UNIVERSITÀ DEGLI STUDI
DI TRENTO

DEPARTMENT OF INFORMATION ENGINEERING AND COMPUTER SCIENCE
ICT International Doctoral School

Adaptation Methods for Statistical Machine Translation
In Business Scenarios

Prashant Mathur

Advisor

Marcello Federico

FBK Trento, Italy

Co-Advisor

Mauro Cettolo

FBK Trento, Italy

March 2017

Abstract

Adaptation methods for phrase-based statistical Machine Translation (MT) have been explored in the literature under different paradigms, such as domain adaptation and topic adaptation, and most of the times in rather ideal experimental set-ups.

We address this subject in three real-life industrial use cases, in which MT has to quickly adapt in accordance with specific operating conditions. In particular, we explore domain adaptation when no in-domain parallel data are available, which is a typical use case of MT service providers. Then, we investigate topic adaptation for the translation of short highly ambiguous item titles in an e-commerce setting.

Finally, we consider the Computer Assisted Translation (CAT) scenario, in which MT interacts with a human translator by providing them with translation drafts and by adapting from their post-editions. In this scenario, we investigate online adaptation from human post-editions, respectively, in a single-user setting and in a multi-user setting, in which multiple translators are working on different parts of the same document. In addition, for the single-user case we also discuss the optimisation of the hyper-parameters of the employed online adaptation method.

Keywords

Statistical Machine Translation, Adaptation Computer Assisted Translation, Online Learning Parameter Estimation, Sparse Features, Multi-Task Learning

Acknowledgements

This PhD is dedicate to my childhood hero, my grandpa - Babaji who always wanted me to be a PhD holder.
Another dedication goes out to my parents. Mummy-Papa I did it, I finally finished my thesis.

I would like to express my sincere gratitude to my advisors Marcello and Mauro. Without their guidance I would not be where I am now.

Marcello took me under his wing in my first stint at FBK as an intern. The short period in Trento was amazing, which led me to make an easy decision to come again to Trento and this stint was a long and memorable one. It was in my later years of PhD that I built a strong connection with Marcello and his family due to our stay in San Jose. During my PhD, I embodied Mauro's persona that kept me calm and composed. His zeal to question everything made me into a good researcher today. Thank you Marcello and Mauro.

I would like to thank Nicola Cancedda and Sriram Venkatapathy for the internship in Xerox. I got to meet amazing people in Xerox and live in the beautiful city of Grenoble. Sriram Sir, I cannot thank you enough. It was because of you I started working in Machine Translation and I still am. Thank you to Diana and Kunal for making my stay awesome.

I would like to thank Hassan Sawaf for giving me internship opportunity at eBay. Hassan is an epitome of hard-work and a true people's person and I surely look upto him. The internship did not just helped me career wise but my stay in San Jose helped me in reconnecting with my good old friends.

Rolling back in time, the first year of PhD was difficult. I had to settle

down in a new city, a new country. Amazingly, Trento had a big Indian community that made me feel like home. I will miss all the festivals we celebrated, the dances we organized. I will especially miss you people - Parveen bhai, Maitreyi bhabhi, Raju, Soudip bhai, Niyati bhabhi.

I would like to thank the people in Trento who came across my PhD life. José - The closest friend from Trento, we studied together, lived together and now we even work together. Nick - We started our Trento expedition at the same time and we are ending it at the same time. I'll miss working with you man. Arianna - Thank you for being an inspiration throughout my PhD, I looked up to you and you always set the bar high. Gözde, Serra, Kostas - My first connections to Turkey and Greece. I will miss all the countless dinners, apertivos, after lunch coffees and specially the holiday in Turkey. Thanks for the endless MT discussions Amin, Marco, Matteo, Nicola, Roldano.

A big thanks to Duygu. Though I met her in the end of my PhD cycle but without her completing this PhD would have been a total nightmare. Thank you General!

Last but not the least a big thanks to my family. Special thanks to my inspirational Chandni didi. I am waiting for your book to come out. Thank you Nani, Kanu bhaiya, Deepika didi, Devika, Taiji, Tauji, Mama, Mami, Mausi, Mausaji.

And to all the unnamed people who helped in any way in getting me here, thank you!

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Machine Translation (MT) is the field of Computational Linguistics that makes use of computers to translate text or speech from one language to another. Initial efforts to build automated translation engines have started almost as soon as electronic computers came into existence. The approach of what we call MT in today's world are mostly the same decoding algorithms developed by the British army in order to decode the German Enigma codes. Moving from the World War II to the Cold War, the underlying principle in approaching the production of such systems is ideally summarized by Warren Weaver, one of the pioneering minds in MT, who wrote in 1947:

"*When I look at an article in Russian, I say: This is really written in English, but it has been coded in some strange symbols. I will now proceed to decode.*"

Born in a divided world as a necessity of war, MT today promises to respond to the needs of a globalized world and allows better communication between countries, international corporates and civil organizations. With the advancement of technology and computational resources, recent approaches in MT has followed statistical or machine learning approaches. Statistical Machine Translation (SMT) constitutes the current state-of-

the-art in MT: based on the principle of using collections of translation examples to translate new text, these systems are typically good enough to produce at least a gist of translation of the source text.

The use of MT is predominantly classified in two broad categories [Hovy, 1996]:

1. Assimilation Tasks: Low to medium quality translation aiming to understand the gist of a document, e.g. translation of news, reviews, etc.

2. Dissemination Tasks: High quality translation for publishing content in other language, e.g. translation of manuals and business letters.

Our focus in this thesis will be on high quality translation for *dissemination* purpose. The importance of high quality translation can be judged by the fact that the language service (i.e. localization) industry is now worth more than US $38 billion, according to the information collected by Common Sense Advisory (CSA) [DePalma et al., 2015]. The same article also states that the total revenue of the top 100 language service providers is at least US $4 million. Moreover, the size of the industry is growing rapidly; CSA has predicted the language services market growth at 6% in 2016.

Localization is the process of adapting a globalized product in a locale. This detailed process involves the translation of product names, manuals and their online descriptions along with the modification of the time zone and currency that the product is serviced, such that the product looks as if it is being developed within the locale. Translation is a major part of localizing any product, thus, it is vital that the quality is perfect such that the resulting product image is reliable.

In localization companies, professional translators use computer assisted translation (CAT) tools, which are software packages consisting of special

text editors that handle different document formats, splits their content into segments to be translated, and finally reproduce the original layout and format from the (manually) translated segments. CAT tools also include useful linguistic resources such as a translation memory (TM), spell checkers, dictionaries, and terminology lists which can aid translators while working on the translation of a segment. A translation memory is a database of segments which were already translated by users. It can belong to any individual translator or to the company they work for. In the latter case, the TM would be a collection of translations carried out by different translators. In a standard CAT process each source segment is searched for in the TM and the best matches are shown with their stored translation to the user, who then decides whether to ignore, accept or post-edit them.

More often than not TMs fail to produce even a partial match in the database and the translators end up translating source sentences from scratch. The lack of reliability of TMs, due to their limited coverage has led to recent research efforts, which aim at integrating MT technology in the back-end of CAT tools in order to increase the efficiency of translators. With the surge in automatic translation accessibility, especially due to the public availability of the gisting and the customized MT systems, this research has picked up the pace. However, there are several reasons due to which MT is still not a popular choice amongst a few translators, who still prefer translating from scratch rather than post-editing automatic translations.

A translator typically expects an automatic system to provide a consistent level of quality and to be self-learning (i.e. adaptive) and not produce the same errors repetitively, especially after they have been fixed during the translation process. The lack of reliability in the non-adaptive MT systems is the major problem that makes translators hesitate to leverage the MT technology and instead prefer to use their own TMs. However, a

recent study by Green et al. [2013a] showed the efficacy of human post-editing for the language translation process. This study highlighted the fact that post-editing not only speeds up the process of translation but also improves the quality of it.

Statistical Machine Translation is a good fit in the post-editing framework because it makes use of a large available parallel corpus to generate statistical models for translation, thus has a higher generalization capability than the translation memories. In the cases where TM fails to produce a match, SMT is able to provide translation suggestions. A seamless integration of SMT engines in the post-editing framework have shown to increase the translator's productivity [Federico et al., 2012]. To date, the research has been more focused on this integration of MT with TM [Koehn and Senellart, 2010] and TM with MT [Zhechev and van Genabith, 2010].

Adaptive SMT systems are required in order to make the MT systems self-learning and thereby gain the trust of the translator during the process of post-editing. In this thesis, we focus on building these adaptive statistical MT systems in several real world scenarios.

1.1 Problem Scenarios

In this section we describe the business scenarios that this thesis considers and has the motivation to provide solutions.

Scenario 1: An MT service provider (MTSP) receives a request for translation of a project from a client who wants to globalize their product. As a typical case in the market, the client usually has the option to choose one MTSP among many competitors. To pick the best MTSP that fits its requirements, the client may opt to send a sample text to all the providers and make a decision based on the quality of the text translated by all the

MTSPs. This is a challenging situation for the MTSP; in spite of the fact that they have a library of ready MT models built on different domains, they do not have any information on the domain of the sample text which they will be translating. The goal of this phase is to build a domain-specific translation system so as to win the bid for translating the project.

Scenario 2: Translation in the e-commerce domain includes translating product descriptions (i.e. item titles) with concise user-generated texts describing each item put on sale. Item titles are generally short texts full of jargon and proper names. The challenge in this scenario is to correctly translate proper names which could be mistaken for common names as well as product features with multiple senses. One problem with translating item titles is that their size is usually short, 10 to 15 words, which represents a rather limited context.

Scenario 3: Scenario 3 requires the MTSP to develop a new MT technology, a back-end adaptive MT engine, for the localization company in order to facilitate the translation process for the translators with the CAT (post-editing) framework. The requirement is a two-fold adaptive MT system where the MT system adapts to the domain of the project before the actual translation starts and then adapts to the translator's correction or stylistics in real time during the actual translation of project.

Scenario 4: This scenario extends the previous one for the cases where a project is being translated by several translators and all of their feedback is provided to the MT system. An online adaptive system with many users can overlap the corrections and stylistics of several translators and end up not learning anything. The solution to be developed aims to prevent this risky situation and maximize the gain that can be obtained by using

post-edits in the multi-user CAT tools.

Scenario 5: Any online learning algorithm uses several hyperparameters which, if not properly tuned or optimized, can harm the learning process. The requirement in Scenario 5 is a new framework for optimizing the hyperparameters for the specific task of online learning in the CAT environment.

1.2 Structure of Thesis

We have set-up this thesis in a modularized manner in order to help the reader gain a wider insight into adaptation methods for SMT. The thesis has been structured as a tree, where each node is a different chapter. We start with this chapter by introducing the problem statements and its business use cases. Next chapter builds up the plot by shedding some light on the background that is needed to understand the following chapters. Rest of the thesis is divided into two different levels:

1. Connectivity
2. Type of Adaptation

In the following sections, we provide some details on both of these levels.

1.2.1 Connectivity

Connectivity refers to the state of deployment of a machine translation system. If the translation system is live on site then it is *online state* otherwise *offline state*.

Offline State In offline state, prior to deployment of the MT engine, the system is generally provided with a sample of already translated text in

the target domain of translation for adaptation purpose. The advantage of being in the offline state is that the adaptation of models does not have to be real-time, so one does not have to worry about speed issues while adapting an MT system. Rather, the focus is on improving the performance of the MT system on a given in-domain text.

Online State In online state, post deployment of the MT engine, the MT system adapts on the incoming text which could be in a different domain than the domain of training data. This adaptation brings another set of challenges. First, the adaptation of models needs to be in real-time, so there needs to be a trade-off between the speed and the amount of adaptation a translation system can do. Second, online adaptation occurs sentence by sentence, and adapting to such a small sample reduces the stability which could lead to overfitting or underfitting.

1.2.2 Type of Adaptation

The standard MT system is a linear combination of several features as represented in Equation 1.1.

$$Score(\tilde{S}, \tilde{T}) \quad = \quad \Sigma_i^m \lambda_i \cdot h_i(\tilde{S}, \tilde{T}) \tag{1.1}$$

Here, $\tilde{S}$ is the source segment, $\tilde{T}$ is the translation candidate, $Score()$ is the score computed by the model, $h_i()$ are the *features* associated with the model, and λ are the *weights* associated with the features. Adaptation of SMT system can be done in two ways as described below.

Feature Adaptation In offline connectivity there are several ways in which in-domain data can be used to adapt a generic model (i.e. background model), such as: data selection, adding provenance features and linear

mixture of models. For online adaptation, one can extract features based on the information at hand such as: the source segment, the translation suggestion and the post-edited segment. Once the features are extracted, they can be used to modify the existing features in the translation model.

Weight Adaptation In a standard MT pipeline, after the MT models are built, weights for these models are learnt via a discriminative learning process such as minimum error rate training (MERT) [Och, 2003a]. Several discriminative learning processes can be applied in place of MERT to optimize the parameters. For the first scenario, we present a novel idea of predicting the weights (λs) of the model. For the latter, we exploit the use of existing online algorithms.

1.3 Contributions

The main original contributions of this thesis are:

- Casting MT adaptation in challenging business use cases.
- Offline model weights prediction for domain adaptation.
- On-line unsupervised topic-adaptation of model features.
- On-line adaptation of model weights in a single-user setting.
- On-line adaptation of model weights in a multi-user setting.
- Optimization of online adaptation hyper-parameters

1.4 Related Publications

- Prashant Mathur, Marcello Federico, Selçuk Köprü, Shahram Khadivi, Hassan Sawaf. Topic Adaptation for Machine Translation of e-Commerce

Content. In Proceedings of MT Summit 2015. [Mathur et al., 2015]

- Prashant Mathur, Mauro Cettolo. Optimized MT Online Learning in Computer Assisted Translation. In Proceedings of Interactive and Adaptive MT workshop in AMTA 2014. [Mathur and Cettolo, 2014]

- Prashant Mathur, Mauro Cettolo, Marcello Federico, Jose Guilherme Camargo De Souza. Online Multi-User Adaptive SMT. In Proceedings of AMTA 2014. [Mathur et al., 2014a]

- Prashant Mathur, Sriram Venkatapathy, Nicola Cancedda. Fast domain adaptation of SMT models without in-domain parallel data. In Proceedings of Conference on Computational Linguistics 2014. Dublin, Ireland. [Mathur et al., 2014b]

- Prashant Mathur, Mauro Cettolo, Marcello Federico. Online learning approaches in Computer Assisted Translation. ACL Workshop on Machine Translation (WMT), Sofia, Bulgaria, August 2013. [Mathur et al., 2013]

Chapter 2

Background on Adaptation in Statistical MT

The aspect of natural language processing where machines automatically translate a text from one language to another is called Machine Translation (MT).

Several paradigms of machine translation have been developed over the last 50 years. The most prominent are:

- **Example based MT** where the translation is done via analogy [Nagao, 1984]. This type of translation system can be seen as translation via concatenation of translation fragments automatically extracted from the translation samples.

- **Rule based MT** where manually designed linguistic rules are applied to translate syntactic structures within sentences; the resulting system being specific to the source and target language pair [Nirenburg et al., 1992].

- **Statistical MT** sees translation as a machine learning problem; a translation system learns to translate automatically by analyzing large amounts of already available human translated texts [Koehn, 2010].

- **Hybrid MT** is a popular view within commercial MT systems (e.g. Systran Inc.) where statistical translation rules are combined with linguistic translation rules in order to build a robust MT system.

- **Neural MT** is basically an interlingua approach to translation. A Neural MT system is a concatenation of an Encoder, a Recurrent Neural Net (RNN) that encodes the information of a source sentence in a vector, and a Decoder (RNN) that decodes this information and predicts the target sentence Sutskever et al. [2014].

Our focus in this thesis will be on statistical MT, where translation of a sentence is modeled as a stochastic process generating either words [Brown et al., 1993], phrases [Koehn et al., 2003] or partial parse trees [Yamada and Knight, 2001, Chiang et al., 2005].[1] In particular, we work on phrase based SMT; the next section describes it in detail.

2.1 Phrase based SMT

Statistical MT can be seen as the problem of predicting a target sentence e given a foreign sentence f. Mathematically we can write it as:

$$e^* = arg\max_e P(e|f) \tag{2.1}$$

where e^* is the most probable English sentence given the foreign sentence f.

SMT is modeled with a noisy channel model, where instead of figuring out e from f directly, we see the problem as if a person X spoke a sentence e and by the time it reached Y it was garbled with noise and converted to f. The problem reduces to Y deducing the actual sentence e spoken by X.

[1] In this thesis, since our focus is on statistical MT, we sometimes refer statistical MT as just MT.

For noisy channel modeling, we apply Bayes rule to the above equation:

$$e^* = arg\max_e P(f|e) \times P(e) \tag{2.2}$$

Hence, we can split the problem into two parts: first, we deduce how the English sentence e garbled into the noisy foreign sentence f; second, we check if the English e is spoken correctly. In other words, $P(f|e)$ is the translation model which models the adequacy of the translation, while $P(e)$ is the language model which models the fluency of the translation e.

Word based model: In early 1990's SMT approaches were based on word based models [Brown et al., 1993]. Under word based SMT, both translation and language models are factored at the level of words. The language model is a stochastic n-gram model, where each word in the target language is assigned a probability given its preceeding n-1 words. The translation model, factored at word level, is first arbitrated through a word alignment model as in Equation 2.3.

$$P(f|e) = \sum_a P(f, a|e) \tag{2.3}$$

Alignment a is a hidden variable which maps each source word $(f_j; j \in 1, \ldots, J\ )$ in the source sentence with length J to a target word $(e_i; i \in 1, \ldots, I)$ in the target sentence with a length of I. $a_j = i$ represents a alignment mapping from j^{th} source word to the i^{th} target word.

In the beginning of training the SMT system, a parallel corpus aligned at the sentence level is provided. A first step towards building a translation model, thus, is word aligning the sentences in the source and the target language. Brown et al. [1993] proposed several *IBM models* to learn the latent word alignments from a sentence aligned parallel corpus via Expectation-Maximization algorithm.

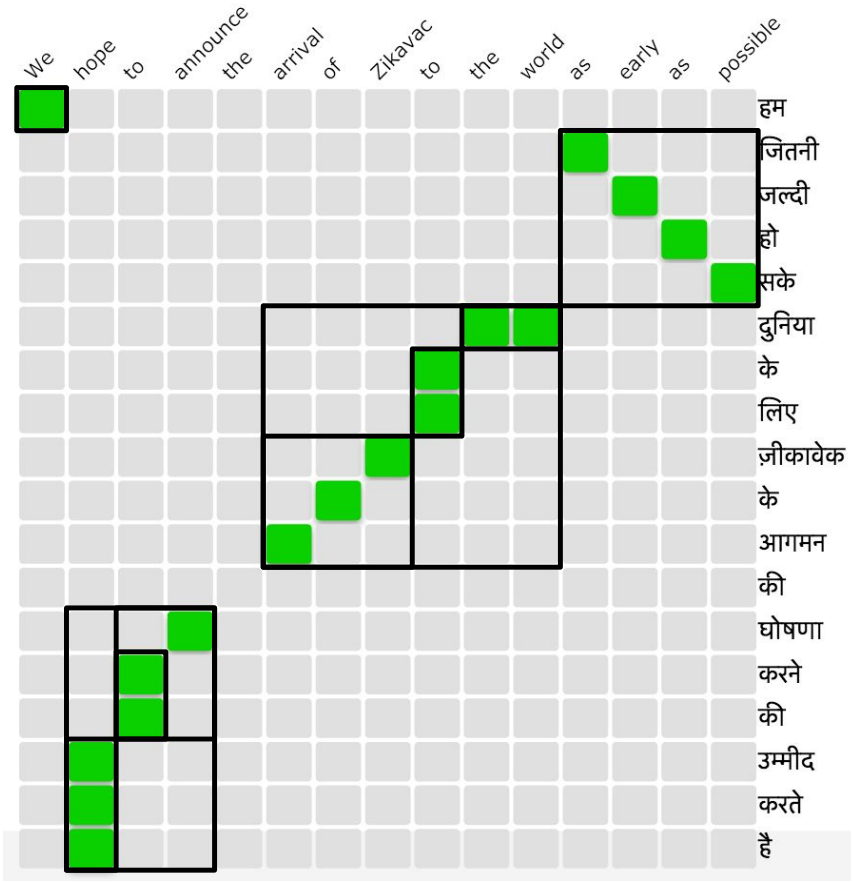

Figure 2.1: An English-Hindi translation example with alignments, phrase boundaries and phrase reorderings.

An alignment between words of an English-Hindi sentence pair is shown in Figure 2.1, where a green cell marks an alignment point $a_j = i$. In the top right black block in the example, the four green cells show a *monotonic* behavior i.e. the source words (*as early as possible*) translate one after another. This monotonic behavior of moving down and right ($\searrow$) in the matrix is a type of word reordering. A contrary up-right movement ($\nearrow$) is observed when a source word (*arrival*) translates after the successive source word (*of*) in the sentence; this type of movement of words is called a *swap* reordering. We also see a non-contiguous movement of words when *as* (... early as possible) is translated after *We* (... hope to announce). This type of movement is called *discontinuous* reordering.

Phrase based model: In PBSMT, the decomposition occurs at the level of *phrases* instead of words; a phrase being a word or a contiguous sequence of words. It is common for a contiguous sequence of words to translate as a unit in a real translation scenario. For example in Figure 2.1, the green cells form blocks that can be translated as a unit. The top right black block form a contiguous sequence, and with each green cell representing word alignment, we call the black box as a phrase alignment between the source and target phrase. This means that the source phrase *as early as possible* is aligned to the target phrase *jitnii jaldii ho sake* (as soon as possible).[2]

Reorderings such as monotone, swap, discontinuous for words can also be observed at phrase level in the figure. A phrase based monotone reordering would mean that the contiguous phrases translate in sequence, a swap reordering where a successive phrase is translated before a phrase, a discontinuous reordering where two discontiguous phrases are translated in sequence.

Contrary to word based SMT, the conditional probability $P(f|e)$ in PBSMT is decomposed differently than the Equation 2.2 as:

$$e^* = arg\max_{e^*} P(e|f) = arg\max_{e} P(e_1^I) \times P(f_1^J, b_1^I|e_1^I) \tag{2.4}$$

where b_1^I represents the derivation of the target string from f_1^J to e_1^I i.e. it embeds the phrase segmentation scheme ($\tilde{f} \in f_1^J, \tilde{e} \in e_1^I$) on source string f_1^J and the alignment between the source phrase ($\tilde{f}$) and the target phrase ($\tilde{e}$) such that the target string generated is e_1^I. Thus, b_i represents the phrase alignment between source phrase $\tilde{f}_i$ and target phrase $\tilde{e}_i$.

To include the reordering phenomenon in Equation 2.4, $P(f_1^J, b_1^J|e_1^I)$ further breaks down into phrase (surface) translation model, which captures surface level translation, and phrase reordering model, which captures

[2]Instead of the wx notation for Hindi the final submission will contain the devnagri script.

the distance between phrase being translated and the previous translated phrase:

$$P(f_1^J, b_1^J|e_1^I) = \prod_{i=1}^{I} \phi(d_{\tilde{f}_i} - d_{\tilde{f}_{i-1}}) \times P(\tilde{f}_i, b_i|\tilde{e}_i) \tag{2.5}$$

where $d_{\tilde{f}_i}$ represents the start position of the source phrase for alignment b_i and $d_{\tilde{f}_{i-1}}$ represents the end position of source phrase in phrase alignment b_{i-1}.

