percentage, and the x axis shows BLEU (1a), TER (1b) or TERp-A (1c) scores. It can be seen from Figure 6.2 that the major parts of points (the densest areas) in all three cases conform the common tendency: In Figure 6.2a, the higher "G" percentage, the higher BLEU is, leading to a high positive overall correlation score of 0.7874. The high negative correlations can also be observed in Figure 6.2b (Figure 6.2c), where the higher "G" percentage, the lower TER (TERp-A) is and the overall scores are -0.8687 and -0.7968, respectively. We notice some outliers, i.e. sentences with most or almost words labeled "good", yet still have low BLEU or high TER (TERp-A) scores. This phenomenon is to be expected when many (unknown) source words are not translated or when the (unique) reference is simply too far from the hypothesis. Nevertheless, the information extracted from oracle WCE labels seems useful to build an efficient re-ranker.

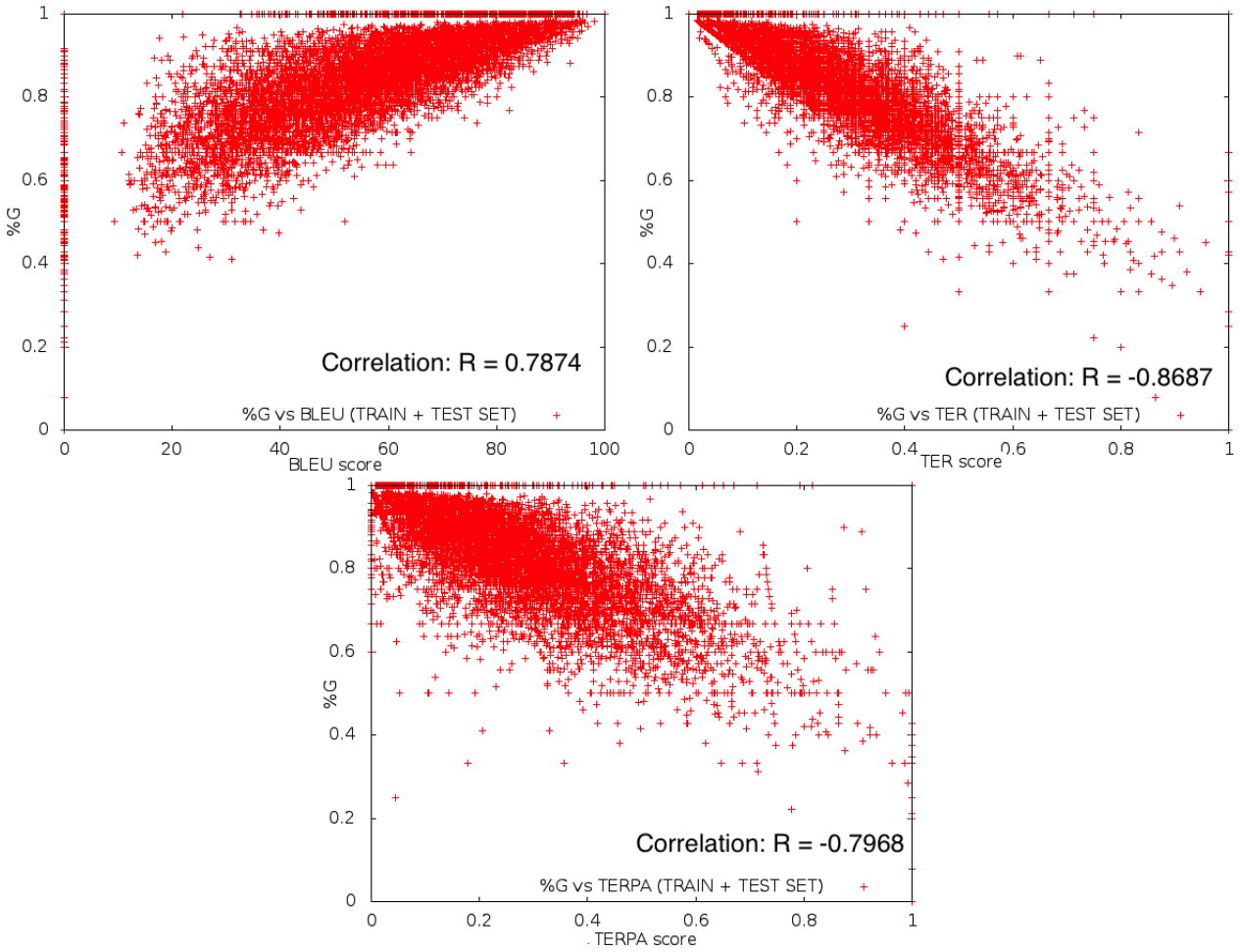

Figure 6.2 – The correlation between WQS in a sentence and its overall quality measured by : (a) BLEU, (b) TER and (c) TERp-A metrics

3.3 WCE System Preparation

The N-best list re-ranking using WCE needs all N translation candidates for each source sentence available along with predicted labels. However, due to the inability of obtaining these data in **en-es(WMT2013)** and **en-es(WMT2014)** systems, we will focus only on using **fr-en** system.

We use the entire feature set of the baseline **fr-en** SMT system (as described in Chapter 3) for experiments, along with one additional new feature: *"Occurrence in Google Translate"*[1]. In other words, the whole 25 feature types of the baseline system, plus this above one will be used to build the SMT system for our experiments. For the sake of understandability, we briefly list them as follows:

[1]We did not think about this feature at the time of building the baseline **fr-en** SMT system, it is added during our N-best list re-ranking studies

- Target Side: target word; bigram (trigram) backward sequences; number of occurrences
- Source Side: source word(s) aligned to the target word
- Alignment Context: the combinations of the target (source) word and all aligned source (target) words in the window ±2
- Word posterior probability
- Pseudo-reference (Google Translate): whether the current word appears in the pseudo reference or not[2]?
- Graph topology: number of alternative paths in the confusion set, maximum and minimum values of posterior probability distribution
- Language model (LM) based: length of the longest sequence of the current word and its previous ones in the target (resp. source) LM. For example, with the target word w_i: if the sequence $w_{i-2}w_{i-1}w_i$ appears in the target LM but the sequence $w_{i-3}w_{i-2}w_{i-1}w_i$ does not, the n-gram value for w_i will be 3.
- Lexical Features: word's Part-Of-Speech (POS); sequence of POS of all its aligned source words; POS bigram (trigram) backward sequences; punctuation; proper name; numerical
- Syntactic Features: Null link; constituent label; depth in the constituent tree
- Semantic Features: number of word senses in WordNet.

Once having the prediction model built with these above features, we apply it on the test set (881 x 1000 best = 881000 sentences) and get needed WCE labels. According to the Precision (Pr), Recall (Rc) and F-score (F) shown in Table 6.2, our WCE system reaches very promising and appealing performance in predicting *"G"* label, and acceptable for *"B"* label. These labels will be used to calculate our proposed features (section 3.3).

Label	Pr(%)	Rc(%)	F(%)
Good (G)	84.36	91.22	**87.65**
Bad (B)	51.34	35.95	**42.29**

Table 6.2 – Pr, Rc and F for "G" and "B" labels of our fr-en WCE system

3.4 Proposed Re-ranking Features

Since the scores resulted from the WCE system are for words, we have to synthesize them in sentence level scores for integrating with the 14 decoder scores. Six proposed scores involve:

[2]This is our first-time experimented feature and it does not appear in (Luong et al., 2013a)

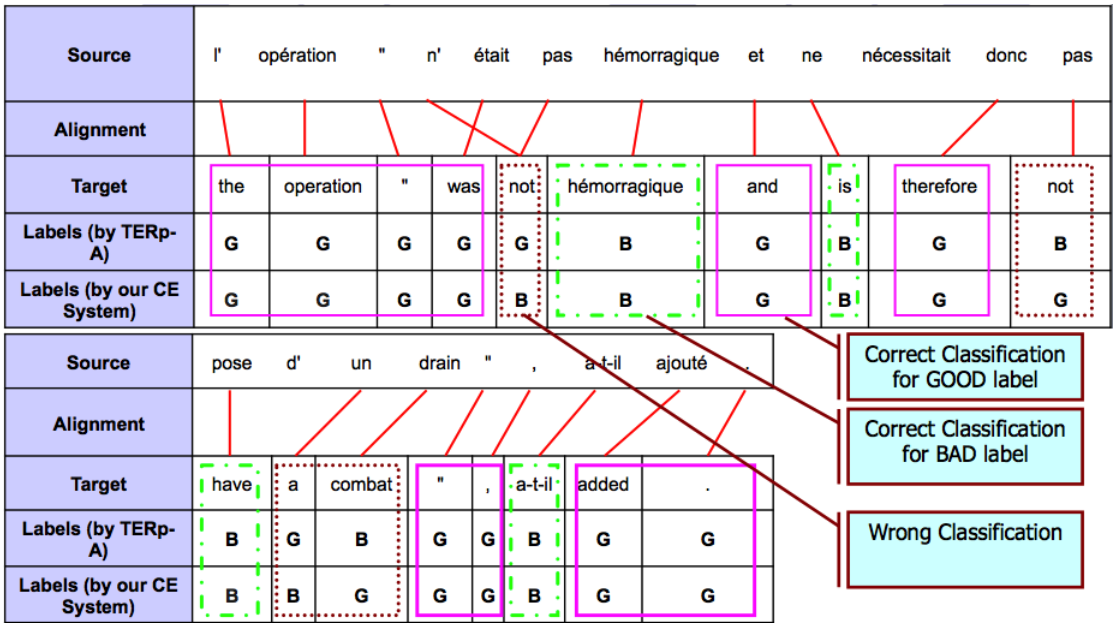

Figure 6.3 – Example of our WCE classification results for one MT hypothesis

- The ratio of number of good words to total number of words. (1 score)
- The ratio of number of good nouns (verbs) to total number of nouns (verbs)[3]. (2 scores)
- The ratio of number of n consecutive good word sequences to the total number of consecutive word sequences ; n=2, n=3 and n=4. (3 scores)

For instance, in case of the hypothesis in Figure 6.3: among the total of 18 words, we have 12 labeled as *"G"*; and 7 out of 17 word pairs (bigram) are labeled as *"GG"*, etc. Hence, some of the above scores can be written as:

$$\begin{aligned} \frac{\#good\ words}{\#words} &= \frac{12}{18} = 0.667 \\ \frac{\#good\ bigrams}{\#bigrams} &= \frac{7}{17} = 0.4118 \\ \frac{\#good\ trigrams}{\#trigrams} &= \frac{3}{16} = 0.1875 \end{aligned} \tag{6.2}$$

With the features simply derived from WCE labels and not from CRF model scores (i.e. the probability $p(G)$, $p(B)$) , we expect to stretch the evaluation up to the "oracle" setting, where the users validate a word as "G" or "B" without providing any confidence score.

[3]We decide not to experiment with adjectives, adverbs and conjunctions since their number can be 0 in many cases.

4 Experiments

4.1 Experimental Settings

As described in Chapter 3, our SMT system generates 1000-best list for each source sentence, and among them, the best hypothesis was determined by using the objective function based on 14 decoder scores, including: 7 reordering scores, 1 language model score, 5 translation model scores and 1 word penalty score. After combining with six proposed re-ranking scores, we have a total of 20 scores per hypothesis. Table 6.3 gives an example of how these scores are organized in the data for the hypothesis: *"the operation " was not hémorragique and is therefore not have a combat , " a-t-il added . "* in the N-best list data file. On this table, the red part represents decoder scores while the blue part contains re-ranking scores.

12 ||| the operation " was not hémorragique and is therefore not have a combat , " a-t-il added . ||| **d: 0 -5.30284 0 0 -6.60458 0 0 lm: -277.497 tm: -50.4321 -58.0075 -14.9024 -16.6772 10.9989 w: -18 goodness: 0.6666667 noun: 0.33333334 verb: 0.5 bigram: 0.4117647 trigram: 0.1875 quadgram: 0.13333334** ||| -350.125 ||| 0-1=0-1 2=2 3-5=3-4 6=5 7=6 8-9=7 10-12=8-9 13=10 14-15=11 16=12 17-18=13-14 19=15 20-21=16-17 ||| 0=0 1=1 2=2 3=4 4=3 5=4 6=5 7=6 8=7 9=-1 10=8 11=9 12=-1 13=10 14=-1 15=11 16=12 17=14 18=13 19=15 20=16 21=17 ||| 0=0 1=1 2=2 3=4 4=3,5 5=6 6=7 7=8 8=10 9=11 10=13 11=15 12=16 13=18 14=17 15=19 16=20 17=21

Table 6.3 – Example of one hypothesis with 20 scores (14 model scores + 6 proposed scores)

Initially, all six additional WCE-based scores are weighted as 1.0. Then, two optimization methods: MERT and Margin Infused Relaxed Algorithm (MIRA) are applied to optimize the weights of all 20 scores of the re-ranker.

- MERT (Och, 2003) This is an algorithm for optimizing the un-smoothed error count of the feature function. It starts at a random point in the K-dimentional parameter space and try to find a better scoring point in the parameter space. To do this, it makes a one-dimensional line minimization along the pre-defined directions by optimizing one parameter while keeping all others fixed. The algorithm can start from different initial parameter values. It is proven to be much faster and more stable than the grid-based line optimization method (which suffers from the shortcoming of balancing the grid size and the optimal solution guarantee).
- MIRA (Margin Infused Relaxed Algorithm) (Crammer et al., 2006): This is an online version of the large-margin training algorithm for structured classification that has been successfully used for dependency parsing, joint-labeling or chunking. The principle idea of

this algorithm is to keep the norm of the updates to the weight vector as small as possible, and considering a margin at least as large as the loss of the incorrect classification. A larger error means a larger distance between the score of the correct and incorrect translations. It is important to note that only a number of *"active"* features satisfying some specific constraints are kept and updated, unlike the offline training where all possible features are required to be extracted and selected in advance.

In both methods, we carry out a **2-fold cross validation** on the N-best test set. In other words, we split our N-best test set into two equivalent subsets: $S1$ and $S2$. Playing the role of a development set, $S1$ will be used to optimize the 20 weights for re-ranking $S2$ (and vice versa). Finally two result subsets (all the new 1-best sentences after re-ranking process on $S1$ and $S2$) are concatenated for evaluation. The evaluation is based on three metrics: BLEU, TER and TERp-A. To better acknowledge the impact of the proposed re-ranking scores, we re-rank the N-best list not only by using our *"real"* WCE system labels, but also by using the *"oracle"* (or ideal WCE) labels (henceforth called "WCE scores" and "oracle scores", respectively). To summarize, we experiment with the three following systems:

- **BL**: Baseline SMT system with 14 decoder scores (as mentioned above).
- **BL+WCE**: Baseline with 14 score + 6 *"real"* WCE scores
- **BL+OR**: Baseline with 14 scores + 6 *"oracle"* scores (for simulating an interactive scenario where the user's validation on the current hypothesis helps the system to generate a better one).

4.2 Results and Analysis

The translation quality of **BL**, **BL+WCE** and **BL+OR**, optimized by MERT and MIRA method are reported in Table 6.4. Meanwhile, Table 6.5 depicts in details the number of sentences in the two integrated systems which outperform, remain equivalent or degrade the baseline hypothesis (when matching against the references, measured by TER).

It can be observed from Table 6.4 that the integration of **oracle scores** significantly boosts the MT output quality, measured by all three metrics and optimized by both methods employed. We gained 5.79 and 4.72 points in BLEU score, by MERT and MIRA (respectively). With TER, **BL+OR** helps to gain 0.03 point in both two methods. Meanwhile, in case of TERp-A, the improvement is 0.05 point for MERT and 0.03 point for MIRA. It is worthy to mention that the possibility of obtaining such oracle labels is definitely doable through a human-interaction scenario (which could be built from a tool like PET (Post-Editing Tool) (Aziz et al., 2012) for

Systems	MERT			MIRA		
	BLEU	**TER**	**TERp-A**	**BLEU**	**TER**	**TERp-A**
BL	52.31	0.2905	0.3058	50.69	0.3087	0.3036
BL+OR	**58.10**	**0.2551**	**0.2544**	**55.41**	**0.2778**	**0.2682**
BL+WCE	52.77	0.2891	0.3025	51.01	0.3055	0.3012
Oracle BLEU score	**BLEU=60.48**					

Table 6.4 – Translation quality of the baseline system (using only decoder scores) and that with additional scores from "real" WCE or "oracle" WCE system

System	MERT		
	Better	Equivalent	Worse
BL+WCE	159	601	121
BL+OR	**517**	261	103

Table 6.5 – Quality comparison (measured by TER) between the baseline and two integrated systems in details (How many sentences are improved, kept equivalent or degraded, out of 881 test sentences?

instance). In such an environment, ***once having the hypothesis produced by the first pass (translation task)***, the human editor could simply click on words considered as bad (B), the other words being implicitly considered as correct (G). This validation helps the system to generate the better hypothesis.

Breaking down the analysis into sentence level, as described on Table 6.5, **BL+OR** (MERT) yields nearly 59% (517 over 881) better outputs than the baseline, 24% of equivalent and only 17% of worse ones. Furthermore, Table 6.4 shows that in case of our test set, optimizing by MERT is pretty more beneficial than MIRA. The reason for this should be investigated in the future work.

Compared to the *"oracle scores"*, the contribution of *"real"* WCE scores seems more modest: **BL+WCE** marginally increases BLEU scores of **BL** (0.46 gain in case of optimizing by MERT and 0.32 by MIRA). The gains measured by TER and TERp-A are also trivial: only 0.0014 TER and 0.0033 TERp-A points reduced (MERT); and those by MIRA are almost analogous: 0.0032 and 0.0024 (in that order). Since the progressions are negligible, we raise a question about the signification of the real WCE contribution to the MT quality: do these small gain really come from WCE scores or simply from the optimizer instability or other factors? In order to verify this, we estimate the p-value between BLEU of **BL+WCE** system and BLEU of baseline **BL** relying on Approximate Randomization (AR) method (Clark et al., 2011) which indicates if the improvement yielded by the optimized system is likely to be generated again by some random processes (randomized optimizers). The p-value can vary widely from statistically insignificant (at $p > 0.05$) to highly significant ($0.01 < p < 0.05$) or strongly significant ($p <=$

0.01).

After various optimizer runs, we selected randomly from the large pool a total of $\boldsymbol{n}$ optimizer outputs and perform the AR test to verify the significance. The AR tests are conducted thanks to MUTLVAL[4] toolkit. Finally, all the corresponding p-values to different values of $\boldsymbol{n}$ (from 2 to 5) are reported on Table 6.6 .

Number of randomly selected optimizers (n=)	***p*-value (MERT)**
2	0.00
3	0.01
4	0.01
5	0.01

Table 6.6 – The p-values between **BL+WCE** and **BL** with different number of optimizer runs

This results (small p-values in all cases) reveals that the improvement yielded by **BL+WCE** is significative although small, and they are originated from the contribution of WCE scores, not by any optimizer variance.

