The Integration of Machine Translation and Translation Memory

Yifan He

B.A., M.A.

A dissertation submitted in fulfilment of the requirements for the award of

Doctor of Philosophy (Ph.D.)

to the

Dublin City University

School of Computing

Supervisors: Prof. Andy Way and Prof. Josef van Genabith

May 2011

I hereby certify that this material, which I now submit for assessment on the programme of study leading to the award of Ph.D. is entirely my own work, that I have exercised reasonable care to ensure that the work is original, and does not to the best of my knowledge breach any law of copyright, and has not been taken from the work of others save and to the extent that such work has been cited and acknowledged within the text of my work.

Signed:

(Candidate) ID No.:

Date:

Abstract

We design and evaluate several models for integrating Machine Translation (MT) output into a Translation Memory (TM) environment to facilitate the adoption of MT technology in the localization industry.

We begin with the integration on the segment level via translation recommendation and translation reranking. Given an input to be translated, our translation recommendation model compares the output from the MT and the TM systems, and presents the better one to the post-editor. Our translation reranking model combines k-best lists from both systems, and generates a new list according to estimated post-editing effort. We perform both automatic and human evaluation on these models. When measured against the consensus of human judgement, the recommendation model obtains 0.91 precision at 0.93 recall, and the reranking model obtains 0.86 precision at 0.59 recall. The high precision of these models indicates that they can be integrated into TM environments without the risk of deteriorating the quality of the post-editing candidate, and can thereby preserve TM assets and established cost estimation methods associated with TMs.

We then explore methods for a deeper integration of translation memory and machine translation on the sub-segment level. We predict whether phrase pairs derived from fuzzy matches could be used to constrain the translation of an input segment. Using a series of novel linguistically-motivated features, our constraints lead both to more consistent translation output, and to improved translation quality, reflected by a 1.2 improvement in BLEU score and a 0.72 reduction in TER score, both of statistical significance ($p < 0.01$).

In sum, we present our work in three aspects: 1) translation recommendation and translation reranking models that can access high quality MT outputs in the TM environment, 2) a sub-segment translation memory and machine translation integration model that improves both translation consistency and translation quality, and 3) a human evaluation pipeline to validate the effectiveness of our models with human judgements.

Acknowledgments

I would first like to thank my fantastic supervisors Prof. Andy Way and Prof. Josef van Genabith for their guidance in the course of my PhD research, and for pointing me to the emerging field of TM-MT integration. Their sharp insight and enormous support not only help to shape the work reported in this thesis, but also build outstanding examples for my future career. I should also thank my examiners Jesús Giménez and Dorothy Kenny for their insightful feedback that both make this thesis more solid and point out many possibilities for future work. I am grateful to Dr. Gareth Jones, Prof. Harold Somers, and Dr. Martin Crane who provided insightful suggestions on my transfer report, much of which is incorporated into this thesis. I thank Prof. Mikel Forcada for his insightful comments on our human evaluation approaches.

I am also indebted to the post-doctoral researchers I work with. First of all, most of the work reported in this thesis would be impossible without the collaboration with Yanjun Ma, who has always been a keen researcher and a warm-hearted friend. I am fortunate to have Jinhua Du to guide me into the field of machine translation, and to give me advice on both life and research when I first arrived in Ireland. I owe another big thank you to Sudip Naskar for advising me when I just began to write my first research papers.

I have an excellent bunch of colleagues in the MT group around me to build a great atmosphere for MT research. I am thus grateful to my current and past colleagues Jie Jiang, Junhui Li, Sara Morrissey, Pavel Pecina, John Tinsley, Antonio Toral, Xiaofeng Wu, Hala Al-Maghout, Hanna Bechara, Sandipan Dandapat, Tsuyoshi Okita, Robert Smith, Ankit Srivastava, Ventsislav Zhechev, and Patrick Lambert. I should especially thank Sergio Penkale, Pratyush Banerjee, and Rejwanul Haque, the people with whom I share the block in L2.08, as well as Marianna Apidianaki for being an excellent co-author. I received considerable help from the "larger" research group as well. I especially thank Jennifer Foster and Özlem Çetinoğlu for their help in dependency parsing.

This work would not be in its current shape without the unique academia-industry alliance setting provided by CNGL. I should first thank Johann Roturier from Symantec who

helped to conduct human evaluations in this thesis and provide comments from the industrial perspective. I should also thank Hilary McDonald for arranging demos to present this work to the industrial audience, and Páraic Sheridan, Steve Gotz, and Stephen Roantree for their valuable feedback on intellectual property and commercialization. I thank the linux guru Joachim Wagner for maintaining our computing cluster in excellent shape and consistently providing unix tips, as well as Ríona Finn, Fiona Maguire, and Eithne McCann for their kind help since the first day I arrived in Ireland.

I am very fortunate to have had the chance to visit the NLP group at Institute for Computing Technology, Chinese Academy of Sciences (CAS-ICT) for three months during my PhD study. I should thank Prof. Andy Way, Prof. Josef van Genabith, and Prof. Qun Liu for making this possible. I appreciate other faculties in the NLP group, including Prof. Yajuan Lü, Prof. Yang Liu, and Hongmei Zhao, for their advice and help during my visit. And it is the excellent colleagues Jinsong Su, Yang Feng, Jun Xie, Wenbin Jiang, Xinyan Xiao, Hui Yu, Zhiyang Wang, Hao Xiong, Zhaopeng Tu, Daqi Zheng, Tian Xia, Kai Liu, Wei Luo, and Miao Yu who both made my stay enjoyable and taught me much that I did not know.

Last but certainly not least, I hope to extend my deepest gratitude to my family, who have been supporting me through the ups and downs in the course of my education. And finally, to Miaomin Zhu, who makes this work meaningful.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Since the publication of [Brown et al., 1993], statistical machine translation (SMT) has made significant progress, both in terms of translation quality and ease of deployment and maintenance. SMT technologies are beginning to make inroads into the localization industry:[1] successful integration of SMT into localization workflows can help reduce the amount of human labor involved in localization and drive down costs.

Despite its promise, however, SMT has been embraced somewhat more reluctantly by some parts of the localization community than some SMT proponents may have hoped. There are several important reasons for this:

Firstly, translation memories (TMs), rather than machine translation, are the main-stay technology used in the localization industry. TMs are databases consisting of previously human translated segments. Given new text to be translated, the TM is searched for matching (source) text segments and the associated (human) translations are reused "recycled" in the jargon used in the localization industry). Given the repetitive nature of the (often) technical text processed in many localization workflows, TM hit rates can be up to 30% of new text to be translated (cf. e.g. [2]), and TMs can thus provide considerable savings. In the absence

[1] Localization is the industrial process of adapting digital content to culture, locale and linguistic environment. A core part of localization is translation of (often large amounts of usually technical) text. Localization is a global business with an estimated turnover of 12 Billion US $ in 2010 (Common Sense Advisory – Research and Consulting).

[2] `http://www.iai-sb.de/docs/aslib-js.pdf`, for reports of TM hit rates

of a full match, TMs provide a fuzzy match[3] facility, where the closest match in the TM given some input is retrieved and the translation associated with the closest match is presented to the professional human translator to be post-edited (i.e. adapted to a translation of the input), again with the potential for considerable savings over a manual translation from scratch. TMs thus represent considerable value and previous investment in translation, and TMs are assets that the industry does not want to abandon.

Secondly, in the localization industry, translation cost estimation is based on TM hit rates and fuzzy match scores, with full rates paid for segments which require translation from scratch (these are segments with low fuzzy match scores in the TM), reduced rates for segments with high fuzzy match scores that need to be post-edited and a small fee for proofing segments that have a full match in the TM. In contrast to TMs, SMT does not yet have a reliable translation cost estimation method, and this creates a difficulty for the industry to prepare accurate project plans.

Finally, acceptance of SMT (and other MT) technologies is still somewhat mixed, as some professional translators are reluctant to embrace new and unfamiliar technologies, especially if they are perceived as a potential threat to employment and/or human creativity.

TMs are used throughout the localization industry. First proposed by Kay [1980], this paradigm is well established, and has been serving translation professionals and the industry well (cf. Somers [2003]).

At the same time, advances in SMT have shown a strong potential to further improve the productivity of translators and post-editors, as SMT output is now quite acceptable for certain language pairs and applications, especially in domains where large parallel training corpora are available. Furthermore, SMT and TM technologies are complementary in that (i) SMT models can easily be trained on TM data; (ii) while TM translations are always fluent (they are, after all, human translations), for fuzzy matches TM translations are not actually translations of the input (but of the fuzzy match); (iii) while SMT output is not always fluent, it is a genuine attempt at translating the input; and (iv) unlike most SMT

[3] Usually a version of string edit distance.

technologies, TM technologies always support and closely integrate the human translator and post-editor into the translation workflow. Because of this, research on combining TM with SMT technologies is important. Ideally, such a combination should preserve what is best in the TM and SMT paradigms, exploiting their complementary strengths.

1.1 Research Questions

Given the crucial role of TMs in the workflows of localization industry, as well as the advancements of SMT systems in recent years, it is quite natural for us to firstly focus on using high-quality SMT outputs to enrich the TM environment, which leads to the first research question of this thesis:

> (**RQ1**) Can we provide translators with high-quality MT segments in a TM environment, without sacrificing the strengths of TMs?

Considering that most modern TM and SMT systems are able to produce k-best outputs, **RQ1** actually has to handle two sub-problems: 1) to enrich TMs with 1-best MT outputs, and 2) to enrich TMs with k-best MT outputs. We will handle both cases in this thesis.

In **RQ1**, we mainly consider TM-MT integration on the segment level. However, we can also integrate these two paradigms more tightly, on the sub-segment level, so that even when the whole TM segment is not good enough, we may still be able to reuse parts of it to improve translation consistency and quality. This leads to our second research question:

> (**RQ2**) Can we reuse sub-segment chunks from TMs to improve SMT consistency and quality?

Last but not least, we have to keep in mind that, while in MT research the focus is often on automatic evaluation metrics (e.g. to support parameter tuning), the TM-MT integration research requires human validation to support its effectiveness. After all, the purpose of TM-MT integration is to reduce the workload of human translators and the cost of localization vendors. This observation leads to the final research question of this thesis:

(**RQ3**) Can we validate our TM-MT integration models with human evaluation?

1.2 Thesis Structure

In this thesis we will tackle the research questions proposed in Section 1.1. We will also provide necessary background information on TM, MT, and translation quality estimation to make the thesis self-contained. We will present the material as follows:

In Chapter 2, we introduce the two paradigms used in the localization industry: the TM paradigm and the MT paradigm. We discuss how candidate translations are chosen or generated in these two paradigms, and we will show the strengths of each paradigm, namely the ability to reuse previously translated segments and to perform more reliable confidence estimation and cost estimation for the TM system, and the ability to produce fully automatic, high-coverage end-to-end translation for the MT system. We will briefly discuss how these two paradigms can complement each other.

In Chapter 3, we focus on existing quality estimation techniques for TM and MT systems. When integrating MT outputs into the TM environment, we are essentially comparing the quality of MT outputs with the TM outputs, and select the ones that are better. The methods and linguistic features used in translation quality estimation are a major inspiration of our work on TM-MT integration. Moreover, we analyze the DCU-DEP metric, a linguistically-inspired metric, as an example to show how linguistic features can be used to evaluate MT quality. We will use similar features for sub-segment TM-MT integration in Chapter 6.

In Chapter 4, we present the translation recommendation model which integrates TM and MT systems by automatically recommending 1-best MT outputs that are more suitable for post-editing to translators working in a TM environment (**RQ1**). We will show that the recommendation model has high precision, so that TM-based cost estimations are still valid as an upperbound if the recommendation model is applied. The recommendation model

can also produce a recommendation confidence score, on which the translators can set the threshold, and control how progressive/conservative the recommendations should be.

In Chapter 5, we extend the recommendation model in Chapter 4 to k-best lists of MT and TM system outputs. By reranking the k-best outputs from TM and MT systems, we provide a larger set of translation candidates for translators to choose from, and the translated segments in TMs will not be wasted, as they are all kept in the reranked k-best list.

In Chapter 6, we validate our models proposed in Chapter 4 and Chapter 5 with judgements provided by human translators (**RQ3**). We collect preferences of human translators and compare them with the recommendation and reranking produced by our models. We also analyze the behavior of the translators in the course of this user study, and hear their feedback. The results and user feedback will confirm the effectiveness of our models, and the necessity to perform TM-MT integration.

After tackling segment-level TM-MT integration in Chapters 4, 5, and 6, we move on to perform sub-segment level TM-MT integration in Chapter 7 (**RQ2**). We automatically select high quality chunks from TM fuzzy matches, and use them to constrain SMT. Experiments show that this approach not only ensures translation consistency, but also leads to a significant improvement in translation quality.

Finally, we summarize our work and point out avenues for future research in Chapter 8.

Chapter 2

Translation Memory and Statistical Machine Translation in Localization

In this chapter, we review two technologies used in the localization industry that help translators to finish their tasks more efficiently: Translation Memories (TMs) and Statistical Machine Translation (SMT). We also briefly discuss why and how we would propose an integrated paradigm that combines these two systems. More specifically, we will cover:

- The TM paradigm and the reason for its popularity in the localization industry.
- The SMT paradigm: its components, workflow, strength, and weakness.
- How SMT can potentially further improve the efficiency of translators, if they are properly integrated into the MT workflow.

2.1 Translation Memory

TMs are databases that store a translation history, i.e. source sentences and their translations as produced by humans. When there is a new segment to translate, a TM system will present the entry in the database to the translator, whose source side is most similar to the new segment.

This similarity is often measured using the fuzzy match score, which in turn is based on Levenshtein Distance [Levenshtein, 1966] as in (2.1):

$$FuzzyMatch(t) = 1 - \min_{e} \frac{LevenshteinDistance(s, e)}{Len(s)} \tag{2.1}$$

where s is the source side of the TM hit t, and e is the source side of an entry in the TM.

When exactly the same segment can be found (i.e. an exact 100% match), the translation of this segment can be directly reused, without any extra work, otherwise the translation retrieved from the database may still be used as a skeleton translation, which translators *post-edit* to produce the correct translation.

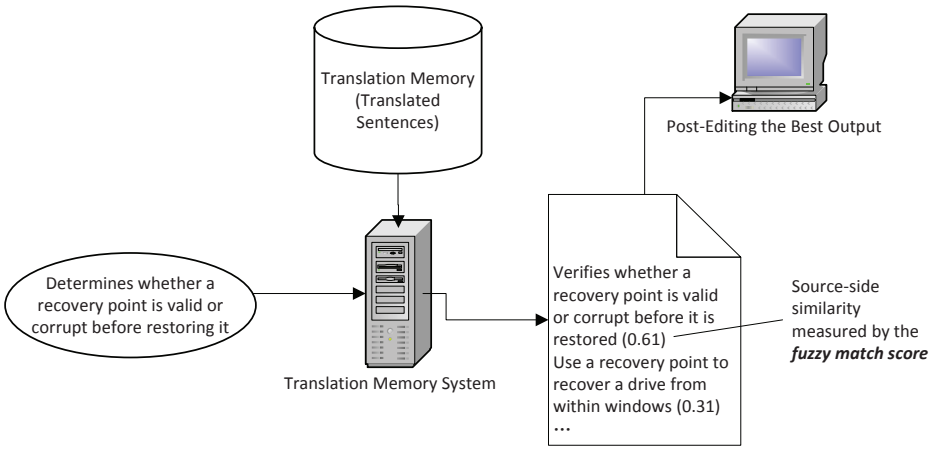

Figure 2.1: The TM Paradigm

We depict an example of this paradigm in Figure 2.1. If we have a source segment to translate:

> **Source Segment**:*Determines whether a recovery point is valid or corrupt before restoring it*

The TM system would query the TM consisting of previously translated sentence pairs, and would select the segment whose source side is most similar to this segment measured by the fuzzy match score. In our example, the following source segment will be selected,

with a fuzzy match score of 0.61:

> **The Fuzzy Match**:*Verifies whether a recovery point is valid or corrupt before it is restored*

The translator will be presented with this fuzzy match segment and its human translation, so that instead of translating from scratch, they only need to post-edit a human translated segment in French:

> **Translation of the Fuzzy Match**:*Vérifie si un point de récupération est valide ou endommagé avant la restauration.*

Usually the matched chunks in the source and fuzzy match segments (underlined in the examples) are color-coded or highlighted in the frontend computer-aided translation system for the translator to find where to post-edit. The translator will change *Vérifie* to *Détermine*, and finish translating this segment.

2.1.1 Advantages of the TM Paradigm

As we can see, although the TM paradigm could be as simple as querying a database and presenting the user with the most similar translated segment, it can significantly help the work of a translator with respect to the following aspects::

- **Leveraging legacy materials**. With translation memory, translators in the localization industry will not need to work on materials that have already been translated before. In turn, localization companies and customers do not need to pay for these materials. This significantly reduces the cost for the localization industry.
- **Estimating localization cost**. The fuzzy match score measures the source side similarity, and can thus be computed before translation actually begins. This helps localization vendors to effectively estimate the cost before they set out to work.

- **Friendly Computer-Aided Translation(CAT) environment**. The fuzzy matched chunks in a segment can be highlighted in the CAT environment, which helps the translators to find where to post-edit.

In the following sections, we further review the intuition and techniques behind the TM paradigm.

2.1.2 The Origin of the TM Paradigm

The TM paradigm emerged when localization and translation professionals began to realize the limitations of MT and realized the necessity of reusing previously translated material to reduce translation workload. In one of the earliest papers that inspired today's TMs, Kay [1980] analyzes the limitations of MT in both the cognitive/linguistic sense and the resource/computer science sense:

- **The Linguistic Point of View**. Kay [1980] uses the example of pronominal reference (anaphora resolution) in translation to illustrate the difficulty of making translation decisions. The large number of such problems renders it difficult for machines at that time to obtain high-quality translation without human intervention.

- **The Computer Science Point of View**. From the computer science point of view, Kay [1980] compares the complexity of dictionary search and translation, and conjectures that there will hardly be an efficient enough algorithm for MT at that time. It would be proved later in [Knight, 1999], that the problem of exact MT-decoding is NP-complete.

Although these arguments were made at a time when our understanding of computer science and the ability of hardware were much inferior compared to that of today, the major points still hold. Based on the above analysis, Kay [1980] proposes to build a human-centric paradigm, in which a computer begins by offering help to the translator on the lexical level. As more data is gathered during the translation process, the translator will later be able to

"call for a display of all the units in the text that contain a certain word, phrase, string of characters, or whatever", but the human translators can always intervene if the translation is of inferior quality.

In some sense, this thesis also follows the spirit of this proposal to build a "*translator's amanuensis*", but the work in this thesis now has access to SMT systems that are much more powerful than those 30 years ago.

2.1.3 TM Technologies

The success of modern TM systems – the extent to which this mechanism can help human translators – relies mainly on two technologies: 1) efficient storage and acquisition of existing translation data, and 2) fast and intelligent searching of the database.

2.1.3.1 Building and Exchanging Data

The success of a TM application depends very much on whether there is enough in-domain exact or high fuzzy match data in the database. It is reported that TMs are most useful when there is a large portion of exact matches (which often occurs when the translation task is to update an old version of a document to a new version), and TMs full of low fuzzy matches may well be useless [Sofer, 2006].

It is therefore very important to collect enough translation data for TMs to work properly. TM users have two options to obtain the database:

- **Internal Collection**. Obviously, the data can be collected in the translation process itself. This is preferable in many circumstances, because this way the information in the data is kept secure. However, it is a time consuming process, and it is quite natural for users to consider sharing some TMs.

- **Sharing and Exchanging**. Most of the TMs used in the industry today conform to the TMX (Translation Memory eXchange) format,[1] an XML-based format created

[1] `http://www.lisa.org/tmx/`

to encode TM information, and can be shared on professional localization web sites such as TDA[2]. Therefore, it is now entirely possible technically to share TM data with other parties.

Comparing these two approaches of TM data collection, sharing and exchanging can obviously collect required amounts of data more efficiently. But in the real world, not all TM data is suitable to share, and the translation industry still has to look for other methods that can improve translation efficiency.

2.1.3.2 Searching Techniques

Another factor affecting the performance of TMs is the search technique. The first consideration in searching is obviously speed, so that TM systems can retrieve the best fuzzy match in real time. This remains an area under active optimization. For example, in [Koehn and Senellart, 2010a], matching is first performed on the n-gram level to find the potential candidates, then A* search-based filtering is applied, and finally A* parsing (instead of directly computing the Levenshtein distance [Levenshtein, 1966]) is used to validate the matched segment. This is a typical example of the techniques used to ensure efficient searching in translation memories.

The other consideration is how fuzzy the source-side match can be. In the strictest sense, two words are considered to match only if these two words have exactly the same surface forms. Using our example in Section 2.1, words "*restoring*" and "*restored*" will not be considered as matched, because their forms are different.

However, now some TM systems (e.g. SDL Trados[3]) will give credit to partially matched words, so that Trados will consider "*restoring*" and "*restored*" as partially matched, and add a fraction into the segment level fuzzy match score.

[2] `http://www.translationautomation.com/`
[3] `http://www.sdl.com/en/language-technology/products/translation-memory/`

2.2 Statistical Machine Translation

Another paradigm that has the potential to aid the work of translators is MT. In contrast to TMs that facilitate *human* translation by reusing translated segments, MT systems aim to provide end-to-end translation solutions without human intervention.

Many approaches have been proposed for MT. One paradigm that has previously served translators is rule-based MT. Rule-based MT translates a source sentence to the target language by using hand-crafted transformation rules, and has the advantage of usually producing more grammatical and consistent translations (even in the sense that the translation errors are consistent, and are thus easier to identify in the post-editing process). When the hand-crafted rules do not cover the material being translated well enough, one can use statistical post-editing [Dugast et al., 2007], which automatically makes changes on the outputs of rule-based MT to further reduce potential workload, before the translation is finalized by human translators.

Although rule-based MT is still in active use in the localization industry, there is now a growing interest from the industry to leverage SMT systems in the workflow, with promising results. For example, Flournoy and Duran [2009] report that using the Language Weaver[4] SMT system, post-editing MT outputs achieves 4-fold speed-up in a pilot study to translate product documents compared to translating from scratch.

The interest and positive feedback on the SMT paradigm from the localization industry can be reduced to two reasons. From the translation quality perspective, SMT is able to provide translation for segments that TMs might not be able to cover, and from the cost perspective, the extra cost of introducing SMT into the localization workflow is reasonable.

- **Improved Coverage**. As we have discussed in Section 2.1.3.1, one of the challenges that the TM paradigm is facing is to construct an effectively large database of translated segments, otherwise many of the segments will be matched poorly and are less valuable for post-editing. Using an MT system can provide good translation candi-

[4] `http://www.sdl.com/en/language-technology/products/automated-translation/` (Now part of the SDL product line)

dates for these uncovered segments.

Moreover, statistical models used in SMT are language neutral, meaning that one can easily build SMT systems for any language pair as long as (general purpose) parallel corpora exist. Even for translations between low-resource languages for which the initial translation database hardly exists, it is still possible to use a high-resource language as a pivot and produce usable translations [Wu and Wang, 2007]. This property is growing in importance as the localization industry targets an ever-increasing number of languages.

- **Low Extra Cost**. Assuming that the translators are already using TM tools, the extra cost of introducing an extra SMT layer to reduce translation workload can be absorbed reasonably quickly by the cost it saves. For example, one can resort to third-party SMT services, such as Google[5] and Bing[6] translation, which are both provided as free services. It would therefore be worthwhile to test with these services, as long as the cost they save can compensate for the integration cost involved. Localization vendors can also use out-of-the-box open source toolkits such as Moses[7] to build in-house SMT systems with a small maintenance team, without having to continuously support bi-lingual grammarians capable of writing transformation rules to keep rule-based MT systems in good shape. In-house systems can be built using internal data, and can potentially save more translation cost than public translation services in the long run.

Although SMT has the potential to improve the localization workflow, it is unlikely that SMT output can be used without review, especially in applications where high quality translations are required. Furthermore, current state-of-the-art SMT also lacks a confidence estimation method as reliable as the fuzzy match score in TMs, and often is not integrated well enough in CAT tools. This thesis will therefore focus on the integration of SMT and

[5] http://translate.google.com
[6] http://translate.bing.com
[7] http://www.statmt.org/moses

TM, in which the strengths of both sides can be preserved.

In the following sections, we introduce the SMT paradigm in more detail.

2.2.1 The SMT Workflow

Given a source segment **f**, the SMT paradigm models the translation problem as the task of finding the translation **e** which maximizes the probability of **e** given **f**, as in (2.2):

$$\mathbf{e} = \arg\max_{\mathbf{e}} P(\mathbf{e}|\mathbf{f}) \tag{2.2}$$

However, the direct translation model in (2.2) rarely works well by itself, because the model is too coarse and the search space for the $\arg\max$ operator is too large. Therefore, (2.2) is usually formulated in terms of the noisy-channel model [Brown et al., 1993] using Bayes' theorem, as in (2.3):

$$\mathbf{e} = \arg\max_{\mathbf{e}} \frac{P_{TM}(\mathbf{f}|\mathbf{e})P_{LM}(\mathbf{e})}{P(\mathbf{f})} = \arg\max_{\mathbf{e}} P_{TM}(\mathbf{f}|\mathbf{e})P_{LM}(\mathbf{e}) \tag{2.3}$$

where P_{TM} is the translation model and P_{LM} is the language model probability. Note that the second equation in (2.3) is valid because when **f** is given, $P(\mathbf{f})$ becomes a constant and does not impact on the $\arg\max$ operator.

A further step in statistical modeling of MT comes from the intuition that using more features will help to improve translation quality, which leads researchers away from the noisy-channel model towards the log-linear translation model. In log-linear SMT [Och and Ney, 2002], $P_{TM}(\mathbf{f}|\mathbf{e})$ is further estimated using a log-linear combination of translation features. For example, in phrase-based SMT, the translation model is estimated using a combination of (direct and inverse) phrase translation probabilities, (direct and inverse) lexical translation probabilities, position- and lexical-based distortion probabilities, the word penalty and the phrase penalty, so that different aspects of translation choices (word translation, reordering, etc.) can be modeled directly and put together as a model of translation. Furthermore, these features are assigned different weights according to their importance

in the translation model, and a weight is also assigned to the language model. After this decomposition, the translation process can be represented by the $\arg\max$ operation in (2.4).

$$\mathbf{e} = \arg\max_{\mathbf{e}} \Pi_{i=1}^{n} \lambda_i P_{TM(i)} \lambda_{LM} P_{LM} \tag{2.4}$$

For convenience of computation and presentation, we usually take log on the right side of (2.4). Let $h_i = log(P_i)$, for each P in (2.4), and we can rewrite (2.4), as in (2.5).

$$\mathbf{e} = \arg\max_{\mathbf{e}} \Sigma_{i=1}^{n} \lambda_i h_i \tag{2.5}$$

Using the representation in (2.5), we can identify three iterative components in the SMT workflow:

- **Training** finds the feature functions h_i
- **Tuning** finds the weights for features λ_i
- **Evaluation**, or quality estimation, measures the quality of the output, and points out direction for further training and tuning.

After the models are built and the parameters are tuned, a *decoder* can decode new source sentences into their translations in the target language.

We depict the workflow of SMT systems in Figure 2.2: given a parallel corpus, we first train the language model and translation models. Then, based on some quality estimator, we tune the models to find a set of parameters. Using the models and parameters we decode the new sentences.

Suppose we need to translate the segment from Section 2.1 from English to French using SMT, as in (2.6):

Determines whether a recovery point is valid or corrupt before restoring it (2.6)

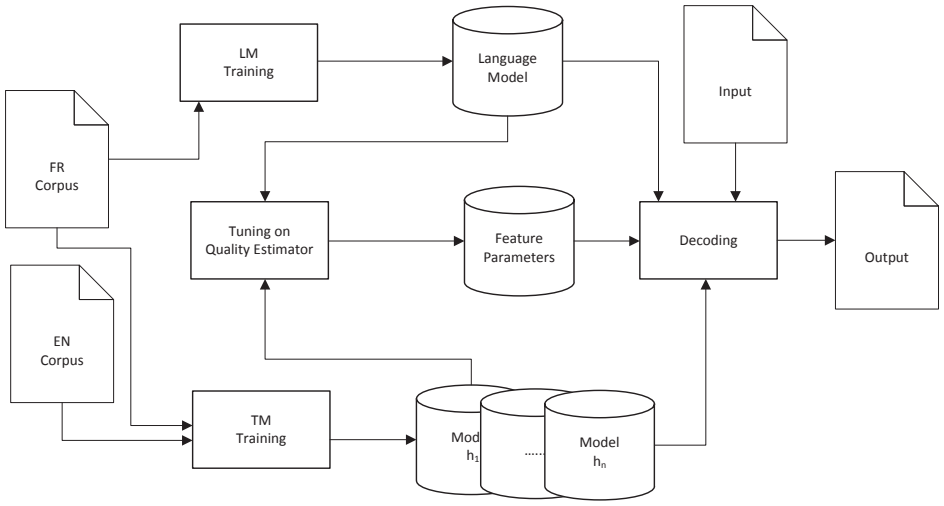

Figure 2.2: The SMT Workflow for EN-FR

First of all, we have to have the models and parameters ready. We train a French language model using the French corpus which ensures the fluency of our output (upper left of Figure 2.2). We also estimate a series of translation model feature functions using the parallel English–French corpus (lower left of Figure 2.2). When these features are ready, we tune the weights of these features against a development set (middle of Figure 2.2).

Now, suppose we have the feature functions and parameters ready for a phrase-based SMT system. The system will then split the source segment into several phrases, translate the phrases using the features, and re-combine them to produce the output. In the example, we have rules as in (2.7):

$$\begin{gathered} \textit{Determines whether} \mapsto \textit{détermine si} \\ \textit{a recovery point is} \mapsto \textit{un point de récupération est} \\ \textit{valid or corrupt} \mapsto \textit{valide ou endommagée} \\ \textit{before restoring it} \mapsto \textit{avant la restauration} \end{gathered} \tag{2.7}$$

And we obtain the translation by combing these translated phrases, and the SMT system will choose the phrasal translation and recombination that maximizes (2.5) as the output (with translation errors), as in (2.8):

détermine si un point de récupération est valide ou endommagée avant la restauration.