To make the decoding tractable, Och and Ney [2002] moved towards log-linear modeling framework for statistical machine translation. In a log-linear framework the generative model probabilities are replaced with feature functions h. Every feature function h maps each source-target phrase pair and the alignment to a non-negative value. Each of these non-negative values are weighted with a scalar (λ), known as feature weights or scaling factors. Log-linear model basically alters the Equation 2.4 and the probabilities are replaced with the weighted sum of feature functions as shown in Equation 2.6:

$$e^* = arg\max_{e} \prod_{i=1}^{I} \sum_{k=1}^{K} \lambda_k \cdot h_k(f_i, e_i, b_i) \tag{2.6}$$

During decoding, each sentence is decomposed into all possible combinations of source phrases. Since every source phrase can have multiple target phrases (translation options), and each translation option can be individually scored for all feature functions; the decoder job is then to recombine these target phrases (while covering all source words) in such a way that the resulting target sentence achieves the highest score by the log-linear model (amongst all possible generated target translations). This recombination can be achieved in a **tractable** way with the use of dynamic programming.

A basic log linear model for the PBSMT system is represented with the

Model	Features	Representation ($h(\tilde{f}, \tilde{e})$)
Translation Model	phrase translation	$\phi(\tilde{f}, \tilde{b}\|\tilde{e})$, $\phi(\tilde{e}, \tilde{b}\|\tilde{f})$
	lexical translation	$\ell(\tilde{f}, \tilde{b} \mid \tilde{e})$, $\ell(\tilde{e}, \tilde{b} \mid \tilde{f})$
Reordering Model	Monotone	$d_m(\tilde{f}, \tilde{b}\|\tilde{e})$, $d_m(\tilde{e}, \tilde{b}\|\tilde{f})$
	Swap	$d_s(\tilde{f}, \tilde{b}\|\tilde{e})$, $d_s(\tilde{e}, \tilde{b}\|\tilde{f})$
	Discontinuous	$d_{dis}(\tilde{f}, \tilde{b}\|\tilde{e})$, $d_m(\tilde{e}, \tilde{b}\|\tilde{f})$
Penalty	Phrase Penalty	$\eta_{phr}(\tilde{f}, \tilde{e})$
	Word Penalty	$\eta_{word}(\tilde{f}, \tilde{e})$
	Distortion	$dist(b)$
Language Model	Language Model	$P(\tilde{e})$

Table 2.1: Standard models and features used in PBSMT systems.

following core features: $P(\tilde{e})$ represents a language model, $\phi(\tilde{f}, \tilde{b}|\tilde{e})$ represents a phrase translation model, $l(\tilde{f}, \tilde{b}|\tilde{e})$ represents a lexical translation model, $d_m(\tilde{f}, \tilde{b}|\tilde{e})$ represents a monotone reordering model, $d_s(\tilde{f}, \tilde{b}|\tilde{e})$ represents a swap reordering model and $d_{dis}(\tilde{f}, \tilde{b}|\tilde{e})$ represents a discontinuous reordering model.

In addition to these core features, a PBSMT system typically includes features in inverse direction (from e to f) too; all these features are collected in Table 2.1. Penalty features (η) restrict the PBSMT system to produce sentences with short phrases, while the distortion feature ($dist$) restricts longer jumps between successive phrase translations.

So far, we reviewed the theoretical aspect of a SMT system. Practically, in a standard pipeline for building a SMT system a bitext is supplied to the SMT system; the bitext is tokenized, cleaned and then sent to a word aligning toolkit. Word aligners such as Giza++ [Och and Ney, 2003] or Fast Align [Dyer et al., 2013] are leveraged to align the words in the sentences of the parallel bitext. Phrase pairs are heuristically extracted from the aligned corpora and then scored to create a phrase table [Koehn et al., 2007a]. Simultaneously, a language model is built on the target language using a large monolingual target side corpus. Weights of these models (λ) are then tuned on a development set held-out from the training set. At

translation time, the decoder collects the feature values and weights them with λ to compute a score for each hypothesis.

2.2 Problem of Adaptation in SMT

The parallel corpora for building an SMT system can be collected from various sources of text such as web blogs, news websites, legal documents from European parliament etc. A *domain* can be seen as the type of content collected from one single source. Thus, a collection of texts from various sources represents several domains. Machine translation systems, as any other machine learning system, can suffer from a mismatch between training domain and application domain [Rogati, 2009]. For example, a domain mismatch can occur if one trains a MT system on a mix of news and legal documents and then applies it to translate text from the information technology manuals. To counter this mismatch one can drive the MT system to *adapt* to the target domain.

In this chapter, we shed light on a few adaptation problems and review the previous works for the same in machine translation. We classify our review of the adaptation techniques in two different time-lines: 1) when we are building the SMT system and 2) when the system is built and deployed to the user.

2.3 Offline Adaptation

This section describes the research on adaptation methods, that are applied before deploying the SMT system. In an offline state of a SMT system, one has access to the parallel corpora, the MT models and the feature weights, which can be modified to adapt to the evaluation domain.

Within offline adaptation, our focus is mainly on translation models.

The translation model captures the adequacy of a translation and can be seen as a dictionary of source and target phrases (as shown in Table 2.1). A language model (henceforth LM) on the other hand captures the fluency of the text and can be seen as a probability distribution over a sequence of words (called n-grams), assigning a probability to a sequence of words with length n. This section talks about altering these features and their weights by augmenting and modifying existing features, adding new features and tuning the feature weights to better fit to evaluation task.

We further describe this alteration technique at two different levels of information granularity i.e. domain and topic. Domain adaptation and topic adaptation can be seen as complementary methods to cope with the variability between training and testing data in machine translation. Under this perspective, domain modeling typically assumes training data partitioned or hard clustered according to human defined labels, while topic modeling builds on fuzzy clustering of the data based on automatically learned labels.

2.3.1 Domain Adaptation

Domain adaptation has been widely studied under different styles of adaptation techniques for phrase based SMT. Initial work in domain adaptation dates back to adding more training data to the word level SMT system [Koehn and Knight, 2001]. The idea of this work was to add more training data to the SMT system in order provide more accurate word level translations following the principle of "*no data is better than more data*". Lately there has been a surge in the amount of monolingual and bilingual corpora, and the above principle was proven to not hold true always [Moore and Lewis, 2010, Axelrod et al., 2011].

In a typical SMT setting, we assume to have a small in-domain (IN) corpus and a large collection of generic data from various domains (OUT).

The works by Moore and Lewis [2010] and by Axelrod et al. [2011] focus on the selection of a subset of the available parallel corpus to train the SMT system instead of using it entirely. Moore and Lewis [2010] introduced a cross entropy based data selection approach for language model adaptation. In-domain (IN) and out-domain (OUT) language models are built on the corresponding corpora for the target language (E). Each sentence e in the out-domain target corpus is ranked according to the difference between the cross entropies (H) of the two language models (as in Equation 2.7), with lower score being ranked higher. Top-k sentences are then chosen as the "nearest" subset to the in-domain corpus and the SMT system is trained on this subset.

$$score(e) = H_{IN,E}(e) - H_{OUT,E}(e) \tag{2.7}$$

Axelrod et al. [2011] extended the data selection techniques by adding two more variants of selection criteria. First, for each sentence in the target side of out-domain corpus, perplexity based on the in-domain LM is calculated. Sentences are ranked based on the perplexity score and the top N are extracted. Second, sentence pairs in both source and target languages are ranked, instead of just the target side. Each sentence pair is scored with the cross-entropy difference of in-domain LM and out-domain LM. The idea is that the cross-entropy of the in-domain LM and out-domain LM should be similar, i.e. the lower the difference the closer the out-domain sentence pair is to the in-domain sentence pair. Hence a sentence pair f, e is scored with:

$$score(f, e) = [H_{IN,F}(f) - H_{OUT,F}(f)] + [H_{IN,E}(e) - H_{OUT,E}(e)] \tag{2.8}$$

Hildebrand et al. [2005] is one of the first works in translation model adaptation making use of information retrieval (IR) methods. The idea is to collect subsets from training data similar to the test set using IR

techniques and build translation models on those subsets for translations. For each document (in this case one document is one sentence) in the training and the evaluation set, they create a tf-idf vector [Salton and McGill, 1986] and retrieve the N most similar sentences from the parallel training corpus using the cosine similarity. The SMT system is then trained on the concatenation of the similar parallel sentences.

Mixture modelling for heterogeneous translation models was first proposed by Foster and Kuhn [2007]. They showed various ways of computing mixing coefficients for linear interpolation using several distance based metrics borrowed from information theory. Note that to calculate any such metrics it is required that one has an access to the source/target training corpus and source/target development corpus.

Koehn and Schroeder [2007a] built domain specific models which they combined as components in a standard log-linear SMT framework. Mixture modelling has also been applied to improve word alignment [Civera and Juan, 2007], to create an ensemble of translation models at decoding time [Razmara et al., 2012] with Minimum Bayes Risk decoding [Duan et al., 2010].

There are several methods that alter the phrase dictionary by adding in-domain information within each phrase pair. Bisazza et al. [2011a] introduces a *fill-up* approach which merges in-domain and out-domain models into one model and adds a binary feature (in addition to the standard features) for all the merged phrase pairs indicating whether they came from an in-domain corpus or out-domain. If there are conflicting phrase pairs from in-domain and out-domain phrase table, in-domain phrase table features are preferred. A similar approach to fill-up is the *backoff* approach where the provenance feature is skipped and just the translation features are copied.

[Banerjee et al., 2010] built several domain specific translation systems,

trained a classifier to assign each incoming sentence to a domain and used the domain specific system to translate the sentence. They assume that each sentence in the test set belongs to one of the already existing domains which means it would fail in the case where the sentence does not belong to any of the existing domains.

More recently, [Sennrich, 2012a] designed an approach to calculate mixing coefficients by minimizing the perplexity of translation models over an **aligned** development set for mixture modelling via linear interpolation or by weighting the corpora. Sennrich et. al. [Sennrich, 2012b] clustered a large heterogeneous development corpus and tuned a translation system on each cluster. In the decoding phase, each sentence was assigned to a cluster and the translation system tuned on that cluster was used to translate that sentence.

Domains as discussed before are a type of content collected from one single source and hence, hard labeled by humans. Topics on the other hand are modeled as hidden variables associated to the document and can be automatically learned without any supervision. The model that uncovers the latent thematic structure in a document collection is called *topic model*. In the following section, we review approaches to adapt the MT system by leveraging the latent information of topics in texts.

2.3.2 Topic Adaptation

Topic adaptation for statistical MT have been studied in the literature to enhance phrase-based models with additional topical information derived from the training and testing data.

Previous works have approached this issue under different perspectives, also providing different levels of integration of topic information. For instance, in [Gong et al., 2011] and [Ruiz and Federico, 2011] authors devised and exploited cross-lingual topic models to generate topic relevant target

words for an entire document or sentence.

In Eidelman et al. [2012], lexical probabilities are conditioned on topics and inserted alongside with the standard lexical features in a phrase-based SMT system. The authors infer topic distributions at the document level (here one sentence is considered as one document) on the source side of the parallel data. After extraction of the translation rules, topic-conditional lexical translation probabilities are inferred by computing the expectation over the topic vectors observed in all the sentences where each translation rule was extracted from. At decoding time, these probabilities are weighted by the topic prior inferred on the test document. This results in a set of sparse features, one per topic, which are tuned on a development set.

In Su et al. [2012] topic models are used to *off-line* adapt a phrase table trained on out-domain parallel corpora by using in-domain monolingual corpora. Topic distributions are inferred through *Hidden Topic Markov Model* [Gruber et al., 2007] trained on monolingual sentences. They leverage the topic information from topic models and marginalise the phrase translation probability over these topics to compute a topic-conditioned translation probability. Finally, the topic-conditioned phrase translation probabilities trained on out-domain data are weighted with the in-domain topic prior probabilities. This way, topic information is statically integrated in the training phase to bias the translation model towards the in-domain data.

In [Xiao et al., 2012] monolingual topic models are trained on the source and target side with LDA and topic vectors are associated with each rule in hierarchical machine translation [Chiang, 2007]. Differently from [Eidelman et al., 2012], during decoding topic posterior distribution on phrase-pairs are matched against the topic distribution of the test sentence. Matching is performed with the Hellinger divergence [Hellinger, 1909]. During decoding when one does not have access to the target side

topic vector, a projection matrix is learnt from the training data which allows a one-to-many source to target topic mapping. This projection allows the decoder to perform "matching" of topic distribution on the target side too. In addition, two features for source and target language for each phrase-pair are included indicating whether the phrase-pair is sensitive to a topic or not based on an entropy function.

In [Hewavitharana et al., 2013] topic adaptation is performed in context of machine translation of task-driven conversations. Topic vectors are inferred via Latent Dirichlet Allocation Blei et al. [2003] on the source side of in-domain parallel training data. At test time, the topic model of the conversation is incrementally updated with the updating topic vectors. During decoding, with a Moses-like phrase-based system, each candidate phrase-pair activates a feature function measuring the highest similarity between the current topic vector and all topic vectors associated to the occurrences of the phrase-pair in the training corpus. Similarity is computed by taking the complement of the Jensen-Shannon divergence [Lin, 2006].

In [Hasler et al., 2014] the authors combine domain adaptation and topic adaptation methods in a phrase-based statistical MT for the translation of texts from three different domains. Domains are hard labels defined beforehand but topics are soft clusters of automatically generated labels; the idea of this work is that when domain adaptation fails, topic adaptation can detect structures within the text and perform better than domain adaptation. For topic adaptation, the authors developed three sets of features associated to the phrase table apart from the standard Moses phrase-based feature functions (i) two topic-conditioned translation probabilities, (ii) a topic based unigram language model score and (iii) three topic distribution similarity feature functions. The first set of features introduces source-to-target phrase probabilities that account for topic information; the second set scores unigrams of the target phrase according

to their topic relevance; the last one measures the similarity between the input topic distribution and the topic distribution associated, respectively, to the whole phrase-pair, to the target phrase, and to the most representative target of the phrase. Topic distributions on the test set are inferred at the level of whole document. Particular care is taken about sampling data from different domains, to avoid domain biases, as well as sampling the same amount of context data for each phrase-pair, to avoid context bias. The main conclusion is that topic adaptation helps especially if there is a high divergence between training and testing domains. Moreover, topic vectors can be helpful also to predict the domain of test data when topic domain vectors are used as proxy of data.

The next section overviews various ways we can adopt in order to adapt the MT system when it is deployed.

2.4 Online Adaptation

This section focuses on adaptation of features and weights while the MT system is deployed online. While online, a traditional MT system is not able to quickly update the models (translation/language models). This is important because during translation of large text there often occurs a change in a domain or a topic with the incoming stream of data. Here, adapting online can be advantageous as MT can dynamically adapt to an evolving topic/domain and thereby improve its performance in real time.

Most of the research on online adaptation has been done either to change the structure of the translation or language model [Bertoldi et al., 2013, Denkowski et al., 2014] so as to adapt the features in the model, or just adapt the feature weights of the system to an incoming stream of supervised data [Ortiz-Martínez et al., 2010, Martínez-Gómez et al., 2011]. In the following sections we explain several adaptation strategies under different

perspectives.

2.4.1 Cache based Models

Cache is like a temporary storage for frequently accessed items. The principle of caching is basically to speed up the performance of any system by quick retrieval the frequent items from temporary storage instead of reading it from the disk. This principle applies in natural language tasks because when humans speak or write they tend to maintain their vocabulary and style from one sentence to another. This is particularly true in the translation of technical documentation, which shows a higher repetitiveness [Cettolo et al., 2014] with respect to other text genres, such as news.

One of the first works by Nepveu et al. [2004] introduced a caching mechanism for both translation and language models in a interactive machine translation framework. For language model adaptation, the new text being post-edited was inserted in a cache in the form of n-grams with n ranging from one to three. Hence, the language model score was computed as a linear interpolation of a conventional LM and a cache-based LM. Concerning the translation model, previously post-edited text was word aligned and the word pairs were inserted in a cache in order to retrieve translations for later segments.

A cache based translation model was also introduced in Tiedemann [2010] who proposed to add the recent translation options used in recent translations in an exponentially decaying cache LM and TM. If an entry is present in the cache, a linear interpolation of cache probability and actual TM/LM probability is computed and returned to the decoder.

Hardt and Elming [2010] leverage what they call the *file-context* effect: adapting SMT on the text while it is being translated can have an enormous benefit on the translation quality of the successive sentences in the

same text. Adaptation on the test set in question was done via a four-steps process: 1) translate the source segment 2) receive the post-edited segment 3) greedily align the source with the post-edit 4) run the extract and the score steps of the MT training pipeline to get a new local phrase table. This cache like phrase table is then used in combination with the background phrase table for the translation of successive segments. While experimenting, each test set was split in 20 equal parts, each part was translated with a background and a foreground phrase table trained on the last 10 parts. Creating this foreground table for each part of the test set in practice can be too costly and hence this idea is not really viable in production systems.

Inspired by [Nepveu et al., 2004] and [Hardt and Elming, 2010], Bertoldi et al. [2013] designed online adaptation of the translation and language models via a caching mechanism that allows to define and dynamically adapt a small and local model (with respect to the document at hand); during decoding, the local model is combined with the global SMT model estimated on the training data. The cache-based translation model (CBTM) is intended for integrating new translation alternatives suggested by the user and for rewarding those approved, with the ultimate goal of translating the successive segments more consistently with respect to the user preferences. The added advantage of this method is that one can insert new phrase pairs instead of just word pairs as in Nepveu et al. [2004]. The CBTM is implemented as an additional phrase table providing one score. This model dynamically changes over time in two ways: (i) new phrase-pairs can be inserted with an initial score, and (ii) scores of all current entries decay when new pairs are added. The authors also proposed a cache based language model which is built to reward recent n-grams (size can vary from one to the sentence length) found in post-edited translation. This model is implemented as an additional feature of the log-linear model, which pro-

vides a score for each translation option, based on a cache storing target n-grams.

In cache based TM, there are several ways of adding phrase pairs from the new aligned source and post-edited segment. Wäschle et al. [2013] force aligned the source and post-edited segment with a constrained search approach described in Cettolo et al. [2010]. The search uses the translation options from the decoder, the source and the post-edited segment and provides one exact phrase segmentation and alignment. This phrase alignment provides new source-target phrase pair which are then inserted in the CBTM.

So far, we analysed how a different architecture like the cache mechanism can help the MT system in adapting to new incoming data. Learning from new data in post-editing scenario can be classified on the basis of the amount of data used for the learning process. In particular, an update step can either be performed once the whole document is translated or every time a single sentence is post-edited. We call the former as *incremental learning* and the latter as *online learning*. In the following section, we first look at the various incremental adaptation strategies.

2.4.2 Incremental Learning

In a post-editing scenario, under *incremental learning* purview the MT system adapts the MT models only after a day of translation or when a whole document is translated. In theory, *incremental learning* has access to more statistics in one shot and so the convergence rate of the learning algorithm should be faster than *online learning* which has access to less statistics. Incremental learning has been mainly studied in the context of aligning incoming stream of data on the fly, and then adapting from it the models in SMT.

Liang and Klein [2009] compared two online variants of EM algorithm, *stepwise online EM* [Sato and Ishii, 2000, Cappé and Moulines, 2007] and *incremental EM* [Neal and Hinton, 1998] for various NLP tasks such as POS tagging, word alignment, document classification and word segmentation. The difference between the two online EMs is that stepwise EM interpolates the current sufficient statistics with the old sufficient statistics and then progressively forgets about the old ones. While incremental EM interpolates the current sufficient statistics with all the previous sufficient statistics seen so far making it more stable. Findings of the paper showed that stepwise EM requires two hyper-parameters, a step-size parameter and a mini-batch size parameter, which makes it prone to failure if these parameters are not chosen correctly. A high step-size parameter can converge faster but the final result might not be optimal, whereas, the larger the mini-batch size the better the model is, but with a large batch size the stepwise EM behaves like a standard EM. Incremental EM on the other hand requires substantial storage costs as it has to store sufficient statistics for each sample it has seen so far.

Levenberg et al. [2010] proposed an incremental adaptation technique for the core generative component of the SMT system, i.e. word alignments using the above stepwise EM approach. Liang and Klein [2009] implemented the stepwise EM approach in IBM-1 model whereas this work extends the implementation to hidden markov models for word alignments. They provide a more dynamic way to adapt the incoming stream of parallel data in the background translation model with the help of dynamic suffix arrays [Lopez, 2008]. Dynamic suffix arrays can efficiently store the aligned parallel corpus in an efficient data structure, thereby, reducing the computational cost of searching the phrase pairs during the translation process. When a phrase translation is asked by the decoder, the corpus is searched, counts are collected and probabilities are computed on the fly.

This dynamic adaptation approach can also work on per-sentence basis, thus fits the post-editing scenario.

Another work on incremental retraining of the SMT system is that of Potet et al. [2011] where they adapted a baseline static SMT system by adding new post-edited data to the training corpus and then retraining the whole MT pipeline. The size of the new post-edit data of course is smaller than the actual training data, so they replicate the post-edits N (here 10,000) times before adding it to the training corpus, in order to have a bias towards the new data. However, restarting the whole training process every time takes a lot of time which renders this adaptation approach inviable to business cases where MT system needs to quickly adapt to the new data.

We can safely assume that most of the incremental retraining procedures are unaffordable in the post-editing scenario; they can be only useful if we work at sentence level. The following section takes on the topic of online learning where the SMT system adapts to the post-edit after each sentence is translated.

2.4.3 Online Learning

SMT system under the purview of *online learning* in statistical machine translation can be adapted at sentence level (post-editing scenario) or phrase/word level (interactive machine translation). There have been various studies on the topic of online learning in interactive MT learning [Nepveu et al., 2004, Ortiz-Martínez et al., 2010, Martínez-Gómez et al., 2011] and for post-editing scenario [Cesa-Bianchi et al., 2008, Martínez-Gómez et al., 2012, Green et al., 2013b, Denkowski et al., 2014].

Unlike *incremental learning*, online learning takes advantage of the document being translated at the moment. The idea of using online learning in post-editing framework is to adapt the SMT system with the feedback

from the post-editors, thereby, enriching the user experience by making fewer translation errors in the future. Repeated correction of same errors committed by the SMT system can lose post-editors' trust and slow down the translation process of the document, consequently render the SMT system useless to the post-editor. This slow down of the process of post-editing is completely opposite to the idea of using SMT system in the background. Adaptation to post-edited segments can allow SMT system to learn the stylistics of the translator and the terminology and vocabulary of the document being translated, thus, improving the quality of translation.

Previous works in online learning can be classified according to the adaptation strategy they apply:

- User adaptation: SMT system can adapt to the translator's corrections in two ways:
  - Adapting the features in the MT model to better adapt to the vocabulary and the style of correction of the translator;
  - Adapting the feature weights because importance of each feature may vary with time.
- Multi-User adaptation: This strategy is applied in the case where the same MT system is deployed to serve multiple users. Adaptation of feature weights is performed by trying to simultaneously adapt to multiple users.
- Hyperparameter Optimization: For online learning to converge one has to carefully choose a set of hyperparameters used in the learning algorithm; there have been previous works in MT which focus on the optimization of these free parameters (hyperparameters).

Minimum error rate training (MERT) in SMT [Och and Ney, 2003] is a standard process in the SMT pipeline to tune the weights of the MT

models on a development set. Following MERT procedure, a few more search algorithms such as max-margin algorithm [Crammer and Singer, 2003], gradient descent approaches [Green et al., 2013b], re-ranking approaches [Hopkins and May, 2011] and derivative free optimization approaches [Cettolo and Federico, 2004] were implemented to optimize the weights (λ).

Liang et al. [2006] describe an end-to-end discriminative SMT system which adapts the feature weights leveraging an averaged Perceptron algorithm [Collins, 2002]. Several strategies were suggested to update the weights of SMT models towards the reference or the oracle translation (closest translation in the n-best list to the reference translation): (1) aggressively update towards the reference translation, *bold update*; (2) update towards the oracle translation in n-best list, *local update*; (3) either a *bold* update is performed, when the reference is reachable, or otherwise a *local* update is performed, *hybrid update*. A caveat of this approach which the authors discuss is that the update towards the reference is prone to overfitting the development set.