4.3 Comparison to Oracle BLEU Score

For more insightful understanding about WCE scores' acuteness, we make a comparison with the optimal $BLEU$ score that could be obtained from the N-best list. We apply the sentence-level BLEU+1 (Nakov et al., 2012) metric over candidates in the list. Why $BLEU+1$ but not $BLEU$ metric is applicable in sentence level? According to these authors, we can explain by taking a look at the definition of $BLEU$ metric:

$$BLEU = BP.(\prod_{n=1}^{N} p_n)^{\frac{1}{N}} \tag{6.3}$$

$BLEU$ consists of two components: the brevity penalty (BP) and precision component (PC), the geometric mean of n-gram precisions $p_n, 1 \leq n \leq N$. The BP can be defined as following:

$$BP = \begin{cases} 1 & \text{if } c > r \\ exp(1 - \frac{r}{c}) & \text{if } c \leq r \end{cases} \tag{6.4}$$

where c is the length of the candidate, and r is the effective reference corpus length. Since BLEU is defined at the corpus level, p_n, r and c are sums over all sentences of the corpus.

[4]Source code to carry out the AR test : http://github.com/jhclark/multeval.

- $p_n = \frac{\sum_i m_{in}}{\sum_i h_{in}}$: where m_{in} is the number of n-gram matches between a translation and the references for sentence i, and h_{in} is the number of n-grams in the hypothesis.

- $c = \sum_i c_i$: where c_i is the length of the MT output for sentence i

- $r = \sum_i r_i$: where r_i is the reference length for sentence i. In case of multiple references, this is the closest reference sentence length. It is straightforward to see that when BLEU score is applied for sentence, p_n, c and r become $p_n = \frac{m_{in}}{h_{in}}$, $c = c_i$ and $r = r_i$.

With the above definition, $BLEU$ scores can be flawlessly applied at corpus level, since its large size ensures that the n-gram will always be found. Nevertheless, at sentence level (the one that we desire to use BLEU for), this causes a problem which might unfortunately bias the *"really bad"* and *"pretty good"* sentences. In Formula 6.3, if at least one p_n is zero, the whole score will be zero. And this possibility often occurs, for instance in case where there is no 4-gram match is found. That would lead to a fact that the hypothesis which has no matches at all will possess the same BLEU score with one having unigram, bigram, trigram matches yet no 4-gram, resulting in a very inaccurate sentence quality evaluation. Therefore, a strategy to solve this problem is to use an add-one smoothed version of $BLEU$, called $BLEU+1$, in which p_n is redefined as follows: $p_n = \frac{m_{in}+1}{h_{in}+1}$. Beside this, we also apply the add-one smoothing for the length of the reference translation, i.e. use $r = r_i + 1$. The main idea is to smooth the PC and the BP consistently, thus maintaining the balance between them: if we assume an extra n-gram in the reference, then we also should assume that the reference contains an extra word.

After calculating the above sentence-level $BLEU + 1$ for N-best candidates in the list, we are able to select the one with highest score to be the *"oracle"* translation of the source sentence. Once having all of them for the entire test set, we combine them to form the new 1-best translations. The oracle BLEU score is finally calculated by comparing this new version of translation and the post-editions, and the score obtained is **60.48**.

This score accounts for a fact that the simulated interactive scenario (**BL+OR**) lacks only 2.38 points (in case of MERT) to be optimal and clearly overpass the baseline (8.17 points below the best score). This result also reveals that with the help of *real* WCE labels, we are able to choose a very small portion of better candidates among N-bests to replace the current decoder-best hypothesis. However, the improvement, although small yet significant, encourages us to investigate and analyze further about WCE scores' impact, supposing WCE performance is getting better. More in-depth analysis is presented in the next section.

5 Further Understanding of WCE scores role in *N*-best Re-ranking via Improvement Simulation

We think it would be very interesting and useful to answer the following question: do WCE scores really effectively help to increase MT output quality when the WCE system is getting better and better? To do this, our proposition is as follows: firstly, by using the oracle labels, we filter out all wrongly classified words in the test set and push them into a temporary set, called **T**. Then, we correct randomly a percentage (25%, 50%, or 75%) of labels in **T**. Finally, the altered **T** will be integrated back with the correctly predicted part (by the WCE system) in order to form a new "simulated" result set. This strategy results in three "virtual" WCE systems called **"WCE+N%"** (N=25, 50 or 75), which use 14 decoder scores and 6 "simulated" WCE scores. The steps in detail are described in Figure 6.4. In this process, **Filter** splits correctly predicted labels from wrong ones, and **Corrector(N%)** plays the job of replacing randomly N% of wrong labels by the corresponding oracle ones. It is important to note that, when the corrected file is combined with the correct part to form the new result file, the position (index) of labels in each sentence is kept intact as in the original file, to ensure the accuracy of re-ranking feature values.

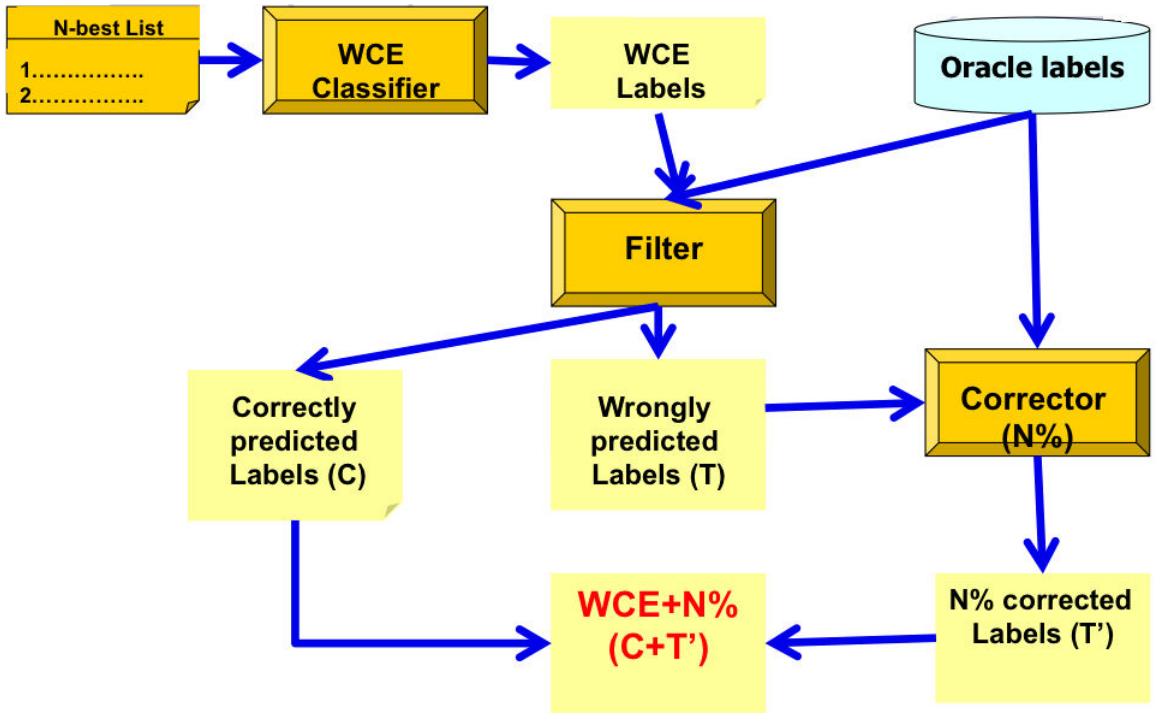

Figure 6.4 – The simulation of WCE's improvement, by replacing a part of wrongly predicted labels by corresponding oracle ones.

Table 6.7 shows the performance of these **WCE+N%** systems in term of the average weighted F scores (%). From each of the above systems, the whole experimental setting is

System	F("G")	F("B")	Overall F
WCE+25%	89.87	58.84	63.51
WCE+50%	93.21	73.09	76.11
WCE+75%	96.58	86.87	88.33
Oracle labels	100	100	100

Table 6.7 – The performances (Fscore) of simulated WCE systems

identical to what we did with the original **BL+WCE** and oracle **BL+OR** systems: six scores are built and combined with existing 14 system scores for each hypothesis in the N-best list. After that, MERT and MIRA methods are invoked to optimize their weights, and finally the reordering is performed thanks to these scores and appropriate optimal weights. The translation quality measured by BLEU, TER and TERp-A after re-ranking using **"WCE+N%"** (N=25,50,75) can be seen also in Table 6.8. The number of translations which outperform, keep intact and decline in comparison to the baseline are shown in Table 6.9 for MERT optimization. We note that all obtained scores fit our guess and expectation: the better performance WCE

Systems	MERT			MIRA		
	BLEU	TER	TERp-A	BLEU	TER	TERp-A
BL	52.31	0.2905	0.3058	50.69	0.3087	0.3036
BL+OR	**58.10**	**0.2551**	**0.2544**	**55.41**	**0.2778**	**0.2682**
BL+WCE	52.77	0.2891	0.3025	51.01	0.3055	0.3012
WCE + 25%	53.45	0.2866	0.2903	51.33	0.3010	0.2987
WCE + 50%	55.77	0.2730	0.2745	53.63	0.2933	0.2903
WCE + 75%	56.40	0.2687	0.2669	54.35	0.2848	0.2822
Oracle BLEU score	**BLEU=60.48**					

Table 6.8 – Translation quality of the three *"simulated"* WCE systems (green zone)

system reaches, the clearer its role in improving MT output quality. Diminishing 25% of the wrongly predicted words leads to a gain 0.68 point (by MERT) and 0.32 (by MIRA) in BLEU score, and help to select 28.72% (253 out of 881) better candidate than the current 1-bests. More significant increases of BLEU 3.00 and BLEU 3.63 (MERT) can be achieved when prediction errors are cut off up to 50% and 75%; with the percentage of better hypotheses generated are 36.32% and 52.33% in that order. Figure 6.5 presents an overview of the results obtained and helps us to predict the SMT quality improvements expected as the WCE systems are more and more improved in the future.

System	MERT		
	Better	Equivalent	Worse
BL+WCE	159	601	121
BL+OR	**517**	261	103
WCE+25%	253	436	192
WCE+50%	320	449	112
WCE+75%	461	243	177

Table 6.9 – Quality comparison (measured by TER) between the baseline and three *"simulated"* systems (green zone) in details (How many sentences are improved, kept equivalent or degraded, out of 881 test sentences?

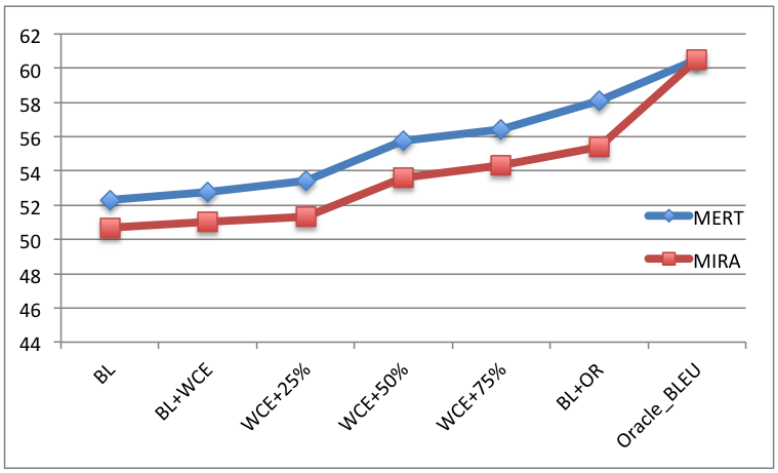

Figure 6.5 – Comparison of the performance of various systems: the integrations of WCE features, which the quality increases gradually, lead to the linear improvement of translation outputs.

To end up this section, we show on Table 6.10 several examples where WCE scores drive SMT system to better reference-correlated hypothesis. In the first example, the baseline generates the hypothesis in which the source phrase *"pour sa part"* remains untranslated. On the contrary, **WCE+50%** overcomes this drawback by resulting in a correct translation phrase: *"for his part"*. The latter translation needs only one edit operation (shift for *"Bettencourt-Meyers"*) to become its reference, then outperforms the hypothesis. In example 2, **BL+OR** selects the better hypothesis, in which the phrases *"creuser la tombe"* and *'"pme du secteur"* are translated into *"digging the grave"* and *"medium-sized businesses"*, respectively, better than those of the baseline (*"deepen the grave"* and *"smes in the sector"*).

Example 1 (from WCE+50%)	
Source	Pour sa part , l' avocat de Françoise Bettencourt-Meyers , Olivier Metzner , s' est félicité de la décision du tribunal .
Hypothesis (Baseline SMT)	The lawyer of *Bettencourt-Meyers* Françoise , Olivier Metzner , welcomed the court 's decision .
Hypothesis (SMT+WCE scores)	*For his part ,* the lawyer of *Bettencourt-Meyers* Françoise , Olivier Metzner , welcomed the court 's decision .
Post-edition	*For his part ,* the lawyer of Françoise Bettencourt-Meyers , Olivier Metzner , welcomed the court 's decision .
Example 2 (from BL+OR)	
Source	Pour l' otre , l' accord risque " de creuser la tombe d' un très grand nombre de pme du secteur dans les 12 prochains mois " .
Hypothesis (Baseline MT)	For the otre the agreement is likely to *deepen the grave* of a very large number of *smes in the sector* in the next 12 months " .
Hypothesis (SMT+WCE scores)	For the otre agreement , the risk " *digging the grave* of a very large number of *medium-sized businesses* in the next 12 months " .
Post-edition	For the otre , the agreement risks " *digging the grave* of a very large number of *small- and medium-sized businesses* in the next 12 months " .

Table 6.10 – Examples of MT hypothesis before and after reranking using the additional scores from WCE+50% (Example 1) and BL+OR (Example 2) system

6 Conclusions

So far, the word confidence scores have been exploited in several applications, e.g. post-editing, sentence quality assessment or multiple MT-system combination, yet very few studies (except Nguyen et al. (2011)) propose to investigate them for boosting MT quality. Thus, in this chapter, we proposed a number of features extracted from a WCE system and combined them with existing decoder scores for re-ranking N-best lists. Due to the WCE classifier's limitations in predicting translation errors ("B" label), WCE scores ensured only a modest improvement in translation quality over the baseline SMT. Nevertheless, further experiments about the simulation of WCE performance suggested that such types of score would contribute dramatically if they are built from an accurate WCE system. They also showed that with the help of an "ideal" WCE, the MT system reached quite close to its most optimal possible quality. These scores were totally independent from the decoder, they can be seen as a way to introduce lexical, syntactic and semantic information (used for WCE) in a SMT pipeline.

Chapter 7

WCE for SMT Search Graph Redecoding

1 Introduction

Conventionally, during SMT's decoding, the decoder travels over all complete paths on the Search Graph (SG), seeks those with cheapest costs and backtracks to read off the best translations. Although these winners beat the rest in model scores, they might not have the highest quality with respect to the human references (the most crucial criteria to judge MT quality). In reality, there are many cases in which the top-ranked sentence is sub-optimal, as shown in Chapter 6. Beside N-best list re-ranking, the integration of more system-independent scores **directly** into the current transition cost of the edges of the SG would be another potential solution to redirect the decoder to another optimal path, then improve the MT quality. This problem is also known as the re-decoding. The principal discrepancy between ***N*-best list re-ranking** and **Re-decoding** is that the former method re-calculates the scores for N-best candidates to order it again, meanwhile the latter one updates the scores for not only them, but also all those containing *at least one word* in the N-best list. Therefore, the former one offers much more selection, yet requests more computational complexity than the latter one.

In this chapter, we exploit WCE outputs (labels or probabilities) in the second pass of decoding to enhance the MT quality. By using the confidence score of each word in the N-best list to update the cost of SG hypotheses containing it, we hope to "reinforce" or "weaken" them depending on word quality estimation. After the update, new best translations are re-determined using updated costs. In the experiments on our *real WCE scores* and *ideal (oracle) ones*, the latter significantly boosts one-pass decoder by 7.87 BLEU points, meanwhile the former yields an improvement of 1.49 points for the same metric.

In Section 2, we introduce the general concept as well as several approaches dealing with this issue, followed by the description of how the SG is organized and represented in Section 3. Section 4 details the proposition of integrating WCE based scores into the nodes of SG, the way

to calculate them, the way to update the 1-pass SG and seek for the new hypothesis, followed by an example to illustrate. Next, we conduct the settings for all experimental re-decoders in Section 5 in both real and oracle scenarios. The results, analysis and comparisons with re-ranking approach are depicted in Section 6.

2 Re-decoding: General Concept and Related Work

Beside plenty of commendable achievements, the conventional one-pass SMT decoders are still not sufficient yet in yielding human-acceptable translations (Venugopal et al., 2007; Zhang et al., 2006). Therefore, a number of methods to enhance them are proposed, such as: post-editing, re-ranking or re-decoding, etc. Post-editing (Parton et al., 2012) is in fact known to be a human-inspired task where the machine post edits translations in a second automatic pass. In re-ranking (Duh and Kirchhoff, 2008; Nguyen et al., 2011; Zhang et al., 2006), more features are integrated with the existing multiple model scores for re-selecting the best candidate among N-best list. Meanwhile, the re-decoding process intervenes directly into the decoder's search graph (SG), driving it to the optimal path (cheapest hypothesis).

The two-pass decoder has been built by several discrepant ways in the past. Zens and Ney (2006) introduce n-gram and sentence length posterior probabilities and demonstrate their usefulness in the 2-pass decoding. The idea to calculate n-gram posterior probabilities is essentially similar to the WPP (Word Posterior Probabilities - see Chapter 2), that we sum up the sentence probabilities for each occurrence of an n-gram. Meanwhile, the sentence length posterior probability of a specific target sentence length I is computed by summing the posterior probabilities only over all target sentences with this length in the list . These authors create the 2-pass decoder using these additional scores on the Chinese - English NIST task (from 2002 to 2005) and get a significant and promising improvement of the translation quality. Successively adding higher (than 1) order n-gram posterior probabilities, the quality (in BLEU) increases consistently across all evaluation sets (up to 1.2%). More interesting, adding the sentence length posterior probability boosts the overall quality to 1.5% on the development set, and between 1.1% to 1.6% on the test set. Finally, another interesting property of this method is that they do not require any additional knowledge source.

Meanwhile, Tromble et al. (2008) present a new approach for multiple-pass decoding: performing Minimum Bayes-Risk Decoding (MBR) on the translation lattices (a compact representation for very large N-best lists of MT hypotheses and their likelihoods). MBR aims at finding the candidate hypothesis that has at least expected loss under the probability model.

According to this approach, firstly all the set of n-gram that occur in the *evidence lattice* (used for computing the Bayes-risk) are extracted. Then, after computing the posterior probability for each of them, we intersect each n-gram with an appropriate weight to form the weighted copy of this lattice. Finally, the best path is obtained by seeking on that new word lattice. Their experiments show that the Lattice MBR decoder yields moderate yet consistent gains over N-best MBR decoding on Arabic - English, English - Chinese and Chinese - English translation tasks.

In another attempt, Venugopal et al. (2007) do a first pass translation without LM, but use it to score the pruned search hyper-graph in the second pass of a probabilistic synchronous context-free grammar based (PSCFG-based) SMT. Actually, this approach is built upon the concept of a second pass but uses the N-gram LM to search for alternative, better translations. The first pass corresponds to a severe parameterization of Cube Pruning considering only the first-best (LM integrated) chart item in each cell while maintaining unexplored alternatives for second-pass consideration. The second pass allows the integration of long distance and flexible history n-gram LMs to drive the search process, rather than simply using such models for hypothesis re-scoring. The results suggest that the syntactic grammar's score are more useful than hierarchical grammar (1.5 BLEU higher) and the syntax based translation quality is very robust to differences in the number of N-gram LM options explored.