(2.8)

We discuss training, tuning, and quality estimation in more detail in following sections.

2.2.2 Training

In the context of modern SMT, *training* usually means the process of finding translation and language model feature functions, usually consisting of three components.

2.2.2.1 Language Modeling

Language Modeling estimates the language model probability P_{LM}. Most often, n-gram language models are used in SMT, which predict one word at a time based on the history of preceding words, following the Markovian assumption, as in (2.9)

$$P(w_1, w_2, \cdots, w_n) = p(w_1)p(w_2|w_1)\cdots p(w_n|w_1 w_2 \cdots w_{n-1}) \tag{2.9}$$

where $w_1 \cdots w_n$ is a sequence of n words, and the conditional probabilities $p(w_n|w_1 w_2 \cdots w_{n-1})$ are estimated using relative frequency, usually with some kind of smoothing (e.g. modified Kneser-Ney smoothing [Kneser and Ney, 1995]).

2.2.2.2 Word Alignment

Word alignment builds word-level correspondences between words in the source and their corresponding translations. Let a be an alignment function that maps the target word at position j to the source word at position i, as in (2.10):

$$i = a(j) \tag{2.10}$$

It follows that the word alignment process is to find an alignment a that maximizes $P(a|\mathbf{e}, \mathbf{f})$, as in (2.11).

$$a = \arg\max_{a} P(a|\mathbf{e}, \mathbf{f}) \tag{2.11}$$

Using the IBM word-based translation models [Brown et al., 1993], a can be found implicitly in an Expectation-Maximization [Baum, 1972] (EM) procedure that at the same time determines word-level translation probabilities. For ease of discussion, we use IBM Model 1 as an example. Besides, the IBM Model 1 alignment probability is also a feature used in our translation recommendation/reranking models. The notations and presentation in this section basically follows that of [Koehn, 2010], rather than that of [Brown et al., 1993].

IBM Model 1 is a word to word translation model, in the sense that the translation probability $P(\mathbf{e}|a, \mathbf{f})$ is estimated only via word translation probabilities $t(e_j|f_i)$, where e_j is the jth source word and f_i is the ith target word.

IBM Model 1 defines the translation **e** and alignment a given the source f as follows, as in (2.12):

$$P(\mathbf{e}, a|\mathbf{f}) = \frac{\epsilon}{(l_f + 1)^{l_e}} \Pi_{j=1}^{l_e} t(e_j|f_{a(j)}) \tag{2.12}$$

which is based on the product over all t, with $\frac{\epsilon}{(l_f+1)^{l_e}}$ for normalization, so that the probabilities can sum to 1.

Following this definition, we have:

$$P(\mathbf{e}|\mathbf{f}) = \Sigma_a P(\mathbf{e}, a|\mathbf{f}) = \frac{\epsilon}{(l_f + 1)^{l_e}} \Pi_{j=1}^{l_e} \Sigma_{i=1}^{l_f} t(e_j|f_{a(j)}) \tag{2.13}$$

Using (2.12) and (2.13), we can calculate $P(a|\mathbf{e}, \mathbf{f})$, as in (2.14):

$$P(a|\mathbf{e}, \mathbf{f}) = \frac{P(\mathbf{e}, a|\mathbf{f})}{P(\mathbf{e}|\mathbf{f})} = \Pi_{j=1}^{l_e} \frac{t(e_j|f_{a(j)})}{\Sigma_{i=0}^{l_f} t(e_j|f_i)} \tag{2.14}$$

Here we obtain the probability of a, which finishes the E-step in the EM procedure. On the other hand, we can also define a count function, based on which we can perform the

M-step, which re-estimates $t(e|f)$, as in (2.15):

$$c(e|f;\mathbf{e},\mathbf{f}) = \Sigma_a P(a|\mathbf{e},\mathbf{f})\Sigma_{j=1}^{l_e}\delta(e,e_j)\delta(f,f_{a(j)}) \tag{2.15}$$

where δ is the Kronecker function which is equal to 1 if $a = b$ in $\delta(a,b)$, and 0 otherwise. Then we can re-estimate $t(e|f)$, as in (2.16):

$$t(e|f;\mathbf{e},\mathbf{f}) = \frac{\Sigma_{(\mathbf{e},\mathbf{f})}c(e|f;\mathbf{e},\mathbf{f})}{\Sigma_e\Sigma_{(\mathbf{e},\mathbf{f})}c(e|f,\mathbf{e};\mathbf{f})} \tag{2.16}$$

Accordingly $c(e|f;\mathbf{e},\mathbf{f})$ and $t(e|f)$ can be iteratively determined in the EM procedure. In practical SMT, the probabilities estimated by IBM model 1 are too coarse, and are used to find good initial starts for higher order IBM models. However, the EM scheme does not change in these models. IBM model 1 can also be used as an estimator for word-to-word translation quality in MT quality estimation, as we do in this thesis.

One limitation of IBM models is that they only allow one-to-many alignment. To fix this, SMT developers usually merge alignments in two directions and apply some kind of heuristics, such as intersection and union [Och and Ney, 2003].

2.2.2.3 Translation Rule Extraction

Although it is possible to extract translation rules directly from corpora (e.g. phrasal translation rules in [Marcu and Wong, 2002]), most popular translation rule extraction techniques, both phrasal and syntactic, rely on the symmetric alignment between the source and the target sentences. We briefly review the phrase-based rule extraction method as an example as we mainly rely on phrase-based models to build MT systems in this thesis. Note that in TM-MT integration, the probabilities from the phrase-based models are also used as features to estimate translation quality.

In phrase-based translation models, the translation model is first decomposed into a phrasal translation model and a reordering model, as in (2.17):

$$P(\bar{f}_1^I|\bar{e}_1^I) = \Pi_{i=1}^I \phi(\bar{f}_i, \bar{e}_i) d(start_i - end_{i-1} - 1) \tag{2.17}$$

where ϕ is the phrasal translation model and d is the position-based reordering model. d can usually be estimated with a decay function in distance-based reordering models, such that $d(x) = \alpha^{|x|}$, where $\alpha \in [0, 1]$.

Estimating ϕ requires extracting phrase pairs from the symmetrically aligned corpus and calculating their relative count. The phrase pairs extracted have the constraint that they should be consistent with the alignment, such that given alignment a, if $\bar{e}$ is aligned to $\bar{f}$, all words from $\bar{e}$ that have alignment points in a should have their corresponding aligned words in $\bar{f}$, and vice versa. $\bar{e}$ and $\bar{f}$ should also contain at least one alignment point.

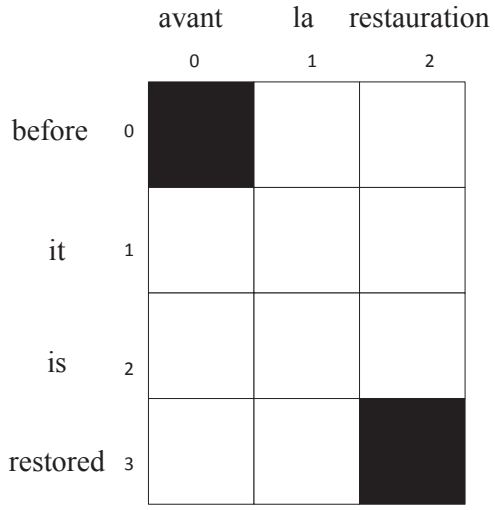

Figure 2.3: Phrasal Translation Rule Extraction

The idea of phrase pairs consistent with the alignment can be illustrated by the following example. Suppose we have the alignment in Figure 2.3.

In this example, if we start from alignment point (3, 2), we can find that *restored* $\mapsto$ *restauration* is a valid translation rule, as it corresponds to an alignment point. The following rules in (2.18) are also valid, because they are all consistent with alignment point (3, 2), and do not involve other alignment points:

$$\begin{gathered} \textit{it is restored} \mapsto \textit{la restauration} \\ \textit{is restored} \mapsto \textit{la restauration} \\ \textit{it is restored} \mapsto \textit{restauration} \\ \ldots \end{gathered} \tag{2.18}$$

However, the following two rules in (2.19) are not valid, as the first is inconsistent with the alignment point (0, 0), and the second does not cover any alignment point:

$$\begin{gathered} \textit{before it is restored} \mapsto \textit{la restauration} \\ \textit{it is} \mapsto \textit{la} \end{gathered} \tag{2.19}$$

After the phrases are extracted, the calculation of ϕ is straightforward using relative frequency, as in (2.20):

$$\phi(\bar{f}_i|\bar{e_i}) = \frac{count(\bar{f}_i, \bar{e_i})}{\Sigma_f count(\bar{f}, \bar{e_i})} \tag{2.20}$$

This estimation does not perform any smoothing, and is therefore prone to bias. There is evidence that smoothing translation rule probabilities can further improve the performance of SMT [Foster et al., 2006, Duan et al., 2010]. However, in our integration models we still stick to the unsmoothed probabilities which are more widely used.

2.2.3 Tuning

Given the translation feature functions $h_{1\cdots n}$, their weights $\lambda_{1\cdots n}$ can be determined in a discriminative learning process, the most popular of which is Minimum Error Rate Training (MERT). MERT [Och, 2003] tunes the weights λ_i of the features h_i in (2.5) to minimize the error function on the error surface of the N-best list of a development (or 'dev') set, as in (2.21):

$$\lambda = \arg\min_{\lambda} Err(e^*(\lambda); \mathbf{ref}) \tag{2.21}$$

where e^* is the 1-best translation. In practice, the function Err is actually approximated by a specific automatic evaluation metric E, in which case MERT is actually optimizing on (2.22):

$$\lambda = \arg\min_{\lambda} err_E(e^*(\lambda); \mathbf{ref}) \tag{2.22}$$

where err_E in (2.22) is a specific automatic evaluation metric used to approximate Err in (2.21). Och [2003] uses an improved version of Powell's line search to find the optimal λ. Besides MERT, new training schemes such as the Margin Infused Relaxed Algorithm (MIRA: Crammer et al. [2006]) have been introduced to MT (cf. [Watanabe et al., 2007], [Chiang et al., 2008] and [Chiang et al., 2009]), so that more features can be tuned.

2.2.4 The Role of Quality Estimation in the MT Workflow

The techniques used for translation quality estimation will be discussed in more detail in Chapter 3. However, quality estimation has some direct impact on the development of MT itself, and is essential to the success to some of the MT tasks, such as tuning and reranking, which we discuss below.

- **Tuning**. As is shown in (2.22), MERT relies on the choice of error function err_E. In practice, BLEU [Papineni et al., 2002] is often used as the error function, despite the fact that it has been shown to have a lower correlation with human judgement than other metrics such as METEOR [Banerjee and Lavie, 2005] and TER [Snover et al., 2006]. It is shown in [Cer et al., 2010] that when presented with multiple references, tuning on BLEU leads to more consistent results than tuning on other metrics. However, as we reported in [He and Way, 2009], tuning on BLEU is not that stable when only a single reference is available.

- **Reranking**. Another aspect where quality estimation techniques have a direct impact on SMT performance is reranking. The idea behind reranking is to take the N-best outputs from an SMT system, judging them with some quality estimation method,

and selecting the best translation from this N-best list. Reranking is shown to lead to significant improvements in translation quality [Shen et al., 2004]. It is quite clear that the performance of the reranking process is determined by the size, quality, and diversity of the N-best list, as well as how well the quality/confidence estimation metric can capture the quality of these candidate translations.

- **Implications for TM-MT Integration**. The impact of quality estimation methods on SMT performance has a considerable impact on TM-MT integration. TM-MT integration also relies on accurately determining the quality of translations (in fact, one of our integration models performs reranking on a combined k-best list of TM and MT outputs, much like SMT reranking). With this in mind, the TM-MT integration models presented in this thesis formulate many integration problems as quality comparison or quality ranking problems, and follow many of the standard practices in MT quality estimation.

2.3 The Convergence of TM and SMT Paradigms

In the previous sections, we reviewed the both the TM and the SMT paradigms. Both paradigms have strengths and weaknesses, as we enumerate in Table 2.1.

Table 2.1: Comparison of the TM and the SMT paradigms

	SMT	TM
Process	Fully automatic	Computer assisted human-translation
Adequacy	Real translation	Not translation per se
Fluency	No guarantee	Human translation
Environment	N/A	Color-coded
Cost Estimation	N/A	Fuzzy match score
Investment	MT software	Human translation collection

We see that the TM paradigm has several advantages that SMT systems currently cannot provide, such as color-coded post-editing environments and fuzzy match-based localization cost estimation. However, we can also see that SMT systems can complement some of TM's shortcomings (especially on coverage) and improve localization efficiency by providing au-

tomatic end-to-end translation to any input segment. This leads us to devise mechanisms that can help translators to access SMT outputs in the TM environment, which would preserve the strengths of TMs and leverage the advances of SMT.

2.4 Summary

In this chapter, we reviewed two paradigms that facilitate translation tasks using computing technology. In the TM paradigm, the system queries a database of previously translated segments and sends them back to translators for post-editing. In an MT system, the end-to-end system translates the segment without human intervention. We show that TM paradigms have several attractive properties for the localization workflow, but if we introduce MT outputs into the pipeline, we can potentially obtain better efficiency as we will have better coverage on the localization material.

We also looked at the features TM systems and (phrase-based statistical) MT systems use to find the best translation: the fuzzy match score and the translation and language model features. These features will be used as a starting point in our TM-MT integration research.

Based on the analysis of the TM and the MT paradigms, we present our proposal to perform TM-MT integration by integrating MT outputs into the TM environment. We will discuss the details of this proposal in Chapters 4, 5, 6, and 7.

Chapter 3

Translation Quality Estimation

In this chapter, we present existing technologies in the field of MT quality estimation. We briefly describe both methods using surface-level features and methods trying to apply deep features. We analyze the DCU-DEP metric as an example, and discuss potential pros and cons of surface and deep features. These insights will help us to design better features to integrate TM and MT.

After demonstrating existing technologies, we discuss the potential of combining the best from both TM and MT, on a segment or sub-segment level, by automatically choosing the translation segment/chunk of better quality using feature functions inspired by translation quality estimation. Finally, we present the blueprints for our TM-MT integration models based on the techniques we review.[1]

3.1 From Human to Automatic Estimation of Translation Quality

Developers and users of TM or MT systems rely on quality estimation techniques to quickly and easily estimate the quality of an MT output. Arguably, the ideal estimation method is judgement made by bilingual translators, as the effectiveness of an MT system (like all

[1] Part of the research presented in this chapter has been published in [He et al., 2010a].

systems) should eventually be judged by the people who use it, though human judges still have their limitations.

Firstly, the human judgement may vary from task to task. For example, for information retrieval applications, translation adequacy should be more important than grammaticality, while for post-editing, a half well-translated segment is much better than a translation that is correct in meaning but has grammatical errors scattered everywhere. Therefore, human judgement is not that consistent an evaluation measure.

Furthermore, human judges do not always agree with each other, making people question the reliability of human judgement results. In one evaluation task (WMT 2007), the inter annotator agreement of human judges measured by the Kappa score is 0.37 when ranking sentence pairs [Callison-Burch et al., 2007], suggesting only a *fair* correlation. This shows that human judges can reach a consensus quite often, but they also make conflicting decisions a substantial amount of times.

That said, human judgement is still the best resource we can resort to when we need to assess the quality of a translation, or validate an automatic quality estimation method. Very often, however, time and economic constraints render this option impossible. In such cases, automatic translation quality estimation methods have to be relied upon to obtain an approximation of output quality.

Automatic translation quality estimation methods can be categorized into two families:

- **Translation Evaluation Metrics.** For translation output (hypothesis) hyp, a source src and a set of human translations of src (references) ref, an MT evaluation metric m produces a metric score s_E, which aims to reproduce the scores given by bilingual human judges to hyp given src.

- **Translation Confidence Estimations.** For translation output hyp and a source src, a confidence estimation C produces a confidence score s_C, which aims to reproduce the scores given by bilingual human judges to hyp given src.

Here src, ref and hyp can be a sentence, a document, or a set of system outputs com-

prising several documents. Accordingly, quality estimation could happen at sentence-level, document-level and/or system-level. The most obvious difference between evaluation metrics and confidence estimations is that confidence estimations do not rely on human reference translations ref, but evaluation metrics do.

In the following sections, we first review confidence estimation methods used by the MT and the TM community, and then review MT evaluation metrics.

3.2 Target-Driven Translation Confidence Estimation in MT

Confidence estimation is the technique used to assess translation quality given the src and the hyp. However, the MT and the TM communities take very different approaches to the prediction of translation confidence.

Often, the focus of the MT community is to apply prior or posterior knowledge to predict the quality given a particular hyp. This strand of research was initiated by [Ueffing et al., 2003], in which posterior probabilities on the word graph or N-best list are used to estimate the quality of MT outputs. The idea is explored more comprehensively in [Blatz et al., 2004]. These estimations are often used to rerank the MT output and to optimize it directly. Extensions of this strand are presented in [Quirk, 2004] and [Ueffing and Ney, 2005]. The former experimented with confidence estimation with several different learning algorithms; the latter use word-level confidence measures to determine whether a particular translation choice should be accepted or rejected in an interactive translation system.

In the context of TM-MT integration, efforts have been made to incorporate confidence measures into a post-editing environment. To the best of our knowledge, the first paper in this area is [Specia et al., 2009a]. Instead of modeling on translation quality (often measured by automatic evaluation scores), this research uses regression on both the automatic scores and scores assigned by post-editors. The method is improved in [Specia et al., 2009b], which applies Inductive Confidence Machines (ICMs) [Vovk et al., 2005] and a larger set of features to model post-editors' judgement of the translation quality between "good" and

"bad", or among three levels of post-editing effort.

3.3 Source-Driven Translation Confidence Estimation in TM

The TM community, on the other hand, relies on the similarity of the source side to judge whether a translation retrieved from the TM database could be useful to translate a new segment.

The calculation of fuzzy match score itself is one of the core technologies in TM systems and varies among different vendors, but most often the calculation is based on Levenshtein Distance [Levenshtein, 1966], as in (3.1):

$$FuzzyMatch(s) = 1 - \min_{e} \frac{LevenshteinDistance(s, e)}{Len(s)} \tag{3.1}$$

where s is the input, and e is the source side of an entry in the TM.

Despite its simplicity, the fuzzy match score used in TMs offers a good approximation of post-editing effort, which is useful both for translators and translation cost estimation, while current SMT translation confidence estimation measures are not as robust as TM fuzzy match scores in this respect. Consequently professional translators are not yet ready to replace fuzzy match scores with SMT-oriented confidence measures.

3.4 MT Evaluation Methods

3.4.1 Surface-Level MT Evaluation

Many of the evaluation metrics used in day-to-day MT development are surface-level, or string-based metrics. Here we review three representative metrics: BLEU [Papineni et al., 2002], METEOR [Banerjee and Lavie, 2005], and TER [Snover et al., 2006], as they represent three different design considerations: BLEU uses n-gram precision to ensure translation fluency and fidelity; METEOR, by contrast, relies on unigrams and linguistic resources; and TER is modeled after post-editing operations, therefore TER scores can have the most intu-

itive interpretation for translation and post-editing tasks.

There are other string-based MT evaluation metrics that introduce novel string matching techniques and are of interest in the MT community, including GTM [Turian et al., 2003], which pioneered the idea of balancing precision and recall, ROUGE [Lin and Och, 2004], which models MT evaluation as the longest common subsequence matching, and MAXSIM [Chan and Ng, 2008], which fomulates MT evaluation as a bipartite graph match.

BLEU BLEU is the most popular evaluation metric in MT development. Although it suffers from several shortcomings, such as low correlation with human judgement on the sentence level, preference to statistical systems [Callison-Burch et al., 2006] and inconsistency in related evaluation scenarios [Chiang et al., 2008], it is still the most popular automatic evaluation metric used in many translation campaigns and remains the most often used loss function in discriminative training of MT models.

BLEU performs *n*-gram matching between the output and the reference, using *n*-gram precision with a brevity penalty as the score, as in (3.2):

$$\text{BLEU}(n) = \prod_{i=1}^{n} PREC_i{}^{\frac{1}{n}} \cdot exp(min(1 - \frac{len(ref)}{len(hyp)}, 0)) \tag{3.2}$$

where n is the order of n-gram, $PREC_i$ is the i-gram precision, $len(ref)$ is the length of the reference, and $len(hyp)$ is the length of the output. It has been shown in evaluation tasks [Callison-Burch et al., 2008] that BLEU has a lower correlation with human judgement than newer metrics that make use of more linguistic resources and better matching strategies, including METEOR and TER.

METEOR METEOR tries to solve the problems of BLEU by performing multi-stage unigram matching and adding recall into consideration. With the use of unigram matching, METEOR is less sensitive to variations in word order, and with multi-stage matching, METEOR can consider stemming and WordNet ([Fellbaum, 1998], currently for English only) semantic information. The METEOR score is calculated as in (3.3):

$$\text{METEOR} = \frac{PR}{\alpha P + (1-\alpha)R} \cdot (1 - cp) \text{ in which } cp = \gamma \cdot (\frac{\#chunks}{\#matches})^{\beta} \quad (3.3)$$

where P is the unigram precision, R is the unigram recall and cp is the chunk penalty, which is used to penalize disfluent outputs.

TER TER is a Levenshtein Distance-style evaluation metric. It calculates how many insertions, deletions, substitutions and sequence shifts are needed to make the output and reference token sequences identical. The difference between TER and the classical Levenshtein Distance [Levenshtein, 1966] is the sequence shift operation, which allows phrasal shifts in the hypothesis to be captured. TER is calculated as in (3.4). There is also a version of TER in which references are not predefined but created by the human annotators based on the MT output. This version (called HTER) measures post-editing effort directly.

$$\text{TER} = \frac{\#INS + \#DEL + \#SUB + \#SHIFT}{len(ref)} \quad (3.4)$$

One advantage of surface-level metrics is that they can be easily enhanced with lexical or shallow syntactic features, such as POS tags or paraphrases. For example, POS-BLEU [Popović and Ney, 2009], uses POS tags to enhance BLEU, while METEOR-NEXT [Denkowski and Lavie, 2010] and TERP [Snover et al., 2009] rely on paraphrases to improve the coverage of METEOR and TER, respectively. Using such resources leads to improved correlation with human judgement, as might be expected.

3.4.2 Deep Features in MT Evaluation

Some researchers have gone beyond the surface level and designed metrics that incorporate syntactic features. The first step in this direction was by Liu and Gildea [2005], who used syntactic structure and dependency information in order to see past the surface phenomena. Two of these metrics are based on matching syntactic subtrees between the translation and

the reference, and the third is based on matching headword chains, but only for *unlabelled* dependencies.

Since then, Owczarzak et al. [2007] have extended this line of research with the use of a term-based encoding of LFG *labelled* dependency graphs into unordered sets of dependency triples, and calculating precision, recall, and f-measure on the sets corresponding to the translation and reference sentences. With the addition of partial matching and *n*-best parses, [Owczarzak et al., 2007] considerably outperform [Liu and Gildea, 2005] with respect to correlation with human judgement. We will use an extension of [Owczarzak et al., 2007] as a case study in the contribution of surface/linguistic features in MT evaluation (cf. Section 3.5).

Instead of relying solely on one type of deep linguistic feature, some researchers evaluate and combine many heterogeneous linguistically motivated metrics. The best example of this strand of research is perhaps [Giménez and Màrquez, 2008, Giménez and Màrquez, 2010], where the linguistic analysis applied in MT evaluation includes constituency parses, dependency parses, semantic roles, and discourse representations. In their experiments, dependency parses and discourse representations all lead to promising correlation with human judgement.

3.4.3 Convergence of Surface and Deep Features in MT Evaluation

Given that both surface- and deep- level metrics have achieved promising correlation results in the literature, it is quite natural that researchers have begun to compare and combine these two approaches in search of even better MT evaluation metrics.

In one such effort, [Amigó et al., 2009] systematically compare the strength and weakness of n-gram and linguistic-driven metrics. They observe that linguistically motivated metrics can outperform n-gram metrics at system level and avoid rewarding poor translations that happen to have surface-level overlapping with the reference, as more linguistic constraints are introduced in the alignment process. They also show that a linear combination of these two types of metric can obtain the highest correlation with human judgement

among the metrics they have evaluated.

Besides explicit combinations, one can also apply features from different levels inherently by virtue of text entailment systems. Pado et al. [2009] evaluate translation outputs by examining whether the source and the reference entail each other. This metric is built upon the Stanford RTE system [Raina et al., 2005], and is also able to achieve state-of-the-art correlation performance.

The most obvious drawback of these methods is that, as they require a large amount of potentially computationally expensive linguistic analysis, they are thus often slow and resource-consuming. This renders these all-in-one metrics less useful in certain tasks, such as MT tuning. Such metrics are also more restricted to specific output languages.

3.4.4 Evaluation of Translation Quality Estimation

As mentioned in Section 3.1, when evaluated *intrinsically*, the performance of translation quality estimation can be assessed by how well it conforms to judgements by human raters. When comparing two MT outputs, we can calculate accuracy, precision, and recall by using human judgement as the gold standard. In this thesis, we apply these criteria to evaluate the quality of our integration model against judgements made by human translators.

Let A be the set of system outputs, and B be the set of gold standards. We standardly define precision P, recall R and F-value as in (3.5):

$$P = \frac{|A \bigcap B|}{|A|} \text{ and } R = \frac{|A \bigcap B|}{|B|} \text{ and } F = \frac{2PR}{P+R} \tag{3.5}$$

When rating more than two MT systems, the performance of a quality estimation technique is often measured by its correlation with human judgement. If we have gold standard human evaluation scores, we can compute Pearson's correlation [Hollander and Wolfe, 1999]. Given a sequence of quality estimation scores (such as automatic evaluation scores) $\mathbf{X} = \{x_1 ... x_i ... x_n\}$ and a sequence of gold standard scores (such as human evaluation scores) $\mathbf{Y} = \{y_1 ... y_i ... y_n\}$, we compute Pearson's correlation score, as in (3.6):

$$r = \frac{1}{n-1} \sum (\frac{x_i - \bar{\mathbf{X}}}{s_{\mathbf{X}}})(\frac{y_i - \bar{\mathbf{Y}}}{s_{\mathbf{Y}}}) \tag{3.6}$$

where x_i is the value of the i^{th} score, $\bar{X}$ is the mean score and s_X is the standard deviation. r is a real value in the range $[-1, 1]$. The value 0 implies that **X** and **Y** are independent, and 1 or -1 implies a perfect relationship (positively or negatively).

It is also possible to measure Spearman's correlation [Hollander and Wolfe, 1999] when only human rankings (instead of human scores) are available. Spearman's correlation is defined in (3.7), where d is the difference between corresponding values in rankings and *n* is the length of the rankings:

$$\rho = 1 - (\frac{6 \sum d^2}{n(n^2 - 1)}) \tag{3.7}$$

Another way to measure ranking correlation is Kendall's τ coefficient.

Kendall's τ measures the relevance of two rankings by comparing the number of concordant and discordant pairs in these rankings, as in (3.8)

$$\tau(r_a, r_b) = \frac{P - Q}{P + Q} \tag{3.8}$$

where P and Q are the amount of concordant and discordant pairs in r_a and r_b.

There is also the option to evaluate the performance of translation quality estimation *extrinsically*, which means evaluating it in specific use cases. For example, MT evaluation metrics or confidence estimation methods can be evaluated by how much they can boost the performance of MERT, or MT reranking. In this thesis, we also apply this type of evaluation, and we would evaluate how good our quality estimation is by measuring the improved translation quality/reduced post-editing effort obtained using quality-estimation based translation recommendation and reranking.

3.5 The DCU DEP-based Metric

In this section we present our extension to [Owczarzak et al., 2007] as an example of how the combination of surface and deep features can improve a pure syntax-based evaluation metric. Furthermore, many of the features used in this metric have been successfully applied in sub-sentential integration of TM and MT paradigms.

3.5.1 Background

Our DCU-DEP metric is based on [Owczarzak et al., 2007], which uses a term-based encoding of LFG (Lexical-Functional Grammar) *labelled* dependency graphs into unordered sets of dependency triples, and calculates precision, recall, and F-score on the sets corresponding to the translation and reference sentences.

The line of research is extended by the EDPM metric [Kahn et al., 2010] which uses arc labels derived from a PCFG parse to replace the LFG labels, so that a PCFG parser is sufficient for preprocessing. EDPM also incorporates more information sources: e.g. the parser confidence, the Porter stemmer, WordNet synonyms and paraphrases.

Besides these, information from the dependency parser is a component of some other metrics that use a larger knowledge source, such as the textual entailment-based metric [Pado et al., 2009].

Here we present another extension of the work of [Owczarzak et al., 2007]. We use the Stanford parser[2] to obtain Stanford dependencies and merge some labels whose granularity is too fine for the MT evaluation task. We incorporate the stemming, synonym and paraphrase information as in [Kahn et al., 2010], and at the same time we introduce a chunk penalty in the spirit of METEOR to punish discontinuous matches. We sort the matches according to the match level and the dependency type, and weight the matches to maximize the correlation with human judgement.

[2] `http://nlp.stanford.edu/software/lex-parser.shtml`

3.5.2 The Dependency-based Metric

In this section, we briefly review the metric presented in [Owczarzak et al., 2007]. The basic method can be illustrated by the example in Table 3.1.