Following MERT procedure, Hopkins and May [2011] describe a pairwise ranking approach where they see tuning of the MT system on the development set as a ranking task. In this approach once the development set is translated, random pairs are extracted from each of the n-best translations and then compared against each other in terms of BLEU score [Papineni et al., 2001] and model score. If the two scores diverge, weights are updated with a Perceptron update step [Rosenblatt, 1958]. Hasler et al. [2011] implemented margin infused relaxed algorithm (MIRA) [Crammer and Singer, 2003] an online learning algorithm in Moses [Koehn et al., 2007a] to tune feature weights, and their results were comparable with that of MERT. MIRA and Perceptron were devised for online learning problems, and hence suitable fit for the post-edit scenario.

Ortiz-Martínez et al. [2010] presented an online learning framework for interactive machine translation via feature adaptation. All the standard features in the phrase table are probabilities, therefore, in an online framework where the system has to adapt to the incoming stream of parallel data, one has to keep track of the *sufficient statistics* for calculating these probabilities. The authors kept the counts of phrases and words in both languages in order to compute the phrase and lexical translation probabilities. Similarly, for other standard set of features *sufficient statistics* are recorded and updated as the new data comes. There still exists a problem of aligning the words in the new sentence pairs for which the authors leverage incremental EM [Neal and Hinton, 1998]. Once the sentences are aligned, the sufficient statistics are updated for all the features. However, storage of these *sufficient statistics* can be quite costly and makes the decoding slower.

Martínez-Gómez et al. [2012] presented a comparison of online adaptation techniques in post editing scenario. They compared various learning algorithms such as Passive-Aggressive [Crammer and Singer, 2003], Perceptron [Rosenblatt, 1958], Bayesian predictive adaptation [Duda et al., 2001] and Ridge Regression for both feature and weight adaptation. Cesa-Bianchi et al. [2008] proposed a slightly different online learning approach during decoding. The authors construct a layer of "online weights" over the standard weights and update these weights at sentence level with MIRA; to our knowledge, this is the first work on online adaptation during decoding.

Recent works on online adaptation in post-editing scenario have made major advancements in this field. Green et al. [2013b] proposed AdaGrad [Duchi et al., 2011], a fast online learning adaptive algorithm for tuning the feature weights. AdaGrad is a type of stochastic gradient descent (SGD) algorithm which in addition to SGD keeps track of per-feature previous gradients by keeping track of the previous updates of each fea-

ture in the MT model. In addition to AdaGrad as the learning algorithm, Green et al. [2013b] uses pairwise ranking optimization (PRO) to tune the feature weights of the SMT system.

Denkowski et al. [2014] presented the online learning framework for hierarchical machine translation [Chiang, 2007]. Their online approach included extraction of new hierarchical rules from the incoming source and post-edit segments, online adaptation of weights of the MT models with MIRA and using the updated model for the translation of next source segments.

Later same year, Germann [2014] proposed a simulated post-editing framework with the help of dynamic suffix arrays (DSA) as data structures. DSA also allows the addition of new parallel text along with the word alignments, which makes it suitable for the post-editing scenario. In this approach, the source and target corpora are stored in these DSA along with the word alignments, and as soon as the new post-edited segment comes, it is word aligned with the source segment and the tuple of <source, target, alignment> is added to the dynamic data structure. This way the models are up to date with the incoming stream of text.

So far, we have talked about online learning works, whose application is user adaptive SMT, where an SMT system is adapted to the stylistics and terminology of a particular user. In the next section, we review various works related to **multi** user adaptive SMT.

2.4.4 Online Multi-User Adaptation

The SMT systems featuring online adaptation from a post-editor are not necessarily optimized for handling the feeback from multiple users at the same time on the same translation project. Online multi-user learning scenario can be considered as an online *multi-task* learning problem [Caruana, 1993] where each user is seen as one task and so the idea is to learn from all

the tasks/users at the same time. In this section we review previous works in SMT under the umbrella of multi-task learning, though task unspecific.

Duh et al. [2010] approached the problem of re-ranking an n-best list generated with sparse features as a multi-task learning problem. They proposed a meta-algorithm to discover common feature representations across large number of n-best list. Once these features are extracted, a normal re-ranker on these common features can be used to tune the MT models. The authors' argument for their proposed approach is that with a large sparse feature set it is rare to find **sparse** feature sets which are common across the input which itself can vary a lot. This variance in the input and the n-best list (features) can be captured by joint regularization.

Simianer et al. [2011] casted MERT training procedure on a patent translation domain as a multi-task learning problem. Multi-task learning allows to address commonalities through shared parameters and model differences through task-specific parameters. They addressed a particular task of patent translation where the shared features model the patent jargon and textual structure, while task-specific features capture the terminology for each class of patents.

Multi-task learning has been explored in SMT in the context of tuning the sparse log linear weights by Simianer et al. [2012] where they split the training set in random shards and perform a joint feature selection over these shards using ℓ_1/ℓ_2 regularization. In this way after each epoch, the size of feature vector decreases and only the important features are taken into account.

In multi-task learning each task can be correlated with the other tasks and Cavallanti et al. [2010] exploited this behaviour by explicitly encoding the task relationships in a matrix which is assumed to be known beforehand. They presented an online multi-task learning approach where after each training sample all the models are updated leveraging the task rela-

tionship factor given in the matrix. However, an apriori assumption on the nature or extent of relatedness can often be restrictive. Saha et al. [2011] argued the same and that the task relatedness can alter with time and also extended the method of Cavallanti et al. [2010] by actively updating the task relationship matrix after seeing each training sample.

A recent application of multi-task learning has been quality estimation for machine translation by [Cohn and Specia, 2013] where the authors model annotator bias using multi-task Gaussian processes. Their model outperforms annotator specific models and thus boosting the use of Multi-Task learning in NLP applications. Another application of multi task learning has been in supervised domain adaptation for quality estimation [C. de Souza et al., 2014]. In this work the authors leverage all available training labels from different domains in order to learn a robust model for a target domain with very little labeled data. The proposed approach outperforms independent models trained separately on each domain.

With the adoption of the above online learning algorithms, hyperparameters such as learning rates, number of iterations, regularization constants etc. become an integral part of the learning process. In the next section we review previous works on optimization of hyperparameters in learning algorithms in context of machine translation.

2.4.5 Tuning of Hyperparameters

In machine translation, the hyperparameters of a learning algorithm (eg. learning rates), once optimized, are assumed to not have any effect on the actual translation. However, these hyperparameters could be sensitive to fluctuation in their values, thus, there is a need for direct search methods [Lewis et al., 2000]. This type of methods requires the availability of objective function values but no derivative information. The objective function takes in a set of source and reference sentences, a set of parame-

ters (weights and hyperparameters) and produces an evaluation score. In the case of SMT, the objective function could be any MT evaluation metric such as BLEU, TER etc. Derivative of these functions is impractical, hence, derivative free optimizers can be applied to tune the parameters.

The most notable work in the field of optimization of hyperparameters in machine translation is that by Chung and Galley [2012] where the decoder is integrated with a minimizer so that they can optimize the values of free parameters such as beam size and distortion limit. This minimizer runs derivative free optimization (DFO) techniques, such as Powell and Nelder-Mead methods, to optimize log-linear weights as well as the hyperparameters. They also argue that this integrated minimizer measures the *true error rate* whereas MERT minimizes the *artificial error rate* computed on a n-best list.

Another motivation to use the DFO methods was the work of Stymne et al. [2013] who focused on using a tunable distortion limit in the document level decoder Docent [Hardmeier et al., 2013]. Their system provided better BLEU scores when the distortion limit was tunable rather than when it is not tunable. This experiment further supports the need of optimization of hyperparameters in order to gain performance.

Learning of hyperparameters has been a widely studied topic in machine learning. Grid search, random search [Bergstra and Bengio, 2012], Gaussian process [Snoek et al., 2012] are only a few methods that have been used in the past for hyperparameter optimization. Gradient-based hyperparameter learning algorithms have been proposed for a variety of supervised learning models such as neural networks [Larsen et al., 1996]. In cases where the evaluation of the loss function is a costly procedure, requiring the translation of the whole development set; the application of the above approaches can then be unfeasible unless one leverages a model selection technique such as *Racing* [Moore and Lee, 1994] along with *Lattice*

based decoding as shown in Chung and Galley [2012].

2.5 Problems with state-of-the-art systems

This chapter throws light on important concepts, models, features and parameters of Machine Translation (MT). We reviewed various studies when adaptation for MT is necessary in an *offline state*:

1. There is no parallel in-domain data available for adaptation and there is a domain mismatch between training and evaluation data;

2. The topic of conversation continuously evolves in the evaluation data, and the MT system cannot adapt to the topical information.

Concerning the first point, we reviewed how most of the previous works either require a parallel in-domain training data or a target language in-domain monolingual data for adaptation procedure to work; contradicting our problem scenario where there is no or little source in-domain sample data is available. In theory, the approaches discussed in *Domain Adaptation* section are well suited for solving the problem of domain adaptation, but during the deployment of SMT systems when the client is unable to deliver the parallel in-domain data the same approaches fail to provide a quick solution.

As for the second point, we described various adaptation techniques at the *topical* level. We presented an in-depth review of topic adaptation approaches such as source side topic modeling and adaptation, dynamic adaptation using topic similarity between input and source-target phrase pair during decoding etc. However, our focus in this work has been to translate the item titles in an e-commerce inventory leveraging the context information in it.

We now focus on the issues arising when the MT system is deployed, it is in an *online state* and adaptation to the continuous feedback (in the form of post-editions) from translators becomes important. The online adaptation issues are categorized as:

1. Adapting the features of translation model;

2. Adapting the feature weights of MT models;

3. Optimization of hyperparameters such as learning rates of online adaptation algorithms;

4. Adapting a single MT system to multiple translators with different styles of corrections.

We overviewed various previous works on topics such as Cache based models [Nepveu et al., 2004, Hardt and Elming, 2010] addressing the problem of new terminology adaptation, incremental learning [Levenberg et al., 2010] addressing the problem of continuous adaptation to an incoming stream of data, interactive learning [Ortiz-Martínez et al., 2010, Martínez-Gómez et al., 2011] addressing the issue of predictive typing in CAT tools and learning from post-editions sentence by sentence [Cesa-Bianchi et al., 2008]. These works either addressed the problem of adapting the translation model or adapting the feature weights of the MT models or their implementation was not practical [Hardt and Elming, 2010], while we focus on a practical adaptive MT system which addresses both issues of adaptation of translation model and feature weights of MT models.

We then also reviewed a few works related to optimization of hyperparameters used in MT systems such as distortion limit and beam size. However, our task is to optimize the learning rates and stopping criteria in the online algorithm for MT, which has not been addressed before.

Later, we will address a difficult task of learning from multiple post-editions coming from different users at the same time. Since, this is a multi-task learning problem, we reviewed a few works which fall in the intersection of multi-task learning and Machine Translation. However, none of the works could directly be applied or address the problem of learning from multiple users for Machine Translation in CAT scenario.

Chapter 3

Domain Adaptation without in-domain parallel data

Domain adaptation is the process of adapting a SMT system trained on a generic domain data to a specific in-domain data such that an enhanced performance can be observed on unseen in-domain texts. In this chapter, in particular we assume that there is no in-domain parallel data available but only a small source monolingual text.

This is a challenging problem frequently faced by MT service providers who have to build a domain-specific SMT system exploiting only a pure source-monolingual sample of text from the domain. We address this problem by introducing methods for domain adaptation requiring no in-domain parallel data. Our approach yields results comparable to the state-of-the-art *oracle* systems (with the knowledge of the in-domain parallel set); experiments show a drop of as little as 0.5 BLEU points across 4 domains.

3.1 Overview

In this chapter, we consider the problem of creating the best possible statistical machine translation (SMT) system for a specific domain when no parallel sample or training data from such domain is available. We as-

sume that we have access to a collection of phrase tables (PT) and other models independently created from now **unavailable** corpora, and we receive a monolingual source language sample we would like to optimize the performance on.

This problem is an abstraction of a number of real-world situations where some user has recurring needs for translations in a specific domain or for a specific task, and it is therefore cost-effective to create a dedicated SMT system, but has no access to any parallel sample because the same task was either not done at all before or it was completed through a different process without resorting to translation.

For a MT provider to deliver a SMT system tailored to a customer's domain, a sample dataset is requested. In most cases, the customer is able to provide an in-domain monolingual sample from his operations. However, it is generally not feasible for the customer to provide the translations as well because the customer needs to hire professional translators to do that. In such a scenario, the translations have to be generated by the MT service provider itself by hiring human translators thus requiring an upfront investment. The methods proposed in this chapter aim to avoid that by building a good quality **pilot** SMT system leveraging only sample monolingual source corpus, and previously trained library of models. This in turn postpones the task of generating in-domain parallel data to a later date when there is a commitment by the customer.

Unavailability of the raw parallel data could derive from a collaboration or trading model where data owners share intermediate-level resources like PTs, Reordering Models (RM) and Language Models (LM), but can not, or do not want to, share the textual data such resources were derived from. Cancedda [2012] presents this particular scenario in detail.

This scenario is also similar to the multi-model framework studied in Sennrich et al. [2013], with the additional challenge that no parallel devel-

opment set is available. We build on the linear mixture model combination of the cited work, extending it to our more challenging environment:

1. We propose a new measure derived from the popular BLEU score [Papineni et al., 2002a] to assess the fitness of a PT to cope with a given monolingual sample S. This measure is computed from n-gram statistics that can be easily extracted from a PT, and is amenable to a very efficient implementation via finite-state transducer (FST).

2. We propose a new method for tuning the parameters of a log linear model that does not require an in-domain parallel development set, and yet achieves results very close to traditional tuning on parallel in-domain data.

We present the formulation of the *BLEU-PT* metric and the computation of multi-model in Section 3.2. The parameter estimation of log-linear parameters of the SMT system is described in Section 3.3. We describe the datasets and compare the proposed metric *BLEU-PT* with Cross-Entropy in Section 3.4. MT experiments and results are presented in Section 3.5 and Section 3.6 respectively. Later, we provide a use case scenario for our application of domain adaptation in Section 3.7 and then conclude the chapter.

The main ideas of the presented approach have been published in [Mathur et al., 2014b].

3.2 Building Multi-Model

Given a library of phrase tables from various domains, the goal is to first generate a subset of relevant phrase tables that need to be retrieved from the library and then create a domain adapted multi-model (phrase table)

from the retrieved set. There are two reasons that motivate the retrieval of relevant phrase tables:

1. Out of the entire library, only a subset might be useful for a particular domain;

2. It is computational expensive to use a combined model of a large number of phrase tables.

The challenging aspect in our scenario is the lack of in-domain parallel data, as well as the absence of original parallel corpora corresponding to the library of models. This rules out the possibility of using metrics such as cross-entropy [Sennrich, 2012a] or LM-perplexity for computing the mixing coefficients [Foster and Kuhn, 2007]. We present our proposed metric in the following section and interpolation of the models in Section 3.2.2.

3.2.1 Retrieval of Phrase Tables

Given a source sample corpus s, and a set of phrase tables $\{pt_1, pt_2, \ldots, pt_n\}$, the goal is to measure the similarity of each of these tables with s. We adapt the popular BLEU score [Papineni et al., 2001] to BLEU-PT (BLEU of a Phrase Table) for measuring the similarity between a corpus and a phrase table. The metric BLEU-PT is defined in Equation 3.1:

$$\text{BLEU-PT}(PT, S) = \left(\prod_{n=1}^{4} \frac{match(n|pt, s)}{total(n|s)} \right)^{1/4} \tag{3.1}$$

where $match(n|pt, s)$ is the count of n-grams of order n in the source sample corpus s that exist in the source side of the phrase table pt. $total(n|s)$ is the number of n-grams of order n in the source corpus.

The bottleneck in computation of BLEU-PT is the calculation of the number of matches of n-grams between source phrases in the sample corpus

source phrase	phrase table index
out for the dinner	1,2
out for the shopping	2
at your service ma'am	1

Table 3.1: Source phrases from different phrase tables.

	TRIE	FST
Storage	1 GB	84 MB
Time	5 secs	13 secs

Table 3.2: Table shows the pros and cons of using a TRIE v/s a FST. For comparison purpose, we indexed a 250 MB phrase table as train FST and queried with a source sample corpus. The phrase table took 1 GB of storage space when indexed with a trie and 84 MB when indexed with a FST. Total time of querying the data structure with a chosen source sample was 5 seconds for the trie vs 13 seconds for the FST.

and in the phrase table. The goal is to first store the source phrases coming from multiple models or phrase tables and then retrieve them when the source phrases from sample corpus are queried. Since a source phrase can occur in multiple phrase tables (as shown in Table 3.1), we can store source phrases and their phrase table IDs. The problem is to find a data structure that allows to store the pairs at a low cost and at the same time be efficient in computation of the *match* of n-grams.

In the past, FSTs have been successfully applied in NLP and string matching problems [Mohri, 1997]. We ran a few experiments comparing the feasibility of a FST versus a trie. Results of the experiments are shown in Table 3.2. For data structure, we used FST because there is a highly optimized OpenFST library which makes it easier to store, manage and compress the FSTs with the help of the inbuilt functions.[1]

Figure 3.1 shows the indexed FST (henceforth, *train* FST) which encodes the source phrases from different phrase tables as shown in Table 3.1. The input symbols are the vocabulary of the source language and output symbols are the phrase table IDs. For example, in the transition from

[1] `http://www.openfst.org/`

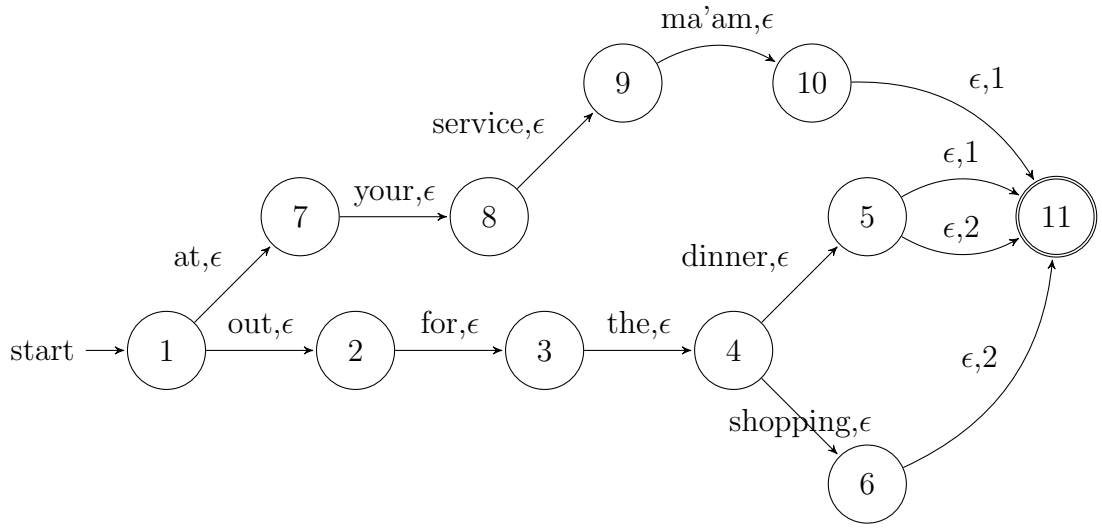

Figure 3.1: Indexed train FST.

state 1 to state 7, FST consumes the input *at* and spits an empty output symbol ϵ. In the transition from the state 10 to the state 11 (an accepted final state) FST consumes an empty input ϵ and spits 1 as the output. This basically means that the source phrase consumed until state 10 (i.e. *at your service ma'am*) comes from the phrase table ID 1. Once all the phrase tables in the library are read and indexed in the train FST, the FST is determinized and stored in a binary format on disk.

When the source sample arrives to the MT provider, all the n-grams in the sample are indexed in a *query* FST (as shown in Figure 3.2). In the query FST, the output symbols are the vocabulary of the source sample and the input symbol is an empty ϵ. Once all the n-grams in the source sample are indexed, the query FST is composed with the train FST. The *composed* FST is shown in Figure 3.3. The idea of composition is that the train FST consumes an input phrase from the source sample and outputs the indices of all the phrase tables where the input phrase occurs.

In the composed FST we check for all the transitions where the input and the output label are equal in order to identify a "match" between the n-grams. Once accepted state is reached, the output labels from the last

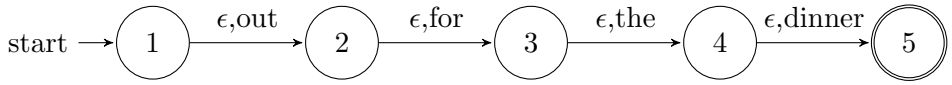

Figure 3.2: Query FST.

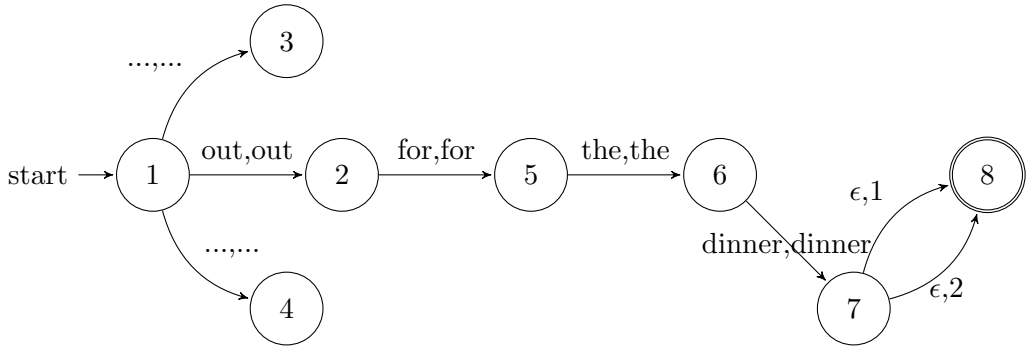

Figure 3.3: Composed FST.

transition are collected. These matches are computed over one pass of the source corpus s. At each position in the corpus, all n-grams up to the desired order are looked up in the FST index, and the identifiers of the PTs holding matching phrases are retrieved.

For example, the transducer in Figure 3.1 would accept an input phrase "at your service ma'am" and output 1 indicating that this phrase is present only in phrase table 1, while accepting an input phrase "out for the dinner" to return 1 and 2 indicating that this phrase is present in both the tables 1 and 2. The value of $match(n|pt, s)$ is incremented for each of the matching phrase tables. The number of matches is then used to compute BLEU-PT as presented in Equation 3.1.

After calculation of BLEU-PT, phrase tables indices are sorted in the descending order of their BLEU-PT score. Depending on the size of phrase table library, the most relevant phrase tables are selected for building the multi-model. Simultaneously, the corresponding RMs and LMs are also retrieved from the library. Once the subset of models are retrieved the problem of creating a multi-model remains. The next section describes the ways of creating the multi-model by first computing the mixing coefficients

and then linearly combining multiple models.

3.2.2 Interpolating Models

A state-of-the-art approach for building multi-models is through linear interpolation of component models, exemplified in Equation 3.2 for the case of the forward conditional phrase translation feature ($\phi(\tilde{e}|\tilde{f})$):

$$\phi(\tilde{e}|\tilde{f}) = \sum_{j=1}^{N} \eta_j \phi_j(\tilde{e}|\tilde{f}) \tag{3.2}$$

where N is the number of models under linear interpolation. Various approaches have been suggested for computing the coefficients η of the interpolated model, the most recent being perplexity minimization described in Sennrich [2012a], where each translation model feature is optimized separately on the parallel development set. Our work is set in a scenario where no parallel development set is available for optimizing the interpolation coefficients. We have also observed that perplexity minimization is computationally intensive, requires aligned parallel development set, and the optimization time increases rapidly with increasing number of component models (for details, see Section 3.4.2).

We propose a simple approach for the computation of the mixing coefficients that relies on the similarity of each model with respect to the test set. The mixing coefficients are obtained by normalizing similarity values. The similarity between a model (phrase table) and a corpus is computed using the BLEU-PT metric proposed in the previous section. Another similarity metric that could be used is the *source LM Perplexity*. We could compute the perplexity on the given source sample of a language model trained on the source side of parallel text. However, in the current scenario we do not have resources (training data) to build a source side LM.