Distinct from the above work, this thesis concentrates on a second automatic pass where the costs of all hypotheses in the decoder's SG containing the words of the N-best list will be updated regarding the word quality predicted by Word Confidence Estimation (Ueffing and Ney, 2005) (WCE) system. In single-pass decoding, the decoder searches among complete paths (i.e. those who cover all source words) for obtaining the optimal-cost ones. Essentially, the hypothesis cost is a composite score, synthesized from various SMT models (reordering, translation, LMs etc.). Although the N-bests beat other SG hypotheses in term of model scores, there is no certain clue that they will be the closest to the human references. As the reference closeness is the users' most pivotal concern on SMT decoder, this work establishes one second pass where model-independent scores related to word confidence prediction are integrated into the first-pass SG to re-determine the best hypothesis. Inheriting the first pass's N-best list, the second one involves three additional steps:

- Firstly, apply a WCE classifier on the N-best list to assign the quality labels ("Good" or "Bad") along with the confidence probabilities for each word.

- Secondly, for each word in the N-best list, update the cost of all SG's hypotheses containing it by adding the update score to their current cost (see Section 3.2 for update

score detailed definitions).

- Thirdly, search again on the updated SG for the cheapest-cost hypothesis and trace backward to form the new best translation.

Basically, this initiative originates from an intuition that all parts of hypotheses corresponding to correct (predicted) words should be appreciated while those containing wrong ones must be weakened. At the end of the second pass, the costs of complete hypotheses are updated and the best translation will be selected again with the same protocol as in the first pass. The use of novel decoder-independent and objective features like WCE scores is expected to raise up the better candidate, rather than accepting the current sub-optimal one. The new decoder can therefore use both *real* and *oracle* word confidence estimates.

3 Search Graph Structure

The SMT decoder's Search Graph (SG) can be roughly considered as a "vast warehouse" storing all possible hypotheses generated by the SMT system during decoding for a given source sentence. In this large directed acyclic graph, each hypothesis is represented by a path, carrying all nodes between its begin and end ones, along with the edges connecting adjacent nodes. One hypothesis is called *complete* when all the source words are covered and *incomplete* otherwise. Starting from the empty initial node, the SG is gradually enlarged by expanding hypotheses during decoding. To avoid the explosion of search space, some weak hypotheses can be pruned or recombined. In order to facilitate the access and the cost calculation, each hypothesis **H** is further characterized by the following fields (we can access the value of the field f of hypothesis **H** by using the notion $f(H)$):

- **hyp**: hypothesis ID
- **stack**: the stack (ID) where the hypothesis is placed, also the number of foreign (source) words translated so far.
- **back**: the backpointer pointing to its previous cheapest path.
- **transition** : the cost to expand from the previous hypothesis (denoted by **pre(H)**) to this one.
- **score**: the cost of this hypothesis, computed by: $score(H) = score(pre(H)) + transition$.
- **out**: the last output (target) phrase. It is worth to accentuate that **out** can contain multiple words.

- **covered**: the source coverage of **out**, represented by the start and the end position of the source words translated into **out**.
- **forward**: the forward pointer pointing to the cheapest outgoing path expanded from this one.
- **f-score**: estimated future cost from this partial hypothesis to the complete one (end of the SG).
- **recombined**: the pointer pointing to its recombined[1] hypothesis.

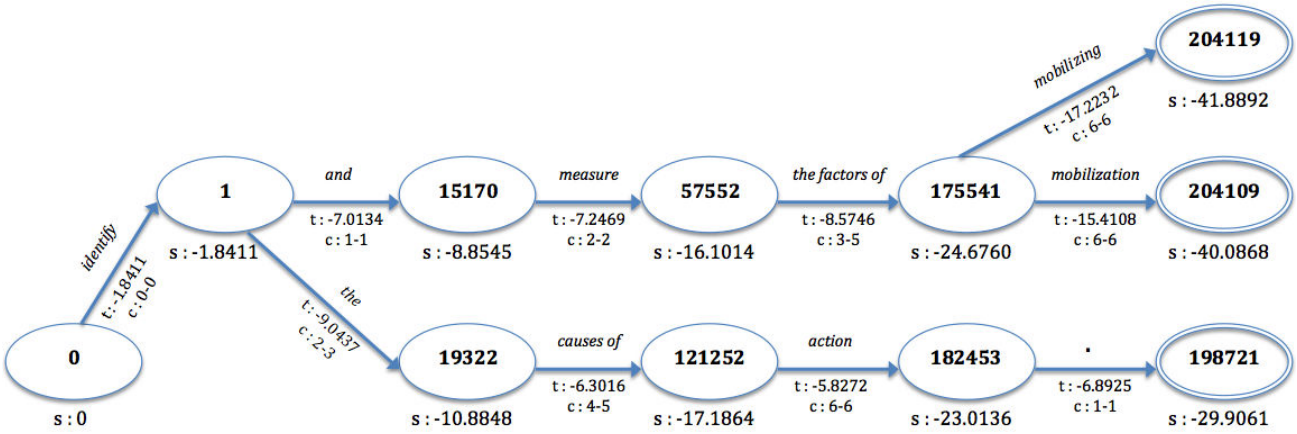

Figure 7.1 – An example of search graph representation

Figure 7.1 illustrates a simple SG generated for the source sentence: ***"identifier et mesurer les facteurs de mobilization"***. The attributes **"t"** and **"c"** refer to the transition cost and the source coverage, respectively. Hypothesis **175541** is extended from **57552**, when the three words from 3rd to 5th of the source sentence (*"les facteurs de"*) are translated into *"the factors of"* with the transition cost of -8.5746. Hence, its cost is: $score(175541) = score(57552) + transition(175541) = -16.1014 + (-8.5746) = -24.6760$. Three rightmost hypotheses: **204119**, **204109** and **198721** are complete since they cover all source words. Among them, the cheapest-cost one[2] is **198721**, from which the model-best translation is read off by following the track back to the initial node **0**: ***"identify the causes of action ."***.

[1] In the SG, sometimes we recombine hypotheses to reduce the search space in a risk-free way. Two hypotheses can be recombined if they agree in (1) the source word covered so far (2) the last two target words generated, and (3) the end of the last source phrase covered.

[2] It is important to note that the concept **cheapest cost hypothesis** means that it has the highest model's score value. In other words, the higher the model score value, the "cheaper" the hypothesis is.

4 Our Approach: Integrating WCE Scores into SG

In this section, we present the idea of using additional scores computed from WCE output (labels and confidence probabilities) to update the SG. We also depict the way update scores are defined. Finally, the detailed algorithm followed by an example illustrates the approach.

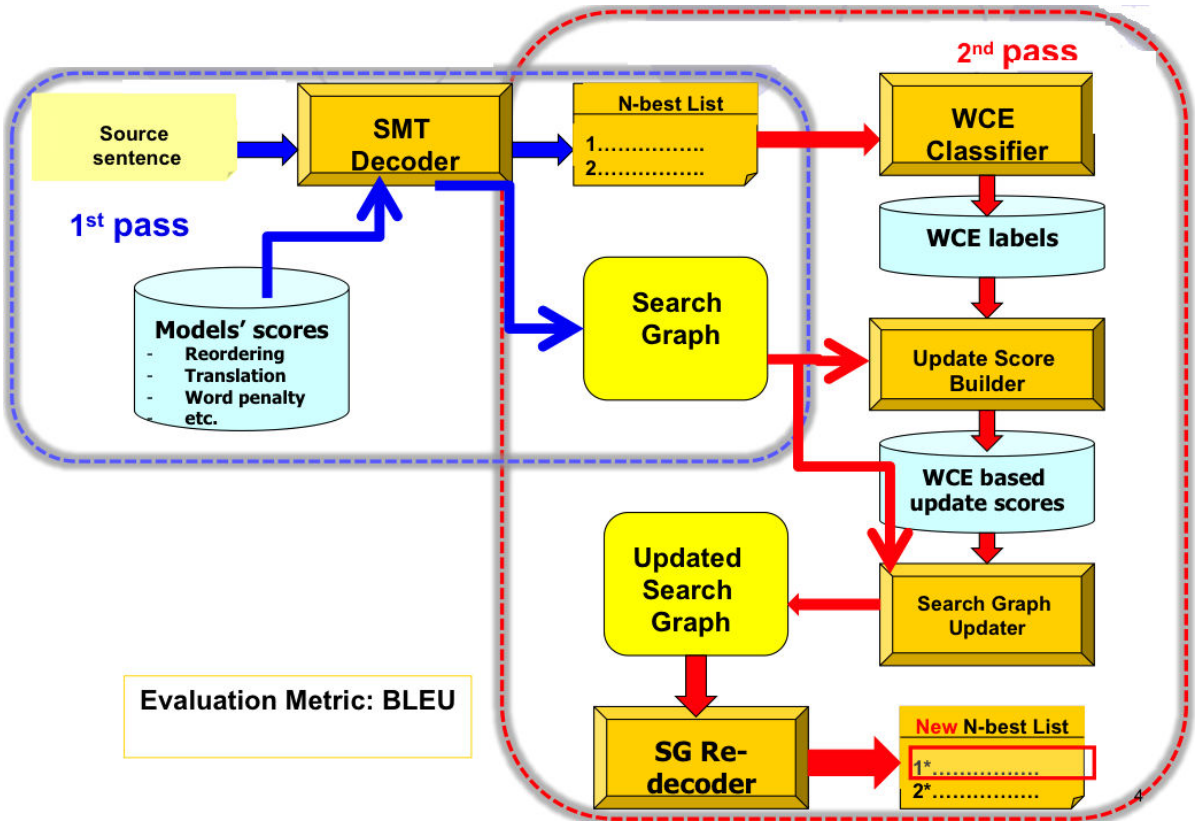

Figure 7.2 – Re-decoding Process

As shown in Figure 7.2, firstly, we inherit the N-best list and the SG outputted by the first pass of decoding. Next, the WCE system is used to generate the quality label and the probabilities (for each class "G", "B") of all words. The reward or penalty scores for word will be calculated right after by using several definitions (which are detailed in Section 4.2). These scores are used to alter the cost of all hypotheses containing this word, resulting in a new SG. The cheapest hypothesis will be selected as new output of the MT system.

4.1 Principal Idea

We assume that the decoder generates N best hypotheses $T = \{T_1, T_2, ..., T_N\}$ at the end of the first pass. Using the WCE system (which can only be applied to sequences of words - and not

directly to the search graph - that is why N best hypotheses are used), we are able to assign the *j-th* word in the hypothesis T_i, denoted by t_{ij}, with one appropriate quality label, c_{ij} (e.g. *"G"* (Good: no translation error), *"B"* (Bad: need to be edited)), followed by the confidence probabilities ($P_{ij}(G), P_{ij}(B)$ or $P(G), P(B)$ for short). Then, the second pass is carried out by considering every word t_{ij} and its labels and scores $c_{ij}, P(G), P(B)$. Our principal idea is that, if t_{ij} is a *positive* (good) translation, i.e. $c_{ij} =$ "G'' or $P(G) \approx 1$, all hypotheses $H_k \in SG$ containing it in the SG should be "rewarded" by reducing their cost. On the contrary, those containing *negative* (bad) translation will be "penalized". Let $reward(t_{ij})$ and $penalty(t_{ij})$ denote the reward or penalty score of t_{ij}. The new **transition** cost of H_k after being updated is formally defined by:

$$transition'(H_k) = transition(H_k) + \begin{cases} reward(t_{ij}) & \text{if } t_{ij} = good\ translation \\ penalty(t_{ij}) & \text{if } otherwise \end{cases} \tag{7.1}$$

The update finishes when all words in the N-best list have been considered. We then re-compute the new score of complete hypotheses by tracing backward via back-pointers and aggregating the **transition cost** of all their edges. Essentially, the re-decoding pass reorders SG hypotheses in term of the more *"G"* words (predicted by WCE system) they contain, the more cost reduction will be made and consequentially, the more opportunity they get to be admitted in the N-best list. The re-decoding performance depends largely on the accuracy of confidence scores, or in other words, the WCE quality.

It is vital to note that, during the update process, we might face a phenomena that the word t_{ij} (corresponds to the same source words) occurs in different sentences of the N-best list. In this case, for the sake of simplicity, we process it only at its first occurrence (in the highest rank sentence) instead of updating the hypotheses containing it multiple times. In other words, if we meet the exact t_{ij} once again in the next N-best sentence(s), no further score update will be done in the SG.

4.2 Update Score Definitions

Defining the update scores is obviously a nontrivial task as there is no correlation between WCE labels and the SG costs. Furthermore, we have no clue about how proportional the SMT model and WCE (penalty or reward) scores should share in order to ensure that both of them will be appreciated. In this chapter, we propose several types of update scores, deriving from the global or local cost.

4.2.1 Definition #1: Global Update Score

In this type, an unique score derived from the cost of the current best hypothesis H^* (by the first pass) is used for all updates. This score is normalized by the number of words in this hypothesis. Additional coefficient(s) α (β) will be used to emphasize the level of impact that the *reward* or *penalty* score will make on the corresponding node in the SG. In this work, we propose to compute this score by two ways: (a) exploiting WCE labels $\{c_{ij}\}$ to determine where the score is a *reward* or *penalty* one; *or* (b) using WCE confidence probabilities $\{P(G), P(B)\}$, WCE labels being left aside.

Definition 1a:

$$penalty(t_{ij}) = -reward(t_{ij}) = \alpha * \frac{score(H^*)}{\#words(H^*)} \tag{7.2}$$

Where $\#words(H^*)$ is the number of target words in H^*, the positive coefficient α accounts for the impact level of this score on the hypothesis's final cost and can be optimized during experiments using, for instance, CONDOR toolkit. Here, $penalty(t_{ij})$ gets negative sign (since $score(H^*) < 0$) and will be added to the transition cost of all hypotheses containing t_{ij} in case where this word is labelled as *"B"*; whereas $reward(t_{ij})$ (same value, opposite sign) is used in the other case.

Definition 1b:

$$\begin{aligned} update(t_{ij}) &= \alpha * P(B) * \frac{score(H^*)}{\#words(H^*)} - \beta * P(G) * \frac{score(H^*)}{\#words(H^*)} \\ &= (\alpha * P(B) - \beta * P(G)) * \frac{score(H^*)}{\#words(H^*)} \end{aligned} \tag{7.3}$$

Where $P(G)$, $P(B)$ ($P(G) + P(B) = 1$) are the probabilities of "Good" and "Bad" class of t_{ij}. The positive coefficients α and β can be tuned in the optimization phase. In this definition, the fact that $update(t_{ij})$ is **a reward** ($reward(t_{ij})$) or **a penalty** ($penalty(t_{ij})$) will depend on t_{ij}'s goodness. Indeed, we have: $update(t_{ij}) = reward(t_{ij})$ if $update(t_{ij}) > 0$, which means: $\alpha * [1 - P(G)] - \beta * P(G) < 0$ (since $score(H^*) < 0$), therefore $P(G) > \frac{\alpha}{\alpha+\beta}$. In other words, if the confidence score (to be a "G" label) of the word is greater than this threshold, the corresponding hypothesis will be rewarded. On the contrary, if $P(G)$ is under this threshold, $update(t_{ij})$ takes a negative value and therefore becomes a penalty. We consider an example by taking a look again at the SG on Figure 7.1. The current 1-best hypothesis (*"identify the causes of action ."*) has 6 words with the cost of -29.906. If we apply the coefficient $\alpha = 0.75$, the reward or penalty scores are the same in any word t_{ij} of the SG and take the value of:

$$penalty(t_{ij}) = -reward(t_{ij}) = 0.75 * \frac{-29.906}{6} = -3.73825 \tag{7.4}$$

In another example, we assume that the coefficients $\alpha = 0.75$, $\beta = 0.30$ are used in **Definition 1b**. We consider the hypothesis **182453** and assume that the probabilities for the word *"action"* are : $P(G) = 0.2$, $P(B) = 0.8$. So, the update score computed for the hypothesis **182453** can be written as:

$$\begin{aligned} update(t_{ij}) &= \alpha * P(B) * \frac{score(H^*)}{\#words(H^*)} - \beta * P(G) * \frac{score(H^*)}{\#words(H^*)} \\ &= (0.75 * 0.8 - 0.30 * 0.2) * \frac{-29.9061}{6} = -2.69 \end{aligned} \tag{7.5}$$

Since this update score takes the opposite sign, it is actually the penalty score. It is straightforward to see that all words with $P(G) > \frac{0.75}{0.75+0.30} = 0.71$ will bring the rewards for all hypotheses containing it, and vice versa.

4.2.2 Definition #2: Local Update Score

The update score of each (local) hypothesis H_k derives from its current transition cost, even when they cover the same word t_{ij}. Unlike **Definition 1**, the update score for each word in this definition differs from each other and depends on the hypothesis where it belongs to. Here, we exploit also both WCE labels and probabilities for calculating scores. Two sub-types are defined as follows:

Definition 2a:

$$penalty(t_{ij}) = -reward(t_{ij}) = \alpha * transition(H_k) \tag{7.6}$$

Where $transition(H_k)$ denotes the current transition cost of hypothesis H_k and the coefficient α plays the same role as that in **Definition 1a**. In case of the hypothesis **182453** in Figure 7.1, with = 0.75, transition cost is -5.8272 and *"action"* classified as a "B" label, the penalty equals to : $0.75 * (-5.8272) = -4.3704$.

Definition 2b:

$$\begin{aligned} update(t_{ij}) &= \alpha * P(B) * transition(H_k) - \beta * P(G) * transition(H_k) \\ &= (\alpha * P(B) - \beta * P(G)) * transition(H_k) \end{aligned} \tag{7.7}$$

Where $transition(H_k)$ denotes the current transition cost of hypothesis H_k, and the meanings of

coefficient α (**Definition 2a**) or α, β (**Definition 2b**) are analogous to those of **Definition 1a** (**Definition 1b**), respectively. The update score value for the hypothesis **182453** in Figure 7.1, with $= 0.75$, $\beta = 0.30$, transition cost is -5.8272, probabilities of "action" : $P(G) = 0.2$, $P(B) = 0.8$, is: $(0.75 * 0.8 - 0.30 * 0.2) * (-5.8272) = -3.1466$.