Table 3.1: Sample Hypothesis and Reference

Hypothesis
rice will be held talks in egypt next week
Hyp-Triples
`nsubjpass(held, rice)`
`aux(held, will)`
`auxpass(held, be)`
`dobj(held, talks)`
`nn(week, egypt)`
`amod(week, next)`
`prep-in(talks, week)`
Reference
rice to hold talks in egypt next week
Ref-Triples
`nsubj(hold, rice)`
`aux(hold, to)`
`dobj(hold, talks)`
`nn(week, egypt)`
`nn(week, next)`
`prep-in(talks, week)`

The metric in [Owczarzak et al., 2007] performs triple matching over the Hyp- and Ref-Triples and calculates the metric score using the F-score of matching precision and recall. Let m be the number of matches, h be the number of triples in the hypothesis and e be the number of triples in the reference. Then we have the matching precision $P = m/h$ and recall $R = m/e$. The score of the hypothesis in [Owczarzak et al., 2007] is the F-score based on the precision and recall of matching, as in (3.9):

$$Fscore = \frac{2PR}{P+R} \tag{3.9}$$

3.5.3 Details of the Matching Strategy

Owczarzak et al. [2007] use several techniques to facilitate triple matching. First of all, considering that the MT-generated hypotheses have variable quality and are sometimes ungrammatical, the metric searches the 50-best parses of both the hypothesis and reference and uses the pair that has the highest matching F-score to compensate for parser noise.

Secondly, the metric performs *complete* or *partial* matching according to the dependency labels, so the metric will find more matches on dependency structures that are more informative.

More specifically, for all except the LFG `Predicate-Only` labeled triples of the form `dep(head, modifier)`, the method does not allow a match if the dependency labels (`dep`s) are different, thus enforcing a *complete* match. For the `Predicate-Only` dependencies, *partial* matching is allowed: i.e. two triples are considered identical even if only the `head` or the `modifier` are the same. Predicate-Only dependencies are those relations whose paths end in a predicate-value pair. The role of "predicate" in LFG does not have a direct correspondent in Stanford dependency notations. However, we allow partial matches on labels of the `arg` category, following the spirit of [Owczarzak et al., 2007].

Finally, the metric also uses linguistic resources for better coverage. Besides using WordNet synonyms, the method also uses the lemmatized output of the LFG parser [Cahill et al., 2004], which is equivalent to using an English lemmatizer.

If we do not consider the linguistic resources, the metric would find these matches in the example: `nn(week, egypt)`, `nn(week, next)` and `prep-in(talks, week)`.

We see several points for improvement from the above analysis:

- More linguistic resources. We can use more linguistic resources besides WordNet in pursuit for better coverage, such as a stemmer and paraphrases.

- Simplifying dependency labels. As is shown in Table 3.1, Stanford dependency labels are too fine-grained for our metric, which prevents matching `nsubjpass(held, rice)` to `nsubj(hold, rice)`, even if we use linguistic resources, since the

metric does not allow matching trigrams with different dependency labels.

- Boosting continuous matches. It would be more desirable to reflect the fact that the 3 matches currently found are continuous in Table 3.1.

We introduce our improvements to the metric in response to these observations in the following sections.

3.5.4 Capturing Variations in Language

3.5.4.1 Merging Stanford Dependency Labels

We saw in Section 3.5.2 that the granularity of Stanford dependencies does not fit our dependency-based metric very well. We identify three sets of dependency-types to merge: `subj`, `obj` and `prep`.

The Stanford parser gives a very detailed analysis of `subj` and `obj` dependencies (e.g. active or passive, nominal or clausal, etc.). Though this is preferable behavior of a parser, these details differentiate very similar dependency relations and prevent our metric from capturing useful correspondences. Therefore, we merge the dependency labels under `subj` and `obj`, respectively.

For the `prep` type, the Stanford parser differentiates between the actual preposition and labels such relations as, for example, `prep-in`, so the corresponding triples can match only if the preposition itself is correctly translated. We merge all these labels into a `prep` type.

3.5.4.2 Linguistic Resources

In [Owczarzak et al., 2007], lexical variations at the word-level are captured by WordNet. We use a Porter stemmer and a unigram paraphrase database to allow more lexical variations.

With these two resources combined, there are four stages of word-level matching in our system: *exact* match, *stem* match, *WordNet* match and unigram *paraphrase* match. The

stemming module uses Porter's stemmer implementation[3] and the WordNet module uses the JAWS WordNet interface.[4] Our metric only considers unigram paraphrases, which are extracted from the paraphrase database in TERP[5] using the script in the METEOR[6] metric.

3.5.5 Adding Chunk Penalty to the Dependency-based Metric

The metric described in [Owczarzak et al., 2007] does not explicitly consider word order and fluency. METEOR, on the other hand, utilizes this information through a chunk penalty. We introduce a chunk penalty to our dependency-based metric following METEOR's string-based approach.

Given a reference $r = w_{r1}...w_{rn}$, we denote w_{ri} as 'covered' if it is the head or modifier of a matched triple. We only consider the w_{ri}s that appear as `head` or `modifier` in the reference triples. Given this notation, we follow the approach taken in METEOR by counting the number of chunks in the reference string, where a chunk $w_{rj}...w_{rk}$ is a sequence of adjacent covered words in the reference. Using the hypothesis and reference in Table 3.1 as an example, the three matched triples `adjunct(talks, in)`, `obj(in, egypt)` and `adjunct(week, next)` will *cover* a continuous word sequence in the reference (underlined), constituting one single chunk:

rice to hold talks (in) egypt next week

Based on this observation, we introduce a similar chunk penalty Pen as in METEOR in our metric, as in (3.10):

$$Pen = \gamma \cdot (\frac{\#chunks}{\#matches})^{\beta} \tag{3.10}$$

where β and γ are free parameters, which we tune in Section 3.5.6.2. We add this penalty to the dependency-based metric (cf. (3.9)), as in (3.11).

[3] `http://tartarus.org/~martin/PorterStemmer/`
[4] `http://lyle.smu.edu/~tspell/jaws/index.html`
[5] `http://www.umiacs.umd.edu/~snover/terp/`
[6] `http://www.cs.cmu.edu/~alavie/METEOR/`

$$score = (1 - Pen) \cdot Fscore \tag{3.11}$$

3.5.6 Parameter Tuning

3.5.6.1 Parameters of the Metric

In this metric, dependency triple matches can be categorized according to many criteria. We assume that some matches are more critical than others and encode the importance of matches by weighting them differently. The final match will be the sum of weighted matches, as in (3.12):

$$m = \lambda_t m_t \tag{3.12}$$

where λ_t and m_t are the weight and number of match category t. We categorize a triple match from three perspectives:

- The level of match $L = \{complete, partial\}$
- The linguistic resource used in matching $R = \{exact, stem, WordNet, paraphrase\}$
- The type of dependency D. If we tune weights for each dependency type, there is the danger that we will overfit on the training data and our model will be very language-specific, so we choose to only discriminate between those that are $argument$ dependencies and those that are not, with $D = \{Arg, NoArg\}$.

Therefore for each triple match m, we can have the type of the match $t \in L \times R \times D$.

3.5.6.2 Tuning

In sum, we have the following parameters to tune in our metric: precision weight α, chunk penalty parameters β , γ and the match type weights $\lambda_1 ... \lambda_n$. We perform Powell's line search[7] on the sufficient statistics of our metric to find the set of parameters that maximizes

[7] Powell's line search optimizes an objective function by first searching along all directions, and then starting again at the linear combination of the optimum found in each direction.

Pearson's ρ on the segment level. We perform the optimization on the MT06 portion of NIST MetricsMATR 2010 development set (consisting of Arabic–English translations from 8 systems on 249 segments) with 2-fold cross validation.

3.5.7 Experiments

We experiment with different settings of the metric: WN-ONLY, WN-STEM-PARA(phrase), WN-STEM-PARA-TYPE and WEIGHTED, in order to validate our enhancements. The first two settings calculate F-scores using the linguistic resources suggested by their names. The third setting merges similar Stanford dependency labels (cf. Section 3.5.4.1) and the final setting uses weighted parameters. All words are lowercased for all settings.

We report Pearson's r, Spearman's ρ and Kendall's τ on segment and system levels using Snover's scoring tool.[8]

Table 3.2: Correlation on the Segment Level

	r	ρ	τ
WN-ONLY	0.606	0.636	0.212
WN-STEM-PARA	0.655	0.664	0.236
WN-STEM-PARA-TYPE	0.655	0.661	0.233
WEIGHTED	0.704	0.715	0.280

In Table 3.2, we see that by incorporating more linguistic resources into the dependency-based metric, we improve the metric's correlation with human judgement according to all correlation scores. The effect of simplifying dependency types is not that clear at system level, but parameter tuning almost boosts Pearson's r as much as linguistic resources. Although the parameters might somehow overfit the data set even if we apply cross-validation, this certainly confirms the necessity of weighting dependency matches according to their types.

When considering the system-level correlation in Table 3.3, the biggest difference to the results on the segment level is that it shows the validity of merging dependency labels: Pearson's r coefficients are close before and after label merging, but the ranking correlations are

[8] `http://www.umiacs.umd.edu/~snover/terp/scoring/`

Table 3.3: Correlation on the System Level

	r	ρ	τ
WN-ONLY	0.961	0.738	0.643
WN-STEM-PARA	0.977	0.881	0.786
WN-STEM-PARA-TYPE	0.978	0.929	0.857
WEIGHTED	0.959	0.929	0.857

much improved, suggesting that a simpler set of dependency labels could be more suitable to evaluate MT outputs. WEIGHTED match types lead to a slightly lower r at system level, but that does not affect ranking accuracy, as suggested by the ρ and τ coefficients.

3.5.8 Discussion

As we review the DCU-DEP metric, we can see that combining surface and deep level features can improve the performance of a syntax-oriented MT evaluation metric. Compared to other metrics, this metric is competitive at the system level, but not as competitive at the segment level. One of the reasons for this could be that on segment-level evaluation, the dependency-based metric filters out some valid matches, which has a negative impact on evaluation performance. On the system level, however, the larger amount of data compensates for the segment-level score fluctuation caused by the dependency-oriented matching scheme. This phenomenon is also observed by Amigó et al. [2009].

Although the improvements brought by dependency matching are not clear on MT evaluation, we suspect that they could be more useful when we need to predict the translation quality of sub-segment chunks, where many fewer lexical features can be explored (the chunks may not be long enough to constitute a valid n-gram, and ideas such as chunk penalty or longest match sequence will be less meaningful). In Chapter 7, we will see the application of dependency-based features in sub-segment translation quality estimation.

We also suspect that the dependency-based method would be more suitable for evaluating more structurally-related properties of translation, such as translation consistency, as is discussed in Chapter 7. Compared to evaluating just translation quality, translation consistency evaluation should also consider whether chunks of the same meaning and sim-

ilar grammatical functions have uniform translations. We will show that deep features can substantially help improve such prediction tasks.

3.6 Bringing the Two Worlds Together via Quality Estimation

As presented in the previous chapter, SMT has achieved huge improvements in recent years. This, in combination with the promising results achieved by recent MT quality estimation methods, leads us to consider the possibility of integrating high-quality MT outputs – in whole or in part – into TM outputs which are used actively by translators. Translation quality estimation plays two roles in this process. Firstly, translation quality estimation is essential in determining whether we should use segments/chunks from MT or from TM. TM-MT integration is only useful when translations having better quality can be selected automatically. Secondly, translation quality estimation is also necessary to provide a confidence score in TM-MT integration. The confidence score is needed as a replacement of the fuzzy match scores in the TM, when we choose to favor segments or chunks from the MT system.

3.6.1 Translation Confidence-Inspired Integration of TM and MT

The successful application of surface-level features in MT evaluation metrics suggests that the quality of translation can be estimated reasonably well even without deep features. In [Specia et al., 2009b], it is also shown that surface features are capable of generating confidence estimation scores for MT outputs.

Based on such evidence, we would first experiment with surface-level features on segment-level TM-MT integration. As our results show, using surface-level features – even if only those features derived from translation models – on the segment level can already achieve satisfactory results, especially on the recommendation task.

3.6.2 From Segment Level Integration to Sub-segment-Level Integration

When we move from segment-level to sub-segment level, however, the surface-level features begin to reach their limit. As we will see in Chapter 7, using only translation model features, such as those used in the segment-level TM-MT integration models, cannot lead to improvements. Therefore, we introduce a much more comprehensive feature set to model the sub-segment-level TM-MT integration, and show that using deep features indeed helps us to capture the properties of translation consistency in this setting.

3.7 Summary

In this chapter, we reviewed MT quality estimation methods, including techniques for human evaluation, automatic MT evaluation, and MT confidence estimation. We compared the use of surface- and deep-level features in MT quality estimation, and used the DCU-DEP metric as an example to put the discussion in context. We analyzed the pros and cons of this metric, and the idea of using linguistically-motivated features to predict translation quality will be applied again in Chapter 7.

Finally, based on the analysis of the TM and the MT paradigms, as well as quality estimation methods, we sketched our proposal to perform TM-MT integration on the segment or sub-segment level using techniques that are similar to MT quality estimation. We also hinted at the choice of surface- or deep-level features according to the characteristics of the integration. We will develop this sketch into a fully functional and human-validated integration scheme in Chapters 4, 5, 6, and 7.

Chapter 4

TM-MT Integration as Translation Recommendation

4.1 Introduction[1]

In this chapter, we begin the integration of TM and MT engines by focusing on the 1-best output of each system. In Table 4.1, we present an example segment from a Symantec translation memory, together with a reference translation produced by a human, and the outputs from the TM and the MT system. Note that a typical TM system will display both the source (**TM Source**) and the target (**TM Target**) side, as translators will use the alignment information (as aligned parts of the source segment are usually color-coded in TM systems) on the source side to identify the spans that need editing.

Table 4.1: An Example of TM and MT Output

Source	*Restore over existing virtual machines .*
TM Source	*Check restore over existing files .*
TM Target	*Cochez la case restaurer sur les fichiers existants .*
MT Output	*Restaurer des machines virtuelles existantes .*
Reference	*Restaurer sur les machines virtuelles existantes .*

In Table 4.1, the TM does not find a translation that is close in meaning to the source, but

[1] Part of the research presented in this chapter has been published in [He et al., 2010c]

there is some similarity between the input and the source side of the TM fuzzy match. From **TM Source**, the translators know that they probably do not need to adjust the translation for *restore*, and should instead pay attention to other parts of the segment. We can also see that in this case the MT output would be much easier to post-edit than the TM output.

In this chapter we present a translation recommendation model where translators can have access to MT segments that are more suitable to post-edit, without having to leave the TM environment, and can still use TM-based cost estimation as an upperbound. To achieve this, we estimate the relative quality of the TM output and the MT output, and present the one that is more suitable for post-editing (the MT output in this example) to translators.

We describe the elements of our translation recommendation model in the following sections: we present the translation recommendation paradigm in Section 4.2, and discuss the details of the paradigm in Section 4.3. We describe the features we use in our recommender in Section 4.4. We present experiments to test the performance of our recommender in Section 4.5, and approximate the reduced post-editing effort in Section. We review related work in Section 4.7 and summarize this chapter in Section 4.8.

4.2 The Translation Recommendation Paradigm

The example in Section 4.1 shows that sometimes current MT systems are capable of producing outputs that are more suitable for post-editing than TM hits. However, MT technology is sometimes adopted only slowly and somewhat reluctantly in the localization industry, because 1) TMs represent considerable effort and investment by a company or (even more so) an individual translator; 2) the fuzzy match score used in TMs offers a good approximation of post-editing effort, which is useful both for translators and translation cost estimation and, 3) current SMT translation confidence estimation measures are not as robust as TM fuzzy match scores and professional translators are thus not ready to replace fuzzy match scores with SMT internal quality measures.

It is therefore important to keep in mind that when integrating MT outputs into TM

systems, the original attractive properties of TMs should be kept intact. Our translation recommendation model presented in this chapter is designed to serve this purpose: given that most post-editing work is (still) based on TM output, we propose to recommend MT outputs which are better than TM hits to translators. In this framework, translators still work with the TM while benefiting from (better) SMT outputs; the assets in TMs are not wasted and TM fuzzy match scores can still be used to estimate (the upper bound of) post-editing labor.

There are three specific goals we need to achieve for the recommendation based TM-MT integration to work smoothly. Firstly, the recommendation should have high precision, otherwise it would be confusing for translators and may negatively affect the lower bound of the post-editing effort. Secondly, although we have full access to the SMT system used in this paper, our method should be able to generalize to cases where SMT is treated as a black-box, which is often the case in the translation industry. Finally, translators should be able to easily adjust the recommendation threshold to particular requirements without having to retrain the recommendation model.

Based on these requirements, we recast translation recommendation as a binary classification (rather than regression) problem using SVMs, perform RBF kernel parameter optimization, employ posterior probability-based confidence estimation to support user-based tuning for precision and recall, experiment with feature sets involving MT-, TM- and system-independent features, and use automatic MT evaluation metrics to simulate post-editing effort.

We depict the translation recommendation paradigm in Figure 4.1: both the TM and the SMT systems are used at the backend. When there is a new segment to translate, we compare the output from the TM and the MT system. Using an SVM-based classifier, we predict which of the two translations is more suitable for post-editing, along with a confidence score. In the TM environment, the translator can set a confidence threshold, and only MT outputs that are predicted to be better than the their TM correspondents with high confidence (above the threshold) will be presented to the translator. Otherwise the translator

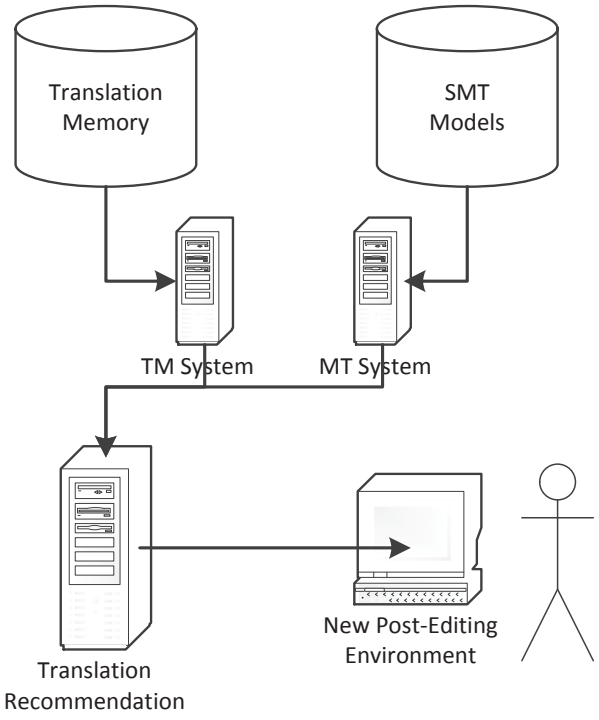

Figure 4.1: The Translation Recommendation Paradigm

will continue to use the TM output.

4.3 The SVM-based Recommendation Model

4.3.1 Support Vector Machines

We train an SVM binary classifier to perform translation recommendation between the TM and the MT output. SVMs [Cortes and Vapnik, 1995] classify an input instance based on decision rules which minimize the regularized error function in (7.5):

$$\begin{aligned} \min_{w,b,\xi} \quad & \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_{i=1}^{l}\xi_i \\ \text{s. t.} \quad & y_i(\mathbf{w}^T\Phi(\mathbf{x}_i) + b) \geqslant 1 - \xi_i \\ & \xi_i \geqslant 0 \end{aligned} \tag{4.1}$$

where $(\mathbf{x}_i, y_i) \in R^n \times \{+1, -1\}$ are l training instances that are mapped by the function Φ to a higher dimensional space. $\mathbf{w}$ is the weight vector, ξ is the relaxation variable and $C > 0$ is the penalty parameter.

Solving SVMs with Φ is performed by finding a kernel function K in (7.5) with $K(\mathbf{x}_i, \mathbf{x}_j) = \Phi(\mathbf{x}_i)^T \Phi(\mathbf{x}_j)$. We perform our experiments with the Radial Basis Function (RBF) kernel, as in (7.6):

$$K(\mathbf{x}_i, \mathbf{x}_j) = exp(-\gamma ||\mathbf{x}_i - \mathbf{x}_j||^2), \gamma > 0 \tag{4.2}$$

When using SVMs with the RBF kernel, we have two free parameters to tune on: the cost parameter C in (7.5) and the radius parameter γ in (7.6). In each of our experimental settings, the parameters C and γ are optimized by a brute-force grid search. The classification result of each set of parameters is evaluated by cross validation on the training set. Note that as we have a relatively small set of features, we rely on the ability of the RBF kernel to map the features to higher dimensional space. This is greatly facilitated using SVMs, where the tuning of C and γ is also important to obtain better prediction performance.

The SVM classifier will predict the relative quality of the MT output, and determine whether it is worthwhile presenting it to the post-editors instead of the TM output. The classifier uses features from the MT system, the TM and additional linguistic features to estimate whether the SMT output is better than the best hit from the TM. Ideally the classifier will recommend the output that needs the least post-editing effort. As large-scale human annotated data is not yet available for this task, we use automatic TER scores [Snover et al., 2006] as the measure for the required post-editing effort. In the future, we hope to train our system on HTER (TER with human-targeted references) scores [Snover et al., 2006] once the necessary human annotations are in place.[2] In the meantime we use TER, as TER is shown to have high correlation with HTER. This method is validated by our human

[2]While our Symantec data set was not annotated by post-editors, some small data sets do exist, e.g. `http://pers-www.wlv.ac.uk/~in1316/resources/datasets_ce_eamt.tar.gz`

evaluation (cf. Section 6.3.4).

We label the training examples as in (7.7):

$$y = \begin{cases} +1 & \text{if } TER(\text{MT}) < TER(\text{TM}) \\ -1 & \text{if } TER(\text{MT}) \geq TER(\text{TM}) \end{cases} \tag{4.3}$$

Each instance is associated with a set of features from both the MT and TM outputs, which are discussed in more detail in Section 4.4.

4.3.2 Recommendation Confidence Estimation

In classical settings involving SVMs, confidence levels are represented as margins of binary predictions. However, these margins provide little insight for our application because the numbers are only meaningful when compared to each other. What is more preferable is a probabilistic confidence score (e.g. 90% confidence) which is better understood by post-editors and translators.

We use the techniques proposed by Platt [1999] and improved by Lin et al. [2007] to convert the classification margin to a posterior probability, which is used as the confidence score in our system.

Platt's method estimates the posterior probability with a sigmoid function, as in (7.8):

$$Pr(y = 1|\mathbf{x}) \approx P_{A,B}(f) \equiv \frac{1}{1 + exp(Af + B)} \tag{4.4}$$

where $f = f(\mathbf{x})$ is the decision function of the estimated SVM. A and B are parameters that minimize the cross-entropy error function F on the training data, as in Eq. (7.9):

$$\begin{gathered} \min_{z=(A,B)} F(z) = -\sum_{i=1}^{l} (t_i log(p_i) + (1 - t_i) log(1 - p_i)), \\ \text{where } p_i = P_{A,B}(f_i), \text{and } t_i = \begin{cases} \frac{N_+ + 1}{N_+ + 2} \text{ if } y_i = +1 \\ \frac{1}{N_- + 2} \text{ if } y_i = -1 \end{cases} \end{gathered} \tag{4.5}$$

where $z = (A, B)$ is a parameter setting, and N_+ and N_- are the numbers of observed

positive and negative examples, respectively, for the label y_i. These numbers are obtained using an internal cross-validation on the training set.

4.4 The Feature Set

We use three types of features in classification: the MT system features, the TM feature and system-independent features.

4.4.1 The MT System Features

The MT system features are derived from the translation model of phrase-based SMT (cf. Chapter 2). We use:

- **Phrase-based Translation Model Scores**. Phrase-based translation model scores are the model scores proposed in [Koehn et al., 2003] as the translation model scores in phrase-based SMT. This includes the direct and reverse phrase translation probability and direct and reverse lexical translation probability.
- **The Language Model (LM) Probability**. This is the language model probability of the MT output.
- **The Distance-based Reordering Score**. This is the distance based reordering score estimated using a decay function in phrase-based SMT.
- **Lexicalized Reordering Model Scores**. These are the lexicalized reordering model scores. These scores estimate the probability of monotone, swap, or discontinuous reordering for a given phrase pair [Och et al., 2004].

In sum, by reusing the feature scores from the standard phrase-based SMT model, we are able to roughly predict the quality of the MT output. Although these features are not that powerful to predict the exact translation quality (otherwise MT reranking should always be able to correctly select the oracle translation, which is not the case, cf. [Shen et al., 2004]),

when combined with the fuzzy match cost feature from TM, we will be able to predict whether the TM or the MT output is of better quality.

4.4.2 The TM Feature

The TM feature is the fuzzy match [Sikes, 2007] cost of the TM hit. The calculation of fuzzy match score itself is one of the core technologies in TM systems and varies among different vendors. We compute fuzzy match cost as the minimum Levenshtein Distance [Levenshtein, 1966] between the source and TM entry, normalized by the length of the source as in (7.10), as most of the current implementations are based on edit distance while allowing some additional flexible matching.

$$h_{fm}(t) = \min_{e} \frac{LevenshteinDistance(s, e)}{Len(s)} \tag{4.6}$$

where s is the source side of t – the sentence to be translated – and e is the source side of an entry in the TM. For fuzzy match scores F, this fuzzy match cost h_{fm} roughly corresponds to $1 - F$. The difference in calculation does not influence classification, and allows direct comparison between a pure TM system and a translation recommendation system in Section 4.5.5.

4.4.3 System-Independent Features

Ideally, localization organizations will train their own MT and translation recommendation systems in order to obtain high quality in-domain translation outputs. However, there is still the choice of using a third party translation service, in which case the system-internal recommendation features from the SMT system will not be available.

To handle this situation, as well as to gather recommendation evidence from rich and varied sources, we use several features that are independent of the translation system, which are useful when a third-party translation service is used, or when the MT system is simply treated as a black-box:

- **Source-Side Language Model Score and Perplexity**. We compute the LM score and perplexity of the input source sentence on an LM trained on the source-side training data of the SMT system. The inputs that have lower perplexity or higher LM score are more similar to the dataset on which the SMT system is built.

- **Target-Side Language Model Perplexity**. We compute the LM probability and perplexity of the target side as a measure of fluency. Language model perplexity of the MT outputs is calculated, and LM probability is already part of the MT system's scores. LM scores on TM outputs are also computed, though they are not as informative as scores on the MT side, as TM outputs are human translations and should be grammatically perfect.

- **The Pseudo-Source Fuzzy Match Score**. We back-translate the output to obtain a pseudo source sentence. We compute the fuzzy match score between the original source sentence and this pseudo-source. If the MT/TM system performs well enough, these two sentences should be the same or very similar. Therefore, the fuzzy match score here gives an estimation of the confidence level of the output. We compute this score for both the MT output and the TM hit. This method is explored previously by Somers [2005] as an independent MT quality estimation measure. Although Somers [2005] does not recommend it as a stand-alone MT confidence estimation measure, we are using it along with other features to exploit useful information from back-translation.

- **The IBM Model 1 Score**. The fuzzy match score does not measure whether the hit could be a correct translation, i.e. it does not take into account the correspondence between the source and target, but rather only the source-side information. For the TM hit, the IBM Model 1 score [Brown et al., 1993] serves as a rough estimation of how good a translation it is on the word level; for the MT output, on the other hand, it is a black-box feature to estimate translation quality when the information from the translation model is not available. We compute bidirectional (source-to-target and

target-to-source) model 1 scores on both TM and MT outputs.

We will show in Section 4.5.3 that we are still able to obtain high recommendation performance only with the system independent features, so that our models still work if the MT system is used as a black box.

4.5 Experiments and Balancing Precision and Recall

We test the precision and recall of our recommendation model before evaluating its impact on post-editing effort to measure whether such a model can be learned well using the SVM framework. More thorough automatic and human evaluations are presented in Section 4.6 and Chapter 6.

4.5.1 Experimental Settings

Our raw data set is an English–French translation memory with technical translations from Symantec, consisting of 51K sentence pairs. This size is smaller than many parallel corpora that are used to train SMT systems, such as Europarl [Koehn, 2005], but it is comparable to the larger TMs used in the localization industry. We randomly selected 43K to train an SMT system and translated the English side of the remaining 8K sentence pairs. The average sentence length of the training set is 13.5 words. Note that we remove exact matches in the TM from our dataset, because exact matches will be reused and not presented to the post-editor in a typical TM setting.

As for the SMT system, we use a standard log-linear PB-SMT model [Och and Ney, 2002]: GIZA++ implementation of IBM word alignment model 4,[3] the refinement and phrase-extraction heuristics described in [Koehn et al., 2003], minimum-error-rate training [Och, 2003], a 5-gram language model with Kneser-Ney smoothing [Kneser and Ney, 1995] trained with SRILM [Stolcke, 2002] on the French side of the training data, and

[3]More specifically, we performed 5 iterations of Model 1, 5 iterations of HMM, 3 iterations of Model 3, and 3 iterations of Model 4.

Moses [Koehn et al., 2007] to decode. We train a system in the opposite direction using the same data to produce the pseudo-source sentences.

We train the SVM classifier using the libSVM [Chang and Lin, 2001] toolkit. The SVM-training and testing is performed on the remaining 8K sentences with 4-fold cross validation. We also report 95% confidence intervals.

The SVM hyper-parameters are tuned using the SVM training data of the first fold in the 4-fold cross validation via a brute force grid search. More specifically, for parameter C in (7.5) we search in the range $[2^{-5}, 2^{15}]$, and for parameter γ in (7.6) we search in the range $[2^{-15}, 2^{3}]$. The step size is 2 on the exponent.

4.5.2 The Evaluation Metrics

We measure the quality of the classification by precision and recall. Let A be the set of recommended MT outputs, and B be the set of MT outputs that have lower TER scores than the corresponding TM hits. We standardly define precision P, recall R and F-value as in (7.11):

$$P = \frac{|A \bigcap B|}{|A|}, R = \frac{|A \bigcap B|}{|B|} \text{ and } F = \frac{2PR}{P+R} \tag{4.7}$$

4.5.3 Recommendation Results

In Table 4.2, we report recommendation performance using MT and TM system features (SYS), system features plus system-independent features (ALL:SYS+SI), and system-independent features only (SI).