In Section 3.6 we empirically compare our method for computing mixing coefficients with the perplexity minimization method. In the end, we also apply the same mixing coefficients obtained by our method as the mixing coefficients for the reordering and the language model.

With the linear combination of several models, once the multi-model is created, next step is to tune the log-linear feature weights of the model. Standard tuning of weights of a model requires access to a development set. In the next section, we describe a way of predicting these weights in the absence of this development set.

3.3 Parameter Estimation

The overall quality of translation is strongly impacted by how the weight vector of the log-linear combination of various translation features are optimized for a domain of interest. MERT [Och, 2003b] and MIRA [Watanabe et al., 2007a] are popular solution to compute an optimal weight vector by minimizing the error on a held-out parallel development set.[2] BLEU and its approximations are commonly used error metrics. In this work we assume the lack of a parallel development set, therefore the above methods cannot be used.

Pecina et al. [2012] showed that the optimized log-linear weight vector of a SMT system does not depend as much on the domain of the development set (on which the system is optimized), as on how "distant" that domain is from the domain of the training corpora used to build the SMT models. This is an important finding. It means that the weight vector of SMT models can be seen as a function of the **distance/similarity** between the domains of the development set and the training corpora on which the SMT models were built on. In this work, we learn this function from examples

[2]Its called held-out because the development set is usually a part of the parallel training corpora and is held out for the unbaised tuning of the MT system.

of previous parameter optimizations, using our BLEU-PT as a similarity metric. Once we have retrieved the most relevant PTs (translation and reordering models) from our library, and we have linearly interpolated them using normalized BLEU-PT, we use the learned model to estimate the optimal value of the log-linear weights, instead of optimizing them.

In order to learn this mapping, we create a dataset of example pairs of the form <BLEU-PT, weight vector> (where BLEU-PT are normalized values) by performing repeated optimizations for out-domain models on a number of parallel development sets (see Section 3.4 for more details of this data) using a traditional optimization method (MIRA in this work). Based on this dataset, the function of our interest can therefore be learnt using a supervised approach. We explore two parametric methods and a non-parametric method. For a monolingual source in a new domain, the BLEU-PT of the multi-model is computed, and then mapped to the appropriate weight vector (λs). In Section 3.3.1 we study the parametric approaches; the non-parametric approach is explained in Section 3.3.2.

3.3.1 Parametric Methods

We considered two distinct parametric methods for estimating the mapping from model/corpus similarity into weight vectors i.e. *Regression* and *Multi-Task learning*. The first approach makes the assumption that parameters can be estimated independently of one another given the similarity, whereas the second tries to leverage the covariance between the parameters in the vector.

3.3.1.1 Linear Regression

Motivated by initial experiments highlighting strong correlation between BLEU-PT and optimal feature weights (see Section 3.5.1 below), we as-

Figure 3.4: Regression Model

sumed here a simple linear relation of the form:

$$\lambda_i^* = W_i \mathrm{X} + b_i \tag{3.3}$$

where λ_i^* is the optimal log-linear weight for feature i, X is the feature vector.[3] W_i and b_i are coefficients to be estimated. While being a strong assumption, this has the advantage of limiting the risk of overfitting in a situation like ours where there is only relatively few data points to learn from. We estimate W_i and b_i by simple least squares regression. Once these are available for all features, we can predict the log linear weights of any model given its BLEU-PT similarity to a monolingual source sample using Equation 3.3. Regression model can be better visualized in Figure 3.4, where we first learn the parameter W of the model given the training data of BLEU-PT as feature and log linear weight as the label and then estimate the labels (λ) based on the learned parameters W.

3.3.1.2 Multi-Task Regression

Optimal log-linear parameters might not be fully independent given BLEU-PT, especially since it is known that model features can be highly correlated. To account for correlation between parameter weights, we explore the use of Multi-Task lasso (MTL) [Caruana, 1997] where several functions corresponding to each parameter are jointly learned considering the

[3]Ideally, X is a vector but in this case we just have one feature i.e. BLEU-PT.

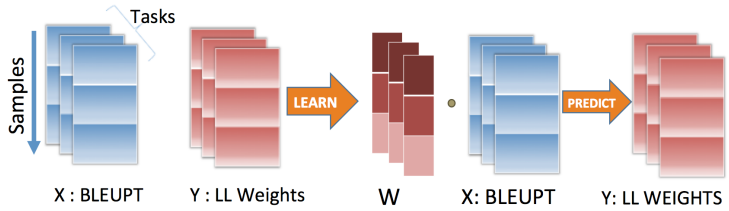

Figure 3.5: Multi-Task Regression Model

correlation between their values observed in the training data.[4] MTL is represented in Figure 3.5: it can be noted that the only difference from single task learning (i.e. Regression) is that MTL learns and predicts all the tasks at once. MTL consists of a least square loss model trained along with a regularizer and the objective is to minimize the following:

$$\arg\min_{W} \frac{1}{2N}||\mathrm{X} \cdot W - \lambda||_2^2 + \alpha||W||_{21}$$

$$where; \quad ||W||_{21} = \sum_{j}^{M} \sqrt{\sum_{i}^{K} w_{ij}^2} \tag{3.4}$$

Here, N is the number of training samples, X is the feature vector λ is the label vector (log linear weights). $||W||_{21}$ is the $\ell 1/\ell 2$ mixed norm regularizer [Yang et al., 2011]. The problem of predicting log linear weights is reduced to prediction of K interlinked tasks where each task has M features. Coefficients are calculated using coordinate descent algorithm in Multi-Task lasso. Once the coefficients are calculated we use Equation 3.3 to predict the log linear weights.

[4] `http://scikit-learn.org/stable/modules/generated/sklearn.linear_model.MultiTaskLasso.html`

3.3.2 Non Parametric

Finally, instead of building a parametric predictor for log linear weights, we experimented with a simple nearest-neighbor approach:

$$\lambda_i^* = \lambda_i(MM_{j^*}) \tag{3.5}$$

where MM_j ranges over the phrase tables of multi-model (MM), and $\lambda_i(MM)$ returns the stored optimal value for the i^{th} log-linear weight, and:

$$j^* = \arg\min_j \min_{s'}(|\text{BLEU-PT}(MM, s) - \text{BLEU-PT}'(MM_j, s')|) \tag{3.6}$$

where s is the monolingual sample on which we want to calculate the BLEU-PT and s' ranges over the source sides of a collection of available parallel development sets.[5] In other words, a BLEU-PT of a model is calculated on the source sample to be translated and the log-linear weight is chosen which corresponds to BLEU-PT′, where BLEU-PT′ is a training data point closest to BLEU-PT. This approach is close to the cross-domain tuning of Pecina et al. [2012] where the MT models trained on medical domain are optimized on medical domain and legal domain development set and it turns out that the system optimized on legal domain performs better than the other.

The nearest neighbor model is visualized in Figure 3.6. The blue dots in the plot show the training data; for each BLEU-PT value there exists a log-linear weight (λ) for the phrase translation feature (ϕ). When the source sample (s) arrives, multi-model MM is created and its BLEU-PT is calculated on s. In the figure, the BLEU-PT of MM on s is 23, and there are two training points with BLEU-PTs of 20 and 25 and corresponding weights of 0.4 and 0.964 near it. So, for the multi-model MM we choose the

[5]We talk about these **available** parallel development sets in the following section.

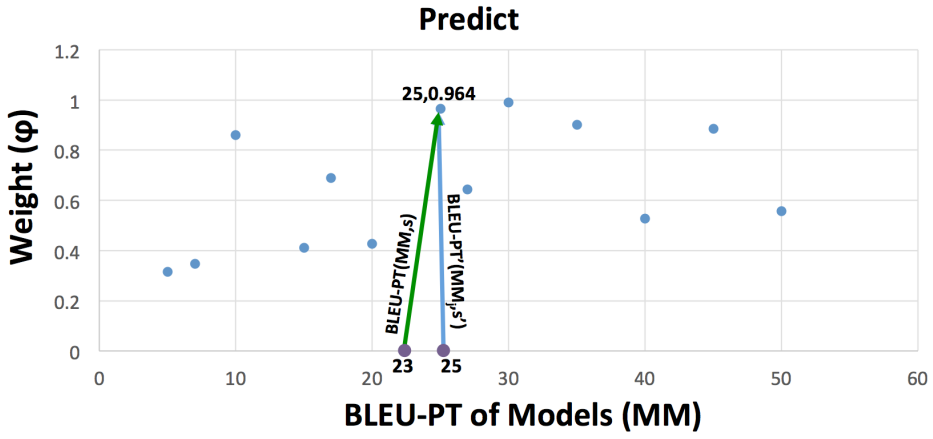

Figure 3.6: Nearest Neighbor Model

closest training point with BLEU-PT of 25 and set the weight of phrase translation feature as 0.964.

3.4 Experimental Program

We conducted a number of experiments for English-French language pair, comparing the methods proposed in the previous sections among one another and against state-of-the-art baselines and oracles.

3.4.1 Datasets

In this section, we present the datasets (EN-FR) that we have used for our experiments and the training data that was created for the purpose of supervised learning. We collected a set of 12 publicly available corpora and one proprietary corpus; statistics of datasets are provided in Table 3.3.

Commoncrawl (CC) [Smith et al., 2013] and News Commentary (NC) [Bojar et al., 2013] corpora were provided in the 2013 shared translation task organized within the workshop on machine translation. TED talks data was released as a part of IWSLT evaluation task [Cettolo et al., 2012a]. ECB, EMEA, EUconst, OpenOffice, OpenSubs 2011, PHP and UN cor-

Corpus	Train	Development	Test
Commoncrawl	78M	12.4K	12.6K
ECB	4.7M	13.9K	14K
EMEA	13.8M	14K	15.7K
EUconst	133K	8K	8.4K
Europarl	52.8M	13.5K	13.5K
P1	5M	35K	14.5K
KDE4	1.5M	12.8K	5.8K
News Comm.	4M	12.7K	65K
OpenOffice	400K	5.4K	5.6K
OpenSubs	156M	16K	15.7K
PHP	314K	3.5K	4K
TED	2.65M	21K	14.6K
UN	1.92M	21K	21K

Table 3.3: Statistics of parallel sets (# of source tokens)

pora are provided as a part of OPUS parallel corpora [Tiedemann, 2012]. The parallel corpora from OPUS were randomly split into training, development and testsets. CC, NC and TED datasets were used as they were provided in the translation tasks of WMT and IWSLT workshops.

Out of 13 different domain datasets we selected 4 datasets randomly: CC, KDE4, TED and UN (in bold Table 3.3), to test our methods.

3.4.2 BLEU-PT vs Cross-Entropy

We compared the overheads of calculating BLEU-PT and a state of the art approach based on Cross-Entropy.[6] We are interested in estimating whether with increasing number of phrase tables the computation of both measures becomes slow or memory intensive.

An advantage of using BLEU-PT is that the indexed FSTs can be stored as a binary on disk. When a source sample comes, we just load the indexed binaries and calculate the BLEU-PT while this cannot be achieved when

[6]We used tmcombine.py script that comes along with the moses package to calculate the mixing coefficients.

we want to calculate cross entropy because we have to do one pass over all the retrieved phrase tables.

Experimental results depicted in Figure 3.7 show that computation of BLEU-PT is fast (160 seconds) while the computation of cross-entropy is slow (42 minutes) when 12 phrase tables are combined with a total size of 4.2GB.

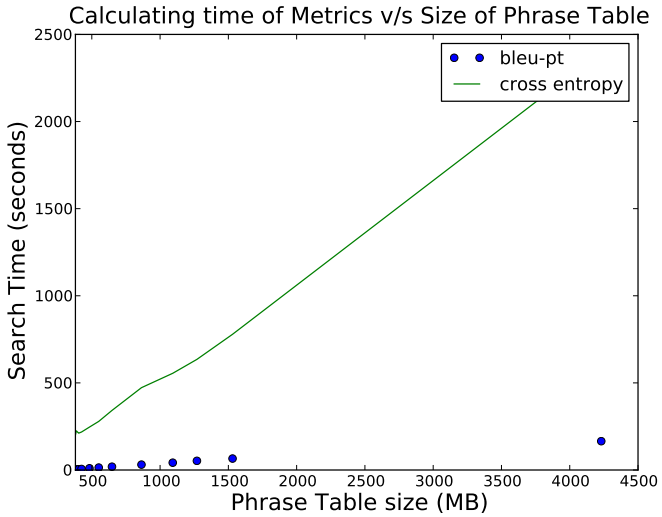

Figure 3.7: BLEU-PT vs. Cross-Entropy

3.4.3 Training data for supervised learning and testing

As mentioned earlier, the estimation of the parameters requires training data containing the tuples of <BLEU-PT, weight vector>. In this section, we explain the collection of such training data. Out of 13 different domain datasets we selected 4 datasets randomly: Commoncrawl, KDE4, TED and UN (in bold in Table 3.3), to perform parameter estimation on them. So, for obtaining the evaluation results on UN, the rest of the resources are used to generate the training data. Our experimental setup can be explained well using the Venn diagram shown in Figure 3.8.

We set one of the four domains as the test domain (in this case, UN)

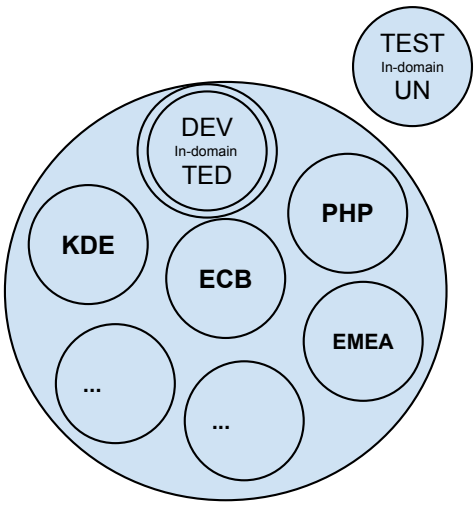

Figure 3.8: Cross domain tuning setup

whose parallel set is not available to us; let this experimental case be called as setup-UN. The training data tuples obtained from the rest of the 12 datasets are used to estimate parameters for the UN domain. From these 12 datasets we perform a round-robin experiment where one by one each dataset is considered as in-domain and the rest as out-domain. In-domain dataset provides the development set and the rest 11 out-domain models are linearly combined to build translation models. In theory, the development set can belong to any domain; for convenience purpose, we use the development sets from the same domains as the models in the library.

In Figure 3.8, for example, the development set from the TED domain is taken as the development set of the multi-model built using the rest 11 models (i.e. excluding TED and UN). This multi-model is built by a weighted linear combination of the 11 out-domain models. The parameters of this multi-model are tuned on the in-domain development set (TED) using MIRA. Simultaneously, we also calculate the BLEU-PT of the linear interpolated model on the source side of the in-domain development set. This provides us the set of tuples of BLEU-PT score and the log linear

weight vector, which is our training data.[7]

So, four sets of experiments are conducted (one each for the four datasets considered for testing), and for each set of experiments, there are 12 training data points. The final evaluation is done by measuring the BLEU score obtained on each test set using the predicted parameter estimates.

Reiterating, our optimizing method is fast, and hence, we are not looking to learn the parameters apriori for all the domains based on a source side of the development set. The goal is to do a fast adaptation by predicting the parameters using statistical models for every new test in a particular domain even in the absence of a parallel development set.

3.4.4 Prediction

To predict the parameters for a new domain, the BLEU-PT of the sample source corpus (UN in our example) is measured with the multi-model built on all the models (all the rest of 12 datasets including the TED model) and then the supervised predictor is applied. In our experiments, we test both parametric and non-parametric methods to estimate the parameters based on the training data obtained using the 12 domains.

3.5 Experiments and Results

3.5.1 Correlation analysis

Before embarking in the actual regression task, we examined the correlation between the similarity values (BLEU-PT) and the various weights in the training data. If there is good correlation between BLEU-PT and a particular parameter, then the linear regressor is expected to fit well and

[7]The weight vector are for the standard phrase based translation features such as forward phrase translation probability, backward phrase translation probability and so on. We have a set of 5 TM features, 6 RM features, 1 LM feature.

then predict an accurate parameter value for a new domain.

For computing the correlation, we use Pearson correlation coefficient (ρ). Figure 3.9 shows the correlation between the feature weights and the BLEU-PT scores. The *tm*'s are the translation model features, *lm* is the language model feature, and *rm*'s are the reordering model features.

We see that there is either a strong positive correlation or a strong negative correlation for most features in both experimental setups shown in Figure 3.9. This validates our hypothesis that optimal parameters for a new test domain can indeed be estimated with good reliability. One can also observe that the correlation level also varies based on the mixture of training models. For example, the correlation of *tm0* feature in setup-UN is much higher than the setup-KDE where there is almost no correlation.

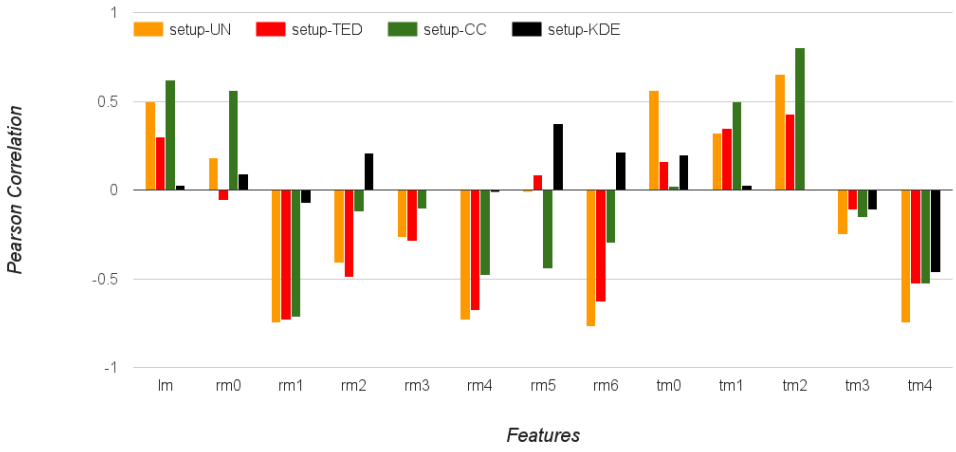

Figure 3.9: Correlation ρ of log linear weights with BLEU-PT for all setups.

In Figure 3.9, it can be noted that $tm0$ (forward phrase conditional probability) and $tm2$ (backward phrase conditional probability) which are shown in previous work [Lopez and Resnik, 2006] to be the two most important features amongst all SMT features in terms of their impact on translation quality, have a high correlation in all setups but KDE. On an

average setup-KDE seems to have almost no correlation with BLEU-PT (except a few features) and setup-UN on the contrary has a high correlation (for most features), hence, it is expected that the linear regression would work better in setup-UN, while the nearest neighbour approach could be a safer bet in for setup-KDE. This point is validated later in the results in Table 3.8. So, in cases where the correlation is low one can resort to the nearest neighbour method.

3.5.2 Systems

All SMT systems were built using the Moses toolkit [Koehn et al., 2007a]. To automatically align the parallel corpora we used MGIZA++ [Gao and Vogel, 2008a]. Aligned training data in each domain was then used to create the corresponding component translation models and lexical reordering models. We created 5-gram language models for every domain using SRILM [Stolcke, 2002] with improved Kneser-Ney smoothing [Chen and Goodman, 1999] on the target side of the training parallel corpora. Log linear weights for the systems were optimized using MIRA [Watanabe et al., 2007a] which is provided in the Moses toolkit. Performance of the systems are measured in terms of BLEU computed using MultEval [Clark et al., 2011a].

		Linear Interpolation		
System	**Train**	**TM(coeff.)**	**RM(coeff.)**	**LM(coeff.)**
in-dom-train	In	N.A	N.A	N.A
mira-bleupt-tm-rm	Out	✓	✓	✗
mira-perp-tm-bleupt-rm	Out	✓(Perp. Min.)	✓	✗
mira-bleupt-tm-rm-perp-lm	Out	✓	✓	✓(LM Perp.)
mira-bleupt-all	Out	✓	✓	✓

Table 3.4: System description for oracle system setups tuned with MIRA on in-domain development sets. ✗ represent log linear interpolation of models while ✓ represents linear interpolation.

We built one *in-dom-train* system where only in-domain training data

System	Parameter Estimation
def-bleupt-all	Default Weights
gen-reg-bleupt-all	Regression
gen-mtl-bleupt-all	Multi-Task
gen-nn-bleupt-all	Nearest Neighbor
top5-reg-bleupt-all	Regression
top5-mtl-bleupt-all	Multi-Task
top5-nn-bleupt-all	Nearest Neighbor

Table 3.5: System description of other systems with the optimizer/predictor is mentioned. *def-bleupt-all* uses default weights from Moses decoder. Multi-model uses normalized BLEU-PT scores.

is taken into account. This system shows the importance of in-domain training data in SMT [Haddow and Koehn, 2012]. Four types of oracle systems are trained on the out-domain training corpus and tuned on the in-domain development data (the four domains we chose to test on are UN, TED, CommonCrawl and KDE4), resulting in four systems for each of the in-domain test sets.[8] All SMT systems are listed in the following and further details provided in Tables 3.4 and 3.5.

1. *mira-bleupt-tm-rm* where mixing coefficients (η) of tm and rm are computed with normalized BLEU-PT score ($normBPT$) but language models are combined in a log-linear way.

2. *mira-perp-tm-bleupt-rm* where η for *tm* are computed with the perplexity minimization method used in Sennrich [2012a]. η for *rm* are computed with normBPT.

3. *mira-bleupt-tm-rm-perp-lm* is same as the first but language models are linearly interpolated with standard perplexity minimization method over the target side of development set.[9]

[8]Oracle systems possess the knowledge of the parallel in-domain data whereas the other systems have no such information.

[9]Linear interpolation of 12 LMs result in one single large LM, thus, one weight. So, a total of 14 weights have to be optimized or predicted.

4. *mira-bleupt-all* same as above but η for LM calculated with normBPT.

5. *def-bleupt-all* same as above but default Moses weights are used instead of tuning the system with MIRA on parallel development set (*a weak baseline*).

6. *gen-<method>-bleupt-all* is a system where all the models are linearly interpolated except the one in test domain. *method* can be either linear regression (*reg*), multi-task Lasso (*mtl*) or nearest neighbor (*nn*).

As mentioned earlier, ideally instead of all the models (such as *gen-** systems) only a subset of all the models closer to the source sample should be taken into account for a **quick** adaptation, so we select the top five domains related to the source sample and interpolate the respective models; we name them as *top5-** systems. Adding more domains would unnecessarily increase the size of the model and add more noise. In the next section we compare the performance of these systems and report the findings.

3.6 Results and Discussion

Table 3.6 presents results of the systems that use an in-domain parallel data. As expected, when an in-domain corpus is used both for training as well as for optimizing the log-linear parameters, the performance is much higher than for the systems built not using in-domain parallel corpus for training [Koehn and Schroeder, 2007b]. We also observe that the use of normalized BLEU-PT for computing mixing coefficients gives comparable performance to using Cross-Entropy. The primary advantage in using BLEU-PT is that it can be computed much faster than Cross-Entropy (as shown in Figure 3.7).

Evidently, normalized BLEU-PT scores as mixing coefficients performs at par with mixing coefficients retrieved by standard perplexity minimiza-

tion method [Bertoldi and Federico, 2009] (comparing systems *mira-bleupt-all* and *mira-bleupt-tm-rm-perp-lm*). One can also use BLEU-PT for LM interpolation in cases where target side in-domain text is not available.

System	UN	TED	CC	KDE
in-dom-train	67.87	29.98	26.62	35.82
mira-bleupt-tm-rm	**44.14**	31.20	17.43	24.25
mira-perp-tm-bleupt-rm	43.56	31.36	17.54	**24.72**
mira-bleupt-tm-rm-perp-lm	43.96	31.85	**18.45**	23.39
mira-bleupt-all	43.66	**32.04**	18.44	23.09

Table 3.6: Comparison of In-Domain system versus the established Oracles in different setups.