4.3 Re-decoding Algorithm

Algorithm 4 Using WCE labels in **SG** decoding

Input: $SG = \{H_k\}$, $T = \{T_1, T_2, ..., T_N\}$, $C = \{c_{ij}\}$
Output: $T^{'} = \{T_1^{'}, T_2^{'}, ..., T_N^{'}\}$

{**Step 1: Update the Search Graph**}
$Processed \leftarrow \emptyset$
for T_i in T **do**
  for t_{ij} in T_i **do**
    $p_{ij} \leftarrow$ position of the source words aligned to t_{ij}
    if $(t_{ij}, p_{ij}) \in Processed$ **then**
      continue; {ignore if t_{ij} appeared in the previous sentences}
    end if
    $Hypos \leftarrow \{H_k \in SG |\ out(H_k) \ni t_{ij}\}$
    if $(c_{ij} =$ "$Good$"$)$ **then**
      for H_k in $Hypos$ **do**
        $transition(H_k) \leftarrow transition(H_k) + reward(t_{ij})$ {reward hypothesis}
      end for
    else
      for H_k in $Hypos$ **do**
        $transition(H_k) \leftarrow transition(H_k) + penalty(t_{ij})$ {penalize hypothesis}
      end for
    end if
    $Processed \leftarrow Processed \cup \{(t_{ij}, p_{ij})\}$
  end for
end for
{**Step 2: Trace back to re-compute the score for all complete hypotheses**}
for H_k in $Final$ (Set of complete hypotheses) **do**
  $score(H_k) \leftarrow 0$
  while $H_k \neq$ initial hypothesis **do**
    $score(H_k) \leftarrow score(H_k) + transition(H_k)$
    $H_k \leftarrow pre(H_k)$
  end while
end for
{**Step 3: Select N cheapest hypotheses and output the new list** $T^{'}$}

This above pseudo-code depicts our re-decoding algorithm using WCE labels (**Definition 1a**

and **Definition 2a**). The algorithm in case of using WCE confidence probabilities (**Definition 1b** and **Definition 2b**) is essentially similar, except the update step (from line 10 to line 18) is replaced by the following part:

---

for H_k in $Hypos$ **do**
  $transition(H_k) \leftarrow transition(H_k) + update(t_{ij})$
end for

During the update process, the pairs including the visited word t_{ij} and the position of its aligned source words p_{ij} is consequentially admitted to $Processed$, so that all the analogous pairs $(t_{ij}^{'}, p_{ij}^{'})$ occuring in the latter sentences can be discarded. For each t_{ij}, a list of hypotheses in the SG containing it, called $Hypo$, is formed, and its confidence score c_{ij} (or $P(G)$) determines whether all members H_k in $Hypo$ will be rewarded or penalized. Once having all words in the N-best list visited, we obtain a new SG with updated transition costs for all edges containing them. The last step is to travel over all complete hypotheses (stored in $Final$) to re-compute their scores and then backtrack the cheapest-cost hypothesis to output the new best translation.

These above depictions can be clarified by taking another look at the example in Figure 7.1:

Rank	**Cost**	**Hypotheses + WCE labels**
1	-29.9061	identify the cause of action . G G G G B B
2	-40.0868	identify and measure the factors of mobilization G G G G G G G

Table 7.1 – The N-best (N=2) list generated by the SG in Figure 7.1 and WCE labels

from this SG, the N-best list (for the sake of simplicity, we choose $N = 2$) is generated as the single-pass decoder's result. According to our approach, the second pass starts by tagging all words in the list with their confidence labels, as seen in Table 7.1. Then, the graph update process is performed for each word in the list, sentence by sentence, which details are tracked in Figure 7.3. In this example, we apply **Definition 1a** to calculate the reward or penalty score, with $\alpha = \frac{1}{2}$, resulting in: $penalty(t_{ij}) = -reward(t_{ij}) = \frac{1}{2} * \frac{-29.9061}{6} = -2.4922$. Firstly, all hypotheses containing words in the 1st ranked sentence are considered. Since the word *"identify"* is labeled as *"G"*, its corresponding edge (connecting two nodes **0** and **1**) is rewarded and updated with a new cost : $t_{new} = t_{old} + reward = -1.8411 + 2.4922 = +0.6511$. On the contrary, the edge between two nodes **121252** and **182453** is penalized and takes new cost: $t_{new} = t_{old} + penalty = -5.8272 + (-2.4922) = -8.3194$, due to the bad quality of the word *"action"*. Obviously, the edges having multiple considered words (e.g. the one between nodes

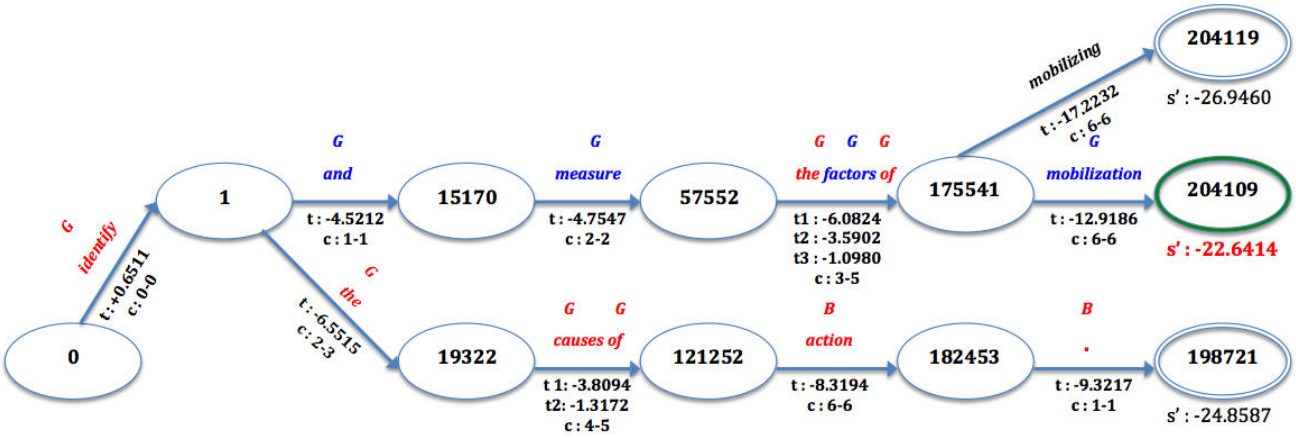

Figure 7.3 – Details of update process for the SG in Figure 7.1. The first loop (when 1st rank hypothesis is used) is represented in red color, while the second one is in blue. For edges with multiple updates, all transition costs after each update are logged. The winning cost is also emphasized by red color.

19322 and **121252**) will be updated multiple times, and the transition costs after each update can be also observed in Figure 7.3 (e.g. $t1$, $t2$, etc). Next, when the 2nd-best is taken into consideration, all repeated words (e.g. *"identify"*, *"the"* and *"of"*) are waived since they have been visited in the first loop, whereas the remaining ones are identically processed. The only untouched edge in this SG corresponds to the word *"mobilizing"*, as this word does not belong to the list. Once having the update process finished, the remaining job is to recalculate the final cost for every complete path and returns the new best translation: ***"identify and measure the factors of mobilization"*** (new cost = −22.6414).

5 Experimental Setup

5.1 Datasets and WCE System

The WCE system used for the experiments in this chapter is identical to the system mentioned in Chapter 6 (for N-best list Re-ranking). The details about the training data, feature set, ML model, annotated (oracle) labels applied to build it, as well as its performance (on the test set) can be found in Chapter 6.

One additional important resource, beside the N-best lists, employed for the re-decoding process, is the Search Graph. During the translation process, the SGs can be extracted by some parameter settings: *"-output-search-graph"*, *"-search-algorithm 1"* (using cube pruning) and *"-cube-pruning-pop-limit 5000"* (adds 5000 hypotheses to each stack). They are compactly encoded under a plain formatted text file that is convenient to transform into user-defined

structures for further processing. We then store the SG for each source sentence in a separated file, and the average size is 43.8 MB.

Table 7.2 shows how the SG is stored in the plain text file. Each line stores one hypothesis, starts by the sentence ID, then lists out all attributes in the common and intuitive format: *attribute_name* = *value* (separated by a space). Please note that not all of them are useful for the re-decoding and only some will be loaded into our data structure for further processing.

0 hyp=83 stack=1 back=0 score=-7.77332 transition=-7.77332 forward=112 fscore=-39.8677 covered=5-5 out=have a , pC=-3.03304, c=-6.55036, f2e: 0=0 , e2f: 0=0 1=-1
0 hyp=77 stack=1 back=0 score=-8.1269 transition=-8.1269 forward=162 fscore=-39.8569 covered=5-5 out=was , pC=-3.45506, c=-5.56398, f2e: 0=0 , e2f: 0=0
0 hyp=87 stack=1 back=0 score=-11.0618 transition=-11.0618 forward=280 fscore=-38.9579 covered=5-5 out=gave , pC=-3.80919, c=-6.73818, f2e: 0=0 , e2f: 0=0
0 hyp=85 stack=1 back=0 score=-9.68428 transition=-9.68428 forward=330 fscore=-38.7896 covered=5-5 out=made , pC=-3.96813, c=-6.58037, f2e: 0=0 , e2f: 0=0
0 hyp=90 stack=1 back=0 score=-9.49315 transition=-9.49315 forward=372 fscore=-41.0202 covered=5-5 out=that have , pC=-3.19534, c=-6.89086, f2e: 0=1 , e2f: 0=-1 1=0
0 hyp=78 stack=1 back=0 score=-10.378 transition=-10.378 forward=491 fscore=-42.0968 covered=5-5 out=, have , pC=-2.26412, c=-5.97011, f2e: 0=1 , e2f: 0=-1 1=0

Table 7.2 – Example of a plain text SG file outputted by Moses

5.2 Experimental Decoders

We would like to investigate the WCE's contributions in two scenarios: real WCE and ideal WCE (where all predicted labels are totally identical to the oracle ones). Therefore, we experiment with the seven following decoders:

- **BL**: Baseline (1-pass decoder)
- **BL+WCE(1a, 1b, 2a, 2b)**: four 2-pass decoders, using our estimated WCE labels and confidence probabilities to update the SGs, and the update scores are calculated by **Definition (1a, 1b, 2a, 2b)**.
- **BL+OR(1a, 2a)**: two 2-pass decoders, computing the reward or penalty scores by **Definition (1a, 2a)** on the oracle labels

It is important to note that, when using oracle labels, **Definition 1b** becomes **Definition 1a** and **Definition 2b** becomes **Definition 2a**, since if a word t_{ij} is labelled as "G", then $P(G) = 1$ and $P(B) = 0$, and vice versa. In order to tune the coefficients α and β, we carry out a **2-fold**

cross validation on the test set. First, the set is split into two equivalent parts: **S1** and **S2**. Playing the role of a development set, **S1** will train the parameter(s) which then be used to compute the update scores on **S2** re-decoding process, and vice versa. The optimization steps are handled by CONDOR toolkit (Berghen, 2004), in which we vary α and β within the interval $[0.00; 5.00]$ (starting point is 1.00), and the maximum number of iterations is fixed as 50. Test set is further divided to launch experiments in parallel on our cluster using an open-source batch scheduler: OAR (Nicolas and Joseph, 2013). This mitigates the overall processing times on such huge SGs. Finally, the re-decoding results for them are properly merged for evaluation.

6 Results

Systems	**Performance**			**Comparison to BL**			***p*-value**
	BLEU ↑	**TER ↓**	**TERp-A ↓**	**B (%)**	**E (%)**	**W (%)**	
BL	52.31	0.2905	0.3058	-	-	-	-
BL+WCE(1a)	**53.80**	**0.2876**	**0.2922**	28.72	57.43	13.85	0.00
BL+WCE(1b)	53.24	0.2896	0.2995	26.45	59.26	14.29	0.00
BL+WCE(2a)	53.32	0.2893	0.3018	23.68	60.11	16.21	0.01
BL+WCE(2b)	53.07	0.2900	0.3006	22.27	55.17	22.56	0.01
BL+OR(1a)	**60.18**	**0.2298**	**0.2264**	62.52	24.36	13.12	-
BL+OR(2a)	59.98	0.2340	0.2355	60.18	28.82	11.00	-
BL+OR(NbestRR)	58.10	0.2551	0.2544	58.68	29.63	11.69	-
BL+WCE(NbestRR)	52.77	0.2891	0.3025	18.04	68.22	13.74	0.01
Oracle BLEU score	**BLEU = 66.48 (from SG)**						

Table 7.3 – Translation quality of the conventional decoder and the 2-pass ones using scores from real or "oracle" WCE, followed by the percentage of better, equivalent or worse sentences compared to **BL** (**B**="Better", **E**="Equal", **W**="Worse")

Table A.1 shows the translation performances of all experimental decoders and their percentages of sentences which outperform, remain equivalent or degrade the baseline hypotheses (when match against the references, measured by TER). Results suggest that using **oracle labels** to re-direct the graph searching boosts dramatically the baseline quality. **BL+OR(1a)** augments 7.87 points in BLEU, and diminishes 0.0607 (0.0794) point in TER(TERp-A), compared to **BL**. Meanwhile, with **BL+OR(2a)**, these gains are 7.67, 0.0565 and 0.0514 (in that order). Besides, the contribution of our real WCE system scores seems less prominent, yet positive: the best performing **BL+WCE(1a)** increases 1.49 BLEU points of **BL** (0.0029 and 0.0136 gained for TER and TERp-A). More remarkable, tiny ***p*-values** (in the range $[0.00; 0.01]$, seen on Table A.1) estimated between BLEU of each **BL+WCE** system and that of **BL** relying

on Approximate Method (Clark et al., 2011) indicate that these performance improvements are significant. Results also reveal that the use of WCE labels are slightly more beneficial than that of confidence probabilities: **BL+WCE(1a)** and **BL+WCE(2a)** outperform **BL+WCE(1b)** and **BL+WCE(2b)**. In both scenarios, we observe that the global update score (**Definition 1**) performs more fruitfully compared to the local one (**Definition 2**).

For more insightful understanding about WCE scores' usefulness, we make a comparison with the best achievable hypotheses in the SG (oracles), based on the "LM Oracle" approximation approach presented in (Sokolov et al., 2012b). This method allows to simplify the oracle decoding to the problem of searching for the cheapest path on a SG where all transition costs are replaced by the n-gram LM scores of the corresponding words. The LM is built for each source sentence using uniquely its target post-edition. We update the SG by assigning all edges with the LM back-off score of the word it contains (instead of using the current transition cost). Finally, we combine the oracles of all sentences, yielding BLEU oracle of **66.48**.

To better understand the benefit of SG re-decoding, we compare the obtained performances with those from our previous attempt in using WCE for N-best list re-ranking (green zone of Table A.1). The idea is to build sentence-level features starting from WCE labels, then integrate them with existing SMT model scores to recalculate the objective function value, thus re-order the N-best list (see the previous chapter). Both approaches are implemented in analogous settings, e.g. identical SMT system, WCE system, and test set. Results suggest that the contribution of WCE in SG re-decoding outperforms that in N-best re-ranking in both "oracle" or real scenarios. **BL+OR(1a)** overpasses its corresponding oracle re-ranker **BL+OR(Nbest_RR)** in 2.08 points of BLEU, diminishes 0.0253 (0.0280) in TER(TERp-A). Meanwhile, **BL+WCE(1a)** wins real WCE re-ranker **BL+WCE(Nbest_RR)** in 1.03 (BLEU), 0.0015 (TER), 0.0103 (TERp-A). These achievements might originate from the following reasons:

- (1) In re-ranking, WCE scores are integrated at sentence level, so word translation errors are not fully penalized; and
- (2) In re-ranking, best translation selection is limited to N-best list, whereas we afford the search over the entire updated SG (on which not only N-best list paths but also those contain at least one word in this list are altered) .

In **Example 1** on Table 7.4, the best translation generated by the re-decoder **BL+WCE(1a)** correctly translated the source phrase *"se sont très vite rendu compte que"* into *"realized very quickly that"*, whereas the **Baseline SMT** does wrongly in terms of voice (passive instead of

active voice): *"are very soon realized that"*. We observe also that in **Example 2**, there are a number of source parts that **BL+OR(1a)** performs better than **Baseline SMT**, such as: *"à une révolution managériale"* → *"for a managerial revolution"*

(better than *"for a revolution managerial skills"*);

or *"au centre des préoccupations de l' entreprise"* →*"at the center of the company 's concerns"*

(better than *"at the center of the company"*).

Example 1 (from BL+WCE(1a))	
Source	les scientifiques se sont très vite rendu compte qu' une partie de ces ossements appartenaient à une nouvelle espèce .
Hypothesis (Baseline SMT)	scientists *are very soon* realised that a portion of *the skeletal remains given* belonged to a new species .
Hypothesis (BL+WCE(1a))	scientists realised very quickly that a portion of *the skeletal remains given* belonged to a new species .
Post-edition	*the* scientists realised very quickly that *a portion of these bones* belonged to a new species .
Example 2 (from BL+OR(1a))	
Source	c' est pourquoi nous appelons à une révolution managériale qui consisterait à mettre les collaborateurs au centre des préoccupations de l' entreprise au même titre que le client .
Hypothesis (Baseline SMT)	this is why we call for a *revolution managerial skills to set* the staff at the center of the company *as the customer* .
Hypothesis (BL+OR)	this is why we call for a managerial revolution *to set* the staff at the center of the company 's concerns *in* the same *way* as the *customer* .
Post-edition	this is why we call for a *managerial revolution which would involve putting the staff at the center of the company 's concerns to the same degree as the client* .

Table 7.4 – Examples of MT hypothesis before and after re-decoding process, given by two re-decoders: BL+WCE(1a) (Example 1) and BL+OR(1a) (Example 2)

7 Conclusion and perspectives

This chapter showed another application of WCE information in making MT outputs better. Different from that in re-ranking process, this time the word quality labels or confidence scores were more *"directly"* used to strengthen or weaken the corresponding node containing it, leading to the re-evaluation of the entire decoder's SG. We started by presenting the idea and motivation about the way that MT decoder can benefit from the predicted qualities of words in

the N-best list. Although the definition of this *"benefit"* was somewhat quite abstract and not straightforward, we tried to quantify into so-called *"update scores"* (or *"re-decoding scores"*). These scores could be derived from the current best hypothesis cost (global) or the current hypothesis's transition cost (local). Furthermore, they could be handled to be a reward or penalty for the SG corresponding nodes by using the predicted labels of probabilities. In total, we proposed four types of update scores. Experiment results showed that the global scores outperformed the local ones; and the use of WCE labels seemed more beneficial than that of predicted probabilities.

The contribution of "real" and "oracle" WCE information was similar to what they played in Re-ranking: while "oracle" WCE labels extraordinarily lifted the MT quality up (7.87 BLEU points to reach the oracle score), real WCE achieved also the positive and promising gains (1.48 BLEU points). These improvements were significative with very low p-value. Furthermore, the re-decoders were also shown to overwhelm the re-rankers in both real and oracle scenarios. The re-decoding method sharpened once again the WCE increasing contributions in every aspect of SMT, as well as opened up a new outlook for multi-pass decoders for MT as a very potential tendency.

Our re-decoding proposition can be further extended by focusing more deeply on the word quality using MQM[3] metric as error typology, making WCE labels more impactful. Besides, the update scores used in this chapter would be further considered towards the consistency with SMT graph scores to obtain a better updated SG.

[3]http://www.qt21.eu/launchpad/content/training

Chapter 8

Conclusion and Perspectives

1 Conclusion

This thesis conducted an insightful and complete research about Word Confidence Estimation (WCE) for Machine Translation, a sub field of Quality Estimation that has not been explored as broadly as the one at sentence level. The thesis spread out over seven chapters, but concentrated on two main issues: how to build and optimize a WCE system? And how to apply the WCE output to improve other fields in MT? As the thesis comes to the last pages, we would like to take a look again at all propositions we made as well as to make some compact conclusions from all the results obtained.

1.1 WCE System Building and Optimization

Walking around many leading and prominent work in the domain, we learned that the *feature processing* and the *Machine Learning (ML) model* play the a pivotal role in building WCE systems. Therefore, we filtered best-performing features of various categories (system-based, lexical, syntactic, semantic) from these previous work as part of our feature set. Besides, we enriched it by proposing a group of other word characteristics that are likely to be good hints for the quality judgment. They were extracted from the word graph merged into Confusion Network, the syntactic parser, reference systems, Language Model (of target word and POS). The entire feature set was then trained on a variety of ML models (Naive Bayes, Logistic Regression, Decision Tree and Conditional Random Fields). The features and models were investigated via three different datasets (fr-en, en-es(WMT13), en-es(WMT14)), making the analysis and conclusion more convincing. The preliminary results showed that the feature set helped to boost the classifier's performance far beyond from the "naive" settings (all-good or random predictions). They also suggested that the CRF model outperformed the others and should be exploited for further experiments.