Table 4.2: Recommendation Results

	Precision	Recall	F-Score
SYS	82.53±1.17	96.44±0.68	88.95±.56
SI	82.56±1.46	95.83±0.52	88.70±.65
ALL	83.45±1.33	95.56±1.33	89.09±.24

From Table 4.2, we observe that MT and TM system-internal features are very useful for producing a stable (as indicated by the smaller confidence interval) recommendation system (SYS). Interestingly, only using some simple system-external features as described in Section 4.4.3 can also yield a system with reasonably good performance (SI). We expect that the performance can be further boosted by adding more syntactic and semantic features. Combining all the system-internal and -external features leads to limited gains in Precision and F-score compared to using system-internal features (SYS) only. This indicates that at the default confidence level of the recommendation system (0.5), current system-external (resp. system-internal) features can only play a limited role in informing the system when current system-internal (resp. system-external) features are available. Additionally, the performance of system SI is promising given the fact that we are using only a limited number of simple features, which demonstrates a good prospect of applying our recommendation system to MT systems where we do not have access to their internal features.

Table 4.3: Contribution of Features

	Precision	Recall	F Score
SYS	82.53±1.17	96.44±0.68	88.95±.56
SYS+M1	82.87±1.26	96.23±0.53	89.05±.52
SYS+LM	82.82±1.16	96.20±1.14	89.01±.23
SYS+PS	83.21±1.33	96.61±0.44	89.41±.84

4.5.4 Contribution of Features

In Section 4.4.3 we suggested three sets of system-independent features: features based on the source- and target-side LM, the IBM Model 1 (M1) and the fuzzy match scores on pseudo-source (PS). We compare the contribution of these features in Table 4.3.

In sum, all three sets of system-independent features improve the precision and F-scores of the MT and TM system features. The improvement is not significant, but improvement on every set of system-independent features gives some credit to the capability of SI features, as does the fact that SI features perform close to SYS features in Table 4.2.

4.5.5 Further Improving Recommendation Precision

Table 4.2 shows that classification recall is very high, which suggests that precision can still be improved, if recall can be compromised to some extent. Considering that TM is the dominant (and tried and trusted) technology used by post-editors, a recommendation to replace the hit from the TM by MT output should require high confidence, i.e. high precision.

4.5.5.1 Adjusting Confidence Levels

We output a confidence score during prediction and threshold recommendation on the confidence score.

We use the SVM confidence estimation techniques in Section 4.3.2 to obtain the confidence level of the recommendation, and change the confidence threshold for recommendation when necessary. This also allows us to compare directly against a simple baseline inspired by TM users. In a TM environment, some users simply ignore TM hits below a certain fuzzy match score F (usually from 0.7 to 0.8). This fuzzy match score reflects the confidence of recommending the TM hits. To obtain the confidence of recommending an SMT output, our baseline (FM) uses fuzzy match costs $h_{FM} \approx 1 - F$ (cf. Section 4.4.2) for the TM hits as the level of confidence. In other words, the higher the fuzzy match cost of the TM hit (lower fuzzy match score), the higher the confidence of recommending the SMT output. We compare this baseline with the three settings in Section 4.5.

Figure 4.2 shows that the precision curve of FM is low and flat when the fuzzy match costs are low (from 0 to 0.6), indicating that it is unwise to recommend an SMT output when the TM hit has a low fuzzy match cost (corresponding to higher fuzzy match score, from 0.4 to 1). We also observe that the precision of the recommendation receives a boost when the fuzzy match costs for the TM hits are above 0.7 (fuzzy match score lower than 0.3), indicating that SMT output should be recommended when the TM hit has a high fuzzy match cost (low fuzzy match score). With this boost, the precision of the baseline system can reach 0.85, demonstrating that a proper thresholding of fuzzy match scores can be used

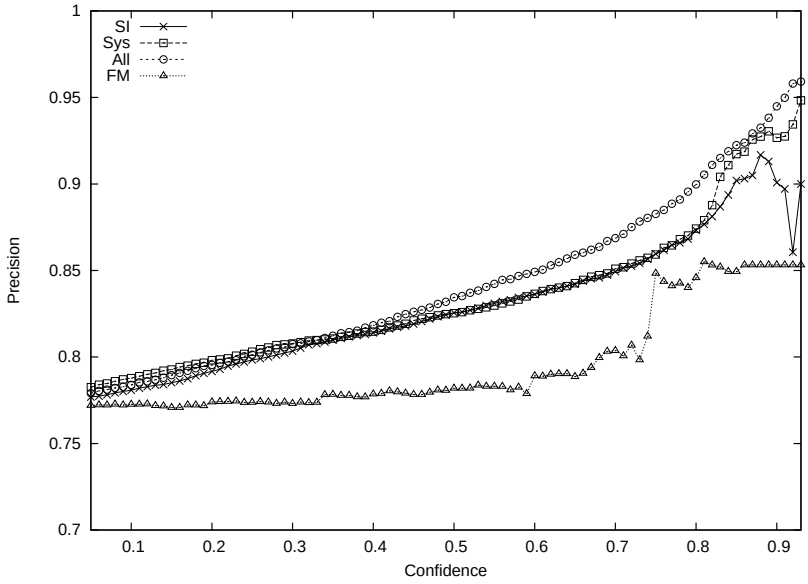

Figure 4.2: Precision Changes with Confidence Level

effectively to discriminate the recommendation of the TM hit from the recommendation of the SMT output.

However, using the TM information only does not always find the easiest-to-edit translation. For example, an excellent SMT output should be recommended even if there exists a good TM hit (e.g. fuzzy match score is 0.7 or more). On the other hand, a misleading SMT output should not be recommended if there exists a poor but useful TM match (e.g. fuzzy match score is 0.2 or below).

Our system is able to address these complications as it incorporates features from the MT and the TM systems simultaneously. Figure 4.2 shows that both the SYS and the ALL settings consistently outperform FM, indicating that our classification scheme can better integrate the MT output into the TM system than our naive FM baseline. The advantage of our method over the TM-cutoff-based FM baseline is further confirmed by human evaluation (cf. Chapter 6).

The SI feature set does not perform well when the confidence level is set above 0.85 (cf. the descending tail of the SI curve in Figure 4.2). This might indicate that this feature set is not reliable enough to extract the best translations. However, when the requirement

on precision is not that high, and the MT-internal features are not available, it would still be desirable to obtain translation recommendations with the black-box SI features. The difference between SYS and ALL is generally small, but ALL performs steadily better in the range [0.5, 0,8].

Table 4.4: Recall at Fixed Precision

	Recall
SYS @85PREC	88.12±1.32
SYS @90PREC	52.73±2.31
SI @85PREC	87.33±1.53
ALL @85PREC	88.57±1.95
ALL @90PREC	51.92±4.28

4.5.5.2 Precision Constraints

In Table 4.4 we also present the recall scores at 0.85 and 0.9 precision for SYS, SI and ALL models to demonstrate our system's performance when there is a hard constraint on precision. Note that our system will return the TM entry when there is an exact match, so the overall precision of the system in a typical mature TM environment is well above the precision score we set here, as a significant portion of the material to be translated will have a complete match in the TM system.

In Table 4.4 for MODEL@K, the recall scores are achieved when the prediction precision is better than K with 0.95 confidence. For each model, precision at 0.85 can be obtained without a very big loss in recall. However, if we want to demand further recommendation precision (corresponding to a more conservative recommendation of SMT output), the recall level will begin to drop more rapidly. If we use only system-independent features (SI), we cannot achieve as high precision as with other models even if we sacrifice more recall.

Based on these results, the users of the integrated TM/MT system can choose between precision and recall according to their own needs. As setting thresholds does not involve retraining of the SMT system or the SVM classifier, the user is able to determine this trade-off at runtime.

4.6 Edit Statistics Using the Recommendation Model

A natural question regarding the integration models is whether recommendation or reranking reduces the effort of the translators and post-editors: after reading the recommended segments or reranked list, will they translate/edit less than they would otherwise have to? In this section, we try to approximate the amount of reduced post-editing effort using the edit operations in the TER automatic MT evaluation metric. We will continue to present evidence from human evaluation that supports validation of the conclusions reported here in Chapter 6. Eventually, we plan to test this method in a full scale industrial TM and post-editing environment.

Table 4.5: Edit Statistics when Recommending MT Outputs in Classification, confidence=0.5

	Insertion	Substitution	Deletion	Shift
MT	0.9849 ± 0.0408	2.2881 ± 0.0672	0.8686 ± 0.0370	1.2500 ± 0.0598
TM	0.7762 ± 0.0408	4.5841 ± 0.1036	3.1567 ± 0.1120	1.2096 ± 0.0554

Table 4.6: Edit Statistics when NOT Recommending MT Outputs in Classification, confidence=0.5

	Insertion	Substitution	Deletion	Shift
MT	1.0830 ± 0.1167	2.2885 ± 0.1376	1.0964 ± 0.1137	1.5381 ± 0.1962
TM	0.7554 ± 0.0376	1.5527 ± 0.1584	1.0090 ± 0.1850	0.4731 ± 0.1083

Table 4.7: Edit Statistics when Recommending MT Outputs in Classification, confidence=0.85

	Insertion	Substitution	Deletion	Shift
MT	1.1665 ± 0.0615	2.7334 ± 0.0969	1.0277 ± 0.0544	1.5549 ± 0.0899
TM	0.8894 ± 0.0594	6.0085 ± 0.1501	4.1770 ± 0.1719	1.6727 ± 0.0846

4.6.1 The Statistics Using the Recommendation Model

For the recommendation model, we provide the statistics of the number of edits for each sentence with 0.95 confidence intervals, sorted by TER edit types. Statistics of positive instances in classification (i.e. the instances in which MT output is recommended over the

TM hit) are given in Table 4.5. These statistics are the avearge number of edits on the segment level.

When an MT output is recommended, its TM counterpart will require a larger average number of total edits, as we expect. If we drill down, however, we also observe that many of the saved edits come from the *Substitution* category, which is the most costly operation from the post-editing perspective. In this case, the recommended MT output actually saves more effort for the editors than what is shown by the TER score. This reflects the fact that often fuzzy match-based TM outputs are not actual translations, and need heavier editing.

Table 4.6 shows the statistics of negative instances in classification (i.e. the instances in which MT output is not recommended over the TM hit). In this case, the MT output requires considerably more edits than the TM hits in terms of all four TER edit types, i.e. insertion, substitution, deletion and shift. This shows that some high-quality TM matches can be very useful as translations in their own right.

4.6.2 The Statistics on Recommendations of Higher Confidence

We present the edit statistics of recommendations with higher confidence in Table 4.7. Comparing Tables 4.5 and 4.7, we see that if recommended with higher confidence, the MT output will need substantially fewer edits than the TM output, e.g. 3.28 fewer substitutions on average.

From the characteristics of the high confidence recommendations, we suspect that these mainly comprise harder to translate (i.e. different from the SMT training set/TM database) sentences, as indicated by the slightly increased edit operations on the MT side. TM produces much worse edit-candidates for such sentences, as indicated by the numbers in Table 4.7, since TM does not usually have the ability to automatically reconstruct an output through the combination of several segments.

4.6.3 A Recommendation Example

From the recommendation precison/recall evaluation and the approximated edit statistics, we can see that the translation recommendation model is able to select the segment that is most suitable to post-edit from the TM and the MT output for translators, and reduce their workload in a TM environment. Before we review related work and conclude this chapter, we walk through the example at the beginning of this chapter to see how the translation recommendation paradigm can help translators in action.

Table 4.8: An Example of TM and MT Output - Revisited

Source	*Restore over existing virtual machines* .
TM Source	*Check restore over existing files* . (Fuzzy Match Score: 0.5)
TM Target	*Cochez la case restaurer sur les fichiers existants* .
MT Output	*Restaurer des machines virtuelles existantes* . (Confidence: 0.8571)
Reference	*Restaurer sur les machines virtuelles existantes* .

In Table 4.8, when we have a source segment to translate, we find both a TM fuzzy match with fuzzy match score 0.5, and an MT output. Our recommender compares these two systems, and recommends the MT output with confidence 0.8571.

Based on the threshold setting of the translator, she can either work on the MT or TM output: given the results in Table 4.7, setting the threshold to 0.85 is very safe for most translators, in the sense that they are very unlikely to miss high quality TM hits. In this example, the translator can benefit from the MT output which is of better quality if the threshold is set to 0.85. However, most conservative translators can still set the threshold even higher, if they feel more comfortable in the traditional TM environment.

4.7 Related Work

To the best of our knowledge, the work reported in this chapter is the first work that performs recommendation between TM and MT output and produces a recommendation confidence score. Previous research relating to this work mainly focuses on predicting MT quality.

The first strand is confidence estimation for MT, initiated by [Ueffing et al., 2003], in

which posterior probabilities on the word graph or N-best list are used to estimate the quality of MT outputs. The idea is explored more comprehensively in [Blatz et al., 2004]. These estimations are often used to rerank the MT output and to optimize it directly. Extensions of this strand are presented in [Quirk, 2004] and [Ueffing and Ney, 2005]. The former experimented with confidence estimation with several different learning algorithms; the latter uses word-level confidence measures to determine whether a particular translation choice should be accepted or rejected in an interactive translation system.

The second strand of research focuses on combining TM information with an SMT system, so that the SMT system can produce better target language output when there is an exact or close match in the TM [Simard and Isabelle, 2009]. This line of research is shown to help the performance of MT, but is less relevant to our task in this chapter.

A third strand of research tries to incorporate confidence measures into a post-editing environment. To the best of our knowledge, the first paper in this area is [Specia et al., 2009a]. Instead of modeling on translation quality (often measured by automatic evaluation scores), this research uses regression on both the automatic scores and scores assigned by translators. The method is improved in [Specia et al., 2009b], which applies Inductive Confidence Machines and a larger set of features to model translators' judgement of the translation quality between 'good' and 'bad', or among three levels of post-editing effort.

Our research is more similar in spirit to the third strand. However, we use outputs and features from the TM explicitly; therefore instead of having to solve a regression problem, we only have to solve a much easier binary prediction problem which can be integrated into TMs in a straightforward manner. Because of this, the precision and recall scores reported in this paper are not directly comparable to those in [Specia et al., 2009b] as the latter are computed on a pure SMT system without a TM in the background.

4.8 Summary

In this chapter we presented a classification model to integrate SMT into a TM system, in order to facilitate the work of translators. In so doing we handled the problem of MT quality estimation as binary prediction instead of regression. From the translators' perspective, they can continue to work in their familiar TM environment, use the same cost-estimation methods, and at the same time benefit from the power of state-of-the-art MT. We used SVMs to make these predictions, and used grid search to find better RBF kernel parameters.

We explored features from inside the MT system, from the TM, as well as features that make no assumption on the translation model for the binary classification. With these features we made glass-box and black-box predictions. Experiments show that the models can achieve 0.85 precision at a level of 0.89 recall, and even higher precision if we sacrifice more recall. With this guarantee on precision, our method can be used in a TM environment without changing the upper-bound of the related cost estimation.

Finally, we analyzed the characteristics of the integrated outputs. We presented results to show that, if measured by number, type and content of edits in TER, the recommended sentences produced by the classification model would bring about less post-editing effort than the TM outputs.

We will extend this model in the following ways. First of all, our current model can handle only 1-best outputs from TM and SMT, while both the localization and the SMT communities have benefited from k-best outputs, so it is worthwhile to extend the recommendation model to the k-best case. Secondly, it is useful to test the model in user studies. A user study can serve two purposes: 1) it can validate the effectiveness of the method by measuring the actual (as opposed to estimated) amount of edit effort it saves, and 2) it can help the creation of human annotated gold standards for us to train better models. Finally, the current model integrates TM and MT systems on the segment level, we will also explore sub-segment level models that can further boost the efficiency of post-editing. We will report advances in these directions in the chapters to follow.

Chapter 5

TM-MT Integration as Translation Reranking

5.1 Introduction[1]

In the previous chapter, we presented a translation recommendation model that automatically selects the better segment from the TM and the MT output for the translator to post-edit. Translation recommendation has the advantage of utilizing high quality MT outputs, while keeping the TM environment (and its cost estimation) intact. However, the translation recommendation paradigm is not able to employ k-best lists, which modern TM and MT systems can both produce.

With this in mind, we continue to investigate a deeper integration of TM and MT paradigms: we now study reranking models that can integrate k-best outputs from TM and MT systems. Presenting k-best output in a TM can provide post-editors with more translation options, though reading and differentiating among closely related options may result in substantial cognitive overhead. This overhead can be alleviated sigificantly if we can rank translations of better quality higher.

In Table 5.1, we compare the k-best output of the TM and the MT system on the same

[1] Part of the research presented in this chapter has been published in [He et al., 2010d]

Table 5.1: An Example of TM and MT 3-best Output

Source	*Restore over existing virtual machines .*
TM 3-BEST	
k=1	*Cochez la case restaurer sur les fichiers existants .*
k=2	*Suppression de machines virtuelles existantes .*
k=3	*Restaurer sur les documents existants .*
MT 3-BEST	
k=1	*Restaurer des machines virtuelles existantes .*
k=2	*Restauration sur des machines virtuelles existantes .*
k=3	*Restaurer par-dessus des machines virtuelles existantes .*
Reference	*Restaurer sur les machines virtuelles existantes .*

segment as in our translation recommendation example in Chapter 4. If we measure the post-editing effort on the output segments using the TER score, we find that all MT outputs are easier to post-edit than the top TM output (TER 0.29 for all MT segments vs. 0.57 for the best TM segment). The second best TM output is also worth editing. Although it has a higher TER score than the MT outputs, its errors are easy to identify in an color-coded environment (*Suppression de* at the beginning of the segment). Our translation reranking model reranks the combined TM-MT k-best list and aims to rank such easier-to-edit segments higher.

The rest of the chapter is organized as follows: we outline the translation reranking paradigm in Section 5.2. The precise formulation of the problem (using Ranking SVM) and experiments with the ranking models are presented in Sections 5.3 and 5.4, respectively. We analyze the post-editing effort approximated by the TER metric in Section 5.5. We review related research in Section 5.6, and summarize in Section 5.7.

5.2 The Translation Reranking Paradigm

In the previous chapter, the recommender is a binary predictor that works on the 1-best output of the MT and the TM system, presenting either the one or the other to the post-editor. In this chapter, we develop the idea further by moving from binary prediction to ranking. We use a reranking model to merge the k-best lists of the two systems, and produce

a ranked merged list for post-editing. As the list is an enriched version of the TM's k-best list, the TM related assets are preserved and TM-based cost estimation is still valid as an upper bound.

More specifically, we recast SMT-TM integration as a ranking problem, where we apply the Ranking SVM technique to produce a ranked list of translations combining the k-best lists of both the MT and the TM systems. We use features independent of the MT and the TM system for ranking, so that outputs from MT and TM can have the same set of features. Ideally the translations should be ranked by their associated post-editing efforts, but given the very limited amounts of human annotated data, we use an automatic MT evaluation metric, TER [Snover et al., 2006], which is specifically designed to simulate post-editing effort to train and test our ranking model.

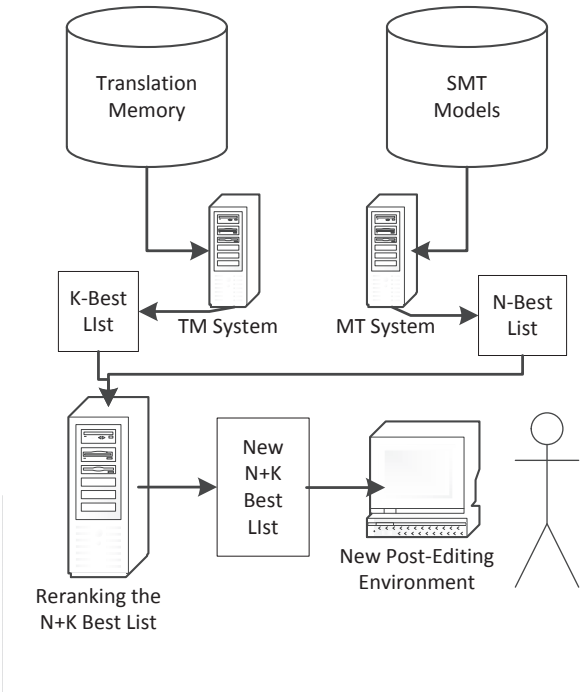

Figure 5.1: The Translation Reranking Paradigm

We depict the Translation Reranking model in Figure 5.1. Like the translation recommendation model, we have both the SMT system and the TM system at the backend. The

main difference is that in the translation reranking model, the reranker will receive k-best list from the systems, rerank them, and provide a new k-best list to the translator. The translator can choose the best translation from the reranked list by herself.

5.3 Ranking SVM for SMT-TM Integration

5.3.1 Problem Formulation with Ranking SVM

SVMs are proposed as binary classifiers in [Cortes and Vapnik, 1995], and were not designed to solve ranking problems in the original setting. However, by modifying the training objective and the constraints, many alternative formulations of SVMs have been proposed for different types of problems. In this chapter, we leverage the ranking SVM algorithm in [Joachims, 2002] to extend our translation recommendation model to handle the ranking case. The idea of the ranking SVM is to produce a ranking r that has the maximum Kendall's τ coefficient with the the gold standard ranking r^*.

Kendall's τ measures the relevance of two rankings: $\tau(r_a, r_b) = \frac{P-Q}{P+Q}$, where P and Q are the amount of concordant and discordant pairs in r_a and r_b. In practice, this is done by building constraints to minimize the discordant pairs Q. Following this basic idea, we show how Ranking SVM can be applied to MT-TM integration as follows.

Assume that for each source sentence s, we have a set of outputs from MT, **M**, and a set of outputs from TM, **T**. If we have a ranking $r(s)$ over translation outputs $\mathbf{M} \bigcup \mathbf{T}$ where for each translation output $d \in \mathbf{M} \bigcup \mathbf{T}$, $(d_i, d_j) \in r(s)$ iff $d_i <_{r(s)} d_j$, we can rewrite the ranking constraints as optimization constraints in an SVM, as in Eq. (5.1).

$$
\begin{aligned}
&\min_{w,b,\xi} \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum\xi \\
&\text{subject to:} \\
&\forall(d_i, d_j) \in r(s_1) : \mathbf{w}(\Phi(s_1, d_i) - \Phi(s_1, d_j)) \geqslant 1 - \xi_{i,j,1} \\
&... \\
&\forall(d_i, d_j) \in r(s_n) : \mathbf{w}(\Phi(s_n, d_i) - \Phi(s_n, d_j)) \geqslant 1 - \xi_{i,j,n} \\
&\xi_{i,j,k} \geqslant 0
\end{aligned}
\tag{5.1}
$$

where $\Phi(s_n, d_i)$ is a feature vector of translation output d_i given source sentence s_n. The Ranking SVM minimizes the discordant number of rankings with the gold standard according to Kendall's τ.

As in Chapter 4, we perform our experiments with the Radial Basis Function (RBF) kernel.

5.3.2 Elements of the Reranking Model

Our reranking model merges the k-best list from TM and MT to produce a new list, which aims to rank segments that are more suitable for post-editing higher, so that the post-editors are offered more and better translation options. The model consists of three elements: The MT k-best list, the TM k-best list, and the reranker.

5.3.2.1 The MT k-best List

The k-best list of the SMT system is generated during decoding according to the internal feature scores. The features include language and translation model probabilities, reordering model scores and a word penalty.

5.3.2.2 The TM k-Best List and the Fuzzy Match Score

The k-best list of the TM system is generated in descending fuzzy match score. The fuzzy match cost [Sikes, 2007] is the similarity of the source sentences used in translation memories, which is the same as we use in Chapter 4.

5.3.2.3 The Reranker

Based on Ranking SVMs [Joachims, 2002] that we introduced in Section 5.3, which have already been applied successfully in machine translation evaluation [Ye et al., 2007], we build a reranker to rerank a merged list of MT and TM outputs, and produce a new reranked k-best list.

5.3.3 The Feature Set

In the previous chapter, we explored using features both from the internals of the TM and the MT system, and features that are independent of the systems. When building features for the Ranking SVM, however, we are limited to features that are independent of the MT and TM systems: we need a set of features that are both applicable to the TM outputs and the MT outputs in reranking, while in recommendation we can extract different features from TM and MT outputs simultaneously.

For the translation reranking model, we experiment with system-independent features that capture translation fluency and adequacy. For more detail, we use source-side LM scores, target-side LM scores, the pseudo-source fuzzy match score and the IBM model 1 score.

- **Source-Side Language Model Score and Perplexity**. We compute the LM score and perplexity of the input source sentence on an LM trained on the source-side training data of the SMT system.
- **Target-Side Language Model Perplexity**. We compute the LM probability and perplexity of both the MT and TM outputs.
- **The Pseudo-Source Fuzzy Match Score**. We back-translate the output to obtain a pseudo source sentence. We compute the fuzzy match score between the original source sentence and this pseudo-source.
- **The IBM Model 1 Score**. We compute the IBM Model 1 score [Brown et al., 1993], which serves as a rough estimation of how good a translation it is on the word level, for both the TM and the MT output.

5.4 Reranking Experiments

As we did in Chapter 4, before we estimate the post-editing effort the reranking model can save, we first evaluate whether ranking SVM and our feature set can model the segment

ranking problem effectively.

5.4.1 The Experimental Settings

We use the same experimental setting as in Chapter 4 to run our experiments: we use the 51K sentence-pair English–French translation memory from Symantec, randomly selected 43K to train an SMT system and translated the English side of the remaining 8K sentence pairs.

We use a standard log-linear PB-SMT model [Och and Ney, 2002] as the SMT engine: GIZA++ implementation of IBM word alignment model 4,[2] the refinement and phrase-extraction heuristics described in [Koehn et al., 2003], minimum-error-rate training [Och, 2003], a 5-gram language model with Kneser-Ney smoothing [Kneser and Ney, 1995] trained with SRILM [Stolcke, 2002] on the French side of the training data, and Moses [Koehn et al., 2007] to decode. We train a system in the opposite direction using the same data to produce the pseudo-source sentences. The only difference from the translation recommendation experiments is that we obtain k-best lists from the TM and the MT systems, and use them as input for the SVM-based reranker.

5.4.2 Training, Tuning and Testing the Ranking SVM

We run training and prediction of the Ranking SVM in 4-fold cross validation. We use the SVM*light*[3] toolkit to perform training and testing.

We optimize C (cost) and γ (radius) meta-parameters of the SVM and the RBF kernel using a brute-force grid search before running cross-validation and maximize precision at top-5, with an inner 3-fold cross validation on the (outer) Fold-1 training set. We search within the range $[2^{-6}, 2^{9}]$ for both C and γ, with a step size of 2 on the exponent.

We rerank the combined list produced with the top-5 distinct outputs from both systems.

[2] More specifically, we performed 5 iterations of Model 1, 5 iterations of HMM, 3 iterations of Model 3, and 3 iterations of Model 4.

[3] `http://svmlight.joachims.org/`

5.4.3 The Gold Standard

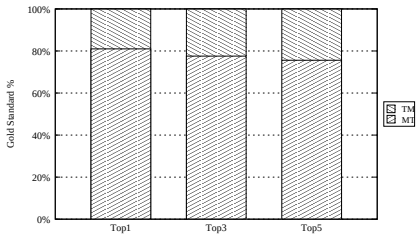

Figure 5.2: MT and TM's percentage in gold standard

Figure 5.2 shows the composition of translations in the gold standard. Each source sentence is associated with a list of translations from two sources, namely MT output and TM matches. This list of translations is ranked from best to worst according to TER scores. The figure shows that over 80% of the translations are from the MT system if we only consider the top-1 translation. As the number of top translations considered increases, more TM matches can be seen. On the one hand, this does show a large gap in quality between MT outputs and TM matches; however, it also reveals that we will have to ensure two objectives in ranking: the first is to rank the 80% MT translations higher and the second is to keep the 20% 'good' TM hits in the Top-5. We design our evaluation metrics accordingly.

5.4.4 Evaluation Metrics

Unlike translation recommendation which chooses the best translation for the post-editor, translation reranking tries to provide post-editors with more translation options. The benefit of the reranking model is that if the better translations are ranked higher, post-editors will be able to find them more easily, compared to an ordinary TM system, where the first candidate will always be the top TM hit. Therefore, the top TM output is the pivot in our evaluation, in the sense that the precision and recall numbers we report are reflecting whether the reranked list can rank higher those translations that are better than the top TM output (the pivot).

Based on this observation, we introduce the idea of *relevant* translations, and our evaluation metrics: PREC@k and HIT@k.

5.4.4.1 Relevant Translations

We borrow the idea of *relevance* from the IR community to define the idea of translations worthy of a high ranking. For a source sentence s which has a top TM hit t, we define an MT/TM output m as relevant, if $TER(m) \leq TER(t)$. According to the definition, relevant translations should need no more post-edits than the original top hit from the TM system. Clearly the top TM hit is always relevant according to this definition.

5.4.4.2 PREC@k

We calculate the precision (PREC@k) of the ranking for evaluation. Assuming that there are n relevant translations in the top-k list for a source sentence s, we have PREC@k$= n/k$ for s. We test PREC@k, for $k = 1 \ldots 10$, in order to evaluate the overall quality of the ranking.

5.4.4.3 HIT@k

We also estimate the probability of having one of the relevant translations in the top k, denoted as HIT@k. For a source sentence s, HIT@k is equal to 1 if there is at least one relevant translation in the top k, and 0 otherwise. This measures the quality of the best translation in the top k, which is the translation the post-editor will find and work on if she reads till the kth place in the list. HIT@k is equal to 1.0 at the end of the list.

5.4.5 Experimental Results

In Table 5.2 we report PREC@k and HIT@k for $k = 1 \ldots 10$. The ranking receives 0.8747 PREC@1, which means that most of the top-ranked translations have at least the same quality as the top TM output. We note that precision remains above 0.8 till $k = 5$, leading us to conclude that most of the *relevant* translations are ranked in the top-5 positions in the list.