System	UN	TED	CC	KDE
mira-bleupt-all	43.66	32.04	18.44	23.09
def-bleupt-all	42.03	30.82	17.97	19.66
gen-reg-bleupt-all	43.27	32.18	17.95	21.05
gen-mtl-bleupt-all	43.35	32.61	18.26	20.67
gen-nn-bleupt-all	42.73	31.04	18.24	21.85

Table 3.7: Performance of generic systems (gen-*) in all setups.

Table 3.7 compares our approach of computing log-linear weights (in the absence of in-domain development set) to the state-of-art weight optimization technique MIRA (which requires an in-domain development set). All the systems presented in this table combine 12 models (except the test domain) and use BLEU-PT for computing mixing coefficients, while the weights are computed using the three techniques that we explored in this work. In case of TED domain, MT performance using multi-task Lasso turns out to be better than the oracle system which has the additional information of parallel in-domain development set (*gen-mtl-bleupt-all* vs *mira-bleupt-all*). Except the KDE domain, in the rest of the cases the results of proposed approach are comparable to that of the oracle. Evidently, the gen-* systems with all learning approaches beat the weak baseline of using default Moses weights for the MT system. This result is notably

good for the MT providers who do not care to tune their system and use default weights in order to provide quick translation services.

System	UN	TED	CC	KDE
def-bleupt-all	42.03	30.82	17.97	19.66
mira-bleupt-all	43.66	32.04	18.44	23.09
top5-reg-bleupt-all	43.39$^{\blacktriangle}$	32.31$^{\blacktriangle}$	18.10	21.54$^{\blacktriangle}$
top5-mtl-bleupt-all	43.56$^{\blacktriangle}$	32.60$^{\blacktriangle}$	18.14	20.91$^{\blacktriangle}$
top5-nn-bleupt-all	42.96$^{\blacktriangle}$	30.89$^{\triangle}$	17.79	22.24$^{\blacktriangle}$

Table 3.8: Comparing the baseline system (def-bleupt-all) and Oracle (mira-bleupt-all) with domain specific multi-model systems trained on top5 domains. $^{\blacktriangle}$and $^{\triangle}$denotes significantly better results in comparison with def-bleupt-all system with p-value < 0.0001 and < 0.05 respectively.

Table 3.8 illustrates the impact of phrase table retrieval on the performance of multi-model. We see that the methods proposed by us perform significantly better than the default weights baseline with an improvement of more than 1.5 BLEU score on an average across the domains. In comparison to oracle system *mira-bleupt-all*, our methods result in an average drop of as little as 0.5 BLEU points across the domains. Among the three approaches for computation of weights, multi-task lasso again performs the best except in setup-KDE where the non-parametric nearest neighbor method performs better. This result lies along the expected lines as multi-task lasso considers the correlation between various features.

These results in Table 3.8 confirm that retrieval helps in building smaller sized multi-models while being more accurate (on an average) at the same time. Phrase table retrieval, thus, can become particularly useful when a multi-model needs to be built from a library of dozens of pre-trained phrase tables of various domains.

The choice of k for our top-k systems is also empirically driven. Figure 3.10 shows the BLEU score curve when we vary the k in top-k systems in the setup-CC. BLEU score curve increases with the number of models up to 5 or 6, then it flattens, which essentially means that selection of $k = 5$ is a good choice. Similar observations were seen in the other setups

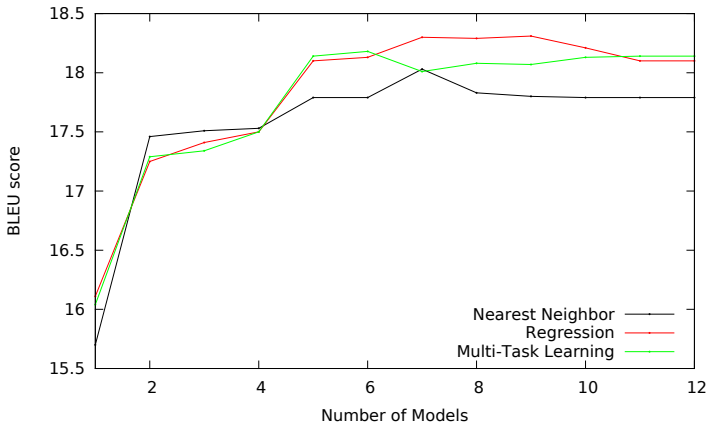

Figure 3.10: BLEU scores when top k models were used to evaluate commoncrawl test set, $k \in 1..12$.

too. For CommonCrawl test set, the top five domains used were Europarl, OpenSubs, NewsCommentary, TED and ECB.

We further analyzed the unexpected BLEU scores in the setup-KDE. Apart from the fact that the feature weights are not well correlated with BLEU-PT in setup-KDE, the results are not good because KDE4 domain is massively different from all the other domains in training. This fact is well reflected in Table 3.9, where the cosine similarity (based on surface matches) of the sample text against the concatenation of source side of the parallel data from the rest of domains is 0.55 while for others it is above 0.7. This means that both the linear regressors were likely extrapolating from, rather than interpolating between, training data-points, which is more difficult and thus, nearest neighbour approach is better in such setup.

Setup	BLEU-PT	Cosine Similarity
UN	47.7	0.74
TED	50.97	0.94
CommonCrawl	31.70	0.87
KDE4	27.64	0.55

Table 3.9: BLEU-PT and cosine similarity scores in different setups.

3.7 Use Case

A simple use case of the proposed domain adaptation approach has been depicted in Figure 3.11. A client X wants to build an in-domain translation system from $\ell 1$ to $\ell 2$ languages but does not have the resources to do so. X has a sample of in-domain text in $\ell 1$ and does not want to pay for the translation cost to create the $\ell 2$ text. X sends out a tender in a MT marketplace for multiple machine translation providers Ys to get a translated bid of the $\ell 1$ text with a sample of translation to judge the quality. Each MT provider (Y) gets the request for the bid from the marketplace, thereon, builds quickly a domain adapted MT system and provides a high quality translation of the sample in $\ell 2$ and bids for the translation of a bigger project. Upon inspecting the bid and the quality of translation sample, X accepts the tender bid from one of the MT providers.

Our method can prove of much convenience to the MT provider Y, as the tender opens only for a short period of time and our method provides a quick solution to building a domain adapted MT system in this short time.

3.8 Conclusion

We present an approach to multi-model domain adaptation in a particularly challenging setting where there is no parallel in-domain data. Parameter estimation without in-domain development set is a problem that, to the best of our knowledge, has not been addressed before. We designed a method for tuning model parameters without parallel development set and validated it through an experimental program for which we compared performance against an array of Oracles and Baselines. The effectiveness of the proposed method empirically supports the findings of Pecina et al. [2012],

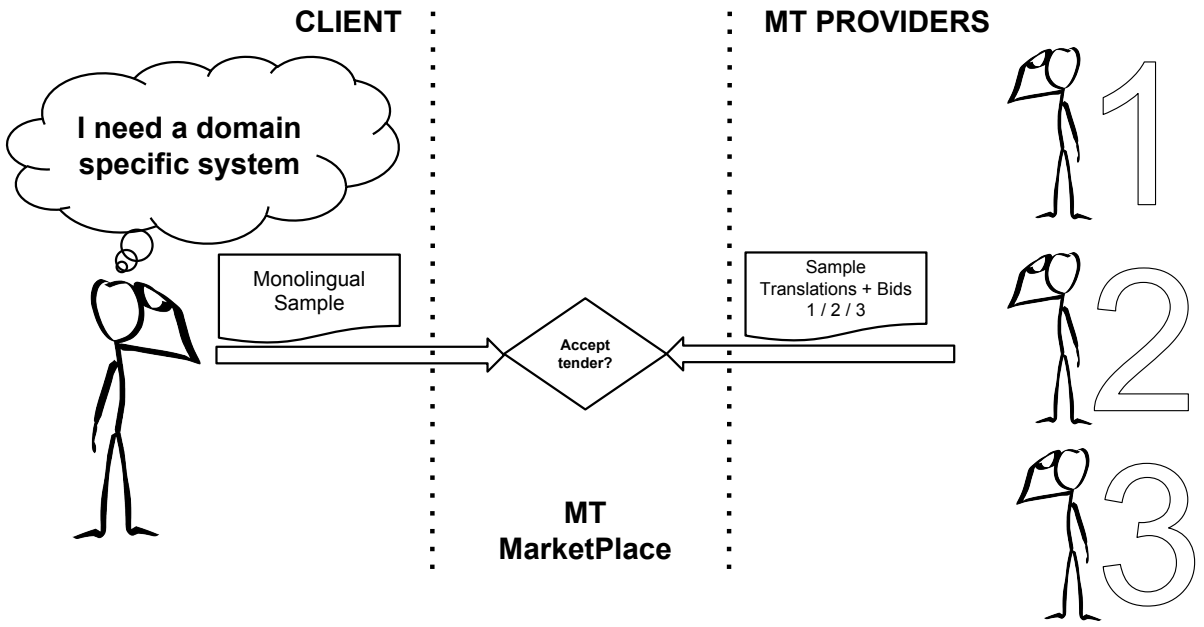

Figure 3.11: Use case

who discovered that the log linear weights largely depend on the **distance** of training domain from the domain on which the models are being optimized on. As a side result, we designed in the process a novel similarity metric between a phrase table and a source sample and implemented it effectively using FSTs. We empirically showed the excellent computation speed of BLEU-PT scores as compared to standard Cross-Entropy measure using standard toolkits. Our results indicate a new theory that one can build a good system for a domain even in the absence of the parallel data in the domain of interest. Re-iterating the words, we believe these advances can drive adoption of specialized MT system in a number of application scenarios.

Chapter 4

Topic Adaptation for e-commerce content

In the previous chapter, we took the assumption that a text coming from one domain (eg. news-wire domain) is homogeneous in nature i.e. the text belongs to same content category. However, this is not always true, documents in the news domain corpus can belong to different content categories such as finance, sports, politics etc. These categories could be referred to as the thematic content or the topics of the document. In statistical machine translation, knowledge about the topics from the input to be translated can be leveraged to adapt the MT system towards training data whose topic is similar to that of the input; this technique is a way of *topic adaptation*.

Domain adaptation and topic adaptation can be seen as complementary methods to cope with the variability between training and testing data in machine translation. Under this perspective, domain modeling typically assumes training data partitioned according to human defined labels, while topic modeling builds on fuzzy clustering of the data and automatically learned labels. The latter type of modeling automatically discovers topic information in a collection of documents or a corpora. Potential advantages of topic adaptation over domain adaptation are that fuzzy clustering can better cope with data sparseness than hard clustering, and that au-

tomatic labels do not require any manual intervention. However, domain adaptation becomes difficult to beat when training data can be effectively and naturally partitioned into in-domain and out-of-domain data.

We exploit the use of the topic modeling approach in this chapter to improve machine translation of item titles found in a large e-commerce inventory via topic adaptation. Item titles are short texts which typically contain brand names that do not have to be translated, and item attributes whose translation often depends on the context. Both of these issues call for robust methods to integrate context information in the machine translation process in order to reduce translation ambiguity.

4.1 Overview

In this work we discuss the application of domain and topic adaptation to an e-commerce online MT system [Guha and Heger, 2014] which we investigated in a research project at eBay Inc. The target is the translation of user queries and all item titles, descriptions, and specifics shown in the search result pages. In particular, our investigation focuses on the translation of item titles, which consist of concise user-generated texts describing each item put on sale. Item titles differ in several ways from text genres typically considered in machine translation research. Titles are usually short texts, of maximum 100 characters, with a simple syntactic structure, and containing brand names, feature values, as well as specific jargon. Two examples of item titles in English and their translations in Italian from a commercial online MT systems follow:

1. "COACH 2014 LE PEANUTS DUFFLE #54 OF 175 MADE"
   → "ALLENATORE 2014 LE ARACHIDI DUFFLE N. 54 DI 175 FATTO"

2. "Delton Locomotive Works Milwaukee Road Long Coach # 7"
   → "Delton Locomotive Works pullman lungo Milwaukee Road # 7"

Translation of item titles poses several challenges [Sanchez and Badeka, 2014], such as:

- the correct rendering of proper names, which can often be confused with common names as in the case of the first example where *COACH* is a brand and should not be translated but copied verbatim, instead it is translated as *ALLENATORE* who is a coach of a team;
- the correct translation of product features, which often depends on the context, as in the case of the second example where *Coach* should be translated as *carrozza* or *vagone*, instead it is translated as *pullman* which is a bus.

From a statistical learning perspective, MT of item titles is also hard because of the large variety of content present in eBay's inventory, based on about 4,000 categories, which are very unevenly populated but also significantly overlapping in terms of linguistic content.

The idea that we follow in this work is to employ topic modeling to better translate English item titles to a foreign language. Since we have a relatively small amount of bilingual in-domain data compared to bilingual out-of-domain data and English in-domain monolingual data, we aim to apply topic adaptation on the bilingual data and to train a topic model on the monolingual data (see Section 4.2). Then, we enrich in-domain and out-of-domain parallel data with topic information and embed this in the translation model of the MT system. At testing time, we infer the topics of the input and use them to dynamically adapt the MT system.

In this work we compare different topic adaptation methods from the recent literature and measure their impact on translation performance with

and without domain adaptation. We report extensive experiments with a Moses-based phrase-based system on the translation of item titles from English into Brazilian Portuguese, and show the impact of topic adaptation both with and without domain adaptation. While for domain adaptation we deploy a state-of-the-art method, for topic adaptation we investigate the use of new sparse features which we compare against other features proposed in the literature.

This approach and the experimental results were, respectively, developed and carried out during an internship at eBay Inc. and were published in [Mathur et al., 2015].

The rest of the chapter is arranged as follows. We first introduce topic modeling applied to our e-commerce content in Section 4.2, after we present our take on topic adaptation in Section 4.3, and finally we report on our experimental set-up and results in Sections 4.4 and 4.5, which are followed by conclusions and future work.

4.2 Topic Modeling

4.2.1 Content

In eBay, machine translation plays an important role to facilitate cross-border trade between sellers and buyers with different languages [Guha and Heger, 2014]. eBay is a marketplace where sellers can advertise their items on the site and buyers can search for the items and then electronically bid for them. To enable a trade between buyers and sellers with different languages, at least four types of texts need to be translated: queries, item specifics, descriptions and item titles.

A query is the information entered by the buyer to search for an item on the eBay's website. Item specifics are details about the item that is sold, such as brand, size type, size, color, and style. These details are

placed at the top of the listing description, in a specific format to provide all essential information about the item for buyers. Description describes the features of the item in detail and is often accompanied by pictures of the product. This work focuses on the translation of item titles, which are concise and usually very informative descriptions of the items put on sale. For instance, the item title:

> *new men's white jekyll & hyde jeans winston designer regular fit shirt size s-xxl*

specifies, in order, the condition, target gender, color, brand, designer, fit, item type, and size of the product. Common challenges in the translation of eBay's user generated content in general, and of titles [Sanchez and Badeka, 2014] and queries [Picinini, 2014] in particular, are the proper rendering of proper names and the translation of words which can have multiple senses, depending on the context in which they appear. For example, the word "age" might have different meanings if context is "baby", "history" or "collectibles". Similarly, the word "bank" might have different meanings if context is "finance" or "river".

The core idea of this work hence is to apply topic modeling to efficiently represent the context of single words or expressions in order to improve the accuracy of their translation by a phrase-based statistical MT system. In the following section, we describe the monolingual data and the topic model that we used for this purpose.

4.2.2 Sampling

The amount of monolingual item titles available to eBay is huge and very unevenly distributed across the 4,000 categories of eBay's inventory. Actually, items in eBay's inventory are classified according to a hierarchical taxonomy. The hierarchy itself contains thirty-four top-level categories

with varying degrees of depth in each category. For example, the top level category "Books" has eleven second-level categories and each of those categories have anywhere from four to thirty categories, with many of these having subcategories as each topic becomes more and more specific. All traded items are placed in the leaves of this hierarchy.

For the sake of experimentation, we performed stratified sub-sampling in order to collect a more balanced and manageable collection of titles. Starting from a collection of billions item titles, after sub-sampling we end up with a collection of 4.3M item titles. We then apply a uniform sampling method, and further sub-sample the data to around 700K item titles, which is actually used to train the topic model. The cascaded sampling is done in order to get a balanced number of samples from all categories within the inventory of eBay.

4.2.3 Models

Topic models can be trained from an arbitrary document or sentence collections with different methods, such as probabilistic latent semantic analysis [Hofmann, 1999], hidden topic Markov models [Gruber et al., 2007], and latent Dirichlet allocation (LDA) [Blei et al., 2003]. For all methods, there are existing software tools that allow to train topic models after specifying the desired number of topics and a few training options. All tools permit then to use a training model to infer a topic distribution for a given sentence.

In our case, we deployed a LDA model trained with the Standford TMT[1] tool [Ramage et al., 2009]. In particular, we worked with a training configuration using 30 topics and 1000 iterations. We also experimented with different number of topics but empirically found 30 to be the optimum number. Moreover, we excluded all item titles with less than 5 words, all

[1] http://nlp.stanford.edu/software/tmt/tmt-0.4/

words occurring less than 3 times, and all words made of less than 3 characters. The pruning steps were performed to exclude from the model item titles providing too little context, words which are too infrequent, and very frequent and short words that do not bear any context information.

In the following table, we report the 15 most relevant words of the first 10 topics trained with the LDA model. As can be seen, some of the topics are easily recognizable, e.g. T01 (toys), T04 (camera and accessories), T05 (photo), TO7 (makeup), T08 (anime) and T09 (fashion), and T10 (hunting and fishing). The other topics look instead combinations of multiple categories, e.g. T02 seems a combination of shipping details of items and fish & aquarium category, and T06 a combination of pet supplies and fashion.

T01	doll high barbie monster little dolls girl lot fashion dress pony and american hair with
T02	free shipping fish aquarium beads tank diy ship round glass pcs wholesale plastic water craft
T03	screen car baby lcd seat glass replacement touch protector cover cloth diaper safety holder glasses
T04	digital control module with remote power sensor switch board lcd meter system arduino air kit
T05	camera lens canon nikon mount digital video with sony gopro dslr camcorder adapter black hero
T06	dress pet dog clothes coat long winter size sweater women puppy warm apparel jacket fashion
T07	nail set art brush hair color makeup gel eye kit cream polish powder tool skins
T08	figure anime movie poster action disney hot mask toys sex toy series japan prop batman
T09	size black jacket mens shoes leather boots large blue womens nwt medium ski white men's
T10	knife steel tool stainless set blade folding with pocket gun black tools handle hunting fishing

Table 4.1: LDA topic model of item titles: top 15 relevant words of the first 10 topics.

Finally, by using the same tool, the topic model is applied to annotate with topics all the sentences in the source side of all the available parallel training data and the evaluation data. As a result, each sentence of the training data and the evaluation set is associated with a topic vector or a distribution. In the next section we describe how the SMT system can dynamically adapt to the topic distributions on the evaluation set at run time.

4.3 Topic Adaptation

4.3.1 Approach

We apply the topic model discussed in Section 4.2.3 to infer the topic distribution (Γ_D) on the source side of all bilingual training data of our statistical MT system (details will follow in Section 4.4.2), on the development set and on the evaluation set (Γ_C). Notice that we inferred the topic distribution on out-of-domain training data at the sentence level rather than at paragraph or document level. This choice is to keep annotation consistent with the training and testing conditions of the topic model, which are performed on item titles.

In this work, we trickle-down topic information from the sentence level to the phrase-pair level. We estimate the conditional probability $P(\tilde{e}|\tilde{f},k)$ of a target phrase $\tilde{e}$ conditioned a source phrase $\tilde{f}$ and the topical information k ($k \in 1 \ldots K$ where K is the total number of topics), through the formula:

$$P(\tilde{e}|\tilde{f},k) = \frac{\sum_d P(k|d) \cdot c(\tilde{e},\tilde{f};d)}{\sum_{\tilde{e}} \sum_d P(k|d) \cdot c(\tilde{e},\tilde{f};d)} \tag{4.1}$$

where $c(\tilde{e},\tilde{f};d)$ is the count of target phrase $\tilde{e}$ and source phrase $\tilde{f}$ being extracted from sentence (our proxy for document) d in the training data D, and $P(k|d)$ (also Γ_{d_k}) is the probability of topic k in sentence d in the training data.

In addition to the probabilities computed with (4.1), we also infer topic vector (distribution) Γ for each phrase-pair ($\Gamma(\tilde{f},\tilde{e})$) extracted from the training data D. Each component of the vector ($\Gamma(\tilde{f},\tilde{e})$) is the average probability $P(k|\tilde{f},\tilde{e})$, which is averaged over topic vectors corresponding to the all the sentences ($d' \subseteq D$) from which the phrase pair ($\tilde{f},\tilde{e}$) was extracted. Equation 4.2 describes the computation of the average topic vector for the phrase pair with a normalization factor Z:

$$\Gamma(\tilde{f},\tilde{e}) = \frac{1}{|Z|}\sum_{d'}\Gamma_{d'}$$
$$where \quad \tilde{f},\tilde{e} \in d' \; and \; d' \subseteq D \tag{4.2}$$

For each phrase pair we only keep the *relevant* topics and set the probability of the other topics to zero. In particular, we compute the perplexity (PP) of the topic distribution and keep only the most probable $\lceil PP \rceil$ topics. To compute perplexity we use the following formulation:

$$H(\Gamma(\tilde{f},\tilde{e})) = -\frac{1}{K}\sum_{k=1}^{K}\Gamma_k(\tilde{f},\tilde{e})log(\Gamma_k(\tilde{f},\tilde{e}))$$
$$PP = 2^{H(\Gamma(\tilde{f},\tilde{e}))} \tag{4.3}$$

In general, the perplexity measure tells how many equally likely topics can be represented with the number of bits of an optimal encoding of a topic distribution. We have empirically observed that the ceiling of the PP ($\lceil PP \rceil$) typically identifies the point in the ranked list of topics after which there is a significant drop in probability. Moreover, PP gives us also a measure of the topic specificity of a phrase pair. In particular, the lower the perplexity of the topic distribution is, the more topic-specific is the translation represented by the phrase-pair phrase. On the contrary, phrase-pairs with high perplexity values reflect translations observed in sentences from many different topics, and so it does not identify translations depending on their context. Therefore, we expect that the translation model will mostly benefit from the topic information only for the phrase-pairs with low perplexity values. For this reason, we add the conditional topic information $P(\tilde{e} \mid \tilde{f}, k)$ and the topic probability for the phrase pair $P(k \mid \tilde{f},\tilde{e})$ to every entry $\tilde{f},\tilde{e}$ of the phrase table only when the perplexity

of $\Gamma(\tilde{f}, \tilde{e})$ is below a given threshold PP_{max} and up to PP topics.

The information entered in the phrase table is exploited to activate feature functions at test time, by combining them with topic information inferred on the input sentence at test time. We denote the input topic information with the vector $\Gamma(c)$, where c stands for context representation. The k^{th} element in the topic vector $\Gamma(c)$ is the probability $P(k \mid c)$. We apply the same topic pruning strategy for the topic vector $\Gamma(c)$ as we apply to the vector $\Gamma(\tilde{f}, \tilde{e})$.

In the following, we present the list of feature functions that we have explored in this work.

4.3.2 Features

1. **Joint Probability**: The topic probability $P(\tilde{e} \mid \tilde{f}, k)$ in itself is not enough to disambiguate between the context. As an example, assume that our English-Portuguese phrase table contains the following two phrase pairs with corresponding probabilities of target given source and topic:

   (a) age ||| idade ||| topic14 0.6 topic04 0.2

   (b) age ||| era ||| topic25 0.5 topic14 0.3

   The two entries show two different translations of the English word **age**, whose probabilities vary according to the context they have been observed with. In particular, the translation **idade** has been observed with *topic14* and *topic04*, while the translation **era** has been observed with *topic25* and *topic14*. However, this information can be properly exploited only once the topic information of the input sentence is known.