Our next contributions lied on the attempts to make the baseline WCE system better, including: discarding poorly-performed features (Feature Selection) and complementing multiple "weak" classifiers for a better composite classifier. In the former attempt, we applied the *"Sequential Backward Algorithm"* on all systems to rank features according to their usefulness. The ordered tables helped not only to clarify the role of each feature on each system, but also to release those who are really beneficial for WCE with their top position in all systems (e.g. Source POS, Target POS, Target word, Target Context, etc). Furthermore, thanks to the selection strategy, we were able to waive redundant features and just retain the best set which yielded the optimal performance. In the second attempt, we boosted the baseline system in another way: from the entire set, multiple subparts were formed to train the basic sub-models, from which we generated the data for a *"Boosting"* model training. Essentially, this training was performed by applying the base ML algorithm (Decision Tree) in order to get a "basic" classifier, and then using the prediction errors from the current classifier to build the next one. By doing so, the future classifier is constructed by taking into account all the errors (wrongly classified words) of the current one, so it is expected to handle better these kind of data. Finally all weak classifiers were combined into a final one which is hopefully much stronger than each of them. Not out of our expectations, dealing with the same test sets of all three datasets, the *"Boosting"* classifiers showed the significant performances over those without applying this method. These improvements lead to a conclusion that by taking advantage of the portion handled successfully by each *"weak"* system and making them complement each other, we could constitute a composite "stronger" predictor.

1.2 WCE Contributions for SMT

Once having the optimized classifier, we turned our work into another stream: applying the output of the WCE system to MT. There were totally three (but not limited to) MT applications that we investigated, including: Sentence Level Confidence Estimation, SMT N-best list Reranking, and SMT search graph re-decoding.

We first proposed seven scores for sentence, synthesized from the word scores (labels) predicted by WCE to train the SCE predictor, and compared with another "pure" SCE system, where all features are at sentence level. We also combined two of them in an appropriate way to see whether WCE features can do anything for SCE. Dealing with the test sets, the WCE-based features were shown to slightly overwhelm pure SCE-based ones. However, the combination between them dramatically boosted the performance beyond each individual. The results suggested that WCE showed a positive impact in predicting the sentence's quality; especially when

it was wisely integrated into the SCE system.

In another exploitation, we also computed several sentence scores based on WCE labels, but for another objective: re-rank the N-best list to look for a better hypothesis. Starting from the observation that the objective function built from multiple model scores selects (in many cases) the sub-optimal hypothesis as the best translation; we would like to add more parameters related to word predicted quality into this function. We integrated them with 14 model scores, resulting in the totality of 20 scores for re-ranking the current N best hypotheses. In our experiment, WCE scores ensured only a modest improvement in translation quality over the baseline SMT. Nevertheless, further experiments about the simulation of WCE's improvement suggested that such types of score contribute dramatically if they are built from an accurate WCE system. They also showed that with the participation of an "ideal" WCE, the MT system reached quite close to its most optimal possible quality. These scores are totally independent from the decoder, they can be seen as a way to introduce lexical, syntactic and semantic information (used for WCE) in a SMT pipeline.

If in these two above applications, WCE labels were used to synthesize the sentence score, in the third scenario, they were used directly to update the corresponding nodes of the SMT search graph. By doing so, we hoped to raise up the paths that cover many *"good"* words as well as weaken those containing many *"bad"* ones. Essentially, the re-decoding pass re-ordered SG hypotheses in term of the more "G" words (predicted by WCE system) they contain, the more cost reduction will be made and consequentially, the more opportunity they get to be admitted in the N-best list. Compared to the Re-ranking approach, this method has much more hypotheses to select (on a huge graph rather than limited N best list), and WCE labels are deeply exploited. This is the clearest and most effective contribution of WCE with the promising gains obtained on both *"real"* and *"oracle"* WCE labels. It sharpened once again the WCE increasing contributions in every aspect of SMT, as well as opened up a new outlook for multi-pass decoders for MT as a very potential tendency.

2 Perspectives

The achievements obtained in this thesis are promising yet still can be much more improved in the future work. Besides, they also open up some new avenues for not only the MT domains, but also for some others in the fields of Natural Language Processing.

2.1 Perspectives for WCE System Building

The first focus should be the investigation of more in-depth linguistic features of word. The knowledge sources that we intend to exploit are: the grammar checker, dependency tree with linguistic hierarchical relations (e.g. father - child, siblings, etc.), semantic similarity checker (synonyms, hypernyms, etc.). From them, we can extract word's syntactic and semantic information that can add more useful knowledge to the prediction model.

Concerning the ML model, we would like to understand more deeply about the reason why CRF method outperforms the other ones (Naive Bayes, Logistic Regression, Decision Tree) on "fr-en" corpus. Similar experiments to train multiple ML models on "en-es" (WMT13+WMT14) will be launched to verify if we get the same tendency. Besides, we also would like to integrate the referential translation model (Bicici, 2013) into all systems as this model performed well in the WMT 2013's QE tasks. In addition, an another "hot" topic: Quality Estimation at **segment level** will be in our focus.

2.2 Perspectives for WCE Contributions

First, we think about an interactive scenario where WCE instantly and effectively assists users and MT system. Specifically, we aim at constructing a system that takes feedbacks from users to regenerate a better hypothesis. With a friendly interface, the system allows them, just by clicking on the word on a given hypothesis displayed on screen to judge it as a translation error (by default, all untouched words are implied correct). Right after the user's judgement, the system has its own method to learn from this information and evaluate again its *"N-best list"* or its *"search graph"* (corresponding paths containing these judged words) in order to display the new (hopefully better) output. This system supports multiple corrections: if users are still not satisfied with the new one, they can continue to specify errors and wait for the new output after their feedbacks are learnt (in the same way) by the system. The process iterates until the users feel happy with the output they get. This sentence pair is then pushed to the *"waiting batch"*, and when the batch is large enough (reaches a pre-defined size), the MT system will be retrained using them. By this way, WCE helps MT system gradually learn from its mistakes.

Beyond the boundary of MT domain, WCE can extend to reach other domains, and one of our targets is the Spoken Language Translation (SLT). We knew that WCE has separately treated in both MT and Automatic Speech Recognition (ASR). We also knew that SLT systems, which takes the job of translating human speech utterances from one language to another, contain both ASR and MT phases. It means that a target word of SLT system is correct if it is correctly translated by the MT system from the source word that has been previously correctly

transcribed (from the speech utterance). So, why do not we try to integrate the predicted scores obtained from MT and ASR modules in an appropriate way to evaluate the word quality for SLT? This task might be more complicated than each individual (WCE for ASR or MT), since it requires the specific corpus for building and testing. So far, we have just made available a French - English corpus of 2643 speech utterances for which a quintuplet containing: ASR output, verbatim transcript, text translation output, speech translation output, post-edition of the translation. With the availability of all other components (ASR, MT, WCE systems), this *"joint estimation"* idea is expected to deploy soon. We believe that the WCE for SLT can help to improve the translators' productivity (in a lecture or movie translation scenario) or it could be useful in interactive speech-to-speech translation.

Moreover, we would like to try the WCE contribution in Active Learning (AL) approach. AL aims at reducing the training corpus size yet still maintaining the SMT performance. The training data reduction can be reached by an intelligent data sampling and is an effective solution to mitigate the annotation (manual) efforts, since more training data needs more human-annotated labels. To do this, AL calculates the usefulness of each sample by classifying it and then measuring how "uncertain" this classification was. We strongly believe that WCE scores can be used as one of the criteria to judge this "uncertainty" to filter out some irrelevant training data and retain only those from which the model benefits most. So far, the concrete steps for this strategy have not been established, yet we hope to get it more and more feasible in the future work starting from this initiative.

By applying these above techniques and ideas, we hope to have better and better CE systems and CE applications to join the upcoming evaluation campaigns (WMT, IWSLT, etc.).

Bibliography

Yaser Al-onaizan, Jan Curin, Michael Jahr, Kevin Knight, John Lafferty, Dan Melamed, Franz-Josef Och, David Purdy, Noah A. Smith, and David Yarowsky. Statistical machine translation. Technical report, Final Report, JHU Summer Workshop, 1999. 26

D. Arnold, L. Balkan, R.L. Humphreys, S. Meijer, and L. Sadler. *Machine Translation: an Introductory Guide*. NCC Blackwell, London, 1994. URL http://www.essex.ac.uk/linguistics/external/clmt/MTbook/PostScript/. 12

Wilker Aziz, Sheila C. M. de Sousa, and Lucia Specia. Pet: a tool for post-editing and assessing machine translation. In *Proceedings of the Eight International Conference on Language Resources and Evaluation (LREC'12)*, Istanbul, Turkey, May 23-25 2012. 28, 113

Satanjeev Banerjee and Alon Lavie. Meteor: An automatic metric for mt evaluation with improved correlation with human judgments. pages 65–72, 2005. 30

Adam L. Berger, Vincent J. Della Pietra, and Stephen A. Della Pietra. A maximum entropy approach to natural language processing. *Comput. Linguist.*, 22(1):39–71, March 1996. ISSN 0891-2017. URL http://dl.acm.org/citation.cfm?id=234285.234289. 40

Frank Vanden Berghen. *CONDOR: a constrained, non-linear, derivative-free parallel optimizer for continuous, high computing load, noisy objective functions*. PhD thesis, University of Brussels (ULB - Université Libre de Bruxelles), Belgium, 2004. 134

Ergun Bicici. Referential translation machines for quality estimation. In *Proceedings of the Eighth Workshop on Statistical Machine Translation*, pages 343–351, Sofia, Bulgaria,

August 2013. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/W13-2242. 41, 42

John Blatz, Erin Fitzgerald, George Foster, Simona Gandrabur, Cyril Goutte, Alex Kulesza, Alberto Sanchis, and Nicola Ueffing. Confidence estimation for machine translation. Technical report, JHU/CLSP Summer Workshop, 2003. 3, 38, 39, 47

John Blatz, Erin Fitzgerald, George Foster, Simona Gandrabur, Cyril Goutte, Alex Kulesza, Alberto Sanchis, and Nicola Ueffing. Confidence estimation for machine translation. In *Proceedings of COLING 2004*, pages 315–321, Geneva, April 2004. 3, 37, 38, 39, 45, 47, 96

Christian Boitet. Les architectures linguistiques et computationnelles en traduction automatique sont indépendantes. In *TALN-08*, Avignon, France, 2008. ATALA. 10

Ondrej Bojar, Christian Buck, Christian Federmann, Barry Haddow, Philipp Koehn, Johannes Leveling, Christof Monz, Pavel Pecina, Matt Post, Herve Saint-Amand, Radu Soricut, Lucia Specia, and Aleš Tamchyna. Findings of the 2014 workshop on statistical machine translation. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*, pages 12–58, Baltimore, Maryland, USA, June 2014. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/W/W14/W14-3302. 53

Ondřej Bojar, Christian Buck, Chris Callison-Burch, Christian Federmann, Barry Haddow, Philipp Koehn, Christof Monz, Matt Post, Radu Soricut, and Lucia Specia. Findings of the 2013 Workshop on Statistical Machine Translation. In *Proceedings of the Eighth Workshop on Statistical Machine Translation*, pages 1–44, Sofia, Bulgaria, August 2013. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/W13-2201. 37, 41, 53, 64, 75

Peter F. Brown, John Cocke, Stephen A. Della Pietra, Vincent J. Della Pietra, Fredrick Jelinek, John D. Lafferty, Robert L. Mercer, and Paul S. Roossin. A statistical approach to machine translation. *Comput. Linguist.*, 16(2):79–85, June 1990. ISSN 0891-2017. URL http://dl.acm.org/citation.cfm?id=92858.92860. 13

PF Brown, A Stephen, V Pietra, and LM Robert. The mathematics of statistical machine translation: parameter estimation. volume 19, 1993a. 13, 17, 19, 39

Chris Callison-Burch and Philipp Koehn. Introduction to statistical machine translation. In European Summer School for Language and Logic (ESSLL) 2005, editors, *European Summer School for Language and Logic (ESSLL) 2005*, 2005. 21, 22, 24, 25, 157

Chris Callison-burch and Miles Osborne. Re-evaluating the role of bleu in machine translation research. In *In EACL*, pages 249–256, 2006. 30

Chris Callison-Burch, Cameron Fordyce, Philipp Koehn, Christof Monz, and Josh Schroeder. (meta-) evaluation of machine translation. In *Proceedings of the Second Workshop on Statistical Machine Translation*, StatMT '07, pages 136–158, Stroudsburg, PA, USA, 2007. Association for Computational Linguistics. URL http://dl.acm.org/citation.cfm?id=1626355.1626373. 27

Chris Callison-Burch, Philipp Koehn, Christof Monz, Matt Post, Radu Soricut, and Lucia Specia. Findings of the 2012 workshop on statistical machine translation. In *Proceedings of the Seventh Workshop on Statistical Machine Translation*, pages 10–51, Montréal, Canada, June 2012. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/W12-3102. 97

Stanley F. Chen and Joshua Goodman. An empirical study of smoothing techniques for language modeling. In *Proceedings of the 34th Annual Meeting on Association for Computational Linguistics*, ACL '96, pages 310–318, Stroudsburg, PA, USA, 1996. Association for Computational Linguistics. doi: 10.3115/981863.981904. URL http://dx.doi.org/10.3115/981863.981904. 16

Jonathan Clark, Chris Dyer, Alon Lavie, and Noah Smith. Better hypothesis testing for statistical machine translation: Controlling for optimizer instability. In *Proceedings of the Association for Computational Lingustics*, 2011. 114, 135

Koby Crammer, Ofer Dekel, Joseph Keshet, Shai Shalev-Shwartz, and Yoram Singer. Online passive-aggressive algorithms. *J. Mach. Learn. Res.*, 7:551–585, December 2006. ISSN 1532-4435. URL http://dl.acm.org/citation.cfm?id=1248547.1248566. 20, 112

George Doddington. Automatic evaluation of machine translation quality using n-gram co-occurrence statistics. In *Proceedings of the Second International Conference on Human Language Technology Research*, HLT '02, pages 138–145, San Francisco, CA, USA, 2002. Morgan Kaufmann Publishers Inc. URL http://dl.acm.org/citation.cfm?id=1289189.1289273. 34

Bonnie J. Dorr, Pamela W. Jordan, and John W. Benoit. A survey of current paradigms in machine translation, 1999. 9

Kevin Duh and Katrin Kirchhoff. Beyond log-linear models: Boosted minimum error rate training for n-best re-ranking. In *Proc. of ACL, Short Papers*, 2008. 37, 103, 106, 122

Yoav Freund. Boosting a weak learning algorithm by majority. *Inf. Comput.*, 121(2):256–285, September 1995. ISSN 0890-5401. doi: 10.1006/inco.1995.1136. URL http://dx.doi.org/10.1006/inco.1995.1136. 87

Yoav Freund and Robert E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. In *Proceedings of the Second European Conference on Computational Learning Theory*, EuroCOLT '95, pages 23–37, London, UK, UK, 1995. Springer-Verlag. ISBN 3-540-59119-2. URL http://dl.acm.org/citation.cfm?id=646943.712093. 87, 88

J. Friedman, T. Hastie, and R. Tibshirani. Additive Logistic Regression: a Statistical View of Boosting. *The Annals of Statistics*, 38(2), 2000. 50

Michael Gamon, Anthony Aue, and Martine Smets. Sentence-level mt evaluation without reference translations: Beyond language modeling. In *In European Association for Machine Translation (EAMT*, 2005. 96

Simona Gandrabur and George Foster. Confidence estimation for text prediction. In *Proceedings of the Conference on Natural Language Learning (CoNLL 2003)*, pages 315–321, Edmonton, May 2003. 36, 38

Ulrich Germann, Michael Jahr, Kevin Knight, Daniel Marcu, and Kenji Yamada. Fast decoding and optimal decoding for machine translation. In *Proceedings of the 39th Annual Meeting on Association for Computational Linguistics*, ACL '01, pages 228–235, Stroudsburg, PA, USA, 2001. Association for Computational Linguistics. doi: 10.3115/1073012.1073042. URL http://dx.doi.org/10.3115/1073012.1073042. 26

Aaron Li-Feng Han, Yi Lu, Derek F. Wong, Lidia S. Chao, Liangye He, and Junwen Xing. Quality estimation for machine translation using the joint method of evaluation criteria and statistical modeling. In *Proceedings of the Eighth Workshop on Statistical Machine Translation*, pages 365–372, Sofia, Bulgaria, August 2013. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/W13-2245. 41, 42

Yifan He, Yanjun Ma, Josef van Genabith, and Andy Way. Bridging smt and tm with translation recommendation. In Jan Hajic, Sandra Carberry, and Stephen Clark, editors, *ACL*, pages

622–630. The Association for Computer Linguistics, 2010. ISBN 978-1-932432-67-1. URL http://dblp.uni-trier.de/db/conf/acl/acl2010.html#HeMGW10. 36

W. John Hutchins. Machine translation: A concise history, 2007. 9, 10

W. John Hutchins and Harold L. Somers. *An Introduction to Machine Translation*. Academic Press, 1992. 1, 7, 11

Cong-Phap Huynh, Christian Boitet, and Hervé Blanchon. Sectra_w.1: an online collaborative system for evaluating, post-editing and presenting mt translation corpora. In *LREC'08, Sixth International Conference on Language Resources and Evaluation*, pages 28–30, Marrakech, Morocco, 2008. 28

F. Jelinek, R. L. Mercer, L. R. Bahl, and J. K. Baker. Perplexity – a measure of the difficulty of speech recognition tasks. *Journal of the Acoustical Society of America*, 62:S63, November 1977. Supplement 1. 16

Ronald M. Kaplan and Joan Bresnan. Lexical-functional grammar: A formal system for grammatical representation, 1995. 12

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*, pages 400–401, 1987. 16

Paul John King. Towards truth in head-driven phrase structure grammar. In Valia Kordoni, editor, *Tübingen Studies in Head-Driven Phrase Structure Grammar*, pages 301–352. Eberhard-Karls-Universität Tübingen, 1999. URL http://www.sfs.uni-tuebingen.de/sfb/reports/berichte/132/king/king.dvi.ps. 12

Katrin Kirchhoff and Mei Yang. Improved language modeling for statistical machine translation. In *Proceedings of the ACL Workshop on Building and Using Parallel Texts*, pages 125–128, Ann Arbor, Michigan, June 2005. 105

Philipp Koehn. *MOSES Statistical Machine Translation System : User Manual and Code Guide*. University of Edinburgh, http://www.statmt.org/moses/manual/manual.pdf. 24

Philipp Koehn. Pharaoh: A beam search decoder for phrase-based statistical machine translation models. In *AMTA*, pages 115–124, 2004. 21, 26

Philipp Koehn. Europarl: A parallel corpus for statistical machine translation, 2005. 26

Philipp Koehn. *Statistical Machine Translation*. Cambridge University Press, New York, NY, USA, 1st edition, 2010. ISBN 0521874157, 9780521874151. 25

Philipp Koehn, Franz Josef Och, and Daniel Marcu. Statistical phrase-based translation. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology - Volume 1*, NAACL '03, pages 48–54, Stroudsburg, PA, USA, 2003. Association for Computational Linguistics. doi: 10.3115/1073445.1073462. URL http://dx.doi.org/10.3115/1073445.1073462. 19, 24