Using the HIT@k scores we can corroborate this argument still further. The HIT@k score grows steadily from 0.8747 to 0.9941 for $k = 1 \ldots 6$, so most often there will be at

Table 5.2: PREC@k and HIT@k of Ranking

	PREC %	HIT %
k=1	87.47±1.60	87.47±1.60
k=2	85.42±1.07	93.36±0.53
k=3	84.13±0.94	95.74±0.61
k=4	82.79±0.57	97.08±0.26
k=5	81.34±0.51	98.04±0.23
k=6	79.26±0.59	99.41±0.25
k=7	74.99±0.53	99.66±0.29
k=8	70.87±0.59	99.84±0.10
k=9	67.23±0.48	99.94±0.08
k=10	64.00±0.46	100.0±0.00

least one *relevant* translation in the top-6 for the post-editor to work with. After that there is very little room left for improvement.

In sum, both the PREC@k scores and HIT@k scores show that the ranking model effectively integrates the two translation sources (MT and TM) into one merged k-best list, and ranks relevant translations higher.

Table 5.3: PREC@k - MT and TM Systems

	MT %	TM %
k=1	85.87±1.32	100.0±0.00
k=2	82.52±1.60	73.58±1.04
k=3	80.05±1.11	62.45±1.14
k=4	77.92±0.95	56.11±1.11
k=5	76.22±0.87	51.78±0.78

To measure whether the ranking model is effective compared to pure MT or TM outputs, we report the PREC@k of those outputs in Table 5.3. On the left are the PREC numbers if we only rely on the Top-5 MT outputs; on the right are the numbers using only the Top-5 TM outputs. We see that the combined and reranked results in Table 5.2 consistently outperform the results in Table 5.3, indicating that our system clearly outperforms these two simple baselines.

The TM outputs alone are generally of much lower quality than the MT and Ranked outputs, as is shown by the precision scores for $k = 2 \ldots 5$. However, TM translations obtain 1.0 for PREC@1 according to the definition of the PREC calculation. Note that

this does not mean that those outputs will need less post-editing (cf. Section 5.5.1); rather, it indicates that each one of these outputs meets the lowest acceptable criterion of being relevant.

Table 5.4: Edit Statistics on Ranked MT and TM Outputs - Single Best

	Insertion	Substitution	Deletion	Shift
TM-Top1	0.7554 ± 0.0376	4.2461 ± 0.0960	2.9173 ± 0.1027	1.1275 ± 0.0509
MT-Top1	0.9959 ± 0.0385	2.2793 ± 0.0628	0.8940 ± 0.0353	1.2821 ± 0.0575
Rank-Top1	1.0674 ± 0.0414	2.6990 ± 0.0699	1.1246 ± 0.0412	1.2800 ± 0.0570

Table 5.5: Edit Statistics on Ranked MT and TM Outputs - Top 3

	Insertion	Substitution	Deletion	Shift
TM-Best-in-Top3	0.4241 ± 0.0250	3.7395 ± 0.0887	2.9561 ± 0.0966	0.9738 ± 0.0505
TM-Mean-Top3	0.6718 ± 0.0200	5.1428 ± 0.0559	3.6192 ± 0.0649	1.3233 ± 0.0310
MT-Best–in-Top3	0.7696 ± 0.0351	1.9210 ± 0.0610	0.7706 ± 0.0332	1.0842 ± 0.0545
MT-Mean-Top3	1.1296 ± 0.0229	2.4405 ± 0.0368	0.9341 ± 0.0209	1.3797 ± 0.0344
Rank-Best-in-Top3	0.8170 ± 0.0355	2.0744 ± 0.0608	0.8410 ± 0.0338	1.0399 ± 0.0529
Rank-Mean-Top3	1.0942 ± 0.0234	2.7437 ± 0.0392	1.0786 ± 0.0231	1.3309 ± 0.0334

Table 5.6: Edit Statistics on Ranked MT and TM Outputs - Top 5

	Insertion	Substitution	Deletion	Shift
TM-Best-in-Top5	0.4239 ± 0.0250	3.7319 ± 0.0885	2.9552 ± 0.0967	0.9673 ± 0.0504
TM-Mean-Top5	0.6143 ± 0.0147	5.5092 ± 0.0473	3.9451 ± 0.0521	1.3737 ± 0.0240
MT-Best-in-Top5	0.7690 ± 0.0351	1.9163 ± 0.0610	0.7685 ± 0.0332	1.0811 ± 0.0544
MT-Mean-Top5	1.1912 ± 0.0182	2.5326 ± 0.0291	0.9487 ± 0.0165	1.4305 ± 0.0272
Rank-Best-in-Top5	0.7246 ± 0.0338*	1.8887 ± 0.0598	0.7562 ± 0.0327	0.9705 ± 0.0515*
Rank-Mean-Top5	1.1173 ± 0.0181	2.8777 ± 0.0312	1.1585 ± 0.0200	1.3675 ± 0.0260

5.5 Edit Statistics Using the Reranking Model

In this section, we move on to approximate the post-editing effort associated with the reranking model using TER operations. We report the results on the Top-1/3/5 candidates of the reranked lists to reflect the performance of the most favorable candidate as well as the overall quality of the list.

5.5.1 Top-1 Edit Statistics

We report the results on the 1-best output of TM, MT and our ranking system in Table 5.4.

In the single best results, it is easy to see that the 1-best output from the MT system requires the least post-editing effort. This is not surprising given the distribution of the gold standard in Section 5.4.3, where most MT outputs are of better quality than the TM hits.

Moreover, since TM translations are generally of much lower quality as is indicated by the numbers in Table 5.4 (e.g. $\sim$ 2x as many substitutions and $\sim$ 3x as many deletions compared to MT), unjustly including very few of them in the ranking output will increase loss in the edit statistics. This explains why the ranking model has better ranking precision in Tables 5.2 and 5.3, but seems to incur more editing effort. However, in practice it is likely that post-editors will be able to dismiss an obviously 'bad' translation very quickly.

5.5.2 Top-k Edit Statistics

We report edit statistics of the Top-3 and Top-5 outputs in Tables 5.5 and 5.6, respectively. For each system we report two sets of statistics: the Best-* statistics calculated on the best output (according to TER score) in the list, and the Mean-* statistics calculated on the whole top-k list.

The Mean- numbers allow us to have a general overview of the ranking quality, but this is strongly influenced by the poor TM hits that can easily be neglected in practice. To control the impact of those TM hits, we rely on the Best- numbers to estimate the edits performed on the translations that are more likely to be used by post-editors, provided that

they can identify the best translation in the top-k list.

In Table 5.5, the ranking output's edit statistics are closer to the MT output than the Top-1 case in Table 5.4. Table 5.6 continues this tendency, in which the Best-in-Top5 Ranking output requires marginally fewer *Substitution* and *Deletion* operations and significantly fewer *Insertion* and *Shift* operations (starred) than its MT counterpart. This shows that when more of the list is explored, the advantage of the ranking model – utilizing multiple translation sources – begins to compensate for the possible large number of edits required by poor TM hits, and finally leads to reduced post-editing effort.

There are several explanations to why the relative performance of the ranking model improves when k increases, as compared to other models. The most obvious explanation is that a single poor translation is less likely to hurt edit statistics on a k-best list with a larger k, if most of the translations in the k-best list are of good quality. We see from Tables 5.2 and 5.3 that the ranking output is of better quality than the MT and TM outputs with regard to precision. For a larger k, the small number of incorrectly ranked translations are less likely to be chosen as the Best-* translation and negatively affect the Best-* numbers.

A further reason is related to our ranking model which optimizes on Kendall's τ score. Accordingly the output might not be optimal when we evaluate the Top-1 output, but it will behave better when we evaluate on the whole list. This is also in accordance with our aim, which is to enrich the TM with MT outputs and help the post-editor, instead of choosing the 1-best translation for the post-editor.

5.5.3 Discussion on the Relative Performance of TM and MT Outputs in Reranking

One of the interesting findings from Tables 5.4 and 5.5 is that according to the TER edit statistics, the MT outputs generally need a smaller number of edits than the TM and Ranking outputs. This certainly confirms the necessity to integrate MT into today's TM systems.

However, this fact should not lead to the conclusion that TMs should be replaced by MT completely. First of all, all of our experiments exclude exact TM matches, as those

translations will simply be reused and not translated. While this is a realistic setting in the translation industry, it removes all sentences for which the TM works best from our evaluations.

Furthermore, Table 5.6 shows that the Best-in-Top5 Ranking output performs better than the MT outputs, hence there are TM outputs that lead to a smaller number of edits. As k increases, the ranking model is able to better utilize these outputs.

Finally, in this task we concentrate on ranking useful translations more highly in the k-best lists, but we are not interested in how very poor translations are ranked. A ranking SVM optimizes on the ranking of the whole list, which is slightly different from what we actually require when calculating edit statistics. One option is to use other optimization techniques that can make use of this property to obtain better top-k edit statistics for a smaller k. Another option is to perform regression directly on the number of edits instead of modeling on the ranking.

5.5.4 A Reranking Example

Before we review related work and conclude, we walk through the example at the beginning of this chapter to see how the reranking model works in a localization environment. For the example in Table 5.7, our reranker will generate the new Top-3 list as in Table 5.8.

Table 5.7: An Example of TM and MT 3-best Output – Revisited

Source	*Restore over existing virtual machines .*
TM 3-BEST	
k=1	*Cochez la case restaurer sur les fichiers existants .*
k=2	*Suppression de machines virtuelles existantes .*
k=3	*Restaurer sur les documents existants .*
MT 3-BEST	
k=1	*Restaurer des machines virtuelles existantes .*
k=2	*Restauration sur des machines virtuelles existantes .*
k=3	*Restaurer par-dessus des machines virtuelles existantes .*
Reference	*Restaurer sur les machines virtuelles existantes .*

As we can see, the new Top-3 list works as we expect. It ranks the top MT output at the top place. From the translator's perspective, this is indeed the translation that requires

Table 5.8: An Example of TM and MT 3-best Output – New Top-3

ORIGIN	Output	Score
MT k=1	*Restaurer des machines virtuelles existantes .*	-0.4645
MT k=2	*Restauration sur des machines virtuelles existantes .*	-0.2620
TM k=2	*Suppression de machines virtuelles existantes .*	-0.2602

minimal effort: the translator only needs to change the function word "*des*" to "*sur les*". It is also worth noting that the one TM segment that translates the tail of the segment correctly is also kept in the new Top-3 list. If the translator is not satisfied with the top 2 MT translated segments, she can still work on the TM segment. This demonstrates the translation reranking model's capability of preserving valuable TM assets for use in the translation workflow.

5.6 Related Work

The work presented in this chapter is an extension of the work in the previous chapter, the aim of which is to integrate high confidence MT outputs into the TM, so that the "good" TM entries will remain untouched. In the previous chapter, we recommend SMT outputs to a TM user when a binary classifier predicts that SMT outputs are more suitable for post-editing for a particular sentence.

The contribution we made in this chapter is that we do not limit ourselves to the 1-best output but try to produce a k-best output in a ranking model. The ranking scheme also enables us to show all TM hits to the user, and thus further protects the TM assets.

There has also been work to improve SMT using the knowledge from the TM. In [Simard and Isabelle, 2009], the SMT system can produce a better translation when there is an exact or close match in the corresponding TM. They use regression Support Vector Machines to model the quality of the TM segments. This is also related to our work in spirit, but our work is in the opposite direction, i.e. using SMT to enrich TM.

Moreover, our ranking model is related to reranking [Shen et al., 2004] in SMT as well. However, our method does not focus on producing better 1-best translation output for an

SMT system, but on improving the overall quality of the k-best list that TM systems present to post-editors. Some features in our work are also different in nature to those used in MT reranking. For instance we cannot use N-best posterior scores as they do not make sense for the TM outputs.

5.7 Summary

In this chapter we present a ranking-based model to integrate SMT into a TM system, in order to facilitate the work of post-editors. In such a model, the user of the TM will be presented with an augmented k-best list, consisting of translations from both the TM and the MT systems, and ranked according to ascending prospective post-editing effort.

From the post-editors' point of view, the TM remains intact. And unlike in the binary translation recommendation, where only one translation recommendation is provided, the ranking model offers k-best post-editing candidates, enabling the user to use more resources when translating. As we do not actually throw away any translation produced from the TM, the assets represented by the TM are preserved and the related estimation of the upper bound cost is still valid.

We extract system independent features from the MT and TM outputs and use Ranking SVMs to train the ranking model, which outperforms both the TM's and MT's k-best list w.r.t. precision at k, for all ks.

We also analyze the edit statistics of the integrated k-best output using the TER edit statistics. Our ranking model results in a slightly increased number of edits compared to the MT output (apparently held back by a small number of poor TM outputs that are ranked high) for a smaller k, but requires fewer edits than both the MT and the TM output for a larger k.

In the next chapter, we will perform human evaluation to validate the translation recommendation model presented in Chapter 4, and the translation reranking model presented in this chapter.

Chapter 6

Human Evaluation of TM-MT Integration

6.1 Introduction[1]

In Chapters 4 and 5, we presented two solutions to promote the application of recent advances in statistical MT (such as [Koehn et al., 2003]) in the localization industry by combining the strengths of both worlds via integrating SMT with TMs.

Given that most post-editing work is based on TM output, we propose to use recommendation or reranking paradigms, in which the translators will only use MT outputs which are better (in terms of estimated post-editing effort) than TM hits to post-editors. In these frameworks, post-editors still work with the TM while benefiting from (better) SMT outputs; the assets in TMs are not wasted and TM fuzzy match scores can still be used to estimate (the upper bound of) post-editing labour.

Chapters 4 and 5 recast TM-MT integration as a binary classification/reranking problem using Support Vector Machine (SVMs: [Cortes and Vapnik, 1995]) algorithms, perform Radial Basis Function (RBF) kernel parameter optimization to find the optimal meta-parameters for the classifier, and use the automatic TER evaluation metric to simulate post-

[1] Part of the research reported in this chapter has been published in [He et al., 2010b]

editing effort.

Despite the fact that the correlations between automatic evaluation metrics and human judgements are improving, professional translators and post-editors are the ones that hold the final verdict over the quality of MT/TM integration. In order to draw grounded conclusions on the performance of our translation recommendation and translation reranking paradigms, it is essential to conduct user studies to show whether or not systems developed using automatic evaluation metrics are confirmed by human judgements.

We conduct human evaluation on both the recommendation and the reranking models with professional post-editors. In this chapter we introduce the evaluation data we use, the post-editors, the evaluation environment, the questionnaire which we give to the post-editors after they have completed the evaluation, and the performance of the recommendation and the reranking models according to the judgement of our post-editors.

The rest of this chapter is organized as follows. We first introduce our evaluation setting in Section 6.2. We then present human evaluation results and our analysis on the translation recommendtion model and the translation reranking model in Section 6.3 and 6.4, respectively. We discuss the post-editors' feedback during the evaluation in Section 6.5. We review related work and summarize this chapter in Sections 6.6 and 6.7.

6.2 The Evaluation Setting

6.2.1 Data

We use the translation recommendation system and the translation reranking system that we built in Chapter 4 and Chapter 5. We randomly pick 300 segments from the first fold test set in the cross-validation data set (cf. Section 4.5.1) to perform human evaluation.

For the translation recommendation model, we use all the features in Chapter 4, we also apply the confidence threshold that we describe in Chapter 4. We choose to use the confidence level instead of the binary classification result so that we can evaluate the performance on varying thresholds.

For the translation reranking evaluation, we use all the features in Chapter 5.

6.2.2 The Post-editors

Five professional post-editors helped us to complete this study. Four of them are full-time post-editors, and one is a part-time post-editor. All of the editors are hired through the localization vendors of Symantec and have experience in post-editing machine-generated segments (including TM, Rule-based MT or Statistical MT).

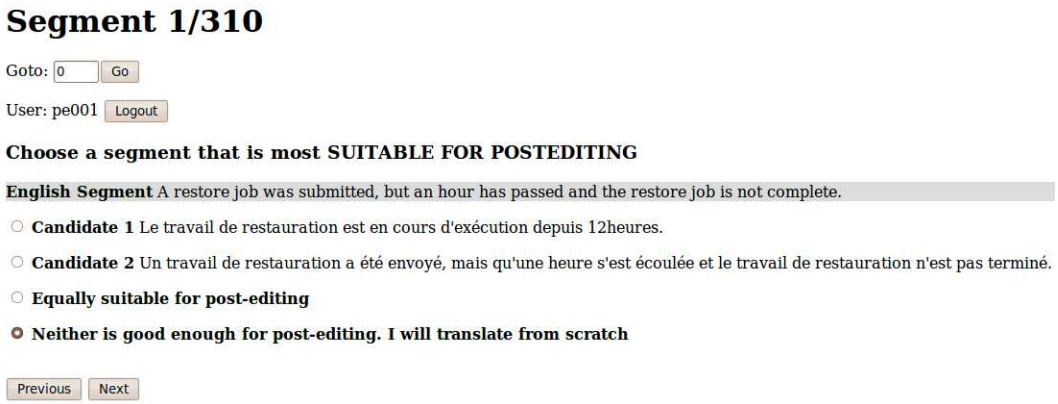

Figure 6.1: Interface of the Evaluation Environment

6.2.3 The Evaluation Environment

We design an evaluation environment to present the 300 English segments translated into French using the TM and MT systems to the post-editors. The environment is a web application developed in Python with the Django framework.[2]

Each post-editor is given a username and password to log into the system. After login, there is only one English segment together with two French translations shown on each page. The two French translations are shuffled randomly, so translation candidate 1 and candidate 2 can both be the MT or the TM output. For the translation recommendation model, one of these two translations is from TM, and another is from MT. However, for

[2] `http://www.djangoproject.com`

the translation reranking model, one translation is the 1-best output of the TM system, and another is an alternative translation that can either be produced by the TM or the MT system.

As this experiment tries to evaluate the performance of the TM/MT integration technique, we need to keep it blind: we do not reveal which engine generates which output to the post-editors. A screenshot of the interface is shown in Figure 6.1.

The post-editors' operations in the system are recorded with a time stamp in the database, which allows us to analyze the time they spend on each segment. The system allows the users to log in and out of the environment so that their previous work is not lost. They are presented with the last segment they worked on once they log in again.

Each post-editor is provided with an introduction to the task before the experiment begins. Note that the post-editors are asked to choose the sentence that is most suitable for post-editing (which is also emphasized in the introduction to the task). The post-editors are told that even if a French translation does not fully translate the English segment, they may still select it because they would spend less time post-editing it into a grammatical French segment whose meaning would match that of the English segment. The original guidelines provided to the post-editors can be found in the Appendix.

To control data quality and to measure intra-annotator correlation, we pre-select 10 segments from the 300 and make them appear twice in the environment. Therefore the post-editors are actually presented with 310 segments.

6.2.4 Questionnaire

After they finished rating the 310 segments, the post-editors were presented with four questions:

- Whether they are full-time post-editors,
- If they are full-time post-editors, how long have they worked as full-time post-editors,
- Whether they have edited MT output professionally,

- What they think of MT (five choices: no idea, very useful, sometimes useful, not useful, and useless).

6.3 Analysis of Recommendation Performance

In this section we investigate the effectiveness of the translation recommendation model according to the judgements of professional post-editors. We also compare these results with the result on a gold standard approximated by TER scores to show whether it is at all valid to use automatic evaluation metric scores to approximate post-editing effort, instead of human judgement in this task.

6.3.1 Precision and Recall of Translation Recommendation

We measure the precision and recall of the automatic translation recommendation, using the judgements of individual post-editors as a gold standard. We report the precision and recall numbers in Table 6.1. The precision can be further improved at the cost of recall, for which we set the confidence threshold to 0.75 in Table 6.2. In these calculations, we discard the segments which the post-editors choose to translate from scratch, as translation recommendation cannot improve the post-editor's productivity in such cases, no matter what it recommends. When the post-editor chooses 'tie', we determine that the TM output should be preserved, in accordance with the gold standard in Chapter 4, where ties on TER scores are regarded as negative examples in recommendation.

Table 6.1: Precision and Recall of Recommendation, Individual Post-editors, confidence = 0.5

Post-Editor ID	Precision	Recall
PE01	0.8812	0.9223
PE02	0.9315	0.9315
PE03	0.8945	0.9138
PE04	0.9123	0.9369
PE05	0.8734	0.9409

In Table 6.1, the automatic recommendation obtains over 0.9 recall according to all post-

Table 6.2: Precision and Recall of Recommendation, Individual Post-editors, confidence = 0.75

Post-Editor ID	Precision	Recall
PE01	0.9379	0.7824
PE02	0.9643	0.7621
PE03	0.9415	0.7629
PE04	0.9500	0.7703
PE05	0.9153	0.7864

editors. The precision of recommendation is always above 0.87. Table 6.2 suggests that if post-editors require higher recommendation confidence, then translation recommendation can obtain 0.9 precision at the cost of reducing recall. With these results on recommendation precision, there is a rather strong guarantee that the integrated MT-TM system will not waste the assets in the TM system and will not change the upper bound of related cost estimation, even at the sentence level, because the recommended SMT outputs are, in fact, more suitable for post-editing from the post-editors' perspective.

6.3.2 Precision and Recall on Consensus Preferences

The localization industry might expect even stronger confidence in the recommendation, so we measure recommendation precision on the segments where there is a consensus preference towards MT outputs among the post-editors.

To reflect consensus, we first discard the segments which the majority of the post-editors (more than 3 in this experiment) choose to post-edit from scratch. For the rest of the remaining segments, we consider that MT output should be recommended if N post-editors prefer to post-edit the MT output. Otherwise, we consider that the TM output should be recommended.

We report the precision and recall numbers on a series of confidence thresholds for $N = 3$ and $N = 4$ post-editors in Tables 6.3 and 6.4, respectively.

Table 6.3 shows that if we consider the consensus among 3 post-editors, precision is still high. This demonstrates that our system correlates quite well with the judgement of the majority of the post-editors. On the other hand, when it comes to a larger majority of

Table 6.3: Precision and Recall of Recommendation, Consensus Preferences of $N = 3$ Post-Editors

Threshold	Precision	Recall
0.5	0.9110	0.9348
0.6	0.9412	0.9043
0.7	0.9606	0.8478
0.8	0.9689	0.6783
0.85	0.9695	0.5522

Table 6.4: Precision and Recall of Recommendation, Consensus Preferences of $N = 4$ Post-Editors

Threshold	Precision	Recall
0.5	0.8263	0.9420
0.6	0.8507	0.9082
0.7	0.8768	0.8599
0.8	0.8944	0.6957
0.85	0.8931	0.5652

the post-editors ($N = 4$), precision begins to drop. Understanding the fact that this is an inherently more complex task than the $N = 3$ case, we also notice some inconsistency of judgements between post-editor PE01 and the other post-editors, which also renders it more difficult to achieve a consensus where $N = 4$ (i.e. all the rest of the editors should have the same judgement), which thus reduces the number of positive examples.

6.3.3 The TER score and the Preference of Post-Editors

We measure the TER score of the TM and MT outputs, and sort them according to the post-editors' preferences in Table 6.5. The TER score is an edit distance-based metric that calculates the number of insertions, deletions, substitutions and shifts required to transform an MT output into a reference sentence, and is therefore expected to be a reasonable automatic metric to approximate post-editing effort. We report the results in Table 6.5, where the scores are averaged among the five post-editors.

In Table 6.5, TER scores are shown to be positively related post-editors' preferences: when the post-editor prefers MT, the MT output obtains a lower TER score, and vice versa. This validates our method in Chapter 4, where the TER score is used to generate a gold

Table 6.5: TER Scores Sorted by Preference

	Post-editors' Selection			
	TM	MT	Tie	Scratch
TM Output	25.00	57.37	19.16	70.33
MT Output	31.85	25.90	20.93	41.74

standard for the translation recommendation system. The TER scores also demonstrate that the sentences which the users would translate from scratch are more difficult to translate in nature than the rest, shown by a big increase in TER points compared to when TM/MT-output (70.33 vs. 25.00/57.37 and 41.74 vs. 31.85/25.90) is chosen.

6.3.4 Comparison with a TER-Approximated Gold Standard

We present the precision and recall numbers at recommendation confidence [0.5, 0.85] in Figure 6.2. Series PE01 – PE05 use the judgement of the corresponding post-editor as the gold standard; series CONSENSUS_3 and CONSENSUS_4 use the consensus of 3 or 4 post-editors as the gold standard; series TER uses the gold standard approximated by TER scores. By presenting results on human-annotated and metric-approximated gold standards head-to-head, we are able to see the relationship between these gold standards.

In Figure 6.2, we find that although the post-editors have different preferences regarding MT and TM outputs (i.e. some reuse MT outputs more than others), the trend of precision on the variation of recommendation confidence remains similar among the post-editors, and also applies to the TER-approximated gold standard. This agrees with our approach in Chapter 4, which uses TER scores to approximate human judgements to prepare the training data and perform evaluation. Note that when calculating precision, the denominator is the total number of segments recommended by the recommendation model, no matter whether the post-editors have consensus judgements on them or not. If we limit the denominator to the number of segments where post-editors do reach a consensus judgement (on whether using the MT or the TM output), the precision will be 0.9641 for CONSENSUS_3 and 0.9848 for CONSENSUS_4. We also note that recall drops quite sharply when we raise the threshold in order to achieve higher precision. Since the majority of the better translations in this

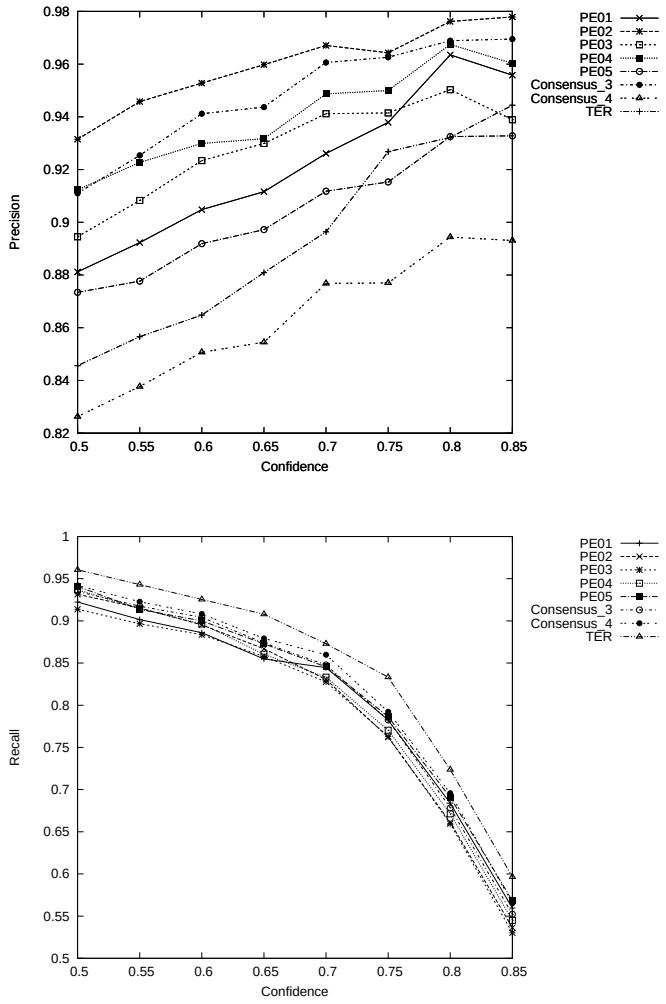

Figure 6.2: Recommendation Precision (upper) and Recall (lower) According to Human-Annotated and TER-Approximated Gold Standards

work come from MT, setting a higher threshold in recommendation will lead us to miss many better translations.

6.3.5 Accuracy on High Fuzzy Match Segments

The localization industry currently uses fuzzy match score to estimate the amount of localization work to be carried out. Specifically, many translators/post-editors set threshold on the fuzzy match, and only reuse those segments whose fuzzy match scores are above that threshold. This can be viewed as a simple baseline setting that recommends MT segments

when the fuzzy match score is below a certain level.

We report in Table 6.6 the recommendation accuracy of our model, in order to enable a direct comparison with this setting. We use the consensus of 3 post-editors as the gold standard of the accuracy calculation.

Table 6.6: Recommendation Accuracy on High Fuzzy Match Segments

	Fuzzy Match Scores						
	0.5	0.55	0.6	0.65	0.7	0.75	0.8
Conf=0.50	0.8265	0.8049	0.7895	0.7557	0.7414	0.7263	0.7237
Conf=0.55	0.8265	0.8049	0.7895	0.7557	0.7414	0.7263	0.7237
Conf=0.60	0.8316	0.8110	0.7961	0.7634	0.7586	0.7368	0.7237
Conf=0.65	0.8010	0.7744	0.7566	0.7252	0.7414	0.7158	0.6974
Conf=0.70	0.7908	0.7622	0.7434	0.7176	0.7328	0.6947	0.6579
Conf=0.75	0.7296	0.6890	0.6711	0.6565	0.6638	0.6421	0.6053
Conf=0.80	0.6224	0.5732	0.5461	0.5267	0.5517	0.5158	0.4868
Conf=0.85	0.5204	0.4939	0.4605	0.4351	0.4483	0.4316	0.4211
Baseline Conf.	0.2296	0.2622	0.2829	0.3282	0.3534	0.3895	0.4079
PreferMT/Total	151/196	121/164	109/152	88/131	75/116	58/95	45/76

In Table 6.6, *Baseline Conf.* is the accuracy of the recommendation of MT output to post-editors using the fuzzy match score as a threshold. The line denoted by *Conf=x* reports the accuracy of our recommendation system at confidence *x*, sorted by fuzzy match levels from 0.5 to 0.8. We see that our recommendation approach outperforms the baseline at any threshold. This can be partly attributed to the fact that MT systems perform very well on this task. As is reported in the line *PreferMT/Total*, the majority of post-editors consistently prefer more MT segments than TM segments, even when the fuzzy match score of the corresponding TM segment is above 0.8.

6.3.6 User Behavior

Besides recommendation performance, we are also interested in the users' reaction to the translation recommendation scheme using this system, as well as what they think about the TM and MT technologies. We report statistics of their behavior along with their ideas and comments on TM and MT.