Let us assume the following two input sentences and (pruned) context vectors:

(a) *early english bronze* **age** *period blade c. 1600 bc* <topic25 0.9>

(b) *supre hempz* **age** *defying moisturizer- 1 bottle* <topic14 0.7>

Our first feature function activates for each translation option a sparse feature for each topic that *occurs in both the context and the phrase-table* with value:

$$f_1^k(\tilde{e}, \tilde{f}, c) \;=\; -\log P(k|c) \cdot P(\tilde{e}|\tilde{f}, k) \tag{4.4}$$

Going back to the first input sentence of our example, translation of **age** with **idade** would not activate any sparse feature, while translation of **age** with **era** would activate sparse features *topic*25 with score $-log(0.45)$. For the second input sentence, both translations of **age** with **idade** and **era** would activate sparse features *topic*14 with scores $-log(0.42)$ and $-log(0.21)$, respectively. The plain interpretation of this example is that our sparse feature function only rewards the translation **era** for the first sentence, while for the second input sentence it rewards both translations but gives a higher score to the translation **idade**.

Our sparse feature is derived from the feature that Eidelman et al. [2012] proposed for hierarchical phrase-based decoding. This feature has a clear probabilistic interpretation: the product $P(k|c) \cdot P(\tilde{e}|\tilde{f}, k)$ computes the joint translation-topic probability $P(\tilde{e}, k \mid \tilde{f})$ by multiplying the translation probability conditioned to topic with the prior topic probability coming from the context.

2. **Geometric Mean**: The same principle of *joint probability* applies with our second basic feature, but this time with the phrase table an-

notated with posterior probabilities $p(k|\tilde{f}, \tilde{e})$. Again, for each phrase-pair and for topics present both in the phrase table and in the context, our sparse feature takes the product of the topic probability on the input and the topic probability of the phrase pair.

$$f_2^k(\tilde{f}, \tilde{e}, c) \;=\; -\log p(k|c) \cdot p(k|\tilde{f}, \tilde{e}) \tag{4.5}$$

Assuming conditional independence of c and $\tilde{e}, \tilde{f}$ given k, as c is the context in the evaluation set and $\tilde{e}, \tilde{f}$ are occurrences in training set, we can show that the following equation holds true:

$$P(k|c) \cdot P(\tilde{e}|\tilde{f}, k) \simeq P(k|c) \cdot P(k|\tilde{f}, \tilde{e}) \tag{4.6}$$

Proof:

$$\begin{aligned}
P(k|c) \cdot P(\tilde{e}|\tilde{f}, k) &= P(k|c, \tilde{e}, \tilde{f}) \\
&= \frac{P(c, \tilde{e}, \tilde{f}|k) \cdot P(k)}{P(c, \tilde{e}, \tilde{f})} \text{ (Bayes rule)} \\
&= \frac{P(c|k) \cdot P(\tilde{e}, \tilde{f}|k) \cdot P(k)}{P(c)P(\tilde{e}, \tilde{f})} \text{ (Independence assumptions)} \\
&= \frac{P(c|k) \cdot P(\tilde{e}, \tilde{f}|k) \cdot P(k)}{P(c)P(\tilde{e}, \tilde{f})} \cdot \frac{P(k)}{P(k)} \\
&= \frac{P(k|c) \cdot P(k|\tilde{e}, \tilde{f})}{P(k)}
\end{aligned}$$

assuming P(k) as constant, we have

$$P(k|c) \cdot P(\tilde{e}|\tilde{f}, k) \simeq P(k|c) \cdot P(k|\tilde{e}, \tilde{f}) \tag{4.7}$$

The intuitive interpretation behind this feature is to measure the level of each *matched* topic with the geometric mean between the probabilities in the context and phrase table. The feature presented in

Equation 4.5 is equivalent to the log of the geometric mean but a constant factor 0.5 which we assume to be absorbed by the feature weight.

While the previous features are sparse, in the sense that they are computed over single topics thus resulting in K distinct features, the following feature are dense, i.e. a single feature envelopes all topics together. In particular, all features try to measure the similarity between the topic distributions on the input and on each phrase pair.

3. **Crude-Count**: In this feature, we simply count the number of topics active in both input and the phrase pair and normalize this count over the total number of topics in both the context and the phrase-pair. Formally:

$$f_3(\tilde{f}, \tilde{e}, c) \;=\; -\log \frac{(|\{k:\; p(k|c) \times p(k|t,s) > 0\}|)}{(|\{k:\; p(k|c) + p(k|t,s) > 0\}|)} \tag{4.8}$$

Notice that this feature as well as the remaining features are activated for phrase-pair only if there is at least one common topic between context and the phrase-pair.

4. **Cosine Similarity**: This feature computes the cosine similarity between the input topic vector ($\Gamma(c)$) and phrase-pair topic vector ($\Gamma(\tilde{f}, \tilde{e})$) .

$$f_4(\tilde{f}, \tilde{e}, c) \;=\; -\log cos(\Gamma(c), \Gamma(\tilde{f}, \tilde{e})) \tag{4.9}$$

This feature was proposed in [Hasler et al., 2014].

5. **JS Divergence**: This feature computes Jensen-Shannon (JS) divergence between input topic vector ($\Gamma(c)$) and phrase-pair topic vector ($\Gamma(\tilde{f}, \tilde{e})$). JS divergence between two vectors is the average sum of the

KL divergences between them.

$$\begin{aligned} KL_c &= \sum_k \Gamma(c)_k \log \frac{\log \Gamma(c)_k}{\log \Gamma(\tilde{f}, \tilde{e})_k} \\ KL_{\tilde{f},\tilde{e}} &= \sum_k \Gamma(\tilde{f}, \tilde{e})_k \log \frac{\log \Gamma(\tilde{f}, \tilde{e})_k}{\log \Gamma(c)_k} \\ f_5(\tilde{f}, \tilde{e}, c) &= -\log \frac{KL_c + KL_{\tilde{f},\tilde{e}}}{2} \end{aligned} \tag{4.10}$$

This feature was proposed in [Hewavitharana et al., 2013].

6. **Hellinger Divergence**: This feature computes the similarity between the input topic vector ($\Gamma(c)$) and phrase-pair topic vector ($\Gamma(\tilde{f}, \tilde{e})$) with Hellinger's divergence (HD).

$$\begin{aligned} f_6(\tilde{f}, \tilde{e}, c) &= -\log HD(\Gamma(c), \Gamma(\tilde{f}, \tilde{e})) \\ &\text{where} \\ HD(\Gamma(c), \Gamma(\tilde{f}, \tilde{e})) &= \frac{1}{\sqrt{2}} \sqrt{\sum_k \left(\sqrt{\Gamma(c)_k} - \sqrt{\Gamma(\tilde{f}, \tilde{e})_k}\right)^2} \end{aligned} \tag{4.11}$$

This feature was proposed in [Xiao et al., 2012].

7. **Sensitivity:** Finally, we measure how *sensitive* the phrase pairs are to the different topics. We measure the sensitivity of the phrase pairs to the topics by measuring the entropy of the topic vector the pair is associated to. High entropy of a topic vector denotes that the phrase pair is susceptible to multiple topics, which means it occurs in multiple context and hence topic vectors are not useful for any disambiguation. Low entropy of the vector on the other hand denotes that the phrase pair is topic specific and hence, discriminative and useful. The idea is

to penalize the phrase pairs with the topic vectors of high entropy.

$$f_7(\tilde{f}, \tilde{e}, c) \;=\; -\log H(\Gamma(\tilde{f}, \tilde{e})) \tag{4.12}$$

This feature can be computed offline as it does not depend on the context. It was also proposed in [Xiao et al., 2012].

The first two sparse features and the other dense features are linearly combined with standard features used in phrase based decoding. Hence, with the notation $f_1 + f_3 + f_4 + f_5 + f_6 + f_7$ we indicate that the standard translation score computed by the decoder is augmented with:

$$\sum_{k=1}^{K} \lambda_{1,k} f^k(s,t) + \sum_{i=3}^{7} \lambda_i f_i(s,t) \tag{4.13}$$

where the $K + 5$ λ-weights are tuned together with the other weights of the translation model.

4.4 Experiments

4.4.1 Task and Data

We evaluated our topic adaptation approach on the translation of item titles from English to Portuguese (Brazilian). Parallel data used to train, tune and evaluate MT systems comes from various publicly available collections, proprietary repositories and in house translated item titles. In particular, in-house translated items, descriptions, and specifics are here considered as in-domain data while all the rest is regarded as out of domain data. For development and testing purposes we use manually translated item titles for which two reference translations are available. Statistics on the amount of parallel data for each category are given in Table 4.2.

	Train (Out-Domain)	Train (In-Domain)	Dev	Test
Segments	5.28M	336K	1631	1000
Tokens EN	69M	2M	27K	10K
Tokens PT	70M	2M	31K	11.6K

Table 4.2: Statistics of English-Portuguese parallel data.

4.4.2 MT Systems

This section describes the topic adapted MT systems and the two baseline MT systems developed for comparison purposes. All MT systems are built using the Moses toolkit [Koehn et al., 2007b] and the linear weights for all systems are optimized using the k-best batch MIRA implementation provided in the Moses toolkit [Cherry and Foster, 2012]. Performance of all systems are reported in terms of BLEU [Papineni et al., 2002b] and TER [Snover et al., 2006a] scores. Statistical significance tests were conducted using approximate randomization tests [Clark et al., 2011a].

Baseline System In-domain and out-domain parallel data were taken in 10:1 ratio for training the word alignments. Translation models along with operation sequence models [Durrani et al., 2011] were trained using the standard pipeline of Moses. Item titles in general has no specific ordering between words or phrases, thus, we do not employ any lexicalized reordering model in the MT system. The distortion limit was set to 6. On the target side, we built a trigram LM, using KenLM [Heafield, 2011] trained with modified Kneser-Ney smoothing [Chen and Goodman, 1996].

Domain Adapted System The domain adapted system (DA) was built on top of the baseline system. An additional translation model was built using the in-domain data and then the fill-up adaptation method [Bisazza et al., 2011b] was applied to combine the in-domain and out-domain phrase tables. Fill-up simply adds a provenance feature in the phrase table with

a score of 1 if the phrase pair is present in in-domain phrase table and 0 if it is from out-domain phrase table.

Topic Adapted Systems To evaluate the performance of the topic adapted systems using the features functions presented in Section 4.3.2 we followed a component analysis approach. Each basic sparse feature was added to the domain adapted system separately, shortly (1) DA+f1, (2) DA+f2. We built two separate systems because when we tag the topics in the phrase table, we can either set them to a distribution of topics over phrase pair ($P(k|s,t)$) or a target-phrase translation probability given the source phrase and the topic ($P(t|s,k)$).

Then we added the other dense features one by one, resulting in 10 distinct systems i.e. 1) DA+f1+f3 2) DA+f1+f4 3) DA+f1+f5 4) DA+f1+f6 5) DA+f1+f7 6) DA+f2+f3 7) DA+f2+f4 8) DA+f2+f5 9) DA+f2+f6 10) DA+f2+f7. Finally, we also combined all dense features together on top of basic sparse features to build 1) DA+f1+f3+f4+f5+f6+f7 2) DA+f2+f3+f4+f5+f6+f7. To evaluate the impact of topic adaptation independently from domain adaptation we performed the same analysis also with the baseline (BA) system.

4.5 Results and Discussion

4.5.1 Topic Model Analysis

Since, our method of topic adaptation depends on the quality of topic labels inferred on training data, development and evaluation sets; first, we analyze the distribution of topic labels. Figure 4.1 shows the average probability mass for each topic in three sets. The plot shows that topics represented in the training are not always very well covered in the development and evaluation sets. Topics 14, 15 and 26 are very frequent in the training data;

as a result, when we project the topic distribution to phrases in the phrase table, these topics occur in more phrase pairs than any other topic. Thus, as a consequence, these topics should have less discrimination power than other topic features. To validate this hypothesis we also plotted the weights of the topic features f1 after tuning them with k-batch MIRA. In fact, we can observe that the tuning algorithm weighs less the more represented topic features compared to other topic features as shown by green lines in the figure.

An interesting note is that for some topic labels the tuning algorithm assigns negative weights. A possible reason is that there is a topic-translation mismatch between training and development data. This can be explained by the fact that the training data contains both in-domain and out-domain data while the development set contains only in-domain data. Hence, the possibility is that topics mostly occurring in phrase-pairs extracted from out-domain data are being penalized to the advantage of topics mostly occurring in in-domain phrase-pairs.

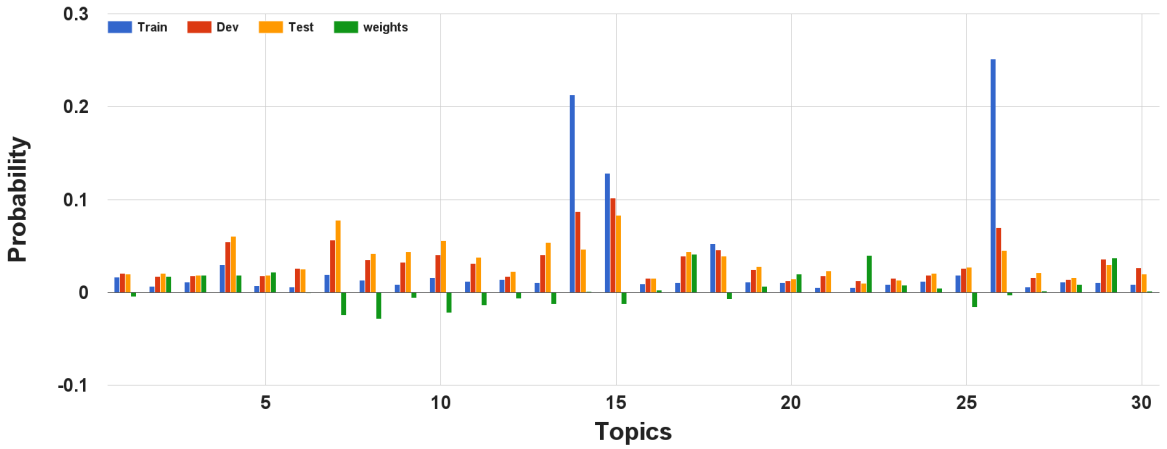

Figure 4.1: Topic distribution on training, development, and test data set. Green bar shows the weight of each topic feature when tuned with MIRA.

4.5.2 MT results

This section presents the results for the baseline, domain adapted and all topic adapted systems. Table 4.3 results shows the impact of adding sparse topic features ($f1$ and $f2$) on top of the baseline (BA) system. Both sparse features improve performance consistently (at least 0.3 gain in BLEU points) over the baseline system and we even observe statistical significant improvements in BLEU score with the Joint Probability feature ($f1$). The Domain Adapted system (DA) however outperforms all other systems in the same table significantly. A reason for this significant gain could be that the training data was partitioned effectively in in-domain and out-of-domain data with the use of an additional in-domain translation model in *fill-up* approach. Since our goal is to improve over domain adaptation, we tested the rest of the features only on top of the domain adapted (DA) system.

System	BLEU	TER
Baseline (BA)	**36.99**	**49.15**
BA + f_1	37.52*	49.09
BA + f_2	37.28	48.93
Domain Adapted (DA)	**39.42***	**48.12***

Table 4.3: BLEU and TER scores of baseline and domain adapted systems on English to Portuguese data set. System performance marked with $*$ show significant different results w.r.t baseline (BA) with p-value < 0.05.

Table 4.4 shows results with the topic adapted system using the *Joint Probability* feature. Individually, the best features which give significant improvements in TER scores over the DA system are the *crude-count*, *cosine similarity* and the *Hellinger's divergence* feature, i.e. DA+f1+f3, DA+f1+f4, DA+f1+f6. Concerning the BLEU scores, only the *cosine similarity* feature is significantly better than the domain adapted but on an average we observe 0.2 BLEU points improvement per feature. We also observe significant gains in TER over the domain adapted system when

System	BLEU	TER
Domain Adapted (DA)	**39.42**	**48.12**
DA + f_1	39.56	48.11
DA + f_1 + f_3	39.66	47.77▲
DA + f_1 + f_4	39.70▲	47.75▲
DA + f_1 + f_5	39.56	47.84
DA + f_1 + f_6	39.66	47.77▲
DA + f_1 + f_7	39.60	47.84
DA + f_1 + f_3 + f_4 + f_5	39.67	47.77▲
DA + f_1 + f_3 + f_4 + f_5 + f_6 + f_7	39.57	47.72▲

Table 4.4: BLEU and TER scores of systems on English to Portuguese data set showing impact of sparse topic feature *Joint Probability* with other dense features against a strong baseline system (DA). System performance marked with ▲show significant different results w.r.t domain adapted system (DA) with p-value < 0.05.

we combine all dense features together (see systems DA+f1+f2+f3+f4+f5 and DA+f1+f2+f4+f5+f6+f7).

The results in Table 4.5 are using the *GeoMean* sparse feature. The component wise analysis shows that the *cosine similarity*, *Hellinger's divergence* and the *Sensitivity* features give statistically significant improvements ($p < 0.05$) over the DA system in terms of TER scores. Average improvements of 0.2 BLEU points are observed across individual feature systems. In terms of BLEU, our best system is the one which combines all the features together without *Hellinger's divergence* achieved statistically significant gains ($p < 0.05$) over the DA system (DA+f2+f3+f4+f5). In terms of TER, we observe statistically significant gains ($p < 0.05$) of 0.34 TER points in the best systems. These systems are the combination of *GeoMean* feature with *Sensitivity* feature (DA+f2+f7) and the one which combines all dense features together with the *GeoMean* feature (DA+f2+f3+f4+f5+f6+f7).

In Table 4.6 we show examples from the evaluation set where our system solved the problems of context disambiguation and proper rendering of proper names. In the first example the source title contains the words

System	BLEU	TER
Domain Adapted (DA)	**39.42**	**48.12**
DA + f_2	39.60	47.86
DA + f_2 + f_3	39.61	47.79
DA + f_2 + f_4	39.60	47.76▲
DA + f_2 + f_5	39.62	47.77▲
DA + f_2 + f_6	39.60	47.78
DA + f_2 + f_7	39.66	47.68▲
DA + f_2 + f_3 + f_4 + f_5	39.71▲	47.77▲
DA + f_2 + f_3 + f_4 + f_5 + f_6 + f_7	39.53	47.68▲

Table 4.5: BLEU and TER scores of systems on English to Portuguese data set showing impact of sparse topic feature *Geometric Mean* with other dense features against a strong baseline system (DA). System performance marked with ▲show significant different results w.r.t domain adapted system (DA) with p-value < 0.05.

endurance which is a brand and **colander** which is the name of a cooking tool. Domain adapted system (DA) incorrectly translates **endurance** as **resistência** while the topic adapted system (TA) correctly took the verbatim translation. In the same sentence, DA translates **colander** as **escorredor** while TA correctly picked the more specific translation **escorredor de macarrão** (colander for pasta). In the second example the source contains: **columbia river crkt** which is a brand name. The DA system erroneously translates **river** as **rio** while the TA system produced the correct verbatim translation of the brand name.

Source	rsvp international 5-qt . endurance colander 1024
DA	rsvp international 5-qt . resistência escorredor 1024
TA	rsvp international 5-qt . endurance escorredor de macarrão 1024
Ref1	rsvp international 5-qt . coador endurance 1024
Ref2	escorredor de macarrão endurance de 5 quartos de galão da rsvp international 1024
Source	columbia river crkt crawford kasper lawks zytel knife !
DA	rio columbia crkt crawford kasper lawks zytel faca !
TA	columbia river crkt crawford kasper lawks zytel faca !
Ref1	faca columbia river crkt crawford kasper lawks zytel !
Ref2	faca canivete columbia river crkt crawford kasper lawks zytel !

Table 4.6: Examples from the domain adapted (DA) and topic adapted (TA) systems.

In the next section we describe a use case of a real life scenario where our topic adaptation methods can be applied.

4.6 Use Case

A buyer in Russia wants to purchase a *toy train* from eBay Russian website (`www.ebay.ru`). As eBay has opened its inventory in US to Russian customers, the search will look for *toy trains* on sale by US sellers. Figure 4.2 describes how search works for Russian queries:

1. Russian buyer types in a Russian query for toy train;
2. eBay SMT system translates the Russian query to English;
3. matching items in US inventory are looked up given the English query;
4. retrieved items have titles and descriptions in English which are then translated by the eBay SMT system;
5. retrieved items are shown to the buyer in Russian.

In this use case, our research focuses on the translation of item titles from Russian to English and improve its translation quality by dynamic topic adaptation.

4.7 Conclusion

An open problem in machine translation is how to effectively handle and incorporate the context information in the translation models that can help the system to properly disambiguate between competing translation alternatives. In this work we presented methods for topic adaptation for phrase-based machine translation that have been experimented in an e-commerce application scenario. In particular, starting from state-of-the-art

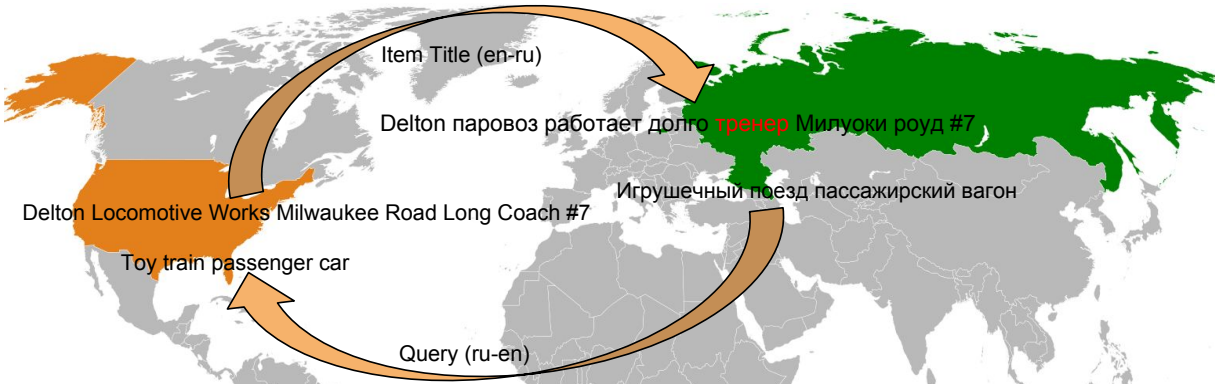

Figure 4.2: eBay's translation service for Russian customers looking for items in US inventory.

LDA topic modeling, we present several feature functions (some of them new and others derived from the literature) that combine topic information integrated in the phrase table with topic information inferred on-the-fly on the input text. We report results on an English-Portuguese translation task of item titles that show consistent and statistical significant improvements through topic adaptation over both a generic baseline MT system and a domain adapted MT system. Our work shows that the use of sparse features permits to identify and cope with topic-translation inconsistencies in the training data, which from one side cannot be avoided when data from multiple and diverse sources are pooled together, but from the other side calls for more refined methods to label such training data with topic labels. Future work will be to provide examples of topic-translation inconsistencies and to investigate feature regularization methods in order to select topic-features with high discrimination power.

Chapter 5

Online User Adaptive MT

In the previous chapter we studied the adaptation methods when the MT systems are not live systems i.e. in *offline state*. In this chapter, we assume that the MT system is live i.e. in *online state*, and now our goal is to adapt the system to the incoming data. We work within the Computer Assisted Translation (CAT) framework to facilitate the translation process with the help of an automatic CAT tool. A CAT tool is a *Translation Memory* (TM) with built-in spell checkers, a dictionary, a concordancer, a *terminology* etc. which helps a translator translate a text. The purpose of CAT tools is to increase the productivity (translation speed) of human translators with the help of these in-built features.

A TM is a database of previously translated text segments by the translators. In the CAT tool, a translator receives suggestions from the TM based on partial matches in the database. Once a new segment is post edited, the source and post-edit pair is stored in the TM database. It is often the case where a TM fails to produce a match because either the domain of translation is drastically different from previously seen domains or the vocabulary/terminology has evolved.

Recent research has led to the integration of CAT tools with statistical machine translation engines. Since MT makes use of a large parallel corpus

to generate statistical models for translation, it has a higher generalization capability than a TM. Thus, MT systems are a good fit in this scenario; moreover, a seamless integration of MT engines in a CAT tool have shown to increase translator's productivity [Federico et al., 2012].