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondrej Bojar, Alexandra Constantin, and Evan Herbst. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics*, pages 177–180, Prague, Czech Republic, June 2007. 21, 26, 53, 63

Anthony Kroch and Aravind Joshi. Linguistic relevance of tree adjoining grammars. Technical report, 1985. 12

John Lafferty, Andrew McCallum, and Fernando Pereira. Conditional random fields: Probabilistic models for segmenting et labeling sequence data. In *Proceedings of ICML-01*, pages 282–289, 2001. 41, 51, 69

David Langlois, Sylvain Raybaud, and Kamel Smaïli. Loria system for the wmt12 quality estimation shared task. In *Proceedings of the Seventh Workshop on Statistical Machine Translation*, pages 114–119, Montréal, Canada, June 2012. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/W12-3113. 41

Thomas Lavergne, Olivier Cappé, and François Yvon. Practical very large scale crfs. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 504–513, 2010. 52, 69

Alon Lavie and Michael J. Denkowski. The meteor metric for automatic evaluation of machine translation. *Machine Translation*, 23(2-3):105–115, September 2009. ISSN 0922-6567. doi: 10.1007/s10590-009-9059-4. URL http://dx.doi.org/10.1007/s10590-009-9059-4. 34

Alon Lavie, Kenji Sagae, and Shyamsundar Jayaraman. The significance of recall in automatic metrics for mt evaluation. In *In Proceedings of the 6th Conference of the Association for Machine Translation in the Americas (AMTA-2004*, 2004. 30

VI Levenshtein. Binary Codes Capable of Correcting Deletions, Insertions and Reversals. *Soviet Physics Doklady*, 10:707, 1966. 28

Daniel Lowd. Naive bayes models for probability estimation. In *Proceedings of the Twentysecond International Conference on Machine Learning*, pages 529–536. ACM Press, 2005. 48

Ngoc-Quang Luong. Integrating lexical, syntactic and system-based features to improve word confidence estimation in smt. In *Proceedings of JEP-TALN-RECITAL*, volume 3 (RECITAL), pages 43–56, Grenoble, France, June 4-8 2012. 45, 55

Ngoc Quang Luong, Laurent Besacier, and Benjamin Lecouteux. Word confidence estimation and its integration in sentence quality estimation for machine translation. In *Proceedings of The Fifth International Conference on Knowledge and Systems Engineering (KSE 2013)*, Hanoi, Vietnam, October 17-19 2013a. 110

Ngoc Quang Luong, Benjamin Lecouteux, and Laurent Besacier. LIG system for WMT13 QE task: Investigating the usefulness of features in word confidence estimation for MT. In *Proceedings of the Eighth Workshop on Statistical Machine Translation*, pages 386–391, Sofia, Bulgaria, August 2013b. Association for Computational Linguistics. URL `http://www.aclweb.org/anthology/W13-2248`. 42

Ngoc Quang Luong, Benjamin Lecouteux, and Laurent Besacier. LIG system for WMT13 QE task: Investigating the usefulness of features in word confidence estimation for MT. In *Proceedings of the Eighth Workshop on Statistical Machine Translation*, pages 396–391, Sofia, Bulgaria, August 2013c. Association for Computational Linguistics. 65

Ngoc Quang Luong, Benjamin Lecouteux, and Laurent Besacier. LIG submissions at WMT14 QE task. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*, pages 396–391, Baltimore, USA, June 2014. Association for Computational Linguistics. 66, 75

Lidia Luminita Mangu. Finding consensus in speech recognition. Technical report, 2000. 58

Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to Information Retrieval*. Cambridge University Press, New York, NY, USA, 2008. ISBN 0521865719, 9780521865715. 14

Daniel Marcu and William Wong. A phrase-based, joint probability model for statistical machine translation. In *In Proceedings of EMNLP*, pages 133–139, 2002. 19

Makoto Nagao. A framework of a mechanical translation between japanese and english by analogy principle. In *Proc. Of the International NATO Symposium on Artificial and Human Intelligence*, pages 173–180, New York, NY, USA, 1984. Elsevier North-Holland, Inc. ISBN 0-444-86545-4. URL http://dl.acm.org/citation.cfm?id=2927.2938. 12

Preslav Nakov, Francisco Guzman, and Stephan Vogel. Optimizing for sentence-level bleu+1 yields short translations. In *Proceedings of COLING 2012*, pages 1979–1994, Mumbai, India, December 8 -15 2012. 108, 115

Bach Nguyen, Fei Huang, and Yaser Al-Onaizan. Goodness: A method for measuring machine translation confidence. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics*, pages 211–219, Portland, Oregon, June 2011. 3, 36, 37, 41, 43, 45, 67, 103, 106, 120, 122

Capit Nicolas and Emeras Joseph. *OAR Documentation - User Guide*. LIG laboratory, Laboratoire d'Informatique de Grenoble Bat. ENSIMAG - antenne de Montbonnot ZIRST 51, avenue Jean Kuntzmann 38330 MONTBONNOT SAINT MARTIN, 2013. 134

Franz Josef Och. Minimum error rate training in statistical machine translation. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, pages 160–167, July 2003. 20, 106, 112

Franz Josef Och and Hermann Ney. A systematic comparison of various statistical alignment models. *Computational Linguistics*, 29(1):19–51, 2003. 26, 64

Franz Josef Och and Hermann Ney. The alignment template approach to statistical machine translation. *Comput. Linguist.*, 30(4):417–449, December 2004. ISSN 0891-2017. doi: 10.1162/0891201042544884. URL http://dx.doi.org/10.1162/0891201042544884. 19

Kishore Papineni, Salim Roukos, Todd Ard, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, 2002. 29, 34, 108

Kristen Parton, Nizar Habash, Kathleen McKeown, Gonzalo Iglesias, and Adrià de Gispert. Can automatic post-editing make mt more meaningful? In *Proceedings of the 16th EAMT*, pages 111–118, Trento, Italy, 28-30 May 2012. 28, 103, 122

M Potet, L Besacier, and H Blanchon. The lig machine translation system for wmt 2010. In ACL Workshop, editor, *Proceedings of the joint fifth Workshop on Statistical Machine Translation and Metrics MATR (WMT2010)*, Uppsala, Sweden, 11-17 July 2010. 63

M Potet, R Emmanuelle E, L Besacier, and H Blanchon. Collection of a large database of french-english smt output corrections. In *Proceedings of the eighth international conference on Language Resources and Evaluation (LREC)*, Istanbul, Turkey, May 2012. 65

J. R. Quinlan. Induction of decision trees. *Mach. Learn.*, 1(1):81–106, March 1986. ISSN 0885-6125. doi: 10.1023/A:1022643204877. URL http://dx.doi.org/10.1023/A:1022643204877. 51

Christopher B. Quirk. Training a sentence-level machine translation confidence measure. In *In Proceedings of LREC*, 2004. 96

L. R. Rabiner and B. H. Juang. An introduction to hidden markov models. *IEEE ASSp Magazine*, 1986. 18

Lawrence Rabiner and Biing-Hwang Juang. *Fundamentals of Speech Recognition*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1993. ISBN 0-13-015157-2. 14

Sylvain Raybaud, David Langlois, and Kamel Smaïli. "this sentence is wrong." detecting errors in machine-translated sentences. *Machine Translation*, 25(1):1–34, 2011. URL http://dblp.uni-trier.de/db/journals/mt/mt25.html#RaybaudLS11. 45

Alberto Sanchis, Alfons Juan, and Enrique Vidal. Estimation of confidence measures for machine translation. In *Proceedings of the MT Summit XI*, pages 407–412, Copenhagen, Denmark, 2007. 37, 39

Carolina Scarton and Lucia Specia. Document-level translation quality estimation: exploring discourse and pseudo-references. In *17th Annual Conference of the European Association for Machine Translation*, EAMT, Dubrovnik, Croatia, 2014. URL http://hnk.ffzg.hr/eamt2014/EAMT2014_proceedings.pdf. 3

Robert E. Schapire. The strength of weak learnability. *Mach. Learn.*, 5(2):197–227, July 1990. ISSN 0885-6125. doi: 10.1023/A:1022648800760. URL http://dx.doi.org/10.1023/A:1022648800760. 87

Michel Simard. The baf: A corpus of english-french bitext, 1998. 26

Matthew Snover, Bonnie Dorr, Richard Schwartz, Linnea Micciulla, and John Makhoul. A study of translation edit rate with targeted human annotation. In *In Proceedings of Association for Machine Translation in the Americas*, pages 223–231, 2006. 31

Matthew Snover, Nitin Madnani, Bonnie Dorr, and Richard Schwartz. Terp system description. In *MetricsMATR workshop at AMTA*, 2008. 34, 67, 108

Artem Sokolov, Guillaume Wisniewski, and Francois Yvon. Non-linear n-best list reranking with few features. In *Proceedings of AMTA*, 2012a. 105

Artem Sokolov, Guillaume Wisniewski, and Franc ois Yvon. Computing lattice bleu oracle scores for machine translation. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*, pages 120–129, Avignon, France, April 2012b. 135

Radu Soricut and Abdessamad Echihabi. Trustrank: Inducing trust in automatic translations via ranking. In *Proceedings of the 48th ACL (Association for Computational Linguistics)*, pages 612–621, Uppsala, Sweden, July 2010. 40

L Specia, M Turchi, N Cancedda, M Dymetman, and N Cristianini. Estimating the sentence-level quality of machine translation systems. In *13th Conference of the European Association for Machine Translation*, pages 18–37, Barcelona, 2009a. 3

L Specia, Z Wang, M Turchi, J Shawetaylor, and C Saunders. Improving the confidence of machine translation quality estimates. In *Proceedings of the MT Summit XII*, Ottawa, Canada, 2009b. 3

Lucia Specia and Atefeh Farzindar. Estimating machine translation post-editing effort with hter. In *AMTA 2010- workshop, Bringing MT to the User: MT Research and the Translation Industry*. The Ninth Conference of the Association for Machine Translation in the Americas, The Ninth Conference of the Association for Machine Translation in the Americas, nov 2010. 28

Lucia Specia and Jesús Giménez. Combining confidence estimation and reference-based metrics for segment-level mt evaluation. In *Ninth Conference of the Association for Machine Translation in the Americas*, AMTA, Denver, Colorado, 2010. URL http://www.mt-archive.info/AMTA-2010-Specia.pdf. 3

Lucia Specia, Dhwaj Raj, and Marco Turchi. Machine translation evaluation versus quality estimation. *Machine Translation*, 24(1):39–50, 2010. doi: 10.1007/s10590-010-9077-2. 3

Lucia Specia, Najeh Hajlaoui, Catalina Hallett, and Wilker Aziz. Predicting machine translation adequacy. In *Machine Translation Summit XIII*, pages 513–520, 2011. URL http://www.mt-archive.info/MTS-2011-Specia.pdf. 36

Andreas Stolcke. Srilm - an extensible language modeling toolkit. In *Seventh International Conference on Spoken Language Processing*, pages 901–904, Denver, USA, 2002. 26, 45, 60, 63, 64

Roy Tromble, Shankar Kumar, Franz Och, and Wolfgang Macherey. Lattice minimum bayes-risk decoding for statistical machine translation. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 620–629, 2008. URL `http://aclweb.org/anthology-new/D/D08/D08-1065.pdf`. 122

Joseph Turian, Luke Shen, and I. Dan Melamed. Evaluation of machine translation and its evaluation. In *In Proceedings of MT Summit IX*, pages 386–393, 2003. 30

Nicola Ueffing and Hermann Ney. Word-level confidence estimation for machine translation using phrased-based translation models. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pages 763–770, Vancouver, 2005. 3, 37, 39, 123

Nicola Ueffing and Hermann Ney. Word-level confidence estimation for machine translation. In *Computational Linguistics*, volume 33, pages 9–40, 2007. 3, 45

Nicola Ueffing, Franz Josef Och, and Hermann Ney. Generation of word graphs in statistical machine translation. In *Proceedings of the Conference on Empirical Methods for Natural Language Processing (EMNLP 02)*, pages 156–163, Philadelphia, PA, 2002. 45

Nicola Ueffing, Klaus Macherey, and Hermann Ney. Confidence measures for statistical machine translation. In *Proceedings of the MT Summit IX*, pages 394–401, New Orleans, LA, September 2003. 3, 38, 39, 44

Bernard Vauquois. A survey of formal grammars and algorithms for recognition and transformation in mechanical translation. In *IFIP Congress (2)*, pages 1114–1122, 1968. URL `http://dblp.uni-trier.de/db/conf/ifip/ifip1968-2.html#Vauquois68`. 11

Ashish Venugopal, Andreas Zollmann, and Stephan Vogel. An efficient two-pass approach to synchronous-cfg driven statistical mt. In *Proceedings of Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics*, April 2007. 37, 103, 122, 123

John S. White. The arpa mt evaluation methodologies: Evolution, lessons, and further approaches. In *Proceedings of the 1994 Conference of the Association for Machine Translation in the Americas*, pages 193–205, 1994. 27

Deyi Xiong, Min Zhang, and Haizhou Li. Error detection for statistical machine translation using linguistic features. In *Proceedings of the 48th Association for Computational Linguistics*, pages 604–611, Uppsala, Sweden, July 2010. 3, 37, 40, 42, 45, 47, 67

Richard Zens and Hermann Ney. N-gram posterior probabilities for statistical machine translation. In *Proceedings of the Workshop on Statistical Machine Translation*, StatMT '06, pages 72–77, Stroudsburg, PA, USA, 2006. Association for Computational Linguistics. URL http://dl.acm.org/citation.cfm?id=1654650.1654661. 122

Ying Zhang, Almut Silja Hildebrand, and Stephan Vogel. Distributed language modeling for n-best list re-ranking. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing (EMNLP 2006)*, pages 216–223, Sydney, July 2006. 37, 103, 105, 122

List of Figures

List of Tables

Appendix A

Personal Publications

1. **Ngoc-Quang Luong**. Integrating lexical, syntactic and system-based features to improve Word Confidence Estimation in SMT. In *Proceedings of JEP-TALN-RECITAL, volume 3 (RECITAL)*, pages 43–56, Grenoble, France, June 4-8 2012.

> Confidence Estimation at word level is the task of predicting the correct and incorrect words in the target sentence generated by a MT system. It helps to conclude the reliability of a given translation as well as to filter out sentences that are not good enough for post-editing. This paper investigates various types of features to circumvent this issue, including lexical, syntactic and system-based features. A method to set training label for each word in the hypothesis is also presented. A classifier based on conditional random fields (CRF) is employed to integrate features and determine the word's appropriate label. We conducted our preliminary experiment with all features, tracked precision, recall and F-score and we compared with our baseline system. Experimental results of the full combination of all features yield the very encouraging precision, recall and F-score for Good label (F-score: 86.7%), and acceptable scores for Bad label (F-score: 36.8%).

2. BESACIER, L., LECOUTEUX, B., AZOUZI, M. et **LUONG NGOC, Q. (2012)**. The LIG English to French Machine Translation System for IWSLT 2012. In *proceedings of the 9th International Workshop on Spoken Language Translation (IWSLT)*, Hong Kong, 5-7 December.

> This paper presents the LIG participation to the E-F MT task of IWSLT 2012. The primary system proposed made a large improvement (more than 3 point of BLEU on tst2010 set) compared to our last year participation. Part of this improvement was due to the use of an extraction from the Giga-word corpus. We also propose a preliminary adaptation of the driven decoding concept for machine translation. This method allows an efficient combination of machine translation systems, by re-scoring the log-linear model at the N-best list level according to auxiliary systems: the basis technique is essentially guiding the search using one or previous system outputs. The results show that the

approach allows a significant improvement in BLEU score using Google translate to guide our own SMT system. We also try to use a confidence measure as an additional log-linear feature but we could not get any improvement with this technique.

3. **Ngoc-Quang Luong**, Benjamin Lecouteux, Laurent Besacier. LIG System for WMT13 QE Task: Investigating the Usefulness of Features in Word Confidence Estimation for MT. In *Proceedings of the 8th Workshop on Statistical Machine Translation*, Sofia, Bulgaria, aug 2013.

This paper presents the LIG's systems submitted for Task 2 of WMT13 Quality Estimation campaign. This is a word confidence estimation (WCE) task where each participant was asked to label each word in a translated text as a binary (Keep/Change) or multi-class (Keep/Substitute/Delete) category. We integrate a number of features of various types (system-based, lexical, syntactic and semantic) into the conventional feature set, for our baseline classifier training. After the experiments with all features, we deploy a "Feature Selection" strategy to keep only the best performing ones. Then, a method that combines multiple "weak" classifiers to build a strong "composite" classifier by taking advantage of their complementarity is presented and experimented. We then select the best systems for submission and present the official results obtained.

4. **Ngoc-Quang Luong**, Laurent Besacier, Benjamin Lecouteux. Word Confidence Estimation and its Integration in Sentence Quality Estimation for Machine Translation. In *Proceedings of the fifth international conference on knowledge and systems engineering (KSE)*, Hanoi, Vietnam, oct 2013.

This paper proposes some ideas to build an effective estimator, which predicts the quality of words in a Machine Translation (MT) output. We integrate a number of features of various types (system-based, lexical, syntactic and semantic) into the conventional feature set, for our baseline classifier training. After the experiments with all features, we deploy a "Feature Selection" strategy to filter the best performing ones. Then, a method that combines multiple "weak" classifiers to build a strong "composite" classifier by taking advantage of their complementarity allows us to achieve a better performance in term of F score. Finally, we exploit word confidence scores for improving the estimation system at sentence level.

5. **Ngoc-Quang Luong**, Laurent Besacier, Benjamin Lecouteux. Word Confidence Estimation for SMT N-best List Re-ranking. In *Proceedings of the Workshop on Humans and Computer-assisted Translation (HaCaT)*, Gothenburg, Sweden, April 2014.

This paper proposes to use Word Confidence Estimation (WCE) information to improve MT outputs via N-best list re- ranking. From the confidence label assigned for each word in the MT hypothesis, we add six scores to the baseline log-linear model in order to re-rank the N-best list. Firstly, the correlation between the WCE-based sentence-level scores and the conventional evaluation scores (BLEU, TER, TERp-A) is investigated. Then, the N-best list re-ranking is evaluated over different WCE system performance levels: from our real and efficient WCE system (ranked 1st during last WMT 2013 Quality Estimation Task) to an oracle WCE (which simulates an interactive scenario where a user simply validates words of a MT hypothesis and the new output will be automatically re-generated). The results suggest that our real WCE system slightly (but significantly) improves the baseline while the oracle one extremely boosts it; and better WCE leads to better MT quality.

6. **Ngoc-Quang Luong**, Laurent Besacier, Benjamin Lecouteux. An Efficient Two-Pass Decoder for SMT Using Word Confidence Estimation. In *Proceeding of the Seventeenth Annual Conference of the European Association for Machine Translation (EAMT), 16th-18th, June, Dubrovnik, Croatia*

During decoding, the Statistical Machine Translation (SMT) decoder travels over all complete paths on the Search Graph (SG), seeks those with cheapest costs and back-tracks to read off the best translations. Although these winners beat the rest in model scores, there is no certain guarantee that they have the highest quality with respect to the human references. This paper exploits Word Confidence Estimation (WCE) scores in the second pass of decoding to enhance the Machine Translation (MT) quality. By using the confidence score of each word in the N-best list to update the cost of SG hypotheses containing it, we hope to "reinforce" or "weaken" them relied on word quality. After the update, new best translations are re-determined using updated costs. In the experiments on our real WCE scores and ideal (oracle) ones, the latter significantly boosts one-pass decoder by 7.87 BLEU points, meanwhile the former yields an improvement of 1.49 points for the same metric.