6.3.6.1 Experience of Post-Editors

We list the years of experience as translators of the post-editors along with the number of sentences they prefer to translate from scratch in our experiment in Table 6.7, since the latter is an indication of the willingness to reuse a computer-generated translation. We also present the number of MT outputs (out of 300) selected by post-editors to work on.

Table 6.7: Participants' Experience and Preference

Post-Editor ID	Years	Scratch	MT
PE01	5	59	193
PE02	3	11	248
PE03	12	22	232
PE04	8	33	222
PE05	part-time	23	220

The results show that the willingness to reuse automatic output varies considerably among post-editors. PE01 is willing to translate one-fifth of the sentences from scratch in this experiment, which is more than five-times the number of PE02. This preference does not correlate well with the years of experience, suggesting that this is more related to the particular habits of post-editors, rather than to their experience in the industry. The result also shows that all post-editors select more MT outputs to post-edit than the other options.

6.3.6.2 Inter-annotator Agreement

To gauge the validity of human evaluation results, we computed the inter-rater agreement measured by Fleiss' Kappa coefficient [Fleiss, 1981] which can assess the agreement between multiple raters, as opposed to Cohen's Kappa coefficient [Cohen, 1960] which works with just two raters.

Fleiss' Kappa coefficient for our five post-editors is 0.464 ± 0.024, indicating a moderate agreement. We also obtained Fleiss' Kappa coefficient for each category as shown in Table 6.8. From this table, we can observe moderate agreements among post-editors in selecting TM or MT output as the most suitable for post-editing. There is also a moderate agreement in making their decision to translate from scratch. However, there is only a fair

agreement in determining whether TM and MT outputs are equally good for post-editing ("Tie").

Table 6.8: Annotator agreement for each category

Category	Kappa
TM	0.519
MT	0.516
Tie	0.285
Scratch	0.426

6.3.6.3 Intra-annotator Agreement

We have ten duplicate samples in our evaluation intended to measure the level of intrinsic agreement for each post-editor. Both percentage of agreement and Cohen's Kappa are calculated as shown in Table 6.9. From this table, we can observe that all five post-editors achieved almost perfect intrinsic agreement, indicating that the evaluation results are highly reliable.

Table 6.9: Intra-annotator Agreement

Post-Editor ID	Agreement	Kappa
PE01	90%	0.87
PE02	100%	1.0
PE03	90%	0.87
PE04	80%	0.73
PE05	90%	0.87

6.3.6.4 Correlation between Sentence Length and Evaluation Time

Our evaluation interface is capable of logging the time spent by the post-editors in evaluating each sentence. One may expect that post-editors may spend more time in evaluating longer sentences and less time evaluating shorter sentences. We calculated Pearson's product moment correlation between the evaluation time and sentence length as shown in Table 6.10. The results appear to be inconclusive: we observe a high correlation between the evaluation time and sentence length for PE02 and PE05; however, for the other three

post-editors, there is a low correlation. These inconclusive results can partly be attributed to the fact that we did not compel the post-editors to conduct their evaluation in one session. We expect to achieve more conclusive results in future work, which would happen in a real working post-editing environment.

Table 6.10: Pearson's Product Moment Correlation

Post-Editor ID	PMCC (r)	r-square
PE01	0.2246	0.0505
PE02	0.6957	0.4840
PE03	0.3916	0.1534
PE04	0.0746	0.0056
PE05	0.4907	0.2408
Average	0.2274	0.0517

6.4 Analysis of Reranking Performance

Following Section 6.3, in this section we analyze the performance of the reranking model using segments extracted from the same set of data as described in Section 6.2. Due to resource limitations, we do not measure whether the reranking model has produced correct complete rankings, as that would need much more effort for human judges. Instead, we try to focus on whether the translation candidates that are easier to edit than the original top TM outputs are actually ranked higher by our reranking model. Considering this, we ask the post-editors to judge between two segments: one is the Top TM output, and another is an alternative output from either the MT or the TM system, so that we can learn whether the reranking model outperforms the 1-best TM output.

6.4.1 Precision and Recall of Translation Reranking

In Table 6.11 we present the precision and recall of favoring the alternative translation. Compared to Table 6.1, the precision and recall of the reranking model both decline. (Note that in Table 6.1, the human-judged precision ranges from 0.8734 to 0.9315, and the recall ranges from 0.9138 to 0.9409) The precision of the reranking model still remains solid

Table 6.11: Precision and Recall of Reranking, Individual Post-editors

Post-Editor ID	Precision	Recall	Scratch
PE01	0.8345	0.5762	6
PE02	0.9327	0.5879	86
PE03	0.8750	0.5583	83
PE04	0.8250	0.6074	58
PE05	0.8786	0.5829	16

in Table 6.11, but we observe a larger decline in recall, from the 0.9–1.0 range for the recommendation model to the 0.5–0.6 range in the reranking model.

We suspect that the reason for this is because we use a smaller number of features in this task, among which the language model-related features will inherently favor the TM output and will lead our reranking model to be more conservative in favoring MT outputs over the TM outputs.

6.4.2 Precision and Recall on Consensus Preferences

Table 6.12: Precision and Recall of Recommendation, Consensus Preferences of $N = 3, 4$ Post-Editors

N	Precision	Recall
N=3	0.8583	0.5860
N=4	0.7323	0.5886

Following Section 6.3.2, we also calculate the precision and recall of the reranking model against the consensus of post-editors in Table 6.12. The trend of the results is similar to those in Table 6.3 and Table 6.4. In the N=3 case the precision and recall calculated against the judgements of individual post-editors is stable comparing to the baselines, but the precision drops when it comes to N=4, as the impact of the disagreement among the post-editors becomes a major issue.

Table 6.13: Precision and Recall of Recommendation, Consensus Preferences of $N = 3$ Post-Editors Grouped by the Source of the Alternative Translation

Source	Precision	Recall	Segments	Sys-humaneval-Favor	Human-Favor
MT	0.8689	0.5824	238	122	182
TM	0.6000	0.7500	62	5	4

To see how our reranker performs on the segments produced by MT and the segments produced by TM respectively, we group the segments where 3 post-editors reach consensus into two groups according to the source of the alternative translation. We then calculate precision and recall numbers separately within these two groups in Table 6.13. We can interpret the result from three angles: firstly it again confirms that in our task the outputs from the MT prevail over those from the TM, as more than half of the k-best MT outputs are favored over the 1-best TM output; secondly, our model performs steadily when reranking the MT outputs which achieves our aim of TM-MT integration; and finally, the fact that post-editors favor many of the segments from the K-best TM or MT output confirms the necessity to utilize the k-best output in TM-MT integration.

6.4.3 The TER Score and the Preference of Post-Editors

As in Section 6.3.3, we measure the TER score of the TM and the alternative outputs, and sort them according to the post-editors' preferences in Table 6.14, where we still average the scores among the 5 post-editors.

Table 6.14: TER Scores Sorted by Preference

	TM-Top1	Other	Tie	Scratch
TM-Top1	34.99	61.49	32.04	81.76
Other	54.70	36.79	38.18	72.26

In Table 6.14, the trend continues to show that TER is a good predictor for post-editing preference, confirming the results in Table 6.5: the output which is preferred by the post-editors will have the lower TER score. We also note that the TER scores in this table are higher than their counterparts in Table 6.5, because in this task the top-k outputs are included, which are supposed to have lower quality than the Top-1 outputs used in the recommendation model.

6.4.4 Accuracy on High Fuzzy Match Segments

In Table 6.15 we report the accuracy of ranking choices following the setting in Section 6.3.5. The *Sys_Acc* row reports the accuracy of our ranking system, and the *Baseline* row reports the corresponding baseline that uses the fuzzy match score as the choice threshold. We still see that the reranking system outperforms the baseline, though with a smaller margin than the binary recommender. This provides further evidence that the ranking model does offer better translation options, and can still find better translations from the MT and TM k-best lists when the TM segment is of high quality.

Table 6.15: Ranking Accuracy on High Fuzzy Match Segments

	Fuzzy Match						
	0.5	0.55	0.6	0.65	0.7	0.75	0.8
Sys_Acc	0.5471	0.5382	0.5330	0.5349	0.5379	0.5400	0.5406
Baseline	0.3882	0.4236	0.4400	0.4762	0.4889	0.4937	0.5000
PreferMT/Total	104/170	83/144	70/125	55/105	46/90	40/79	31/62

6.4.5 User Behavior

We investigate the users' behavior during the evaluation of the reranking system using similar measures as in Section 6.3.6.

6.4.5.1 Inter-annotator Agreement

Fleiss' Kappa coefficient for our five post-editors is 0.479 ± 0.012, indicating a moderate agreement. We also obtained Fleiss' Kappa coefficient for each category as shown in Table 6.16. From this table, we can observe moderate agreements among post-editors in selecting TM or MT output as the most suitable for post-editing. There is also a moderate agreement in making their decision to translate from scratch. However, there is only a fair agreement in determining whether TM and MT outputs are equally good for post-editing ("Tie").

Table 6.16: Annotator agreement for each category

Category	Kappa
TM	0.516
MT	0.593
Tie	0.344
Scratch	0.328

6.4.5.2 Intra-annotator Agreement

We also have ten duplicate samples in our evaluation intended to measure the level of intrinsic agreement for each post-editor, as in Section 6.3.6. Both percentage of agreement and Cohen's Kappa are calculated as shown in Table 6.17. For the ranking evaluation, we can still observe that all five post-editors achieved almost perfect intrinsic agreement.

Table 6.17: Intra-annotator Agreement

Post-Editor ID	Agreement	Kappa
PE01	80%	0.73
PE02	90%	0.87
PE03	90%	0.87
PE04	100%	1.0
PE05	100%	1.0

6.4.5.3 Correlation between Sentence Length and Evaluation Time

As in Section 6.3.6 we calculated Pearson's product moment correlation between the evaluation time and sentence length as shown in Table 6.18. The trend is similar: this time we observe a high correlation between the evaluation time and sentence length for PE01, PE02 and PE05; for the other two post-editors, there is still a low correlation. As is analyzed in Section 6.3.6, the reason could be that we did not compel the post-editors to conduct their evaluation in one session during the experiments, and expect to achieve more conclusive results in future work, which would happen in a real working post-editing environment.

Table 6.18: Pearson's Product Moment Correlation

Post-Editor ID	PMCC (r)	r-square
PE01	0.4683	0.2193
PE02	0.5439	0.2958
PE03	0.2022	0.0409
PE04	0.1486	0.0221
PE05	0.6242	0.3896
Average	0.4605	0.2120

6.5 Discussions on Feedback from Post-editors

We requested post-editors to comment on their attitude to MT and TM. In our questionnaire, all post-editors claim that they have post-edited MT outputs and think that MT is sometimes useful, which might be said to be representative of the current state of MT penetration in the localization industry.

However, the more interesting comment comes from one of our post-editors in private communication, that we think could be worthwhile to note:

> I think that I managed to detect that the TM-based translation was better. Some segments didn't need any changes (or needed very little changes), that was mainly the case for short segments.

Although the post-editor does not know which of the two candidates we present in the evaluation interface is from the MT system, he claims after completing the evaluation that he has found that the TM outputs are more suitable for post-editing, although in fact every post-editor prefers MT outputs in the experiment (cf. Table 6.7 and Table 6.13).

Although this can only reflect the thinking of a single post-editor, this comment is still revealing for two reasons. First of all, the post-editor obviously mistakes MT outputs for TM outputs, which indicates that in this closed-domain setting mainly composed of simple short sentences, a state-of-the-art phrase-based SMT system is able to produce outputs that are not only correct on the word-to-word level, but also grammatically acceptable enough to be recognized as human translations in the TM, and therefore that the SMT output can be smoothly integrated into the TM environment.

The comment also shows how much the post-editors subconsciously trust the TM. This may be an explanation for the relatively low acceptance of MT technology in the localization industry, and demonstrates the need for TM–MT integration techniques, such as ours.

6.6 Related Work

The translation recommendation system we experiment with is an implementation of the translation recommendation model proposed in [He et al., 2010c], and the reranking model is first proposed in [He et al., 2010d]. Research related to the recommendation and reranking models is already reviewed in Chapter 4 and Chapter 5.

As regards other UIs that are capable of evaluating post-editing efficiency, [Koehn and Haddow, 2009] presents a post-editing environment using information from the phrase-based SMT system Moses [Koehn et al., 2007], instead of the fuzzy match information from TMs. The web-based UI is built with the Ruby on Rails (RoR) framework,[3] and is available online at `http://tool.statmt.org/`.

The research presented in this paper focuses on aspects of a user study of post-editors working with MT and TMs. In this respect, it is related to [Guerberof, 2009], which compares the post-editing effort required for MT and TM outputs respectively, as well as [Tatsumi, 2009], which studies the correlation between automatic evaluation scores and post-editing effort. Our work differs in that our research measures how the integration of TM and MT systems can help post-editors, not how post-editors perform using separate TM or MT systems.

6.7 Summary

In this chapter, we evaluated the effectiveness of translation recommendation and translation reranking in the context of TM–MT integration with professional post-editors. The evaluation results support validation of the utility of both translation recommendation and

[3] `http://rubyonrails.org`

translation reranking paradigms, as well as our approach of using automatic evaluation metrics to approximate actual post-editing effort.

We find that a translation recommendation model trained on automatic evaluation metric scores can obtain a precision above 0.9 and a recall above 0.75 with proper thresholds according to each of the post-editors. The model shows precision above 0.8 when we evaluate against the consensus of post-editors.

For the translation reranking paradigm, altough it tries to present translators with more segments at the cost of possibly including low quality segments, it can still obtain 0.85 precision and 0.58 recall when evaluated against the consensus judgement of 3 translators. It can also outperform the naive baseline which uses TM fuzzy match score as threshold.

From the analysis of user behaviour, we note that the users show consistency in their judgements according to both the inter-annotator agreement and the intra-annotator agreement for both the recommendation and the reranking tasks. The recommended MT outputs are incorrectly recognized as TM outputs by one post-editor, which shows both the potential and the necessity for TM–MT integration.

In future, we can further extend the evaluation in several ways. First of all, in this paper we concentrated on proprietary data and professional post-editors, according to the major paradigm in the localization industry. However, at the same time this limits the number of annotators we can hire, as well as the types of evaluations we can perform. We can obtain more comprehensive results by experimenting on open-domain data sets, and applying crowd-sourcing technologies such as Amazon Mechanical Turk[4] [Callison-Burch, 2009].

Secondly, during the evaluation we were able to collect a number of human judgements for training a new translation recommendation system. We plan to train a new recommendation model and to compare the difference with models trained on automatic metric scores, when we have collected more human-annotated data.

Finally, this experiment can also be extended by measuring the actual post-editing time

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

instead of the judgement time, which can lead to a more precise approximation of reduced post-editing effort when using translation recommendation to integrate MT outputs into a TM system.

Chapter 7

Towards Consistent Sub-Segment MT-TM Integration

7.1 Introduction[1]

In previous chapters, we presented methods to integrate SMT into TM systems, while the strengths of TMs – effective cost estimation, friendly integration with CAT, and highly reusable high fuzzy match chunks – were all kept intact.

Both our translation recommendation and translation reranking schemes operate on the segment level. However, TM fuzzy matches may contain some chunks of higher quality than SMT outputs, while not having enough content words in the input correctly translated or translated at all. In such cases, our recommender or reranker will favor SMT outputs, and is not able to leverage the information (chunks) from TM fuzzy matches.

Let us look at a segment from the Symantec English–Chinese TM database as an example, as in Table 7.1.

In this example, we are able to find a fuzzy match in the TM, which perfectly corresponds to the second part of the source segment, but lacks the first part of the source. If

[1]The idea of selecting TM markups with discriminative learning was first conceived by Yanjun Ma. An earlier version of the research presented in this paper has been published in [Ma et al., 2011]. The feature set we use in this chapter is different from that of [Ma et al., 2011], and leads to stronger results.

Table 7.1: Motivating Example

Source	after policy name , type the name of the policy (it shows new host integrity policy by default) .
TM Source	type the name of the policy (it shows new host integrity policy by default) .
TM Output	键入 策略 名称 （ 默认 显示 “ 新 主机 完整性 策略 ” ） 。
MT Output	在 “ 策略 ” 名称 后面 ， 键入 策略 的 名称 (名称 显示 为 “ 新 主机 完整性 策略 默认 ） 。
Reference	在 “ 策略 名称 ” 后面 ， 键入 策略 名称 （ 默认 显示 “ 新 主机 完整性 策略 ” ） 。

we use the translation recommendation scheme, translators will have to choose between the MT output, which makes several translation mistakes, and the TM output, which misses the beginning of the segment completely.

In a commercial localization setting, however, we would ideally hope to leverage both the TM and the MT outputs, because 1) the combination of the TM translation from the second part of the segment and the MT output from the first part together can produce a better translation, and 2) more importantly, in the context of localization, we hope that translations exhibit *consistency*, so that the same technical phrases in one language always correspond to the same translations in another language.

Following this intuition, we extend the translation recommendation and translation reranking schemes to the sub-segment level, and show that by automatically selecting TM matches that ensure consistent translation, and reusing them in a constrained SMT pipeline, we can obtain better SMT outputs that incorporate the knowledge from such TM matches. We will show by automatic evaluation that, apart from ensuring that consistent translation chunks are reused, our method also produces better translations, reflected by a 1.2 BLEU point improvement (2.62% relative) and a 0.72 TER point reduction (1.81% relative), both of which are statistically significant.

In the following sections, we will first discuss the idea of translation consistency in the TM and the SMT setting in Section 7.2. Then we present the two components of our sub-segment integration scheme: the constrained translation framework using discriminative learning in Section 7.3, and our rich linguistically-motivated feature set in Section 7.4. We present experimental results and compare the effectiveness of different types of features in

Section 7.5. We review previous work in Section 7.6. We conclude and point out possible avenues for future work in Section 7.7.

7.2 Translation Consistency in TM and SMT

Translation consistency is an important factor for large-scale translation, especially for commercial translations in an industrial environment. For example, when translating technical documents (especially those with a large amount of terminology), lexical as well as structural consistency is essential to produce a fluent target-language segment. Moreover, even in the case of translation errors, consistent in the errors (e.g. repetitive error patterns) are easier to diagnose and subsequently correct by translators.

In phrase-based SMT, translation models and language models are automatically learned and/or generalized from the training data, and a translation is produced by maximizing a weighted combination of these models. Given that global contextual information is not normally incorporated, and that training data is usually noisy in nature, there is no guarantee that an SMT system can produce translations in a consistent manner.

On the other hand, TM systems – widely used by translators in industrial environments for enterprise localization by translators – can shed some light on mitigating this limitation. TM systems can assist translators by retrieving and displaying previously translated similar "example" segments (displayed as source-target pairs, widely called 'fuzzy matches' in the localization industry). In TM systems, fuzzy matches are retrieved by calculating the similarity or the so-called 'fuzzy match score' (ranging from 0 to 1 with 0 indicating no matches and 1 indicating a full match) between the input segment and segments in the source side of the translation memory.

When presented with fuzzy matches, translators can then avail of useful chunks in previous translations while composing the translation of a new segment. One might expect that most translators only consider a few segments that are most similar to the current input segment; this process can inherently improve the consistency of translation, given that the

new translations produced by translators are likely to be similar to the target side of the fuzzy match they have consulted.

Previous research (cf. Section 7.6) has focused on using fuzzy match score as a threshold when using the target side of the fuzzy matches to constrain the translation of the input segment. In this chapter, we make two improvements over the state-of-the-art:

- **Discriminative Learning**. As we do in the translation recommendation and translation reranking paradigms, we use a more fine-grained discriminative learning method to determine whether the target side of the fuzzy matches should be used as a constraint in translating the input segment.

- **A Rich Feature Set**. The only factor that prior thresholding methods consider is the fuzzy match score. However, we notice that many factors are relevant in deciding whether the matched TM chunks should be reused in constrained translation: therefore we use translation model, lexical, syntactic (dependency), and semantic features to model translation consistency. This not only leads to translations of better quality, but also provides insight into the linguistic properties of consistent translation chunks.

We will demonstrate that by using discriminative learning and a rich feature set, our method can consistently improve translation quality, and outperform the naive fuzzy match-driven baseline.

7.3 Constrained Translation with Discriminative Learning

We introduce our method to tightly integrate TM with MT at the sub-subsegment level. The basic idea is as follows: given a source segment to translate, we firstly use a TM system to retrieve the most similar "example" source segments together with their translations. If matched chunks between input segment and fuzzy matches can be detected, we can directly reuse the corresponding parts of the translation in the fuzzy matches, and use an MT system to translate the remaining chunks.

As a matter of fact, implementing this idea is pretty straightforward. A TM system can easily detect the word alignment between the input segment and the source side of the fuzzy match by retracing the paths used in calculating the fuzzy match score. To obtain the translation for the matched chunks, we just require the word alignment between source and target TM matches, which can be addressed using state-of-the-art word alignment techniques. More importantly, albeit not explicitly spelled out in previous work (e.g. [Koehn and Senellart, 2010b]), this method can potentially increase the consistency of translation, as the translation of new input segments is closely informed and guided (or constrained) by previously translated segments.

Now we define this idea formally. Given a segment $\mathbf{e}$ to translate, we retrieve the most similar segment $\mathbf{e}'$ from the TM associated with target translation $\mathbf{f}'$. The m common "phrases" $\bar{\mathbf{e}}_{\mathbf{1}}^{\mathbf{m}}$ between $\mathbf{e}$ and $\mathbf{e}'$ can be identified. Given the word alignment information between $\mathbf{e}'$ and $\mathbf{f}'$, one can obtain the corresponding translations $\bar{\mathbf{f}}'^{\mathbf{m}}_{\mathbf{1}}$ for each of the phrases in $\bar{\mathbf{e}}_{\mathbf{1}}^{\mathbf{m}}$ (cf. Section 7.3.1). This process can derive a number of "phrase pairs" $< \bar{e}_m, \bar{f}'_m >$, which can be used to specify the translations of the matched phrases in the input segment. The remaining words without specified translations will be translated by an MT system.

For example, given an input segment $e_1 e_2 \cdots e_i e_{i+1} \cdots e_I$, and a phrase pair $< \bar{e}, \bar{f}' >$, $\bar{e} = e_i e_{i+1}$, $\bar{f}' = f'_j f'_{j+1}$ derived from the fuzzy match, we can mark up the input segment as in (7.1):

$$e_1 e_2 \cdots < \text{tm}=\text{“}f'_j f'_{j+1}\text{”} > e_i e_{i+1} < /\text{tm} > \cdots e_I. \tag{7.1}$$

We decode this segment, and only the unmarked portion $e_1 e_2 \cdots e_{i-1}$ and $e_{i+1} \cdots e_I$.will be translated, while the marked-up portion $e_i e_{i+1}$ will reuse the translation from the TM, which is $f'_j f'_{j+1}$.

7.3.1 Consistent Phrase Pair Extraction

The consistent "phrase pairs" we derive from the symmetric alignment between the TM fuzzy match and its translation are different from the phrase pairs extracted as translation rules in phrase-based translation. To achieve sufficient rule coverage, typical phrase-based SMT systems will extract all the rules that do not conflict with the alignment points, while our "phrase pairs" should directly correspond to alignment points in order to ensure that our phrase pairs represent much more consistent translation options (at the cost of lower coverage) than typical phrasal translation rules.

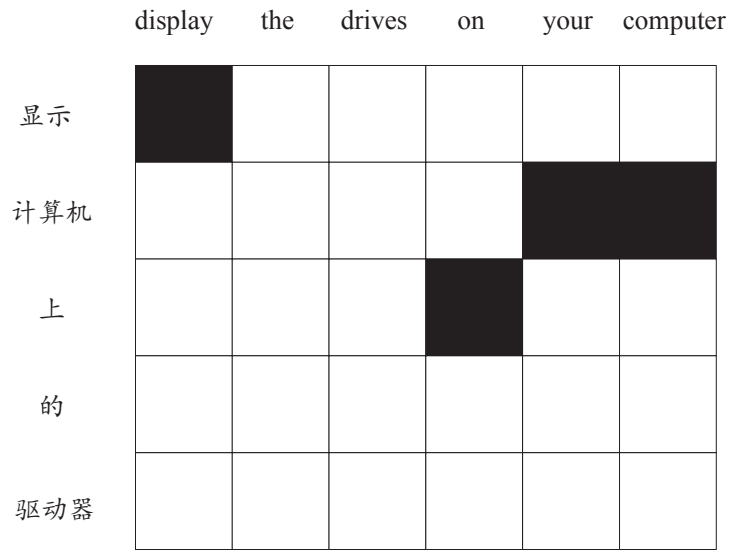

Figure 7.1: Consistent Phrase Pair Extraction

We illustrate this difference using the following example. Suppose that we have an alignment between English and Chinese as in Figure 7.1. Our method to extract consistent phrase pairs only obtains two pairs that are directly derived from this alignment, as in (7.2):

$$\begin{aligned} \text{display} &\mapsto \text{显示} \\ \text{on your computer} &\mapsto \text{计算机 上} \end{aligned} \tag{7.2}$$

Phrasal extraction heuristics in phrase-based SMT [Koehn et al., 2003] have the capa-

bility of extracting a longer consistent phrase pair, as in (7.3)

$$\text{the drives on your computer} \mapsto \text{计算机 上 的 驱动器} \tag{7.3}$$

However, we should not rely on such heuristics in our model because they cannot ensure the consistency of phrase pairs, and we do not have a probability weighting step to rule out any inconsistent pairs. Some of the inconsistent phrase pairs that can be derived from this alignment include those in (7.4):

$$\begin{aligned} &\text{the drives on your computer} \mapsto \text{计算机上} \\ &\text{on your computer} \mapsto \text{计算机上 的 驱动器} \end{aligned} \tag{7.4}$$

The method used to obtain the constrained alignment using TM fuzzy matches is similar to [Koehn and Senellart, 2010b], except that in our case the word alignment between $\mathbf{e}'$ and $\mathbf{f}'$ is the intersection of bidirectional GIZA++ [Och and Ney, 2003] posterior alignments. In marking up the input segment, we use the intersected word alignment to minimize the noise introduced by word alignment in only one direction, so as to ensure translation consistency.

7.3.2 Discriminative Learning

In our approach, whether the translation information from fuzzy matches should be used or not (i.e. whether the input segment should be marked up) is determined by a discriminative learning procedure. We cast this problem as a binary classification problem.

7.3.2.1 Support Vector Machines

Similar to our work on recommendation and ranking for full TM and MT segements, here we use SVMs [Cortes and Vapnik, 1995], binary classifiers that classify an input instance based on decision rules which minimize the regularized error function in (7.5) to determine whether constraining translation with our consistent phrase pairs can help translation

quality:

$$\begin{aligned} \min_{w,b,\xi} \quad & \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_{i=1}^{l}\xi_i \\ \text{s. t.} \quad & y_i(\mathbf{w}^T\phi(\mathbf{x}_i) + b) \geqslant 1 - \xi_i \\ & \xi_i \geqslant 0 \end{aligned} \tag{7.5}$$

where $(\mathbf{x}_i, y_i) \in R^n \times \{+1, -1\}$ are l training instances that are mapped by the function ϕ to a higher dimensional space. $\mathbf{w}$ is the weight vector, ξ is the relaxation variable and $C > 0$ is the penalty parameter.

We perform our experiments with the Radial Basis Function (RBF) kernel, as in (7.6):

$$K(\mathbf{x}_i, \mathbf{x}_j) = exp(-\gamma||\mathbf{x}_i - \mathbf{x}_j||^2), \gamma > 0 \tag{7.6}$$

When using SVMs with the RBF kernel, we have two free parameters to tune on: the cost parameter C in (7.5) and the radius parameter γ in (7.6). We optimize the parameters C and γ by a brute-force grid search. The classification result of each set of parameters is evaluated by cross validation on the training set.

The SVM classifier will thus be able to predict the usefulness of the TM fuzzy match, and determine whether the input segment should be marked up using relevant phrase pairs derived from the fuzzy match before being sent to the SMT system for translation.

When training SVMs, we need gold standard annotations to label training examples. As large-scale manually annotated data is not available for this task, we use automatic TER scores [Snover et al., 2006] as the measure for training data annotation.

We label the training examples as in (7.7):

$$y = \begin{cases} +1 & \text{if } TER(\text{w. markup}) < TER(\text{w/o markup}) \\ -1 & \text{if } TER(\text{w/o markup}) \geq TER(\text{w. markup}) \end{cases} \tag{7.7}$$

Each instance is associated with a set of features which are discussed in more detail in Section 7.4.

7.3.2.2 Classification Confidence Estimation

We use the techniques proposed by Platt [1999] and improved by Lin et al. [2007] to convert classification margin to posterior probability, so that we can easily threshold our classifier (cf. Section 7.5.3.3).

Platt's method estimates the posterior probability with a sigmoid function, as in (7.8):

$$Pr(y = 1|\mathbf{x}) \approx P_{A,B}(f) \equiv \frac{1}{1 + exp(Af + B)} \tag{7.8}$$

where $f = f(\mathbf{x})$ is the decision function of the estimated SVM. A and B are parameters that minimize the cross-entropy error function F on the training data, as in (7.9):

$$\begin{aligned} &\min_{z=(A,B)} F(z) = -\sum_{i=1}^{l}(t_i log(p_i) + (1 - t_i) log(1 - p_i)), \\ &\text{where } p_i = P_{A,B}(f_i), \text{and } t_i = \begin{cases} \frac{N_+ + 1}{N_+ + 2} \text{ if } y_i = +1 \\ \frac{1}{N_- + 2} \text{ if } y_i = -1 \end{cases} \end{aligned} \tag{7.9}$$

where $z = (A, B)$ is a parameter setting, and N_+ and N_- are the numbers of observed positive and negative examples, respectively, for the label y_i. These numbers are obtained using an internal cross-validation on the training set.

7.4 Feature Set

The features used to train the discriminative classifier, all on the segment level, are described in the following sections.

7.4.1 Translation Model Features

We begin with features extracted from the internals of the TM and the MT components, as these are the features that we also use in our translation recommendation and translation reranking models.