#Sentence		Segment
	Source	... Flex System V7000 Storage Node ...
1	Translation	... Flex System V7000 ... Nodo di memoria ...
16		sistema di ... Archiviazione V7000 Flex nodo
32		... Flex Archiviazione V7000 il sistema ... nodo ...
	Postedit	... Flex System V7000 Storage Node ...

Table 5.1: Example translations from a static SMT system of a source segment occurring multiple times in a technical document in information technology domain, showing the repetitive nature of texts. Consistently the SMT system is wrongly translating the same source segment in successive sentences.

In state-of-the-art CAT tools, the problem is that these integrated MT systems are static in nature; it means that once the MT system is live, it cannot be modified and adapted to the post-edits made by the translators. As an example, technical documentation typically contains a lot of *repetitions* due to the employed writing style and the pervasive use of terminology. In order to provide useful hints to the translator, the MT systems are expected to consistently translate the domain-specific terms. However, if the translator edits the translation suggestion of a technical term in the target text, most of the current MT systems are incapable of learning from those corrections; and when the MT system repeats the translation errors, the productivity of the translators as well as their trust in the MT technology is negatively affected. This *negative* effect of the MT system is contrary to the idea of using it in the first place in the CAT tools, where the system is supposed to **increase** the productivity of translators.

The negative effect of translation can be observed in Table 5.1 where the source fragment ("*Flex System V7000 Storage Node*") is translated differently by a static MT system for sentences number 1, 16 and 32. This

means that the translator would have to repeat the same edits to the suggestions from the MT system for these sentences even when the suggestion was correctly post-edited before. The inconvenience of repeating the same corrections will affect the confidence of translators in MT system and they might end up in disregarding all of its future suggestions. The point is that there is a *need* to adapt the MT system on the fly to the corrections made by the translators.
The current framework of the SMT system in CAT scenario is as follows:

1. the SMT system receives the source segment to be translated (input);
2. it outputs the translation suggestion (predict label);
3. then it receives the post-edit from the translator (correct label).

We need a framework where, when the suggestion does not match the post-edit, the SMT system learns from the mistake. If the SMT system learns from the feedback in the end of the cycle it exactly fits the online learning scenario which is defined by the following steps:

1. A learner (L) receives the input X;
2. L predicts a hypothesis $\hat{Y}$;
3. L receives the correct label Y;
4. L learns from the mistake if $\hat{Y}$ and Y are not equal.

Intuitively, the expected usefulness of online learning for translation is related to the amount of repetitions occurring in the text being translated. The higher the number of repetitions in a document the more the SMT system has chances to translate the terminology consistently through the use of online learning.

In this chapter we explore the following two aspects of online adaptation techniques for statistical phrase-based MT [Koehn et al., 2007a]:

- Adaptation of features (§5.1)
- Adaptation of weights (§5.2)

We alter the current log linear models and introduce additional set of dense and sparse adaptive translation model features in Section 5.1. These features are self-updating as in when the post-edit segment is received as a feedback, the feature tune itself to produce better translations in the future. In addition to the translation model features, we also propose a set of sparse discriminative language modeling features. In Section 5.2, we combine these features with different *weight* adaptation techniques using online learning algorithms such as a max-margin based MIRA [Crammer and Singer, 2003], an adaptive sub-gradient descent AdaGrad [Duchi et al., 2011] and a traditional online learning Perceptron algorithm [Rosenblatt, 1958]. Sections 5.3 and 5.4 present the experiments and the results with the above mentioned adaptation approaches, and are followed by conclusions.

The main ideas of this chapter have been published in [Mathur et al., 2013] and in a Technical Report [Mathur, 2015].

5.1 Online Feature Adaptation

This section focuses on feature adaptation by extending the contemporary translation model with four conventional features (namely direct and inverse lexical and phrase translation features) with an additional adaptive feature, namely, *RecencyFeature*. This is an additional *dense* feature which self-updates according to the corrections at the sentence level. A dense feature shares a single weight for all instances whereas a sparse feature has a weight for each instance.

Next, we propose to add a set of *sparse* features to the existing models. Sparse features are more discriminative in nature because they have more

degrees of freedom, as each sparse feature has a separate weight which allows the model to be more flexible by giving more importance to certain features, less to others. This behaviour is unlike that of a dense feature where all the features are given the same importance as they share the same weight.

5.1.1 RecencyFeature (RF)

First, we introduce a dense *Recency Feature* (RF) which represents a *recency* and a *goodness* measure of each phrase-pair in the translation model. RF feature is a self-learning feature based on Perceptron updates (we use the Perceptron algorithm for updates). Here we assume that the decoder, in translating a source sentence x, provides a n-best list of translation hypotheses. As a matter of fact, it may happen that in the n-best list there are translations which are closer to the post-edit than the 1-best translation. The idea is to promote the hypothesis within the n-best list which are better than the 1-best when matched against the post-edit. This feature works as follows:

1. For each n-best translation *candidate* in the search space, we compute a similarity score against the post-edit;

2. We then compare the similarity score of each *candidate* against the similarity score of the 1-best;

3. If the *candidate* scores worse than the 1-best, we promote the building blocks of 1-best that are not in *candidate* and demote those of *candidate* that are not in 1-best.

4. If the *candidate* scores better than the 1-best, we demote the building blocks of 1-best that are not in *candidate* and promote those of *candidate* that are not in 1-best.

These building blocks are the phrase pairs used to generate the translation candidate. To compute the similarity of the candidates and the post-edit we use sentence level BLEU [Lin and Och, 2004], which is a smoothed variant of corpus level BLEU [Papineni et al., 2001]. The promotion/demotion scheme could be implemented by updating the features values of the phrase pairs used in the *candidate* and *bestHyp* translations. However, these features in the translation models are conditional probabilities and perturbing a subset of them by also preserving their normalization constraints would be computationally expensive. For this reason, we prefer adding an extra *RecencyFeature* to the set.

We call the set of phrase pairs used to generate a *candidate* as $candidate_{PP}$ and the set of phrase pairs used to generate the 1-best as $best_{PP}$. The *RecencyFeature* value of each phrase-pair is initialized to a constant (empirically determined) and is updated according to the Perceptron algorithm [Rosenblatt, 1958]. In particular, the amount by which a current feature value is rewarded or penalized depends on a learning rate α and on the difference between the model scores (i.e. $\Delta\lambda \cdot h$) of *candidate* and *bestHyp* as calculated by the MT system. A sketch of our online learning procedure is shown in Algorithm 1.

Here, *score* is the hypothesis score as computed by the decoder times α equals the *margin* by which the online feature score (RF) of the phrasePair pair is modified. We can observe that the feature scores are unbounded and could lead to instability of the algorithm; therefore, we bound the scores via a sigmoid function:

$$f(x) = \frac{2}{1 + exp(x)} - 1 \tag{5.1}$$

This feature has been proposed in [Mathur et al., 2013] and the rest of the features are published in the technical report [Mathur, 2015]. In the

Algorithm 1: Online Feature Adaptation algorithm

```
1  foreach sourceSegment do
2      bestHyp ← Translate(sourceSeg);
3      pe ← PostEdit(bestHyp);
4      for i ← 1 to iterations do
5          n-best ← Nbest(sourceSeg);
6          foreach candidate ∈ n-best do
7              sign ← sgn (sBLEU(candidate, pe) - sBLEU(bestHyp, pe)) ;
8              diffModelScore ← score(candidate) - score(bestHyp) ;
9              foreach phrasePair ∈ candidate do
10                 if phrasePair ∉ bestHyp then
11                     RF_i ← RF_{i-1} + (α· diffModelScore · sign);
12                 end
13             end
14             foreach PP ∈ bestHyp do
15                 if PP ∉ candidate then
16                     RF_i ← RF_{i-1} - (α· diffModelScore · sign);
17                 end
18             end
19         end
20         bestHyp ← Translate(sourceSeg);
21     end
22 end
```

latter, we propose three classes of sparse features that can be added to the existing set of features. The first feature type introduces phrase-pairs as single sparse features (similar to the dense feature), the second feature type introduces features made of target n-grams and the third is a sparse feature extracted from the "new" phrase pairs from incoming data. In the following sections, we detail the working of the three type of sparse features in the online learning paradigm.

5.1.2 Repetition-Correction Phrase Pair (RCPP)

In this feature, we exploit the availability of the post-edit segment to judge the *goodness* of candidates in the n-best list as follows: each phrase-pair

used by the decoder to generate the *oracle* translation, that is the n-best entry closest to the post-edition, is added as a new sparse feature, scored by the learner with a Perceptron update. This idea is very similar to the one of *RecencyFeature*, the difference is that here, instead of using the distance of *candidate* from 1-best translation, we look for the distance between the *oracle* and the *candidate*.

Since the picked phrase pairs are always contained in the original phrase table, it is possible to assign to each of them an initial value which is the product of the forward and backward phrase translation probabilities. This provides a sort of prior value that can be considered as an overall measure of **translatability** of the phrase pair. The weights of the sparse features are initially set to an empirically determined value (which is tuned on a development set).

The Algorithm 2 outlines the update strategy of $RCPP$. During the scoring of the *candidate*, if its metric value is lower than the 1-best translation (i.e. the candidate is worse in quality than best hypothesis), the $RCPP$ value of phrase pairs in the *candidate* (which are not in the *oracle* translation) is decreased by a $margin = score(oracle) - score(candidate)$ (where $score(\cdot)$ is the model score). On the other hand, if the metric value is higher than 1-best translation, the $RCPP$ value is increased by the same $margin$.

When a translation option with source phrase $\widetilde{x}$ and target phrase $\widetilde{y}$ is being evaluated, we retrieve the value of the feature $h(\widetilde{y}, \widetilde{x}) = RCPP(\widetilde{y}, \widetilde{x})$, where $RCPP$ is the current registered score for the phrase pair. In our implementation, each sparse feature shares the same weight initially (λ_{init}); we empirically determine this weight by tuning the system on a development set (D).

Using a combination of source and target phrase pairs as sparse feature has been studied by [Hasler et al., 2012], but there the function of features is

Algorithm 2: RCPP Adaptation algorithm

```
1  foreach sourceSegment do
2      bestHyp ← Translate(sourceSeg);
3      pe ← PostEdit(bestHyp);
4      for i ← 1 to iterations do
5          n-best ← Nbest(sourceSeg);
6          oracle=findOracle(n-best, pe);
7          foreach candidate ∈ n-best do
8              sign ← sgn (sMetric(bestHyp, pe) - sMetric(candidate, pe)) ;
9              diffModelScore ← score(oracle) - score(candidate) ;
10             foreach PP ∈ candidate do
11                 if PP ∉ oracle then
12                     RCPP_i^PP ← RCPP_{i-1}^PP - (α· diffModelScore · sign);
13                 end
14             end
15             foreach PP ∈ oracle do
16                 if PP ∉ candidate then
17                     RCPP_i^PP ← RCPP_{i-1}^PP + (α· diffModelScore · sign);
18                 end
19             end
20         end
21         bestHyp ← Translate(sourceSeg);
22     end
23 end
```

different. Apart the online learning of weights, we reinforce the position of the *oracle* by providing extra evidence coming from these feature values, while in their approach the phrase pairs themselves are used as binary features.

5.1.3 Post-edit Phrase Pairs (PEPP)

We tested another set of sparse features based on the cache based translation model (CBTM) [Bertoldi et al., 2013]. Bertoldi et al. [2013] propose an online adaptation approach via a caching mechanism that allows to define and dynamically adapt a small and local translation models (with respect to the document at hand). At the time of decoding, the local

model is combined with the global (large background) SMT model in the log linear framework. CBTM is intended to integrate new translation options suggested by the user and reward those approved, with an ultimate goal of translating the successive segments more consistently with the user preferences. When the translators complete editing a translation, a set of "new" phrase pairs is extracted from the partial alignment of the source segment and the post-edit segment using the constrained search algorithm (described in Cettolo et al. [2010]). This model dynamically changes over time in two respects: (i) new phrase-pairs can be inserted with an initial score, and (ii) scores of all current entries decay when new pairs are added.

Instead of leveraging the constrained search approach, we employ a five step process to extract "new" pairs:

1. The decoder generates a translation for the current source segement;
2. Translator post-edits the translation and sends back to the decoder;
3. An enhanced online aligner based on MGiza++ [Farajian et al., 2014] aligns the source and the post edit segment;
4. Then a standard phrase extraction process is followed as in the Moses toolkit to extract new phrase pairs from aligned data;
5. These "new" phrase pairs are then added in the local translation model, thereby adapting the models.

In Bertoldi et al. [2013] this feature is a dense feature but in our work each phrase pair added in the cache becomes a sparse feature. The initial weight of all the sparse feature is set to a default value which is determined on a development set as in the case of $RCPP$ feature. Initially, these "new" phrase pairs may or may not exists in the background translation model, so we cannot assign a starting feature value to them as was done in the case of $RCPP$. Therefore, the initial value of the sparse feature is set to

1 and like CBTM, it decays with time when new phrase pairs are inserted in the cache. In the next section, we propose a similar cache based feature for improved language modeling.

5.1.4 Post-edit N-grams (Ngram)

Every post editor reuses the recurring terminology translations while translating a particular document. For example, in Information Technology documents used in our experiments, the English phrase *Volume Copy* occurs frequently in the source text. Most of the times, it is mistranslated as *Volume Copy* in Italian by a static SMT system, while the actual translation *copia del volume* is buried down in the n-best list. The translator corrects the suggestion to *copia del volume* and expects that next time the system provides *copia del volume* as a translation. For this purpose, as soon as the system receives the post-edit segment with the corrected phrases, we promote the target phrases for future translations. With reference to the above example, the idea is to boost the target n-grams that occur in the corrected phrases (i.e. *copia*, *del*, *volume*, and so on), to indirectly reward the hypotheses of the n-best list which include them. This feature resembles the cache based language model (CBLM) described in [Bertoldi et al., 2013] where the authors add "new" n-grams extracted from the post-edited segments using a constrained decoding approach. These n-grams are added in a cache and their (feature) value decreases as the "new" n-grams are inserted in the cache.

We implemented this feature as follows. When the post edition $\hat{y}$ is returned as a feedback to the system for the source sentence x, n-grams with length from 1 to 4 in $\hat{y}$ are inserted in a dynamic vocabulary V. Initially, V is empty; it is gradually populated as new post-editions are made available. During processing, when the decoder score a translation option and if its target phrase is present in the dynamic vocabulary, the

corresponding sparse features activated with feature value of 1; otherwise, their value is set to 0. This set of features is different from that of CBLM in two aspects: 1) these are sparse features, i.e. each feature is independent and has its own weight and 2) the value of CBLM feature decays as the new target phrases are inserted in the cache while *Ngram* feature is a binary value.

Ngram features are even more valuable if used in conjunction with the cache based phrase pairs (empirically proven later in Section 5.4.2). When PEPP scores translation options that are not present in the background translation model (or background model provides with a low score for), the *Ngram* feature can score the target phrases of such translation options, likely better that the background language model.

5.2 Online Weight Adaptation

In addition to adapting the features values in online adaptation scenario, we also adapt feature weights of the log linear combination in an online fashion. In particular, after translating each sentence we can adapt these parameters depending on the quality of the last translated segment. A commonly used algorithm in this online paradigm for tuning of parameters is Margin Infused Relaxed Algorithm (MIRA).

5.2.1 Margin Infused Relaxed Algorithm (MIRA)

MIRA is an online algorithm for multiclass classification that updates the parameter of a given model in accordance to a loss incurred due to an incorrect classification. MIRA for structured prediction was proposed by Taskar et al. [2005] and later Watanabe et al. [2007b] reformulated MIRA for discriminative training of machine translation. The algorithm is ultraconservative, meaning that the update of the current weight vector is the smallest

possible value satisfying the constraint; i.e. the variation incurred by the objective function must not be larger than the variation incurred by the model. This variation is coupled with the following *loss* function (ℓ) based on the complement of any sentence level similarity metric (*sMetric*):

$$\ell_j = sMetric(y_O) - sMetric(y_j) \tag{5.2}$$

where y_O is the *oracle* translation (closest translation to the post-edit at position O in the n-best list) and y_j is the jth translation *candidate* being processed inside an n-best list. In our work, we experiment with two different metrics

1. a sentence level similarity metric BLEU (sBLEU) [Lin and Och, 2004, Nakov et al., 2012] and

2. a sentence level error metric TER (sTER) [Snover et al., 2006b].

Since the latter is an error measure, we use 1-sTER as the sMetric. As said before, weights are updated such that the loss (ℓ_j) is not larger than the difference between the scores given by the model ($\mathbf{w}^T \Delta h_j$). Formally:

$$\ell_j \leq \mathbf{w}^T \Delta h_j \tag{5.3}$$

where, j ranges over all candidates in the n-best list, $\Delta h_j = h_O - h_j$ is the difference (vector) between the feature vectors of the oracle (h_O) and the candidate (h_j) and w is the current weight vector. We can run MIRA until it converges or for a fixed number of iterations. In our case we limit

them to a maximum of 300. The update step in MIRA for i^{th} iteration is:

$$\begin{aligned} \mathbf{w}_i = \arg\min_{\mathbf{w}} \frac{1}{2}||\mathbf{w} - \mathbf{w}_{i-1}||^2 + C\sum_j \xi_j \\ \text{subject to} \\ \mathbf{w}_{i-1}^T \Delta h_j + \xi_j \geq \ell_j \ \text{ and} \\ \xi_j \geq 0 \ \forall \ j \in \{1 \ldots n\} \end{aligned} \tag{5.4}$$

Here, C is a parameter to control the size of an update. ξ is a non-negative *slack variable* and is usually a very small value.The Lagrangian dual form of the Equation 5.4 is:

$$\begin{aligned} \max_{\alpha(\cdot)\geq 0} -\frac{1}{2}||\sum_j \alpha_j \cdot \Delta h_j||^2 + \sum_j \alpha_j \ell_j - \sum_j \alpha_j \mathbf{w}_{i-1}^T \Delta h_j \\ \text{subject to} \qquad \sum_j \alpha_j \leq C \end{aligned} \tag{5.5}$$

where α_j are the Lagrangian multipliers for the j^{th} candidate. This maximization step leads to a quadratic programming (QP) problem and to the weight vector update:

$$\begin{aligned} \mathbf{w}_i = \mathbf{w}_{i-1} + \sum_j \alpha_j \cdot \Delta h_j \\ \textit{where} \\ \alpha_j = \min\left\{C, \frac{\ell_j - \Delta h_j \cdot \mathbf{w}}{||\Delta h_j||^2}\right\} \end{aligned} \tag{5.6}$$

We can determine the Lagrangian multipliers α at each iteration by applying a QP-solver such as Hildreth optimizer [Censor and Zenios, 1997].

The following steps (and Algorithm 3) gives an overview of how MIRA is implemented for updating the weights during decoding.

1. The decoder translates the source segment;

2. Translator post-edits the translation;

3. Decoder finds the oracle translation within the n-best list;

4. Loss (ℓ_j) and the feature difference vector (Δh_j) is computed for j^{th} candidate in the n-best list;

5. If MIRA's constraint (Equation 5.4) is broken then the update is made for j^{th} candidate;

6. To converge, MIRA undergoes up to I iterations;

7. Decoder uses updated weights to translate the next source segment.

Algorithm 3: Margin Infused Relaxed Algorithm for Moses

$\mathbf{w}_cur$ = Initial Weights;
foreach *sourceSegment* **do**
  `bestHyp` ← Translate(*sourceSeg*, $\mathbf{w}_{cur}$);
  `pe` ← PostEdit(`bestHyp`);
  `n-best` ← Nbest(*sourceSeg*, $\mathbf{w}_{cur}$);
  $w_{cur_1} \leftarrow w_{cur}$;
  for $i \leftarrow 1$ **to** I **do**
    `oracle`=findOracle(`n-best`, pe);
    foreach $candidate_j \in$ `n-best` **do**
      ℓ_j = sMetric(`oracle`) - sMetric($candidate_j$);
      $\Delta h_j = h_{oracle}$ - $h_{candidate_j}$;
      if $\Delta h_j \cdot \mathbf{w}_{cur_{i-1}} + \xi_j \geq \ell_j$ **then**
        $\alpha_j \leftarrow$ Optimize(ℓ_j, h_j, $\mathbf{w}_{cur_{i-1}}$, C);
        $\mathbf{w}_{cur_i} \leftarrow \mathbf{w}_{cur_{i-1}} + \eta \cdot \sum_j \alpha_j \Delta h_j$;
      end
    end
    `n-best` ← Nbest(*sourceSeg*, $\mathbf{w}_{cur_i}$);
  end
  $w_{cur} \leftarrow w_{cur_I}$;
end

5.2.2 Sparse weight adaptation

In the online learning with sparse features, the overall procedure entails a feature adaptation step for the sparse features (*RCPP*, *Ngram*, *PEPP*) and a weight adaptation step for the dense and sparse weights with MIRA. These steps are sketched in Algorithm 4.

Algorithm 4: Online Adaptation for Moses

```
1  foreach sourceSegment do
2      for i ← 1 to iterations do
3          bestHyp ← Translate(sourceSeg, w_{i-1});
4          pe ← PostEdit(bestHyp, w_{i-1});
5          n-best ← Nbest(sourceSeg, w_{i-1});
6          oracle=findOracle(n-best, pe);
7          // Feature Update
8          // Weight update
9      end
10 end
```

Since both learning updates in Algorithm 4 aim at promoting the oracle to a better position, we believe it is better to merge the two updates into one single value. Therefore, we designed a protocol sketched in Algorithm 5 whose main stages are:

1. The sparse features (*sparsefeats*) undergo a Perceptron update with a learning rate γ;
2. MIRA updates the weights of sparse features ($\mathbf{w}^{PP}$);
3. The value of each sparse features is updated by taking the product with its corresponding weight;
4. All the weights of the sparse features are reset to 1.

So doing, the feature value incorporates both the information extracted from the current sentence with the MIRA update and the past knowledge of the Perceptron feature update.

Algorithm 5: Modified Sparse Online Adaptation

```
1  foreach sourceSegment do
2      for i ← 1 to iterations do
3          bestHyp ← Translate(sourceSeg, w_{i-1});
4          pe ← PostEdit(bestHyp, w_{i-1});
5          n-best ← Nbest(sourceSeg, w_{i-1});
6          oracle=findOracle(n-best, pe);
7          // Perceptron update
8          Lines 7-20 from Algorithm 2
9          // MIRA update
10         Lines 8-15 from Algorithm 3
11         // Update the feature values
12         foreach PP ∈ sparsefeats do
13             sparsefeats_i^PP ← sparsefeats_i^PP × w_i^PP ;
14             w_i^PP ← 1 ;
15         end
16     end
17 end
```

5.3 Experiment 1

Since, this chapter contain details from two papers, we decided to divide the experiments into two sections. This section details the experiments done with *RecencyFeature* and *MIRA* as published in Mathur et al. [2013]. The next section details the experiments done with the sparse features and sparse weight adaptation as described in Mathur [2015].

5.3.1 Datasets

First, we compare our Recency feature with weight adaptation method (Sections 5.1.1 and 5.2.1) with a state-of-the-art stream based adaptation method [Levenberg et al., 2010] (briefly discussed in the Incremental learning section of Chapter 2). We run evaluations on two datasets from different domains, namely Information Technology (IT) and TED talks, and different language pairs. The IT domain dataset is proprietary; it involves

the translation of technical documents from English to Italian and has been used in the field test carried out under the MateCat [Federico et al., 2014]. Experiments are also conducted on English to French TED talks dataset [Cettolo et al., 2012b] to assess the robustness of the proposed approaches in a different scenario and to provide results on a publicly available dataset for the sake of reproducibility. The training, development (`dev2010`) and evaluation (`tst2010`[1]) sets are the same as used in the IWSLT last evaluation campaigns. In experiments on TED data, we assume that the reference translations are post-edit segments, even if they were actually generated from scratch.