7. **Ngoc-Quang Luong**, Laurent Besacier, Benjamin Lecouteux. LIG System for Word Level QE task at WMT14. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*, Baltimore, MD, USA, June 26th-27th.

This paper describes our Word-level QE system for WMT 2014 shared task on Spanish - English pair. Compared to WMT 2013, this year's task is different due to the lack

of SMT setting information and additional resources. We report how we overcome this challenge to retain most of the important features which performed well last year in our system. Novel features related to the availability of multiple systems output (new point of this year) are also proposed and experimented along with baseline set. The system is optimized by several ways: tuning the classification threshold, combining with WMT 2013 data, and refining using Feature Selection strategy on our development set, before dealing with the test set for submission.

8. **Ngoc-Quang Luong**, Laurent Besacier, Benjamin Lecouteux. Some Propositions to Improve the Prediction Capability of Word Confidence Estimation for Machine Translation. (accepted to *JCSCE Journal*, 2014).

In this article, we propose some ideas to build effective estimators, which predict the quality of words in a Machine Translation (MT) output. We propose a number of novel features of various types (system-based, lexical, syntactic and semantic) and then integrate them into the conventional (previously used) feature set, for our baseline classifier training. The classifiers are built over two different bilingual corpus: French - English (**fr-en**) and English - Spanish (**en-es**). After the experiments with all features, we deploy a "Feature Selection" strategy to filter the best performing ones. Then, a method that combines multiple "weak" classifiers to constitute a strong "composite" classifier by taking advantage of their complementarity allows us to achieve a significant improvement in term of F score, for both **fr-en** and **en-es** systems. Finally, we exploit word confidence scores for improving the estimation system at sentence level.

Appendix B

TERP Toolkit Guidelines

1 Introduction

TERP is an open-source tool to calculate the TER score between two documents, and in MT research we can apply it with the MT output (MT hypothesis) and the references. Besides the TER scores, this tool also assigns for which each word in the MT output with one of the following labels:

- **TERp**: E (exact), S (Substitution), I (Insertion) and D (Deletion)
- **TERp-A**: E (exact), I (Insertion), S (Substitution), T (Stem matches), Y (Synonym matches), and P (phrasal substitutions)

2 Installation and Execution

- For installing the software, please refer to the below webpage which guides you step by step: http://www.umiacs.umd.edu/~snover/terp/doc_v1.html
- Calling script and viewing the result:

```
./bin/terp_ter -r reference_file -h hypothesis_file_to_score
or:
./bin/terpa -r reference_file -h hypothesis_file_to_score
```

Example:

```
/home/potetm/TER-plus/bin/terp_ter -r 100.ref.id -h 100.hyp.id
or:
bin/terpa -r ./test_terpa/100.ref.id -h ./test_terpa/100.hyp.id
```

3 Some Guidelines

It is very important to note that TERP requires the both two reference and hypothesis file are in either **trans** or **sgml** format. If you know how to convert one file into one of these above mentioned formats, please go ahead. Otherwise, you can follow these following instructions to generate it. Suppose that I will generate the **sgml** file from **100.hyp** (containing 100 hypotheses):

1 - Step 1 Check carefully the number of line of this file:

```
wc 100.hyp.id
 => 100.
```

2 - Step 2 Write a script look like this:

```
i=1;
while [ $i -le  100 ]; do
echo "<seg id=\""\$i"\">" >> file1
i=$(($i+1))
done
```

You will receive the file **file1** contains 100 lines, first will be <seg id="1"> and the last will be <seg id="100">

3 - Step 3

Modify a little bit the above script (or you can create another new script) like following:

```
i=1
while [ $i -le  100 ]; do
echo "</seg>" >> file1
i=$(($i+1))
done
```

This yields the file2 with 100 identical rows, each is </seg>

4 -Step 4

Combine the 3 file : file1 , 100.hyp and file2 to form the 100.hyp.id :

```
paste file1 100.hyp  file2 > 100.hyp.id
```

The received file is in the following format:

```
 <seg id="1">    another important step in the balkans   </seg>
<seg id="2">    since the world is focused on iraq , the north-korea and a
possible crisis with iran on the issue of nuclear weapons , kosovo is a
little unremarked .  </seg>
```

and so on...

5 - Step 5 Add the root open and close tag to the 100.hyp.id file

```
<hypset>
<seg id="1">    another important step in the balkans   </seg>
<seg id="2">    since the world is focused on iraq , the north-korea and a
possible crisis with iran on the issue of nuclear weapons , kosovo is a
little unremarked .  </seg>
.............................
</hypset>
```

6 - Step 6

May be your 100.hyp.file contains the symbol "&" which is illegal in XML format; so in this case TERP will generate a error like this : *"Both the reference and hypothesis file have to in the same format"*. To fix it, I suggest that you replace all the symbol "&" by "&*emp*;" , which exactly represents the "&" in an XML file. And then you will get the file ready for TER calculation:

```
sed 's/&/&amp;/g' < ./100.hyp.id > ./100.hyp.tmp
mv ./100.hyp.tmp ./100.hyp.id
```

Attention: Please do not forget that you need repeat from step 1 to 6 with the file 100.ref

The result display final TER score, for instance:

```
bin/terpa_quang -r ./test_terpa/100.ref.id  -h  ./test_terpa/100.hyp.id
Total TER: 0,63 (1643,00 / 2615,00)
```

The alignment information can be refered in the terp.pra (located at your current working directory).

Appendix C

Training and Testing WCE Classifier Using WAPITI

1 Introduction

Wapiti is a *"very fast toolkit for segmenting and labeling sequences with discriminative models. It is based on maxent models, maximum entropy Markov models and linear-chain CRF and proposes various optimization and regularization methods to improve both the computational complexity and the prediction performance of standard models"*(source: http://wapiti.limsi.fr). Wapiti is ranked first on the sequence tagging task for more than a year on MLcomp web site (http://mlcomp.org).

Compare to other Machine Learning models, Wapiti possesses some following advantages:

- Handle large label and feature sets
- L-BFGS, OWL-QN, SGD-L1, BCD, and RPROP training algorithms
- Powerful features extraction system
- N-best Viterbi output
- Compact model creation

To download it, you can visit this GIT repository: https://github.com/Jekub/Wapiti

2 Data Format and Feature Configuration File

The valid training and testing data files for Wapiti should conform the following regulations:

- Each instance (e.g. a sentence under the form of word sequence) is separated with the others by an empty line

- If the instance consists of multiple elements (e.g. words in a sentence), the data representing each element is organized in one line
- In each line, the data consists of two parts: the first part is the feature vector, features separated by a "tab" symbol; the second one is the annotated label of this element

For simplicity, we can imagine the data file looks like a matrix. Columns represent features and annotations. Rows represent data instances (elements). Blank lines are inserted to split instances. The following tables gives an example of our data, in the right format to train with Wapiti. It contains two instances (sentences). Each word in the sentence is characterized by features (8 first columns, they are only a part of our real feature vector) and the training label (always the last column).

yet	RB	B-encore	B-ADV	_x-1	_x-1	une	DET:ART	B
a	DT	B-une	B-DET:ART	encore	ADV	étape	NOM	B
crucial	JJ	B-cruciale	B-ADJ	étape	NOM	pour	PRP	G
step	NN	B-étape	B-NOM	une	DET:ART	cruciale	ADJ	G
for	IN	B-pour	B-PRP	cruciale	ADJ	les	DET:ART	G
the	DT	B-les	B-DET:ART	pour	PRP	balkans	NOM	G
balkans	NNS	B-balkans	B-NOM	les	DET:ART	_x+1	_x+1	G
since	IN	B-depuis|que	B-PRP|PRO:REL	_x-1	_x-1	le	DET:ART	G
the	DT	B-le	B-DET:ART	que	PRO:REL	monde	NOM	G
world	NN	B-monde	B-NOM	le	DET:ART	est	VER:pres	G
is	VBZ	B-est	B-VER:pres	monde	NOM	focalisé	VER:pper	G
focused	VBN	B-focalisé	B-VER:pper	est	VER:pres	sur	PRP	G
on	IN	B-sur	B-PRP	focalisé	VER:pper	l'	DET:ART	G
iraq	NP	B-l'|irak	B-DET:ART|NOM	sur	PRP	,	PUN	G
,	,	B-,	B-PUN	irak	NOM	la	DET:ART	G
north	JJ	B-la|du|nord	B-DET:ART|PRP:det|NOM	,	PUN	et	KON	G
korea	NN	B-corée	B-NOM	la	DET:ART	du	PRP:det	G
and	CC	B-et	B-KON	nord	NOM	une	DET:ART	G
a	DT	B-une	B-DET:ART	et	KON	éventuelle	ADJ	G
possible	JJ	B-éventuelle	B-ADJ	une	DET:ART	crise	NOM	G
crisis	NN	B-crise	B-NOM	éventuelle	ADJ	avec	PRP	G
with	IN	B-avec	B-PRP	crise	NOM	l'	DET:ART	G
iran	NN	B-l'|iran	B-DET:ART|NOM	avec	PRP	au	PRP:det	G
over	IN	B-au|sujet	B-PRP:det|NOM	iran	NOM	des	PRP:det	G
nuclear	JJ	B-nucléaires	B-ADJ	armes	NOM	,	PUN	G
weapons	NNS	B-des|armes	B-PRP:det|NOM	sujet	NOM	nucléaires	ADJ	G
,	,	B-,	B-PUN	nucléaires	ADJ	le	DET:ART	G
kosovo	NN	B-le|kosovo	B-DET:ART|NOM	,	PUN	passe	VER:pres	G
is	VBZ	B-passe	B-VER:pres	kosovo	NOM	un	DET:ART	B
somewhat	RB	B-un|peu	B-DET:ART|ADV	passe	VER:pres	inaperçu	ADJ	G
unnoticed	JJ	B-inaperçu	B-ADJ	peu	ADV	.	SENT	G
.	SENT	B-.	B-SENT	inaperçu	ADJ	_x+1	_x+1	G

Table B.1: Example of valid data file treated by Wapiti

Although the above data file lists all possible features available for use, but the *"real"* features taken into account will be determined in another configuration file. This file enables us to customize which features we want to train, or even lets us combine multiple columns in the data file to create new features. The way to represent them can be seen in Table B.2.
In the configuration file, each feature is placed on one line. We have two types of features: *"unigram"* and *"bigram"*. Each one is represented as follows:

```
[Prefix][Index]:%x[i, j]
where:
- Prefix is a letter to specify the feature type: U (unigram) or B (bigram)
- Index: the index of feature, to differentiate them, e.g. 00,01, etc.
- i: the relative position of the element we want to employ feature
with the current element. For instance: i=0 (the current token), i=-1 (the
previous token), i=1 (the next token).
-j : the zero-based position of the column which we extract data.
```

Let us give an example to illustrate the representation. Supposing that we are now considering the word: *"crucial"* in the following data block:

```
yet     RB      B-encore        B-ADV   _x-1    _x-1    une     DET:ART B
a       DT      B-une   B-DET:ART       encore  ADV     étape   NOM     B
crucial JJ      B-cruciale      B-ADJ   étape   NOM     pour    PRP     G
step    NN      B-étape B-NOM   une     DET:ART cruciale        ADJ     G
for     IN      B-pour  B-PRP   cruciale        ADJ     les     DET:ART G
the     DT      B-les   B-DET:ART       pour    PRP     balkans NOM     G
balkans NNS     B-balkans       B-NOM   les     DET:ART _x+1    _x+1    G
```

and we would like to extract some features for it like:

- The POS of its previous word (1)
- The second word after it (2)
- It's source word combined with the source word aligned to its previous word (3)

For feature (1): the previous word is located at line -1 and the POS is located at the second column (index 1), therefore its representation will be **U01:%x[-1,1]**. Similarly, feature (2) is **U02:%x[2,0]** (target word is the first column). For feature (3), we should use the symbol "/" to combine data, and the representation is **U03:%x[0,2]/x[-1,2]** (source word is the third column).

```
#Unigram
U00:%x[-2,0]
U01:%x[-1,0]
U02:%x[0,0]
U03:%x[1,0]
U04:%x[2,0]
U05:%x[-1,0]/%x[0,0]
U06:%x[0,0]/%x[1,0]
U07:%x[-2,0]/%x[-1,0]/%x[0,0]
U10:%x[-2,1]
U11:%x[-1,1]
U12:%x[0,1]
U13:%x[1,1]
U14:%x[2,1]
U15:%x[-2,1]/%x[-1,1]
U20:%x[-2,1]/%x[-1,1]/%x[0,1]
U21:%x[-1,1]/%x[0,1]/%x[1,1]
U22:%x[0,1]/%x[1,1]/%x[2,1]
U23:%x[0,0]/%x[-1,1]
U24:%x[0,0]/%x[1,1]
U25:%x[0,0]/%x[-2,1]
U26:%x[0,0]/%x[2,1]
U32:%x[0,2]
U33:%x[0,4]
U34:%x[0,6]
U35:%x[0,4]/%x[0,2]
U36:%x[0,2]/%x[0,6]
U37:%x[0,4]/%x[0,2]/%x[0,6]
U42:%x[0,3]
U43:%x[0,5]
U44:%x[0,7]
U45:%x[0,5]/%x[0,3]
U46:%x[0,3]/%x[0,7]
U47:%x[0,5]/%x[0,3]/%x[0,7]
U50:%x[0,4]/%x[0,0]
U51:%x[0,6]/%x[0,0]
U52:%x[-2,0]/%x[0,2]
U53:%x[-1,0]/%x[0,2]
U54:%x[0,2]/%x[1,0]
U61:%x[0,8]
U62:%x[0,9]
U63:%x[0,10]
U64:%x[0,11]
U65:%x[0,12]
U66:%x[0,17]/%x[-1,12]
U67:%x[0,13]
U68:%x[0,14]
U69:%x[0,15]
U70:%x[0,16]
U75:%x[0,17]
U76:%x[0,18]
U77:%x[0,19]
U78:%x[0,20]
U79:%x[0,21]
#Bigram
B
```

Table B.2: The Wapiti's feature configuration file

3 Commands and Options

The common format for all Wapiti modes (the modes can be: “train”, “label”) is:

```
wapiti mode [options] [input] [output]
```

A - Training Mode Options

```
      --me   Activate the pure maxent mode.
      -T | --type <string>
             Select  the  type  of  model  to  train. Can be either "maxent", "memm", or "crf", or "list" to
             get a list  of supported  models types. By default "crf" models are used.
      -a | --algo <string>
             Select the algorithm used to train the model, specify "list" for a list of available algorithms.
      -p | --pattern <file>
             Specify  the  file  containing  the patterns for extracting features.
      -m | --model <file>
             Specify a model file to load and to train again.
      -d | --devel <file>
             Specify the data file to load as a development set.
      -c | --compact
             Enable model compaction at the end of the  training.
      -t | --nthread <integer>
             Set  the  number  of thread to use. Default is 1.
      -j | --jobsize <integer>
             Set  the  size  of  the  job a thread will get each time it have
             nothing more to do.
      -s | --sparse
             Enable the computation of the forward/backward in sparse mode.
      -i | --maxiter <integer>
             Defines  the  maximum  number of iterations done by the training algorithm.
      -1 | --rho1 <float>
             Defines  the L1-component of the elastic-net penalty.
      -2 | --rho2 <float>
             Specifies the L2-component of the elastic-net  penalty. The default value is 0.00001.
       -w | --stopwin <integer>
             Set the window size for the devel stopping criterion. Default value is 5.
      -e | --stopeps <float>
             Set  the  size of the interval for stopping criterion. Default value is 0.02%.
      --clip Enables gradient clipping for the L-BFGS.
      --histsz <integer>
             Specifies the size of the history to keep in L-BFGS. The default is 5.
      --maxls <integer>
             Set  the  maximum number of linesearch step in L-BFGS to perform before giving up.
      --eta0 <float>
             Set the learning rate for SGD trainer.
      --alpha <float>
             Set the alpha value of the exponential decay in SGD trainer.
      --stpmin <float>
      --stpmax <float>
             The minimum/maximum step size  allowed  for the  RPROP trainer. Defaults are 1e-8 and 50.0.
```

B - Label Mode Options

```
Options (some most common used):
       --me   Activate the pure maxent mode.
       -m | --model <file>
              Specifies a model file to load and to  use  for  labeling.  This switch is mandatory.
       -l | --label
              With  this switch, Wapiti will only output the predicted label. Without, it output the full data
              with an additional column  containing the predicted labels.
       -c | --check
              Assume  the data to be labeled are already labeled so during the labeling process we can
              check  our  own  result  displaying  the error rates.
       -s | --score
              Output a line with score before the data.
       -p | --post
              Use posterior decoding instead of the classical  Viterbi  decoding.
       -n | --nbest <int>
              Output  the  N best sequences of labels instead of just the best one. The N sequences of
              labels are output in order in the output file
         --force
              Enable   forced  decoding  for  labeling  data  already  partly labelled.
```

Appendix D

Resumé

Les systèmes de traduction automatique (TA), qui génèrent automatiquement la phrase de la langue cible pour chaque entrée de la langue source, ont fait de bon progrès pendant les dernières décennies et apportent une aide aux utilisateurs dans un monde globalisé multilingue. Néanmoins, en raison de différents facteurs, sa qualité en général est encore loin de la perfection, accentuant le désir des utilisateurs de savoir le niveau de confiance qu'ils peuvent mettre sur une traduction spécifique. La construction d'une méthode qui est capable d'indiquer des bonnes parties ainsi que d'identifier des erreurs de la traduction, et donc d'estimer la qualité globale de chaque hypothèse apporterait un bénéfice pour non seulement les utilisateurs, mais aussi les traducteurs, post-éditeurs, et les systèmes de TA eux-mêmes. Nous appelons cette méthode un estimateur de mesures de confiance (MC). Les motivations de la construction de ces méthodes automatiques proviennent des inconvénients réels des mesures manuelles: elles sont chères en termes de temps et d'efforts humains, et parfois impossible dans le cas où les lecteurs manquent fondamentalement les connaissances de la langue source.

Cette thèse porte principalement sur les méthodes des MC au niveau des mots (MCM). Le système d'estimation de MCM assigne à chaque mot de la phrase cible une étiquette de qualité. Le mécanisme sur lequel ce système repose est simple: il s'agit d'un classificateur appris sur l'ensemble des paramètres en appliquant des méthodes d'apprentissage. Pour chaque mot dans la sortie de TA, il calcule les probabilités (scores de confiance) de tous les labels de qualité, et ensuite choisit celui avec le score le plus élevé comme résultat de classification.

Aujourd'hui, les MCM jouent un rôle croissant dans nombreux aspects de TA. Tout d'abord, elles aident les post-éditeurs à identifier rapidement les erreurs dans la traduction et donc à améliorer leur productivité de travail. De plus, elles informent les lecteurs des portions qui ne sont pas fiables. Troisièmement, elles sélectionnent la meilleure traduction parmi les sorties de plusieurs systèmes de TA. Finalement, et ce qui n'est pas le moins important, les scores MCM peuvent aider à perfectionner la qualité de TA des systèmes automatiques: réordonnance des

listes N-best (liste des N meilleure traductions), ré-décodage du graphe de recherche (GR), etc.