- **The TM Feature**. The TM feature is the fuzzy match score, which indicates the overall similarity between the input segment and the source side of the TM output. If the input segment is similar to the source side of the matching segment, it is more likely that the matching segment can be used to mark up the input segment.

  We compute fuzzy match cost as the minimum Levenshtein Distance [Levenshtein, 1966] between the source and TM entry, normalised by the length of the source as in (7.10), as most of the current implementations are based on edit distance while allowing some additional flexible matching (cf. Chapter 2).

$$h_{fm}(\mathbf{e}) = \min_{\mathbf{s}} \frac{LevenshteinDistance(\mathbf{e}, \mathbf{s})}{Len(\mathbf{e})} \tag{7.10}$$

  where $\mathbf{e}$ is the segment to translate, and $\mathbf{s}$ is the source side of an entry in the TM. For fuzzy match scores F, h_{fm} roughly corresponds to $1 - F$.

- **Translation Features**. We use four features from the SMT translation model: the phrase translation and lexical probabilities for the phrase pairs $< \bar{e}_m, \bar{f'}_m >$ derived using the method in Section 7.3. More specifically, we use the phrase translation probabilities $p(\bar{f'}_m|\bar{e}_m)$ and $p(\bar{e}_m|\bar{f'}_m)$, as well as the lexical translation probabilities $p_{lex}(\bar{f'}_m|\bar{e}_m)$ and $p_{lex}(\bar{e}_m|\bar{f'}_m)$ as calculated in [Koehn et al., 2003]. In cases where multiple phrase pairs are used to mark up one single input segment $\mathbf{e}$, we use a unified score for each of the four features, which is an average over the corresponding feature in each phrase pair. The intuition behind these features is as follows: phrase pairs $< \bar{e}_m, \bar{f'}_m >$ derived from the fuzzy match should also be reliable with respect to statistically produced models.

  We also have a count feature, i.e. the number of phrases used to mark up the input segment, and a binary feature, i.e. whether the phrase table contains at least one phrase pair $< \bar{e}_m, \bar{f'}_m >$ that is used to mark up the input segment.

7.4.2 Linguistic Features

Now we move on to linguistic features ranging from the surface to the semantic level. The linguistic-oriented features measure how well the marked-up portion covers the source segment. The assessments could be (but are not limited to) the percentage of content words that are marked up (lexical level), the number of covered nouns (Part-of-speech (POS) level), the type and number of covered dependency relations (syntactic dependency level), and whether the agent of the main predicate is covered completely (semantic level). We also measure position-related properties, such as whether the marked-up chunk is at the beginning or the end of the segment.

7.4.2.1 Lexical Features

The lexical features reveal the surface-level properties of the marked-up translation. We use the following indicators given a segment and its markup:

- **Coverage**. Coverage measures the percentage of words covered by the marked up segment. We calculate the percentage on both the source and the target side.
- **Alphabetical Words**. This feature measures the percentage of words that are alphabetical (i.e. not numbers and punctuation marks) in the source side of marked up chunks.
- **Punctuation Marks**. This feature in turn measures the percentage of words in the source side of marked up chunks that are punctuation marks.
- **Content Words**. This feature calculates the percentage of content words in the source side of marked up chunks. We use the snowball stop words list[2] as the resource for function words and consider all other words to be content words.
- **Position**. We use two binary features which fire if marked-up chunks cover the head or the tail of the source segment.

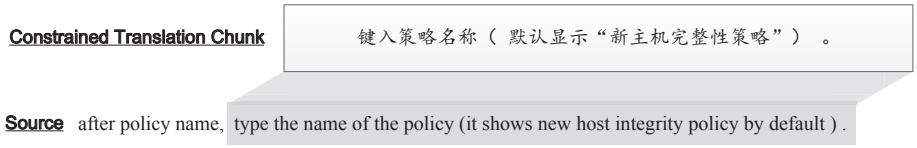

Figure 7.2: Lexical Features

We give an example of these features in Figure 7.2. In this example, the shaded chunk “type the name of the policy (it shows new host integrity policy by default).” is marked up with a corresponding Chinese translation. We extract features on the input segment. The length of the input segment is 21 and the length of the marked up chunk is 17. Therefore we have the coverage feature LEX_COVER=$\frac{17}{21}$. We also calculate the the percentage of alphabetical words, punctuation marks and content words in the marked up chunk. For example, there are 3 punctuation marks in the marked up chunk so we have LEX_PUNCT=$\frac{3}{17}$. Besides, the tail of this segment is covered by the markup, so the position feature will fire.

7.4.2.2 POS Features

For the POS features, we simply extend the calculation of lexical features to the POS level. The POS tags in our experiments are obtained using the Stanford Parser.[3]

The POS features we use are:

- **POS Coverage**. We calculate the percentage of coverage by the markup for each POS tag in the source segment.
- **POS Position**. We also use binary features to indicate whether the head or the tail of the source segment is covered by the markup, sorted by POS tags.

We illustrate the POS features in Figure 7.3. If we look at the VBZ tag, our markup covers the only third person singular verb in the segment, so the POS_COVER_VBZ feature is 1.0. The POS_TAIL_. feature will also fire as the markup covers the full stop at the tail

[2] http://snowball.tartarus.org/algorithms/english/stop.txt
[3] http://nlp.stanford.edu/software/lex-parser.shtml

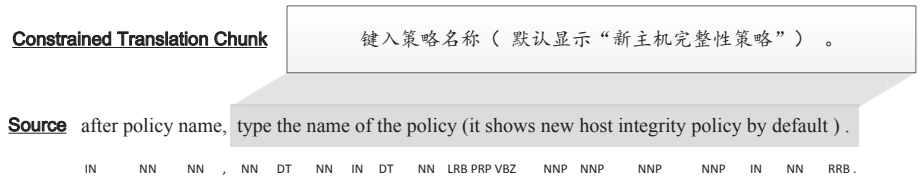

Figure 7.3: Part-of-speech Features

of the segment. Note that the word “type” is mistakenly tagged as NN (instead of VBP), so this will introduce errors in deeper linguistic analysis. This also confirms the necessity of using a richer set of features so that analysis errors can be compensated for by surface-level indicators in the whole feature set.

7.4.2.3 Dependency Features

Given the phrase pairs $< \bar{e}_m, \bar{f'}_m >$ derived from the fuzzy match, and used to translate the corresponding chunks of the input segment (cf. Section 7.3), these translations are more likely to be coherent in the context of the particular input segment if the matched parts on the input side are syntactically related.

We use dependency relations to capture this syntactic relationship. For marked-up phrases $\bar{e}_m$ in the source segment, we use dependency relations between words e_m in $\bar{e}_m$ and the remaining words e_j in the input segment e to determine their syntactic function.

We use the Stanford parser to obtain the dependency structure of the input segment. We add a pseudo-label SYS_PUNCT to punctuation marks, whose governor and dependent are both the punctuation mark. The dependency features designed to capture the context of the matched input phrases $\bar{e}_m$ are as follows:

- **DEP Coverage**. DEP coverage measures the coverage of dependency labels on the input segment in order to obtain a bigger picture of the matched parts in the input. For each dependency label *L*, we consider its head or modifier as *covered* if the corresponding input word e_m is covered by a matched phrase $\bar{e}_m$. Our coverage features are the frequencies of governor and dependent coverage calculated separately for

each dependency label.

- **DEP Position**. DEP position identifies whether the head and the tail of a segment are matched, as these are the cases in which the matched translation is not affected by the preceding words (when it is the head) or following words (when it is the tail), and is therefore more reliable. The feature is set to 1 if this happens, and to 0 otherwise. We distinguish among the possible dependency labels, the head or the tail of the segment, and whether the aligned word is the governor or the dependent just like we do for POS tags. As a result, each permutation of these possibilities constitutes a distinct binary feature.

- **DEP Consistency**. DEP Consistency is a single feature which determines whether matched phrases $\bar{e}_m$ belong to a consistent dependency structure, instead of being distributed discontinuously in the input segment. We assume that a consistent structure is less influenced by its surrounding context. We set this feature to 1 if every word in $\bar{e}_m$ is dependent on another word in $\bar{e}_m$, and to 0 otherwise.

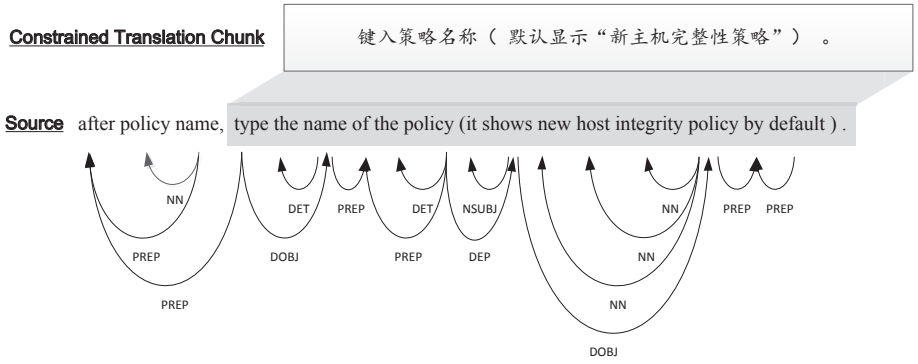

Figure 7.4: Dependency Features

We give an example for dependency features in Figure 7.4. For the dependency label `DOBJ`, we have two relations `DOBJ(type, name)` and `DOBJ(shows, policy)`. The governors “type” and “shows” are both covered by the markup, so we have `DEP_DOBJ_GOV`=1.0. This is also the case for the dependents, so we also have `DEP_DOBJ_DEP`=1.0. There is a

mistakenly annotated PREP arc from the marked-up "type" to the unmarked-up "after", so the consistency feature will not fire. The position feature regarding the tail of the segment will fire.

7.4.2.4 Semantic Role Features

We also suspect that the usefulness of marked-up constrained translations is related to their semantic role [Gildea and Jurafsky, 2002] in the segment. For example, we suspect that if the agent of the predicate is completely marked-up and has a constrained translation, the overall consistency of the segment might improve, especially for the case of Symantec technical documents, as agents are often either user roles (e.g. "the administrator") or product names (e.g. "symantec mail security console") that require a high level of translation consistency.

Our semantic role labels are obtained using the SRL labeler described in [Li et al., 2009], with constituent trees produced by the Stanford parser as input. The labels follow the PropBank [Palmer et al., 2005] annotation. We use the following semantic role features in our system:

- **SEM Coverage**. We calculate the marked-up percentage for each argument label. If there is more than one predicate, the percentage is averaged among argument labels for each predicate. We label these features as SEM_PARTIAL_*.

- **SEM Complete Coverage**. This feature is a binary feature that fires if phrases with argument label ArgN, are completely covered by the markup. If there is more than one predicate, the binary feature requires that all ArgNs are completely covered. In other words, SEM_COMPLETE_ARGN fires if and only if SEM_COVER_ARGN is equal to 1.0.

- **SEM Position**. The SEM position feature fires if an argument at the beginning or the end of the segment is covered by the markup. We also distinguish among cases when the coverage is partial or complete, so if part of an agent (ARG0 in PropBank) chunk

is partially marked up at the head of the segment, the SEM_POSITION_ARG0_HEAD feature will fire.

- **SEM Predicate**. The PropBank-style semantic role labels are predicate driven: the labeler first identifies the predicate of a segment and then labels its arguments. If there is no predicate, the whole segment will not be labeled and our semantic features will not fire. To distinguish this situation from the cases when there are semantic labels but the markup covers none of them, we design a binary feature that fires only if the segment has no predicate.

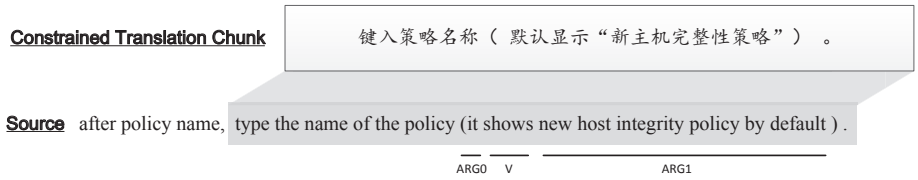

Figure 7.5: Semantic Role Features

We give an example for semantic role features in Figure 7.5. In this example, the ARG0, ARG1, and V roles are all covered by the markup, so we will have SEM_COMPLETE_* and SEM_PARTIAL_* equal to 1.0. The position-based features will not fire, as the ending punctuation marks are not covered by semantic role labels. Note that if analyzed correctly, “type” should also be a predicate and should have its own arguments. If so, we will have an uncovered AM-LOC chunk “after policy name” with coverage features equal to 0.0, but all other features will remain the same.

7.5 Experiments

Our data set is an English–Chinese TM with technical translation from Symantec, consisting of 87K segment pairs. The average segment length of the English training set is 13.3 words and the size of the training set is comparable to the larger TMs used in the industry. Detailed corpus statistics about the training, development and test sets for the SMT system

are shown in Table 7.2.

Table 7.2: Corpus Statistics

	Train	Develop	Test
SEGMENTS	86,602	762	943
ENG. TOKENS	1,148,126	13,955	20,786
ENG. VOC.	13,074	3,212	3,115
CHI. TOKENS	1,171,322	10,791	16,375
CHI. VOC.	12,823	3,212	1,431

The composition of test subsets based on fuzzy match scores is shown in Table 7.3. We can see that segments in the test sets are longer than those in the training data, implying a relatively difficult translation task.

We train the SVM classifier using the libSVM Chang and Lin [2001] toolkit. The SVM-training and validation is on the same training segments[4] as the SMT system with 5-fold cross validation. As for SVM parameters, we set $c = 2.0$ and $\gamma = 0.125$.

Table 7.3: Composition of test subsets based on fuzzy match scores

Scores	segments	Words	W/S
(0.9, 1.0)	80	1526	19.0750
(0.8, 0.9]	96	1430	14.8958
(0.7, 0.8]	110	1596	14.5091
(0.6, 0.7]	74	1031	13.9324
(0.5, 0.6]	104	1811	17.4135
(0, 0.5]	479	8972	18.7307

We conducted experiments using a standard log-linear PB-SMT model: GIZA++ implementation of IBM word alignment model 4 [Och and Ney, 2003], the refinement and phrase-extraction heuristics described in [Koehn et al., 2003], minimum-error-rate training [Och, 2003], a 5-gram language model with Kneser-Ney smoothing [Kneser and Ney, 1995] trained with SRILM [Stolcke, 2002] on the Chinese side of the training data, and Moses [Koehn et al., 2007] which is capable of handling user-specified translations for some portions of the input during decoding. The maximum phrase length is set to 7.

[4]We have around 87K segment pairs in our training data. However, for 67.5% of the input segments, our MT system produces the same translation irrespective of whether the input segment is marked up or not. Having said that, our results show that selecting better translations on the approximately one third of segments to which markup does make a difference, leads to significant improvements on the system level.

7.5.1 Evaluation

The performance of the phrase-based SMT system is measured by BLEU score [Papineni et al., 2002] and TER [Snover et al., 2006]. Significance testing is carried out using approximate randomization [Noreen, 1989] with a 95% confidence level.

We also measure the quality of the classification using precision and recall. Let A be the set of predicted markup input segments, and B be the set of input segments where the markup version has a lower TER score than the plain version. We standardly define precision P and recall R as in (7.11):

$$P = \frac{|A \bigcap B|}{|A|}, R = \frac{|A \bigcap B|}{|B|} \tag{7.11}$$

7.5.2 Cross-fold translation

In order to obtain training samples for the classifier, we need to label each segment in the SMT training data as to whether marking up the segment can produce better translations. To achieve this, we translate both the marked-up versions and plain versions of the segment and compare the two translations using the segment-level evaluation metric TER.

We do not make use of additional training data to translate the segments for SMT training, but instead use cross-fold translation. We create a new training corpus T by keeping 95% of the segments in the original training corpus, and creating a new test corpus H by using the remaining 5% of the segments. Using this scheme we make 20 different pairs of corpora (T_i, H_i) in such a way that each segment from the original training corpus occurs in exactly one H_i for some $1 \leq i \leq 20$. We train 20 different systems using each T_i, and use each system to translate the corresponding H_i as well as the marked-up version of H_i using the procedure described in Section 7.3. The development set is kept the same for all systems.

7.5.3 Experimental Results

7.5.3.1 Feature Validation

Table 7.4: Contribution of Features (%)

	TER	BLEU	P	R
BASELINE	39.82	45.80	N/A	N/A
TRANS	39.80	45.84	66.67	1.02
LEX	39.65	46.20	71.43	10.20
POS	39.30	46.71*	61.54	28.57
DEP	39.81	46.14	58.25	30.61
SEM	39.74	46.35	59.09	19.90
LPDS	39.32	46.81*	61.36	41.33

We first validate the contribution of the feature sets we proposed. The classification and translation results using different features are reported in Table 7.4. Scores marked with "*" are statistically significantly better ($p < 0.01$) than the BASELINE.

First of all, we observe that using translation model-derived features similar to those used in our translation recommendation/reranking models only brings about a trivial difference in translation quality. In fact, very low recall indicates that the SVM actually cannot obtain enough information from this feature set, and has to take advantage of the prior distribution of the samples (where we have more negative examples than positive ones) and reject almost every attempt of markup to obtain the best accuracy. This shows that these features cannot capture the properties of the TM chunks that help translation consistency.

Secondly, we observe that the linguistic features can bring more improvement to classification accuracy and translation quality. The improvement in BLEU scores ranges from 0.36 (DEP) to a statistically significant 0.91 (POS). However, that is not to say that deeper features such as DEP and SEM are much less informative than part-of-speech features. We note that POS features reject more marked-up chunks than deeper features (as is indicated by low recall), which means that only a small number of segments can benefit from this approach if we only use the POS feature set. Besides, the low recall also limits the possibility of pursuing even better translation quality by confidence thresholding (i.e. by sacrificing recall to achieve even higher precision). Therefore it would be worthwhile to combine all

these linguistic-driven features for better classification accuracy, and more importantly, for higher recall.

Finally, we put the LEX, POS, DEP, and POS features together in the LPDS setting. We can see that this setting achieves the best BLEU score among all the settings, which is also significantly better than the baseline. The TER and precision numbers are marginally inferior to those obtained using the POS features alone. However, as we will see, the much higher recall enables us to perform more confidence threshold-based tuning and achieve better results.

7.5.3.2 Translation Results with and without Markup

Table 7.5 contains the translation results of the SMT system when we use discriminative learning with LPDS to mark up the input segment (LPDS). The first row (BASELINE) is the result of translating plain test sets without any markup, while the second row is the result when all the test segments are marked up. We also report the oracle scores, i.e. the upperbound of using our discriminative learning approach. As we can see from this table,

Table 7.5: Performance of Discriminative Learning (%)

	TER	BLEU
BASELINE	39.82	45.80
MARKUP	41.62	44.41
LPDS	39.32	46.81*
ORACLE	37.27	48.32

we obtain significantly inferior results compared to the the Baseline system if we categorically mark up all the input segments using phrase pairs derived from fuzzy matches. This is reflected by an absolute 1.4 point drop in BLEU score and a 1.8 point increase in TER. On the other hand, both the oracle BLEU and TER scores represent as much as a 2.5 point improvement over the baseline. Our discriminative learning method with linguistic features (LPDS), which automatically classifies whether an input segment should be marked up, leads to an increase of 1.01 absolute BLEU points (2.53% relative) over the BASELINE, which is statistically significant. We also observe a 0.5 points (1.10% relative) drop in TER

compared to the BASELINE. This shows that our classifier with linguistic features is capable of judging whether the sub-segment level-constrained translation is helpful for the overall translation quality or not.

7.5.3.3 Translation Results with Confidence Thresholding

To further analyze our discriminative learning approach, we also investigate the use of classification confidence (cf. Section 7.3.2.2) as a threshold to boost classification precision. Table 7.6 shows the classification and translation results when we use different confidence

Table 7.6: The impact of classification confidence thresholding

	BASELINE	0.50	0.55	0.60	0.65	0.70	0.75
BLEU	45.80	46.81*	**47.00***	46.79*	46.47	46.11	46.03
TER	39.82	39.32	**39.10***	39.28	39.45	39.66	39.70
P	N/A	61.36	67.96	71.01	**75.00**	70.97	71.43
R	N/A	**41.33**	35.71	25.00	18.37	11.22	7.65

thresholds, where the scores marked with "*" are significantly better ($p = 0.01$) than the BASELINE. The default classification confidence is 0.50.

We investigate the impact of increasing classification confidence on the performance of the classifier and the translation results using LPDS features. As can be seen from Table 7.6, increasing the classification confidence up to 0.65 leads to a steady increase in classification precision with a corresponding sacrifice in recall. The fluctuation in classification performance has an impact on the translation results as measured by BLEU and TER. We can see that the best BLEU as well as TER scores are achieved when we set the classification confidence to 0.55, representing a further 0.19 points improvement in BLEU score and 0.22 points drop in TER score, compared to the default threshold of 0.50.

Compared to the BASELINE, we obtain a 1.20 (2.62 % relative) BLEU point improvement and 0.72 (1.81 % relative) TER point improvement (with lower TER score), all with statistical significance ($p = 0.01$), when we set the confidence to 0.55. Despite the higher precision when the confidence is set above 0.60, the dramatic decrease in recall cannot be compensated for by the increase in precision.

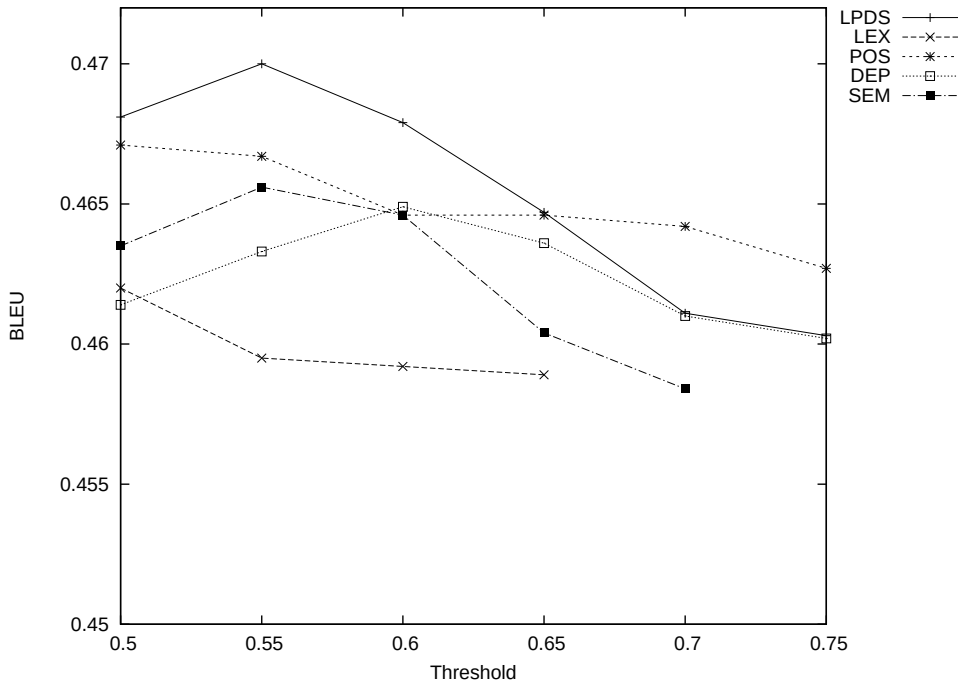

Figure 7.6: Confidence Threshold on Various Feature Sets

We also compare the effect of applying confidence thresholds to all linguistically-motivated feature sets we have proposed in Figure 7.6. Note that the LPDS features obtain the best BLEU scores in the [0.5, 0.65] range and obtain the highest BLEU score at the confidence level of 0.55, which confirms our approach of combining a variety of linguistic features for this task. We also observe that although the BLEU score of POS features is also competitive at the confidence of 0.5, the translation quality will not improve as we set a higher threshold, because its recall is already low initially.

7.5.3.4 Comparison with Previous Work

In previous work (cf. Section 7.6), both Koehn and Senellart [2010b] and Zhechev and van Genabith [2010] used fuzzy match score to determine whether the input segments should be marked up. The input segments are only marked up when the fuzzy match score is above a certain threshold. We present the results using this method in Table 7.7. From this table, we can see an inferior performance compared to the BASELINE results (cf. Table 7.5) when

Table 7.7: Performance using fuzzy match score for classification

	Fuzzy Match Scores				
	0.50	0.60	0.70	0.80	0.90
BLEU	45.13	45.55	45.58	45.84	45.82
TER	40.99	40.62	40.56	40.29	40.07

the fuzzy match score is below 0.70. A modest gain can only be achieved when the fuzzy match score is above 0.8. This is slightly different from the conclusions drawn in [Koehn and Senellart, 2010b], where gains are observed when the fuzzy match score is above 0.7, and in [Zhechev and van Genabith, 2010] where gains are only observed when the score is above 0.9. Comparing Table 7.7 with Table 7.6, we can see that our classification method is more effective. This confirms our argument in the last paragraph of Section 7.6, namely that fuzzy match score is not informative enough to determine the usefulness of the sub-segments in a fuzzy match, and that a more comprehensive set of features, as we have explored in this paper, is essential for the discriminative learning-based method to work.

Table 7.8: Percentage of training segments with markup vs without markup grouped by fuzzy match (FM) score ranges

FM Scores	w. markup	w/o markup
[0,0.5]	37.75	62.24
(0.5,0.6]	40.64	59.36
(0.6,0.7]	40.94	59.06
(0.7,0.8]	46.67	53.33
(0.8,0.9]	54.28	45.72
(0.9,1.0]	44.14	55.86

To further validate our assumption, we analyze the training segments by grouping them according to their fuzzy match score ranges. For each group of segments, we calculate the percentage of segments where markup (and respectively without markup) can produce better translations. The statistics are shown in Table 7.8. We can see that for segments with fuzzy match scores lower than 0.8, more segments can be better translated without markup. For segments where fuzzy match scores are within the range $(0.8, 0.9]$, more segments can be better translated with markup. However, within the range $(0.9, 1.0]$, surprisingly, actually more segments receive better translation without markup. This indicates that fuzzy match score is not a good measure to predict whether fuzzy matches are beneficial when used to

Table 7.9: Translation Examples

Example 1	
w/o markup	after policy name , type the name of the policy (it shows new host integrity policy by default) .
Translation	在 “ 策略 ” 名称 后面 ， 键入 策略 的 名称 (名称 显示 为 “ 新 主机 完整性 策略 默认 ） 。
w. markup	after policy name <tm translation=“， 键入 策略 名称 （ 默认 显示 “ 新 主机 完整性 策略 ” ） 。”>, type the name of the policy (it shows new host integrity policy by default) .< /tm>
Translation	在 “ 策略 ” 名称 后面 ， 键入 策略 名称 （ 默认 显示 “ 新 主机 完整性 策略 ” ） 。
Reference	在 “ 策略 名称 ” 后面 ， 键入 策略 名称 （ 默认 显示 “ 新 主机 完整性 策略 ” ） 。
Example 2	
w/o markup	changes apply only to the specific scan that you select .
Translation	更改 仅 适用于 特定 扫描 的 规则 。
w. markup	changes apply only to the specific scan that you select <tm translation=“。”>.< /tm>
Translation	更改 仅 适用于 您 选择 的 特定 扫描 。
Reference	更改 只 应用于 您 选择 的 特定 扫描 。

constrain the translation of an input segment.

7.5.4 Improved Translations

In order to pinpoint the sources of improvements by marking up the input segment, we performed some manual analysis of the output. We observe that the improvements can broadly be attributed to two reasons: 1) the use of long phrase pairs which are missing in the phrase table, and 2) deterministically using highly reliable phrase pairs.

Phrase-based SMT systems normally impose a limit on the length of phrase pairs for storage and speed considerations. Our method can overcome this limitation by retrieving and reusing long phrase pairs on-the-fly. A similar idea, albeit from a different perspective, was explored by Lopez [2008], where he proposed to construct a phrase table on the fly for each segment to be translated. Differently from his approach, our method directly translates part of the input segment using fuzzy matches retrieved on-the-fly, with the rest of the segment translated by the pre-trained MT system. We offer some more insights into the advantages of our method by means of a few examples.

Example 1 shows translation improvements by using long phrase pairs. Compared to the reference translation, we can see that for the underlined phrase, the translation without markup contains (i) word ordering errors and (ii) a missing right quotation mark. In Ex-

ample 2, by specifying the translation of the final punctuation mark, the system correctly translates the relative clause 'that you select'. The translation of this relative clause is missing when translating the input without markup. This improvement can be partly attributed to the reduction in search errors by specifying the highly reliable translations for phrases in an input segment.

7.6 Related Work

The work in this chapter lies at the intersection of two strands of research. Firstly, it brings our quality-estimation-based TM-MT integration research from the segment level to the sub-segment level. In this chapter, we rely on the SVM classification and confidence estimation schemes in translation recommendation [He et al., 2010c] to predict the the usability of constrained translation chunks.

Secondly, this work also improves upon previous efforts that use TM chunks to improve SMT performance. There are several different ways of using the translation information derived from fuzzy matches, with the following two being the most widely adopted: 1) to add these translations into a phrase table as in [Biçici and Dymetman, 2008, Simard and Isabelle, 2009], or 2) to mark up the input segment using the relevant chunk translations in the fuzzy match, and to use an MT system to translate the parts that are not marked up, as in [Smith and Clark, 2009, Koehn and Senellart, 2010b, Zhechev and van Genabith, 2010]. It is worth mentioning that translation consistency was not explicitly regarded as their primary motivation in this previous work. Our research follows the direction of the second strand given that consistency can no longer be guaranteed by constructing another phrase table.

However, to categorically reuse the translations of matched chunks without any differentiation might generate inferior translations given the fact that the context of these matched chunks in the input segment could be completely different from the source side of the fuzzy match. To address this problem, both Koehn and Senellart [2010b] and Zhechev and van Genabith [2010] used fuzzy match score as a threshold to determine whether to reuse the

translations of the matched chunks. For example, Koehn and Senellart [2010b] showed that reusing these translations as large rules in a hierarchical system [Chiang, 2005] can be beneficial when the fuzzy match score is above 0.7, while Zhechev and van Genabith [2010] reported that it is only beneficial to a phrase-based system when the fuzzy match score is above 0.9.

7.7 Summary

In this chapter, we introduced a discriminative learning method to tightly integrate fuzzy matches retrieved using translation memory technologies with phrase-based SMT systems to improve translation consistency. We used an SVM classifier to predict whether phrase pairs derived from fuzzy matches could be used to constrain the translation of an input segment. A number of feature functions including a series of novel dependency features were used to train the classifier. Experiments demonstrated that discriminative learning and linguistically-motivated features are effective in improving translation quality and are more informative than the fuzzy match score and translation model-based features used in previous research. We report a 1.2 absolute improvement in BLEU score and a 0.72 absolute improvement in TER score, both of statistical significance ($p < 0.01$) when using our approach.

As mentioned in Section 7.6, the potential improvement in segment-level translation consistency using our method can be attributed to the fact that the translation of new input segments is closely informed and guided (or constrained) by previously translated segments using global features such as dependencies. However, it is worth noting that the level of improvment in translation consistency is also dependent on the nature of the TM itself; a self-contained and coherent TM would facilitate consistent translations.

There are many possibilities we can explore along this line of research. We plan to investigate the impact of TM quality on translation consistency when using our approach. Furthermore, we will explore methods to promote translation consistency at document level.

Moreover, we also plan to experiment with phrase-by-phrase classification instead of segment-by-segment classification presented in this paper, in order to obtain more stable classification results. We can also label the training examples using other segment-level evaluation metrics such as Meteor [Banerjee and Lavie, 2005, Denkowski and Lavie, 2010].

Currently, only a standard phrase-based SMT system is used, so we plan to test our method on a hierarchical system [Chiang, 2005] to facilitate direct comparison with [Koehn and Senellart, 2010b]. We will also carry out experiments on other data sets and for more language pairs.

Chapter 8

Conclusion

In this thesis, we explored a series of approaches to integrate MT outputs into TM environments. Using these methods, TM environments are enriched with high quality MT outputs, but the assets and cost estimations associated with TMs are kept intact. Our approaches work both for 1-best and k-best translation candidates, at both segment and sub-segment levels. Most importantly, our approaches are validated by human translators, the target users of our approaches.

We start this thesis in Chapter 2 by reviewing TMs and MTs, the two paradigms that we try to integrate in this thesis. We observe both TM's strengths of precisely reusing previously translated segments and performing reliable translation cost estimation, and MT's capability to produce automatic high quality end-to-end translation. Based on this observation, we propose to integrate high quality MT outputs into the TM environment, so that translators can still work in TMs, but at the same time can benefit from recent advancements in SMT.

In Chapter 3, we review existing methods of translation quality estimation, including the fuzzy match score for TMs, and confidence estimation and automatic evaluation metrics for MT systems. As our TM-MT integration approaches are based on quality comparison, these existing methods are closely related to the work reported in this thesis, and inspired the approaches presented in this thesis, especially the design of linguistically motivated

features.

We begin presenting our TM-MT integration approaches in Chapter 4 by introducing the translation recommendation model. In the translation recommendation model, we only present MT outputs that we predict (with high confidence) to be more suitable for post-editing to translators. At the same time, we also provide a recommendation confidence score, on which the translators can set thresholds by themselves. As only the better MT segments are presented in the TM environment, the assets associated with the TM are kept intact, and the related cost estimation can still be used as an upper bound.

In Chapter 5, we extend our work on translation recommendation with the translation reranking model. While the translation recommendation model focuses only on the 1-best outputs, the reranking model is capable of handling k-best outputs by merging and reranking the TM and MT k-best lists. Using the reranking model, every segment found by the fuzzy match scheme is kept in the environment, but translators have easier access to better quality translations as these are reranked higher in the new k-best lists.

We report the results we collected from a user study to demonstrate that our method is validated by human translators in Chapter 6. We show that our recommendation model can obtain a precision above 0.9 and a recall above 0.75 with proper thresholds, and that our reranking model can obtain 0.85 precision and 0.58 recall when evaluated against the consensus judgement of 3 translators. We also report an interesting user feedback that lends further support to TM-MT integration and acts as implicit endorsement of our integration models.

Finally, in Chapter 7, we develop our TM-MT integration paradigm to the sub-segment level. Instead of comparing the quality of TM and MT output segments and presenting the better one to translators, we explore the possibility of deeper integration by reusing high quality TM sub-segment chunks to enrich SMT systems. Experiments on a real world dataset shows that our method not only better guarantees translation consistency, but also leads to improved translations, reflected by a 1.2 BLEU point improvement (2.62% relative) and a 0.72 TER point reduction (1.81% relative).

Now let us look at the research questions we proposed in Chapter 1.

(**RQ1**) Can we provide translators with high-quality MT segments in a TM environment, without sacrificing the strengths of TMs?

(**RQ2**) Can we reuse sub-segment chunks from TMs to improve SMT consistency and quality?

(**RQ3**) Can we validate our TM-MT integration models with human evaluation?

We tackle **RQ1** with the methods we present in Chapters 4 and 5. The translation recommendation and translation reranking models enable the translators to access SMT outputs in an TM environment, only when the SMT outputs are predicted to be more suitable for post-editing with high confidence. This way, we kept TM's strength of a more user friendly post-editing environment and only when the TM cannot produce a competitive candidate for post-editing, we take advantage of SMT's high coverage and lead the translator to the SMT output.

We bring these integration paradigms to sub-segment level in response to **RQ2** in Chapter 7. We use high confidence TM chunks to mark up and constrain SMT. We also incorporate a rich linguistic feature set inspired by our work on automatic evaluation metrics in Chapter 3 to improve the expressiveness of this model. Our experiments show that both consistency and quality of SMT outputs improve by reusing sub-segment chunks from TM.

In Chapter 6, we perform human evaluation on our recommendation and reranking models. Results from our experiments show that human evaluation support validation of both models, providing a positive answer to **RQ3**.

8.1 Contribution of this Thesis

In sum, we have explored both loose segment-level integration and tight sub-segment-level integration of TM and MT systems, so as to help translators to access the SMT outputs in a TM environment. We have made the following contributions.

- **Segment-level TM-MT Integration Models**. We present two segment-level TM-MT integration models that allow translators to access to high-quality MT, while keeping strengths of the TM environment. The effectiveness of these two models is validated by judgements from human translators.

- **A Sub-Segment level TM-MT Integration Model**. We also present a model to perform sub-segment level integration for TM and MT, so that even if the overall quality of a TM fuzzy match is not good enough, it is still possible to use high-quality sub-segments from it to enrich the SMT engine to produce a more consistent translation of higher quality.

- **Human Evaluation Paradigm for TM-MT Integration**. When evaluating our models against human judgements, we present a paradigm to evaluate TM-MT integration quality against both individual and consensus judgements, and enable comparison with naive fuzzy match thresholding-based methods currently used in the industry. This paradigm can be reused in future research on the topic of TM-MT integration.

8.2 Future Work

The integration of TM and MT paradigms is a field undergoing active research, as is indicated in the related research we discussed in Chapters 4, 5, 6, and 7. The research described in this thesis can be strengthened both by more thorough investigation of the method itself, and by the interaction with other MT-TM integration techniques.

The method presented in this thesis is tested on a proprietary TM in the IT security domain, consisting mainly of short segments. The utility of this approach can be better evaluated by testing on TMs from broader domains and of different characteristics. We also note that while using proprietary TMs enables us to test our models in an industrial setting, it does not always facilitate crowdsourcing as a cheaper approach to perform more extended human evaluations, so testing the method in an open domain could help us to obtain more and better data. Eventually, we hope to tune the system on the human evaluation data in

order to provide better recommendations.

On the other hand, from the perspective of localization vendors, our human evaluation can still be strengthened by statistics collected from a real industrial setting instead of questionnaires. So it would also be interesting to see how this paradigm can improve the efficiency of translators in an industrial localization process.

With regard to the interaction with other methods, it will be very useful to integrate MT confidence estimation scores such as Specia et al. [2009b] into our translation recommendation and translation reranking models, so that the translators can still have a translation confidence score (in addition to a recommendation confidence score), when MT outputs are presented. Moreover, our segment- and sub-segment-integration models can be integrated, so that when the TM output is inferior to the MT output, it can be used to generate an alternative translation, and then the recommender/reranker can predict its quality compared to other "pure" MT outputs.

Finally, our sub-segment integration model is a first step in this direction. Like its segment-level counterparts, this method can be understood much better if human evaluation can be conducted. This method also opens several other possibilities. Firstly, the current model performs classification on a segment-by-segment basis, and we suspect the performance can be further improved if we classify on a markup-to-markup basis. Secondly, as we actually reuse part of the TM fuzzy match, and have the information on alignment and confidence estimation, we can potentially use such information to produce confidence scores for the final translation output, as well as providing a better color-coding scheme to assist translators.

Bibliography

Enrique Amigó, Jesús Giménez, Julio Gonzalo, and Felisa Verdejo. The contribution of linguistic features to automatic machine translation evaluation. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP (ACL-IJCNLP-2009)*, pages 306–314, Suntec, Singapore, 2009.

Satanjeev Banerjee and Alon Lavie. METEOR: An automatic metric for MT evaluation with improved correlation with human judgments. In *Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*, pages 65–72, Ann Arbor, MI, 2005.

Leonard E. Baum. An inequality and associated maximization technique in statistical estimation for probabilistic functions of Markov processes. *Inequalities*, 3:1–8, 1972.

Ergun Biçici and Marc Dymetman. Dynamic translation memory: Using statistical machine translation to improve translation memory. In *Proceedings of the 9th Internation Conference on Intelligent Text Processing and Computational Linguistics (CICLing)*, pages 454–465, Haifa, Israel, 2008.

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 the 20th International Conference on Computational Linguistics (COLING-2004)*, pages 315 – 321, Geneva, Switzerland, 2004.

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

Aoife Cahill, Michael Burke, Ruth O'Donovan, Josef van Genabith, and Andy Way. Long-distance dependency resolution in automatically acquired wide-coverage PCFG-based LFG approximations. In *Proceedings of the 42nd Meeting of the Association for Computational Linguistics (ACL-2004)*, pages 319–326, Barcelona, Spain, 2004.

Chris Callison-Burch. Fast, cheap, and creative: Evaluating translation quality using Amazon's Mechanical Turk. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing (EMNLP-2009)*, pages 286–295, Suntec, Singapore, 2009.

Chris Callison-Burch, Miles Osborne, and Philipp Koehn. Re-evaluation the role of Bleu in machine translation research. In *Proceedings of 11th Conference of the European Chapter of the Association for Computational Linguistics (EACL-2006)*, pages 249–256, Trento, Italy, 2006.

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 (WMT-2007)*, pages 136–158, Prague, Czech Republic, 2007.

Chris Callison-Burch, Cameron Fordyce, Philipp Koehn, Christof Monz, and Josh Schroeder. Further meta-evaluation of machine translation. In *Proceedings of the Third Workshop on Statistical Machine Translation (WMT-2008)*, pages 70–106, Columbus, OH, USA, 2008.

Daniel Cer, Christopher D. Manning, and Daniel Jurafsky. The best lexical metric for phrase-based statistical mt system optimization. In *Proceedings of Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Asso-*

ciation for Computational Linguistics (NAACL-HLT-2010), pages 555–563, Los Angeles, CA, 2010.

Yee Seng Chan and Hwee Tou Ng. MAXSIM: A maximum similarity metric for machine translation evaluation. In *Proceedings of the 46th Meeting of the Association for Computational Linguistics (ACL-2008)*, pages 55–62, Columbus, Ohio, 2008.

Chih-Chung Chang and Chih-Jen Lin. *LIBSVM: a library for support vector machines*, 2001. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

David Chiang. A hierarchical Phrase-Based model for Statistical Machine Translation. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 263–270, Ann Arbor, MI, 2005.

David Chiang, Steve DeNeefe, Yee Seng Chan, and Hwee Tou Ng. Decomposability of translation metrics for improved evaluation and efficient algorithms. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing (EMNLP-2008)*, pages 610–619, Honolulu, Hawaii, 2008.

David Chiang, Kevin Knight, and Wei Wang. 11,001 new features for statistical machine translation. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL-HLT-2009)*, pages 218–226, Boulder, CO, 2009.

Jacob Cohen. A coefficient of agreement for nominal scales. 20(1):37–46, 1960.

Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20(3): 273–297, 1995.

Koby Crammer, Ofer Dekel, Joseph Keshet, Shai Shalev-Shwartz, and Yoram Singer. Online passive-aggressive algorithms. *Journal of Machine Learning Research*, 7:551–585, 2006.

Michael Denkowski and Alon Lavie. Extending the METEOR Machine Translation Evaluation Metric to the Phrase Level. In *Proceedings of Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL-HLT-2010)*, pages 250–253, Los Angeles, CA, 2010.

Nan Duan, Hong Sun, and Ming Zhou. Translation model generalization using probability averaging for machine translation. In *Proceedings of the 23rd International Conference on Computational Linguistics (COLING-2010)*, pages 304–312, Beijing, China, 2010.

Loïc Dugast, Jean Senellart, and Philipp Koehn. Statistical post-editing on SYSTRAN's rule-based translation system. In *Proceedings of the Second Workshop on Statistical Machine Translation (WMT-2007)*, pages 220–223, Prague, Czech Republic, 2007.

Christiane Fellbaum, editor. *WordNet: An Electronic Lexical Database*. MIT Press, 1998.

Joseph L. Fleiss. *Statistical methods for rates and proportions*. John Wiley & Sons, 1981.

Raymond Flournoy and Christine Duran. Machine translation and document localization at adobe: from pilot to production. In *Proceedings of the twelfth Machine Translation Summit (MT-Summit XII)*, pages 425–428, Ottawa, Ontario, Canada, 2009.

George Foster, Roland Kuhn, and Howard Johnson. Phrasetable smoothing for statistical machine translation. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing (EMNLP-2006)*, pages 53–61, Sydney, Australia, 2006.

Daniel Gildea and Daniel Jurafsky. Automatic labeling of semantic roles. *Computational Linguistics*, 28(3):245–288, 2002.

Jesús Giménez and Lluís Màrquez. A smorgasbord of features for automatic MT evaluation. In *Proceedings of the Third Workshop on Statistical Machine Translation (WMT-2008)*, pages 195–198, Columbus, OH, 2008.

Jesús Giménez and Lluís Màrquez. Linguistic measures for automatic machine translation evaluation. *Machine Translation*, 24:209–240, 2010.

Ana Guerberof. Productivity and quality in mt post-editing. In *MT Summit XII - Workshop: Beyond Translation Memories: New Tools for Translators MT*, Ottawa, Ontario, Canada, 2009.

Yifan He and Andy Way. Improving the objective function in minimum error rate training. In *Proceedings of the Twelfth Machine Translation Summit (MT-Summit XII)*, pages 238 – 245, Ottawa, Ontario, Canada, 2009.

Yifan He, Jinhua Du, Andy Way, and Josef van Genabith. The DCU dependency-based metric in WMT-MetricsMATR 2010. In *ACL 2010 Joint Fifth Workshop on Statistical Machine Translation and Metrics MATR (WMT-MetricsMATR-2010)*, Uppsala, Sweden, July 2010a.

Yifan He, Yanjun Ma, Johann Roturier, Andy Way, and Josef van Genabith. Improving the post-editing experience using translation recommendation: a user study. In *Proceedings of the 9th Conference of the Association for Machine Translation in the Americas (AMTA-2010)*, page 10pp, Denver, CO, 2010b.

Yifan He, Yanjun Ma, Josef van Genabith, and Andy Way. Bridging SMT and TM with translation recommendation. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics (ACL-2010)*, pages 622–630, Uppsala, Sweden, 2010c.

Yifan He, Yanjun Ma, Andy Way, and Josef van Genabith. Integrating N-best SMT outputs into a TM system. In *Proceedings of the 23rd International Conference on Computational Linguistics (COLING-2010:Posters)*, pages 374–382, Beijing, China, 2010d.

Myles Hollander and Douglas A. Wolfe. *Nonparametric Statistical Methods*. John Wiley & Sons, 1999.

Thorsten Joachims. Optimizing search engines using clickthrough data. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining (KDD-02)*, pages 133–142, 2002.

Jeremy G. Kahn, Matthew Snover, and Mari Ostendorf. Expected dependency pair match: predicting translation quality with expected syntactic structure. *Machine Translation*, 2010.

Martin Kay. The proper place of men and machines in language translation. *Xerox Palo Alto Research Center*, pages 1–21, 1980.

Reinhard Kneser and Hermann Ney. Improved backing-off for m-gram language modeling. In *The 1995 International Conference on Acoustics, Speech, and Signal Processing (ICASSP-95)*, pages 181 – 184, Detroit, MI, 1995.

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

Philipp Koehn. *Statistical Machine Translation*. Cambridge University Press, 2010.

Philipp Koehn. Europarl: A Parallel Corpus for Statistical Machine Translation. In *Proceedings of MT Summit X*, pages 79–86, Phuket, Thailand, 2005.

Philipp Koehn and Barry Haddow. Interactive assistance to human translators using statistical machine translation methods. In *The Twelfth Machine Translation Summit (MT-Summit XII)*, pages 73 – 80, Ottawa, Ontario, Canada, 2009.

Philipp Koehn and Jean Senellart. Fast approximate string matching with Suffix arrays and A* parsing. In *Proceedings of the 2010 Annual Conference of Association for Machine Translation in the Americas (AMTA-2010)*, Denver, CO, 2010a.

Philipp Koehn and Jean Senellart. Convergence of translation memory and statistical machine translation. In *AMTA Workshop on MT Research and the Translation Industry*, Denver, CO, November 2010b.

Philipp. Koehn, Franz Josef Och, and Daniel Marcu. Statistical phrase-based translation. In *Proceedings of the 2003 Annual Conference of the North American Chapter of the As-*

sociation for Computational Linguistics on Human Language Technology (NAACL/HLT-2003), pages 48 – 54, Edmonton, Alberta, Canada, 2003.

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 Companion Volume Proceedings of the Demo and Poster Sessions (ACL-2007)*, pages 177–180, Prague, Czech Republic, 2007.

Vladimir Iosifovich Levenshtein. Binary codes capable of correcting deletions, insertions, and reversals. *Soviet Physics Doklady*, 10(8):707–710, 1966.

Junhui Li, Guodong Zhou, Hai Zhao, Qiaoming Zhu, and Peide Qian. Improving nominal SRL in Chinese language with verbal SRL information and automatic predicate recognition. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing (EMNLP-2009)*, pages 1280–1288, Suntec, Singapore, 2009.

Chin-Yew Lin and Franz Josef Och. Automatic evaluation of machine translation quality using longest common subsequence and skip-bigram statistics. In *Proceedings of the 42nd Meeting of the Association for Computational Linguistics (ACL-2004)*, pages 605–612, Barcelona, Spain, 2004.

Hsuan-Tien Lin, Chih-Jen Lin, and Ruby C. Weng. A note on Platt's probabilistic outputs for support vector machines. *Machine Learning*, 68(3):267–276, 2007.

Ding Liu and Daniel Gildea. Syntactic features for evaluation of machine translation. In *Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*, pages 25–32, Ann Arbor, MI, 2005.

Adam Lopez. Tera-scale translation models via pattern matching. In *Proceedings of the 22nd International Conference on Computational Linguistics (Coling 2008)*, pages 505–512, Manchester, UK, August 2008.

Yanjun Ma, Yifan He, Andy Way, and Josef van Genabith. Consistent translation using discriminative learning: A translation memory-inspired approach. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics (ACL-2011)*, page forthcoming, Portlan, OR, 2011.

Daniel Marcu and William Wong. A Phrase-Based, joint probability model for Statistical Machine Translation. In *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing (EMNLP-2002)*, pages 133–139, Morristown, NJ, 2002.

Eric W. Noreen. *Computer-Intensive Methods for Testing Hypotheses: An Introduction*. Wiley-Interscience, New York, NY, 1989.

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

Franz Josef Och. Minimum error rate training in statistical machine translation. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics (ACL-2003)*, pages 160–167, Morristown, NJ, 2003.

Franz Josef Och and Hermann Ney. Discriminative training and maximum entropy models for statistical machine translation. In *Proceedings of 40th Annual Meeting of the Association for Computational Linguistics (ACL-2002)*, pages 295–302, Philadelphia, PA, 2002.

Franz Josef Och, Ignacio Thayer, Daniel Marcu, Kevin Knight, Dragos Stefan Munteanu, Quamrul Tipu, Michel Galley, and Mark Hopkins. Arabic and chinese mt at usc/isi. In *Presentation given at NIST Machine Translation Evaluation Workshop*, 2004.

Karolina Owczarzak, Josef van Genabith, and Andy Way. Labelled dependencies in machine translation evaluation. In *Proceedings of the Second Workshop on Statistical Machine Translation (WMT-2007)*, pages 104–111, Prague, Czech Republic, 2007.

Sebastian Pado, Michel Galley, Dan Jurafsky, and Christopher D. Manning. Robust machine translation evaluation with entailment features. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP (ACL-IJCNLP-2009)*, pages 297–305, Suntec, Singapore, 2009.

Martha Palmer, Daniel Gildea, and Paul Kingsbury. The proposition bank: An annotated corpus of semantic roles. *Computational Linguistics*, 31(1):71–106, 2005.

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

John C. Platt. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. *Advances in Large Margin Classifiers*, pages 61–74, 1999.

Maja Popović and Hermann Ney. Syntax-oriented evaluation measures for machine translation output. In *Proceedings of the Fourth Workshop on Statistical Machine Translation (WMT-2009)*, pages 29–32, Athens, Greece, 2009.

Christopher B. Quirk. Training a sentence-level machine translation confidence measure. In *Proceedings of Fourth International Conference on Language Resources and Evaluation (LREC-2004)*, pages 825 – 828, Lisbon, Portugal, 2004.

Rajat Raina, Aria Haghighi, Christopher Cox, Jenny Finkel, Jeff Michels, Kristina Toutanova, Bill MacCartney, Marie-Catherine de Marneffe, Christopher D. Manning, and Andrew Y. Ng. Robust Textual Inference using Diverse Knowledge Sources. In *Proceedings of the First PASCAL Challenge Workshop on Recognizing Textual Entailment*, pages 57 – 60, Southampton, UK, 2005.

Libin Shen, Anoop Sarkar, and Franz Josef Och. Discriminative reranking for machine translation. In *Proceedings of the 2004 Annual Conference of the North American Chap-*

ter of the Association for Computational Linguistics on Human Language Technology (NAACL/HLT-2004), pages 177–184, Boston, MA, 2004.

Richard Sikes. Fuzzy matching in theory and practice. *Multilingual*, 18(6):39 – 43, 2007.

Michel Simard and Pierre Isabelle. Phrase-based machine translation in a computer-assisted translation environment. In *Proceedings of the Twelfth Machine Translation Summit (MT Summit XII)*, pages 120 – 127, Ottawa, Ontario, Canada, 2009.

James Smith and Stephen Clark. EBMT for SMT: A new EBMT-SMT hybrid. In *Proceedings of the 3rd International Workshop on Example-Based Machine Translation*, pages 3–10, Dublin, Ireland, 2009.

Matthew Snover, Bonnie Dorr, Richard Schwartz, Linnea Micciulla, and John Makhoul. A study of translation edit rate with targeted human annotation. In *Proceedings of the 2006 Annual Conference of Association for Machine Translation in the Americas (AMTA-2006)*, pages 223–231, Cambridge, MA, USA, 2006.

Matthew Snover, Nitin Madnani, Bonnie Dorr, and Richard Schwartz. TER-Plus: paraphrase, semantic, and alignment enhancements to translation edit rate. *Machine Translation*, 23:117–127, 2009.

Morry Sofer. *The Translator's Handbook*. Schreiber Publishing, Rockville, MD, 2006.

Harold Somers. Round-trip translation: what is it good for? In *Proceedings of the 2005 Australasian Language Technology Workshop*, Sydney, Australia, 2005.

Harold Somers. Translation memory systems. In Harold Somers, editor, *Computers and Translation: A Translator's Guide*, pages 31–47. John Benjamin's Publishing Co., 2003.

Lucia Specia, Nicola Cancedda, Marc Dymetman, Marco Turchi, and Nello Cristianini. Estimating the sentence-level quality of machine translation systems. In *Proceedings of the 13th Annual Conference of the European Association for Machine Translation (EAMT-2009)*, pages 28 – 35, Barcelona, Spain, 2009a.

Lucia Specia, Craig Saunders, Marco Turchi, Zhuoran Wang, and John Shawe-Taylor. Improving the confidence of machine translation quality estimates. In *Proceedings of the Twelfth Machine Translation Summit (MT Summit XII)*, pages 136 – 143, Ottawa, Ontario, Canada, 2009b.

Andreas Stolcke. SRILM-an extensible language modeling toolkit. In *Proceedings of the Seventh International Conference on Spoken Language Processing*, volume 2, pages 901–904, Denver, CO, USA, 2002.

Midori Tatsumi. Correlation between automatic evaluation metric scores, post-editing speed, and some other factors. In *The Twelfth Machine Translation Summit (MT-Summit XII)*, pages 332 – 339, Ottawa, Ontario, Canada, 2009.

Joseph P. Turian, Luke Shen, and I.Dan Melamed. Evaluation of machine translation and its evaluation. In *Proceedings of the Ninth Machine Translation Summit (MT Summit IX)*, New Orleans, LA, USA, 2003.

Nicola Ueffing and Hermann Ney. Application of word-level confidence measures in interactive statistical machine translation. In *Proceedings of the 9th Annual Conference of the European Association for Machine Translation (EAMT-2005)*, pages 262–270, Budapest, Hungary, 2005.

Nicola Ueffing, Klaus Macherey, and Hermann Ney. Confidence measures for statistical machine translation. In *Proceedings of the Ninth Machine Translation Summit (MT Summit IX)*, pages 394 – 401, New Orleans, LA, 2003.

Vladimir Vovk, Alex Gammerman, and Glenn Shafer. *Algorithmic learning in a random world*. Springer, 2005. ISBN 0387001522.

Taro Watanabe, Jun Suzuki, Hajime Tsukada, and Hideki Isozaki. Online large-margin training for statistical machine translation. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL-2007)*, pages 764–773, Prague, Czech Republic, 2007.

Hua Wu and Haifeng Wang. Pivot language approach for phrase-based statistical machine translation. *Machine Translation*, 21:165–181, 2007.

Yang Ye, Ming Zhou, and Chin-Yew Lin. Sentence level machine translation evaluation as a ranking. In *Proceedings of the Second Workshop on Statistical Machine Translation (WMT-2007)*, pages 240–247, Prague, Czech Republic, 2007.

Ventsislav Zhechev and Josef van Genabith. Seeding statistical machine translation with translation memory output through tree-based structural alignment. In *Proceedings of the Fourth Workshop on Syntax and Structure in Statistical Translation*, pages 43–51, Beijing, China, 2010.

Appendix

Guidelines Provided to Professional Post-Editors during Human Evaluation

Please read this step by step instruction fully and carefully before you conduct the task.

- To log in the evaluation interface, click the following link: `http://eval.yifanhe.org/login/` The server will be up and running from 6:00am 17 May to midnight 19 May, 2010. You should have received your user name and password to log into the server.

- On the login page, input your username and password you obtained and click the "Submit" button, you will be logged into the evaluation page.

- In total, there are 300 English segments translated into French using two different systems. There is only one English segment together with its two French translations shown on each webpage. The two French translations have been shuffled randomly; therefore translation 1 can either be output from translation system 1 or 2 and similarly for translation 2. You will see a snapshot of the interface on the third page.

  You are asked to choose the sentence that is most SUITABLE FOR POST-EDITING. By "suitable for post-editing", you are NOT asked to choose the best French translation Rather, you are asked to choose the French translation that would save you the most time if you were to post-edit it. Therefore, even if a French translation does not

fully translate the English segment, you may still select it because you would spend less time post-editing it into a grammatical French segment, whose meaning would match the English segment's. Please make sure to bear this in mind throughout the task.

Let's take the following example:

> Source: Determines whether a recovery point is valid or corrupt before restoring it.
>
> Candidate 1: Vérifie si un point de récupération est valide ou endommagé avant la restauration.
>
> Candidate 2: détermine si un point de récupération est valide ou endommagée avant la restauration.

Candidate 1 is a grammatical segment but it does not convey the meaning of the source segment ("Determines" is semantically different from "Vérifie"), so an important lexical change would be required. On the other hand candidate 2 is not a grammatical sentence (Lack of initial capitalisation and wrong agreement "endommagée"), so two small changes would be required.

While Candidate 2 is a better translation than Candidate 1 from a semantic perspective, you might consider that it would be quicker to post-edit Candidate 1

There is an option of "Equally suitable for post-editing", please only select this when you are genuinely sure that they are absolutely equally suitable.

There is also an option of "Neither is suitable for post-editing, I will translate from scratch". Please only use this option when you think both candidate translations are not useful. For example:

> Source: IDD _ ADD _ SHARE _ PAGE _ COMPUTER
>
> Candidate 1: IDD _ ADD _ SHARE _ PAGE _ INTRO
>
> Candidate 2: IDD _ ADD _ SHARE _ PAGE _ ordinateur

In this case, you will directly copy the source segment; therefore neither candidate translation is suitable.

- Please complete the selection for all 300 English segments. If necessary, you may take an extra 20 minutes (paid) in order to complete all of them. You will then come to a page showing the following message "Evaluation completed! Thank you!", which is followed by a very short questionnaire. After you finish this, Please click the "Logout" button to log out.

- You may log out in the middle of this task by clicking the "Logout" button in the upper half of your page. Your work will be saved. When you log in next time, it will start from a page you haven't completed last time.

- Whenever you have questions during this task, please send an email to Dr. Yanjun Ma (`yma@computing.dcu.ie`), your query will be replied as soon as we possibly can.

All your appreciated effort in this task will greatly help us to improve our existing technology. Many thanks for your cooperation!