Dans cette thèse, nous visons à renforcer et optimiser notre système de MCM, puis à l'exploiter pour améliorer la qualité de TA ainsi que les mesures de confiance au niveau des phrases (MCP). Comparer avec les approches précédentes, nos nouvelles contributions étalent sur les points principaux comme suivants :

Tout d'abord, nous proposons et intégrons différents types des paramètres: ceux qui sont extraits du système TA, avec des caractéristiques lexicales, syntaxiques et sémantiques pour construire le système MCM de base. L'application et la comparaison entre les performances de différents méthodes d'apprentissage nous permettent d'identifier la meilleure (méthode: "Champs aléatoires conditionnels") qui convient le plus avec nos données.

Ensuite, l'efficacité de tous les paramètres est examinée en utilisant un algorithme heuristique de sélection des paramètres. Troisièmement, nous exploitons l'algorithme Boosting comme méthode d'apprentissage afin de renforcer la contribution des sous-ensembles des paramètres dominants du système MCM, et en conséquence d'améliorer la capacité de prédiction du système MCM. Ensuite, nous évaluons les contributions des MCM pour l'amélioration de la qualité de TA via différents scénarios. Pour le reclassement de la liste N-best, nous synthétisons les scores à partir des sorties du système MCM et puis les intégrons avec les autres scores du décodeur afin de recalculer la valeur de la fonction objective, qui nous permet de réordonner la liste pour l'obtention d'un meilleur candidat. Dans un second temps du ré-décodage du graphe de recherche, nous appliquons des scores de MCM directement aux nœuds contenant chaque mot pour mettre à jour leurs coûts. Une fois la mise à jour effectuée, la recherche du meilleur chemin sur le nouveau graphe nous donne la nouvelle hypothèse de TA. Enfin, les scores de MCM sont aussi utilisés pour renforcer les performances des systèmes de MCP.

La suite de ce résumé présente les points principaux qui sont discutés dans chaque chapitre de la thèse.

Dans le **Chapitre 1**, nous examinons les concepts et théories de base de la TA. Une brève histoire avec les réalisations remarquables pendant sa route de développement est résumée. Les architectures linguistiques et computationnelles de la TA ainsi que toutes les approches connues pour chacune sont mentionnés. Nous détaillons ensuite la TA statistique (TAS) - l'approche la plus populaire et réussie à ce jour, comprenant son modèle mathématique, suivi par les composants indispensables constituant chaque système de TAS: modèle de langage (ML), modèle de traduction et le décodeur. Par ailleurs, nous n'oublions pas de parler de certains outils et ressources utiles qui permettent de construire rapidement et pratiquement un système de TA de base. Les mesures manuelles et automatiques pour évaluer la qualité de la TA sont aussi décrites de ce chapitre.

Le **Chapitre 2** apporte une image complète des mesures de confiance au niveau des mots (MCM) – l'objectif principal de cette thèse. Le mécanisme de fonctionnement d'un système de MCM, ses éléments essentiels et ses applications principales sont tout d'abord présentés. Après avoir cité les travaux les plus importants liés aux MCM, nous nous concentrons sur deux problèmes: **"les paramètres utiles pour la construction d'un prédicteur MCM"** et **" les méthodes d'apprentissage pour entrainer les paramètres"**. Les paramètres largement utilisés dans les travaux précédents, tels que: les informations du côté de source (cible), ceux liés le contexte d'alignement, la probabilité postérieure du mot, le repli du modèle de langage, ceux fondés sur des étiquettes lexicales, et d'autres caractéristiques syntaxiques et sémantiques sont présentés. En outre, nous présentons des méthodes d'apprentissage que nous allons utiliser pour l'entrainement de nos modèles, parmi elles: Naïve Bayes, la régression logistique, l'arbre de décision, et les champs aléatoires conditionnelles. La dernière partie de ce chapitre présente les ateliers sur la TAS (WMT), au quel nous avons participé à la tâche de MCM pour les deux années 2013 et 2014.

Le **Chapitre 3** détaille notre construction du système de MCM de base ainsi que les expérimentations préliminaires pour estimer ses performances. Tout d'abord, notre ensemble des paramètres comporte tous ceux qui sont mentionnés dans le chapitre 2 et les nouveaux que nous avons extrait en exploitant plusieurs ressources, avec l'espoir qu'ils apportent des "connaissances supplémentaires" pour le modèle de classification. Ces paramètres proposés sont:

- Graph Topology: extraits à partir d'une listes des meilleurs hypothèses ('N-best list' en anglais) fusionnée dans un réseau de confusion, comprenant : Nodes (nombre de chemins alternatifs), Min et Max (la valeur min et max des probabilités postérieurs).
- Paramètres syntaxiques: Ils sont obtenus en utilisant les arbres syntaxiques comme la sortie de l'outil "Link Grammar Parser", sachant chaque phrase cible. Nous prenons en compte deux attributs utiles pour chaque mot: le label grammatical (constituant) et la distance avec la racine.
- Les pseudos références: En considérant la sortie de Google Translate comme une pseudo référence, nous obtenons un nouveau paramètre indiquant si chaque mot de la phrase cible se trouve dans cette référence.
- Les paramètres se basant sur le modèle de langage: nous construisons tout d'abord deux modèles de langage 4-gramme pour les deux cotés (source et cible). Après, nous comptons la longueur la plus grande possible du n-gramme couvert par le mot en courant et les mots précédents dans le modèle de langage de coté cible ainsi que coté source.

- Les paramètres se basant sur le modèle de langage des étiquettes lexicales (POS): nous comptons aussi la longueur la plus grande possible du n-gramme constitué par la séquence des POS du mot courant et ceux qui le précèdent.
- Présence dans plusieurs systèmes de référence: Ce paramètre a pour l'objectif de mesurer la qualité d'un mot en utilisant d'autres moteurs de traduction pour la même phase source. Si un mot de notre hypothèse se trouve dans la plupart des autres références, sa chance d'être une bonne traduction est plus élevée.

Nos systèmes de MCM de base sont construits sur trois corpus: (1) le corpus français – anglais (fr-en: 10881 paires, dont 10000 train et 881 test), (2) le corpus anglais - espagnol utilisé dans la campagne WMT2013 (en-es_13: 1087 paires = 750 train + 50 dev + 284 test), (3) le corpus anglais - espagnol utilisé dans la campagne WMT2014 (en-es_14: 2339 paires = 1957 train + 200 dev + 382 test). Par ailleurs, les étiquettes d'apprentissage (oracle) sont obtenues en appliquant l'outil TERp-A puis regroupées dans deux labels "G" (Bon) et "B" (mauvais). Afin d'entraîner les systèmes, nous utilisons la technique d'apprentissage intitulée "Champs aléatoires conditionnels" (CRF) et l'outil correspondant WAPITI. Dans les expérimentations préliminaires, tous les paramètres sont intégrés et chaque système est comparé avec deux "Baselines" naïves. Dans la baseline_1, tous les mots sont classifiés comme "G" et dans la baseline_2, ils sont labellisés aléatoirement dans les deux catégories en respectant les taux (ou la distribution) des labels G/B dans le corpus d'apprentissage.

Les résultats obtenus des trois systèmes en termes de scores de Précision, Rappel et F-score montrent une phénomène commune et très cohérente : les systèmes peuvent identifier les "bonne traductions" beaucoup mieux que les "mauvaises traductions ". De plus, l'utilisation de tous paramètres (l'ensemble entier permet d'améliorer fortement et clairement les performances des classificateur comparant avec les Baselines naïves.

Dans le **Chapitre 4**, nous proposons des techniques exploitées pour perfectionner les systèmes de MCM de base. Sachant que dans ces systèmes, toutes les paramètres sont combinées sans regarder la redondance à cause de certaines inutiles et faibles, dans ce chapitre, nous proposons une stratégie pour enquêter l'utilité de chacune pour chaque système, et en conséquence, retenir celles qui fonctionnent bien, ainsi qu'éliminer toutes les inutiles. Afin de faire ça, nous appliquons l'algorithme "Sequential Backward Selection". Le score pour comparer les systèmes et donc sélectionner les paramètres est une composition linaire entre le F-score du label "G" et celui de "B" en mettant les poids sur chaque élément dans lequel le F-score(B) a plus priorité. Les résultats nous permettent non seulement de mieux comprendre le rôle de chaque type de paramètre mais aussi d'éliminer ceux qui affaiblissent nos systèmes.

La deuxième méthode d'optimisation porte sur l'amélioration du système par renforcement ("Boosting" en anglais). Cette fois ci, nous aimerions exploiter en parallèle les deux côtés, comprenant les paramètres et les techniques d'apprentissage pour améliorer les performances du système. À partir de l'ensemble complet des paramètres, nous construisons plusieurs sous-ensembles afin d'entrainer les modèles faibles qui nous permettent de générer les données pour notre modèle Boosting. Nous avons utilisé la méthode "Arbre de décision" comme l'apprentissage du premier classificateur. Ensuite, le suivant est construit en prenant en compte les erreurs de classification fait par ce qui le précède. C'est pour l'objectif de ne pas répéter les mêmes erreurs dans les classificateurs d'avenir.

Enfin, tous les classificateurs sont combinés dans un système composé. Comme tous les résultats du système fr-en, en-es(WMT2013) et en-es(WMT2014) l'ont montré, cette méthode nous permet d'augmenter la capacité de la détection des erreurs (B) et, en même temps, de maintenir (ou d'augmenter aussi) la bonne performance déjà obtenue avec les bons mots (G). Ces résultats ont montré que le Boosting permet de construire un prédicateur composite "beaucoup plus fort".

Le **Chapitre 5** présente une première application des systèmes MCM : améliorer la performance du système de mesure de confiance au niveau des phrases (MCP). L'objectif de ce système est de prédire un score global indiquant la qualité (précision, cohérence, etc.) de la phrase entière, étant donné une phrase source. Notre idée est d'augmenter la précision dans les scores estimés en intégrant les labels de qualité pour chaque mot comme les paramètres supplémentaires, avec les paramètres au niveau des phrases qui ont été construites précédemment. Pour cela, nous construisons et testons sur trois systèmes de MCP pour les corpus fr-en et en-es_13. Le premier système (SYS1) est seulement un système pur MCP, c'est à dire ce qui est entraîné avec seulement des paramètres au niveau des phrases. Dans ce cas là, nous utilisons 17 paramètres de base qui ont constitué la Baseline dans la campagne WMT12. Dans le deuxième système (SYS2), nous synthétisons des paramètres depuis les labels de qualité au niveau des mots. En d'autres mots, c'est un système MCP basant sur MCM. Nous proposons sept paramètres suivant :

- Le ratio du nombre de mots corrects sur le nombre total de mots
- Le ratio du nombre de noms corrects sur l'ensemble des bons noms présents

Des paramètres similaires sont calculés pour d'autres étiquettes lexicales (verbes, adjectifs et adverbes).

Afin d'observer l'influence des scores MCM dans un système MCP, nous réalisons des expériences sur un troisième système **"SYS1+SYS2"**. Celui-ci prend les résultats prédits par **SYS1** et **SYS2**, et puis poste-traite les résultats pour choisir le label définitif. Autrement dit, le **SYS1+SYS2** récupère les deux scores issus de **SYS1** et **SYS2** pour chaque classe (G, B) et en calcule la moyenne. Finalement, les scores sont comparés et le label correspondant au score le plus élevé est sélectionné. Sur les corpus test, le système SYS2 fonctionne légèrement mieux que le **SYS1** en termes des mesures MAE et RMSE. Par contre, la combinaison (**SYS1+SYS2**) améliore clairement les performances initiales des deux systèmes . Les résultats montrent que les informations de MCM ont joué un rôle prédominant dans l'amélioration des systèmes de MCP.

Dans le chapitre 6, nous nous concentrons sur une autre contribution des systèmes de MCM pour la TA, avec pour objectif d'améliorer les traductions en réordonnant la liste de N meilleurs hypothèses (en anglais : N-best list). Comme nous l'avons déjà vu, les scores du moteur de traduction ne sont pas parfaits, et donc la première hypothèse de la liste N-best n'est pas forcément optimale. L'ajout de scores de confiance sur les mots des N-best est une mesure efficace pour faire émerger d'autres hypothèses candidates qui sont éloitgnées dans la liste. Le réordonnancement des N-best peut être considéré comme une seconde passe de décodage. À partir d'un décodeur de base de Moses, avec 14 scores par défaut, nous proposons des scores supplémentaires se basant sur des labels d'estimation de qualité des mots :

- le ratio du nombre de mots corrects sur le nombre total de mots
- le ratio du nombre de noms corrects sur l'ensemble des bons noms présents

Le ratio du nombre des séquences contenant n bons mots consécutifs sur la totalité des séquences de n mots (dans nos travaux, des séquences de 2,3 et 4 ont été utilisées).

Afin de vérifier l'efficacité des paramètres proposés, nous expérimentons trois systèmes de traduction. Le premier, **BL**, est la Baseline dans laquelle seulement 14 scores du décodeur sont utilisés pour ordonner les hypothèses. Dans le deuxième, **BL+WCE**, le système est enrichi par six scores supplémentaires extraits à partir des labels de confiance générés par notre système WCE réel. Le troisième, BL+WCE, contient les mêmes scores, mais les six scores ajoutés viennent d'un système WCE idéal (qui génère les labels oracles). Ici, **BL** est un décodeur 'single-passe' alors que les autres sont 'double-passe'. Les poids assignés aux paramètres sont optimisés via les méthodes: MERT (Minimum Error Rate Training) ou MIRA (Margin Infused Relaxed Algorithm). Les performances obtenues montrent que les scores oracles apportent informations très importantes qui aident à la sélection de meilleures traductions que la Baseline (5.79 points BLEU gagnés par MERT). Cependant, le rôle des scores réels de MCM est plus léger (avec une

légère amélioration de 0.46 point BLEU par MERT). En supposant que l'efficacité des scores MCM sera plus élevée quand le système de MCM sera plus performant, nous proposons des expérimentations supplémentaires dans lesquelles une partie (N%, N=25, 50 ,75) des mauvais labels générés par le système MCM de base est remplacée par les labels oracles. Ensuite, les scores de ré-ordonnancement sont tous recalculés, constituant trois nouveaux systèmes simulés **WCE+N%** (N=25, 50 ,75). Les nouvelles performances (0.68, 3.00 et 3.63 de points BLEU) montrent une forte corrélation entre performance du système MCM, et son rôle pour améliorer la qualité du réordonnancement des traductions.

La contribution la plus intéressante se trouve dans le dernier chapitre, Chapitre 7, dans lequel nous intégrons les scores MCM directement dans le graphe de recherche (GR) ("Search Graph" en anglais) afin de recalculer les coûts de tous les chemins contenant au moins un mot dans la N-best Liste. Le chemin de meilleur coût après cette mise à jour deviendra la nouvelle traduction. En appliquant cette méthode, nous pouvons élargir l'espace de recherche sur une et avons donc une meilleure chance de choisir la phrase la plus optimale. Pour ce faire, tout d'abord, nous utilisons notre système de MCM pour obtenir les labels de qualité (G, B) ainsi que les scores de confiance (probabilités) concernant tous les mots dans la N-best Liste. Ensuite, en se basant sur ces informations, pour chaque mot, nous modifions les coûts de tous les chemins dans le GR où ils apparaissent. Le coût d'un chemin peut être augmenté ou baissé en fonction de la qualité du mot. C'est à dire, si un chemin contient un bon mot, son coût devrait être baissé et dans le cas contraire, il sera augmenté. Enfin, nous cherchons sur le nouveau GR le meilleur chemin et générerons la meilleure nouvelle traduction. Nous définissons quatre types de score pour mettre a jours le GR :

- un score global : identique pour tous les mots et calculé en utilisant les labels (type 1a) ou la probabilité (type 1b) ;
- un score local : il dépend du coût du chemin actuel et du label du mot (type 2a) ou de sa probabilité (type 2b).

Comme dans le Chapitre 6, nous expérimentons plusieurs systèmes de traduction "double-passe" qui utilisent des scores se basant sur les labels réels du MCM ou les oracles. Sans considérer la baseline, nous avons quatre systèmes basant sur les labels (ou probabilités) qui viennent du système réel de MCM : **BL+WCE(1a,1b,2a,2b)** (les scores sont calculés par type 1a, 1b, 2a ou 2b). De plus, nous avons construit deux autres systèmes **BL+OR(1a, 2a)** dans lesquels les scores sont extraits a partir des labels de confiance oracle. Les performances de tous les systèmes sont affichées dans la Table A.1.

Systems	Performance			Comparison to BL			p-value
	BLEU ↑	TER ↓	TERp-A ↓	B (%)	E (%)	W (%)	
BL	52.31	0.2905	0.3058	-	-	-	-
BL+WCE(1a)	**53.80**	**0.2876**	**0.2922**	28.72	57.43	13.85	0.00
BL+WCE(1b)	53.24	0.2896	0.2995	26.45	59.26	14.29	0.00
BL+WCE(2a)	53.32	0.2893	0.3018	23.68	60.11	16.21	0.01
BL+WCE(2b)	53.07	0.2900	0.3006	22.27	55.17	22.56	0.01
BL+OR(1a)	**60.18**	**0.2298**	**0.2264**	62.52	24.36	13.12	-
BL+OR(2a)	59.98	0.2340	0.2355	60.18	28.82	11.00	-
BL+OR(NbestRR)	58.10	0.2551	0.2544	58.68	29.63	11.69	-
BL+WCE(NbestRR)	52.77	0.2891	0.3025	18.04	68.22	13.74	0.01
Oracle BLEU score	**BLEU = 66.48 (from SG)**						

Table A.1 – Translation quality of the conventional decoder and the 2-pass ones using scores from real or "oracle" WCE, followed by the percentage of better, equivalent or worse sentences compared to **BL** (**B**="Better", **E**="Equal", **W**="Worse")

À nouveau, les résultats obtenus (en terme de score BLEU) montrent que les scores MCM oracle ont amélioré fortement la qualité des traductions (par rapport à la BL) : 7.87 et 7.67 points gagnés par **BL+OR(1a)** et **BL+OR(2a)** respectivement. Concernant les labels réels de MCM, les améliorations sont moins impressionnantes, mais clairement significatives.

Parmi les quatres systèmes, **BL+WCE(1a)** est le meilleur avec 1.49 points BLEU gagnés. Les seuils de significativité (p-value) montrent que les points gagnés grâce aux scores de MCM sont significatifs. Pour mieux comprendre les contributions de cette méthode, nous comparons les performances des systèmes de réordonnancement de N-best Liste et le système deux passes. Le système BL+OR(1a) dépasse le **BL+OR(Nbest_RR)** de 2.08 points BLEU, tandis que le **BL+WCE(1a)** obtient de meilleurs résultats que **BL+WCE(Nbest_RR)** (+1.03 point de BLEU). Ce résultat s'explique qu'en réordonnant, les scores de MCM sont intégrés au niveau de la phrase (en calculant la moyenne), ce qui risque de ne pas pénaliser certaines erreurs de traduction. De plus, en réordonnant, la sélection de meilleure traduction est limitée par la taille de la liste N-best. Lors d'un redécodage l'espace est au contraire élargit à l'ensemble du graphe de recherche.