The extent of usefulness of online learning highly depends on the amount of repetition in the text. A reasonable way to measure the quantity of repetition in each document is through the *repetition rate* [Cettolo et al., 2014]. It computes the rate of non-singleton n-grams, n=1 ... 4, averaging the values over sub-samples `S` of thousand words from the text, and then combining the rate of each n-gram to a single score by using the geometric mean. Equation 5.7 shows the formula for calculating the repetition rate of a document, where `dict(n)` represents the total number of different n-grams and n_r is the number of different n-grams occurring exactly `r` times:

$$RR = \left(\prod_{n=1}^{4} \frac{\sum_S dict(n) - n_1}{\sum_S dict(n)} \right)^{1/4} \tag{5.7}$$

Statistics of the parallel sets and their repetition rate on both sides are reported in Table 5.2.

It can be noted that the repetition rates of IT and TED sets are significantly different, particularly high in IT documents, much lower in the TED talks.

[1] As the size of evaluation set in TED data is too large with respect to the current implementation of our algorithms, we performed evaluation on the first 200 sentences only.

Domain	Set	#srcTok	srcRR	#tgtTok	tgtRR
$IT_{en \rightarrow it}$	Train	57M	na	60M	na
	Dev	3.3k	12.03	3.5k	11.87
	Test	3.3k	15.00	3.3k	14.57
$TED_{en \rightarrow fr}$	Train	2.6M	na	2.8M	na
	Dev	20k	3.43	20k	5.27
	Test	32k	4.08	34k	3.57

Table 5.2: Statistics of the parallel data along with the corresponding repetition rate (RR).

5.3.2 Systems

The SMT systems were built using the Moses toolkit [Koehn et al., 2007a]. Training data in each domain was used to create translation and lexical reordering models. We created a 5-gram language model using IRSTLM [Federico et al., 2008] with improved Kneser-Ney smoothing [Chen and Goodman, 1998] on the target side of the training parallel corpora. The log linear weights for the baseline systems are optimized using MERT [Och, 2003a] provided in the Moses toolkit. To counter the instability of MERT, we averaged the weights of three MERT runs in each case. Performance of the systems are measured in terms of BLEU and TER [Snover et al., 2006b] computed using the MultEval script [Clark et al., 2011b]. Since the implementations of standard Giza and of incremental Giza combined with dynamic suffix arrays are not comparable, we constructed two baselines, a standard phrase based SMT system and one based on incremental Giza baseline [Levenberg et al., 2010]. Details on experimental SMT systems we built follow.

Baseline This system was built on the parallel training data for each domain. We run 5 iterations of model 1, 5 of HMM [Vogel et al., 1996], 3 of model 3, 3 of model 4 [Brown et al., 1993] using MGiza [Gao and Vogel, 2008b] toolkit to align the parallel corpus at word level. Translation and reordering models were built using Moses, while log-linear weights were

optimized with MERT on the corresponding development sets. The same IT baseline system was used in the field test of MateCat and the references in the IT data are actual postedits of its translation.

Incremental Baseline We trained alignment models with incGiza++[2] with 5 iterations of model 1 and 10 iterations of the HMM model. To build incremental baselines, we used dynamic suffix arrays as implemented in Moses which allow the addition of new parallel data during decoding. In the incremental baseline, once a sentence of the test set is translated, the sentence pair (source and target post-edit/reference) along with the alignment provided by incGiza are added to the models.

Online learning systems We created several online systems on top of the two aforementioned baseline systems:

1. +O employ the *RecencyFeature* with Perceptron updates;
2. +O+NS as (1) but feature scores normalized with the sigmoid function;
3. +W weights are updated with MIRA;
4. +O+W is a combination of feature and weight adaptation;
5. +O+NS+W as system 4) but with normalized feature score.

In the online learning system there are three additional parameters: a weight for the online feature, a learning rate for features used in the Perceptron update, and a learning rate for feature weights used in MIRA. These additional parameters were optimized by maximizing the BLEU score on

[2]http://code.google.com/p/inc-giza-pp/

the devset and on top of already optimized feature weights. For practical reasons, optimization of the parameters was run with the Simplex algorithm [Nelder and Mead, 1965].

5.3.3 Results

Tables 5.3 and 5.4 collect results by the systems described in Section 5.3.2 on the IT and TED translation tasks, respectively.

System	Bleu			TER		
	1 Iter	5 Iter	10 Iter	1 Iter	5 Iter	10 Iter
Baseline	38.46	-	-	39.98(1.35)	-	-
+O	39.88	41.22	41.16	38.69	37.78	38.37
+O+NS	39.91	40.54	40.71	38.67	38.21	38.17
+W	39.76	38.16	37.57	38.58	39.53	39.93
+O+W	41.23	40.29	29.36	37.53	38.03	49.08
+O+NS+W	41.19	43.07	**45.13**$^{\blacktriangle}$	37.60	36.43	**34.53**$^{\blacktriangle}$
Incremental Baseline	28.48	-	-	49.23	-	-
+O	29.34	27.80	27.52	47.86	48.20	51.01
+O+NS	28.69	29.68	29.36	48.21	47.51	47.92
+W	28.25	27.68	27.57	49.05	48.74	48.10
+O+W	29.36	29.94	25.95	47.15	46.56	50.31
+O+NS+W	29.76	30.28	**30.83**$^{\blacktriangle}$	46.62	45.60	**46.54**$^{\blacktriangle}$

Table 5.3: Result on the IT domain task (EN>IT). Baseline is a standard phrase based SMT system, +O has the recency feature, +NS normalizes the feature, +W has online weight adaptation. Statistical significance differences are shown by $^{\blacktriangle}$from the corresponding baseline systems.

In Table 5.3, the online system (1st block "+O+NS+W" system with 10 iterations of online learning) shows significant improvements, over 6 BLEU points absolute above the baseline. In this case the recency feature can clearly take advantage of the high repetition rates observed in the IT dev and test sets. Similarly, in the second block, the online system (2nd block "+O+NS+W" with 10 iterations of online learning) outperforms IncGiza baseline too. It is interesting to note that by continuously updating the baseline system after each translation step, even the plain translation models are capable to learn from the correction in the post-edited text.

Figure 5.1 depicts learning curve of *Baseline* system, "+O+NS" (referred as the system with normalized *RecencyFeature*) and "+O+NS+W" (referred as the system with *RecencyFeature* and *MIRA* weight updates). We plotted incremental BLEU scores after translation of each sentence, thereby the last point on the plot shows the corpus level BLEU on the whole test set.

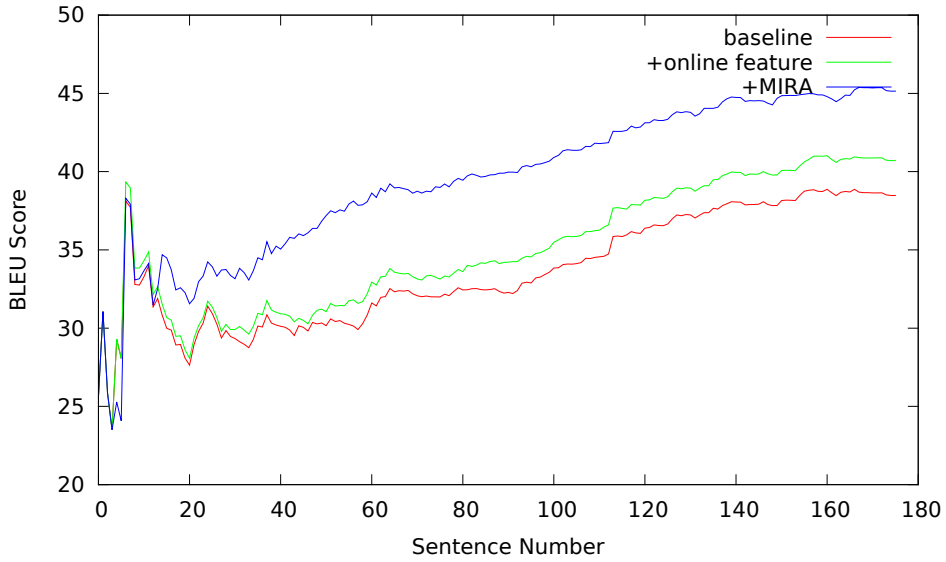

Figure 5.1: Incremental BLEU vs. evaluation test size on the information-technology task. Three systems are tracked: Baseline, +recency feature, +MIRA

In Table 5.4, from the first block we can observe that online learning systems perform only slightly better than the baseline systems, the main reason being the low repetition rate observed in the evaluation set (as shown in Table 5.3.1). The positive results observed in the second block ("+O+W" with 10 iterations) are probably due to the larger room for improvement available for translation models implemented with dynamic suffix arrays, as they only incorporate 3 features instead of 5. Sometimes, online learning systems show worse results with higher numbers of iterations, which seems due to over-fitting. It is also interesting to notice that after optimization the weight value of the recency feature was 0.509 for the

System	Bleu			TER		
	1 Iter	5 Iter	10 Iter	1 Iter	5 Iter	10 Iter
Baseline	22.18	-	-	58.70	-	-
+O	22.17	21.85	21.51	58.75	59.22	60.48
+O+NS	21.97	22.37	22.24	58.86	58.75	59.09
+W	22.39	21.44	21.00	58.96	58.73	58.71
+O+W	22.33	22.11	21.54	58.63	58.31	58.70
+O+NS+W	22.34	22.09	21.62	58.60	58.48	58.40
Incremental Baseline	15.04	-	-	72.64	-	-
+O	15.30	15.47	15.86	72.33	71.68	71.09
+O+NS	15.21	15.48	15.48	72.19	72.06	71.65
+W	14.81	14.61	14.73	73.03	74.69	74.28
+O+W	15.08	15.59	**16.42**$^{\blacktriangle}$	72.55	70.98	**70.07**$^{\blacktriangle}$
+O+NS+W	15.09	15.64	**16.15**$^{\blacktriangle}$	72.57	71.13	**70.61**$^{\blacktriangle}$

Table 5.4: Result on the TED talk task (EN>FR). Baseline is a standard phrase based SMT system, +O has the recency feature, +NS normalizes the feature, +W includes online weight adaptation. Statistical significance differences are shown by $^{\blacktriangle}$from the corresponding baseline systems.

IT task and 0.072 for the TED talk task. This confirms the different use and potential assigned to the recency feature by the SMT systems in the two tasks.

5.4 Experiment 2

5.4.1 Datasets

In Experiment 2, we use the same IT dataset as used in Experiment 1. We also evaluated on two other datasets, WAT collection (Ro→En) and Legal documents (En→Es). The WAT data were released for the word alignment shared task proposed by an ACL 2005 workshop Martin et al. [2005] and contain newspapers, Orwell's 1984 novel and the Romanian Constitution. Legal data as the name suggests contain legal documents and is taken from the JRC-acquis corpus [Steinberger et al., 2006]. Table 5.5 gives the statistics about the latter two datasets used in the experiment.

Task	Set	#srcTok	src-RR	#tgtTok	tgt-RR
$\text{WAT}_{Ro \rightarrow En}$	Train	1M	NA	1M	NA
	Dev	4.3k	6.10	4.6k	7.95
	Test	5.4k	9.71	5.6k	10.18
$\text{Legal}_{En \rightarrow Es}$	Train	55M	NA	58M	NA
	Dev	4.3K	24.09	4.3K	24.46
	Test	2.5K	20.68	2.7K	20.70

Table 5.5: Statistics of the parallel data with their repetition rates. NA stands for Not Applicable.

5.4.2 Systems and Results

In this section we give details of the systems we built and report the results on TER and sBLEU scores of translated outputs over the post-edits. Significance test outcomes with p-value < 0.01 are reported along with the results.

Baselines First of all, let us compare performance of baseline systems, which represent the references for successive experiments. Table 5.6 collects TER and sBLEU scores on the three test sets of four baselines: *Bsl* is the static system whose models are straightforwardly trained on available training corpora. *OriOl* is the best performing online adaptive system in Mathur et al. [2013]. Finally, the system named as *Stc-Rcpp* is the *Bsl* enhanced with the *static* sparse feature function RCPP (Section 5.1.2); static means that its values and weights are not modified during decoding but set respectively to:

- the initial values as mentioned in Section 5.1.2
- the weights are estimated by the means of the Simplex algorithm on the development set.

In general, the caching mechanism (system *Cache*) improves over the baseline on all tasks in terms of at least one metric. Performance on WAT tasks, where TER improves a lot while sBLEU degrades a bit, deserves a

System	IT		Legal		WAT	
	TER	sBLEU	TER	sBLEU	TER	sBLEU
Bsl	39.00	41.44	49.70	37.29	86.87	18.25
Bertoldi et al. [2013]	35.75▲	45.15	50.98▲	39.41	69.26▲	17.68
Mathur et al. [2013]	38.37	43.48	51.53▲	35.42	77.12▲	18.73
Stc-Rcpp	38.96	41.02	50.00	36.81	85.92	18.75

Table 5.6: TER and sBLEU scores of baseline/reference systems on all test sets. Statistical significance differences with $p < 0.01$ are shown by ▲from the reference Bsl system.

comment: *Cache* generates too short sentences (brevity penalty of 0.85), probably because the development set is too small to be really representative; therefore, sBLEU is penalized much more than TER.

The other technique from the literature we built on, that is the online learning algorithm (system *OriOl*), significantly outperforms system *Bsl* on WAT but not on IT. On Legal, it turns out that the *OriOl* system is significantly worse than the *Bsl*.

Finally, the fact that the system *Stc-Rcpp* does not improve over *Bsl* interestingly indicates that the introduction of sparse features must be done with care. Evidently, the offline estimation of RCPP values and of the single weight common to all of them is not effective. In fact, in the next paragraph we will see at what extent the online learning and tuning of sparse features allows to definitely outperform the baseline.

Online and modified sparse online learning Here, the original and the modified online learning algorithms, involving just one kind of sparse features (RCPP), were applied to the standard baseline *Bsl*; the resulting systems are named *Bsl-OriOl-Rcpp* and *Bsl-ModOl-Rcpp*, respectively. Table 5.7 shows performance of the three SMT engines. In general, in both learning schemes, the RCPP feature function improves the baseline performance by a significant amount on Legal and WAT tasks but no significant improvement is observed on IT domain in TER. However, sBLEU improves by a

margin of 1.2 points. It has to be noted that the online learning permits to overcome the problems of *Stc-Rcpp*, as commented in the end of the previous paragraph.

System	IT		Legal		WAT	
	TER	sBLEU	TER	sBLEU	TER	sBLEU
Bsl	39.00	41.44	49.70	37.29	86.87	18.25
Bsl-OriOl-RcppNgram	38.87	42.40	48.72▲	37.94	67.95▲	21.80
Bsl-ModOl-RcppNgram	38.97	42.64	48.60▲	37.57	72.00▲	20.81

Table 5.7: Impact of online learning. Statistical significance differences with $p < 0.01$ are shown by ▲from the reference Bsl system.

Post-edit Ngrams In the previous paragraph, we measured the contribution by the RCPP sparse feature; the goal of experiments reported here is to quantify the impact of the other set of sparse features, i.e Ngram. Table 5.8 provides figures of the standard baseline system equipped either with the original or the modified online learning algorithm, with or without Ngram, in addition to RCPP. In this setup, Ngram does not really boost performance of any tested system. In the next paragraph, we will see if pairing Ngram with the caching mechanism, as suggested in Section 5.2, yields any improvement.

System	IT		Legal		WAT	
	TER	sBLEU	TER	sBLEU	TER	sBLEU
Bsl-OriOl-Rcpp	38.87	42.40	48.72	37.94	67.95	21.80
Bsl-OriOl-RcppNgram	38.63	42.79	48.72	37.60	71.26▲	20.58
Bsl-ModOl-Rcpp	38.98	42.64	48.60	37.57	72.00	20.81
Bsl-ModOl-RcppNgram	38.86	42.56	49.56▲	36.73	70.59▲	20.86

Table 5.8: Effect of adding Ngram features to online learning algorithms. Statistical significance differences with $p < 0.01$ are shown by ▲from the corresponding reference systems i.e. Bsl-OriOl-Rcpp and Bsl-ModOl-Rcpp.

Post-edit phrase pairs Finally, we evaluated the impact of using the methods proposed in this chapter on top of the best performing reference system,

i.e. the cache-enhanced one. Table 5.9 summarizes evaluation scores of the baseline and its versions equipped with either the original or the modified online learning algorithm, in addition to using just one or both the sparse features. On IT and WAT tasks, the baseline is definitely outperformed by the other systems, among which those employing the modified online learning are preferable to the two with the original scheme. On the Legal domain, only the TER is improved, while a quite large degradation of sBLEU is observed. Concerning the contribution of the Ngram feature function, again in two out of three tasks (IT and WAT) it benefits from the pairing with cached models: we think that despite the negligible contribution on the Legal domain, these results support the intuition at the end of Section 5.1.4 about the importance of assisting Ngram with caching.

It is worth noticing that both metric scores in IT domain, and the TER score in WAT task of cache-equipped online learning systems are by far the best of our experiments: focusing on TER, *Cache-ModOl-RcppNgram* improves *Bsl* by 15.2% on IT (33.08 vs. 39.00) and by 24.3% on WAT (65.73 vs. 86.87).

System	IT		Legal		WAT	
	TER	sBLEU	TER	sBLEU	TER	sBLEU
Bertoldi et al. [2013]	35.75	45.15	50.98	39.41	69.26	17.68
Cache-OriOl-Rcpp	34.00$^{\blacktriangle}$	47.18	49.43$^{\blacktriangle}$	36.40	67.96$^{\blacktriangle}$	19.57
Cache-OriOl-RcppNgram	33.90$^{\blacktriangle}$	47.52	49.43$^{\blacktriangle}$	36.42	67.58$^{\blacktriangle}$	19.60
Cache-ModOl-Rcpp	34.79	46.45	49.64$^{\blacktriangle}$	36.37	66.90$^{\blacktriangle}$	19.58
Cache-ModOl-RcppNgram	33.08$^{\blacktriangle}$	46.54	50.32	36.08	65.73$^{\blacktriangle}$	20.20

Table 5.9: Impact of pairing cache based models with the online learning algorithms. Statistical significance differences with $p < 0.01$ are shown by $^{\blacktriangle}$from the reference Cache system.

5.4.3 Insights and Discussion

For each of the three considered tasks, Figures 5.2, 5.3 and 5.4 plot the rate of learning of adaptive systems: *OriOl*, *Cache* and our *Cache-ModOl-*

RcppNgram against the baseline *Bsl*. Rate of learning of a system is calculated as the relative difference of the incremental TER scores from the baseline system. The higher the rate the better is the system. Negative rate implies that the system performs worse than the baseline.

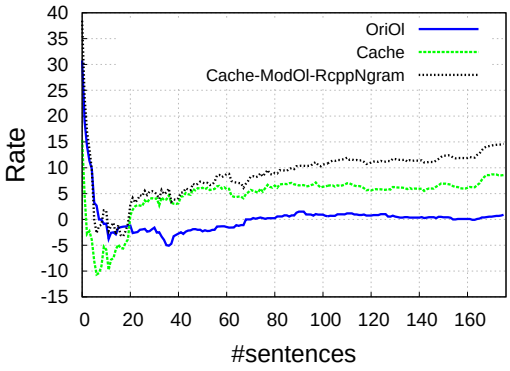

Figure 5.2: Learning curve of *OriOl*, *Cache* and *Cache-ModOl-RcppNgram* systems on IT domain wrt *Bsl* baseline system.

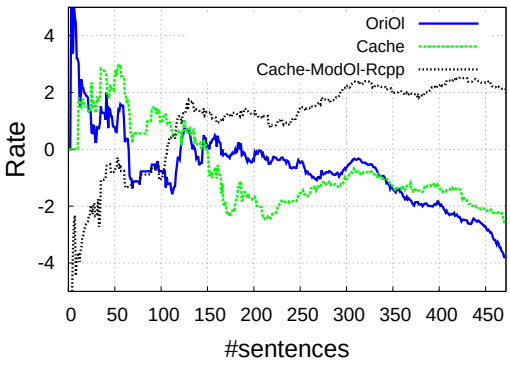

Figure 5.3: Learning curve of *OriOl*, *Cache* and *Cache-ModOl-RcppNgram* systems on Legal domain wrt *Bsl* baseline system.

It is evident that our enhancement via online learning of sparse features of pure caching mechanism (compare black and green lines) is very effective from the beginning (IT and WAT) or at least after a small fraction of the

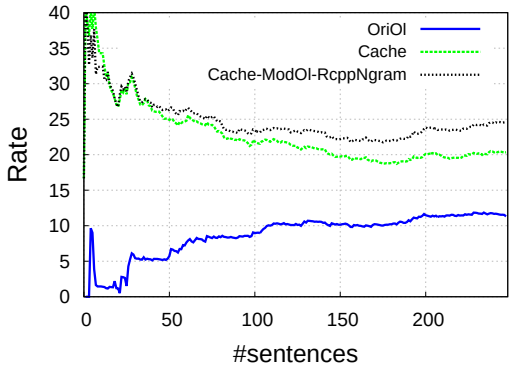

Figure 5.4: Learning curve of *OriOl*, *Cache* and *Cache-ModOl-RcppNgram* systems on WAT task wrt *Bsl* baseline system.

whole test set (Legal). Moreover, the modified online learning scheme together with the new feature functions is able to improve the original learning algorithm even with starting few number of sentences and in a consistent manner over all the considered tasks, even if in different amounts (black vs. blue lines).

Summarizing, the main outcomes of Experiment 2 are:

- sparse features can be effective but attention must be paid on how they are injected into the system, as their estimation and tuning can be problematic due to their intrinsic sparseness: the online learning scheme we have proposed has proved to allow the profitable exploitation of sparse features;

- the two sparse feature functions we have presented allow either consistent performance boost on all considered tasks (RCPP) or interesting improvements provided that they are supported by the caching mechanism (Ngram);

- our full fledged system is able to improve the TER score of the baseline by a large amount on two tasks out of three (15% and 24%); on the

third task, despite the global improvement is quite small, it shows a good dynamic behavior that allows to forecast more significant gains in longer runs.

5.5 Conclusion

We have shown a new way to update the translation model on the fly without changing the original probability distribution. We empirically proved that adding an adaptive feature in SMT system together with MIRA weight updates is robust approach to online adaptation which works for different domain datasets be it Information Technology or TED talks. In addition, if the repetition rate is high in the text, adaptation performance is better than if the rate is low. We tested both with an unbounded and a bounded range on the online feature and found out that bounded values produce more stable and consistent results. From the results, it is evident that we have not used any sort of stopping criterion for online learning; a random of 1, 5 and 10 iterations were chosen in a naive way. It needs to be seen if an optimum stopping criterion and hyperparameter value (such as learning rate in the online algorithm) could be an effective way to improve performance of the online MT systems.

In the latter part of the chapter, we have explored the online learning of sparse features. Two new sparse feature functions are introduced: one aims at promoting the rank of the translation in the n-best list which is closest to the post-edited segment; the other exploits the post edited segments for generating a dynamic list of n-grams used to trigger important target sequences in successive decodings. In addition, a modified online learning scheme has been designed with aggregated feature updates. Experiments are conducted on a number of different tasks and language pairs: TER improvements over the baseline score up to 24% are observed.

Chapter 6

Optimized Online User Adaptive MT

In this chapter we propose a cascading framework to optimize online learning in machine translation for computer assisted translation scenario.

Online learning algorithms have parameters such as learning rates and clipping values; we call them *hyperparameters* henceforth. As seen in the previous chapters, these hyperparameters if not optimized can lead to unstable results due to over or underfitting.

Convergence of online learning algorithms i.e. stopping criteria is another problem we focus on in this chapter.

We discuss these issues in this chapter and provide two different frameworks to optimize the hyperparameters and the stopping criterion in online algorithms for Machine Translation. Empirical results show that an optimized adaptive system yields consistent improvement against a strong adaptive baseline with default values for hyperparameters.

6.1 Introduction

The optimization of hyperparameters has been a well studied problem in machine learning community under the umbrella of *Hyperparameter Search* [Bergstra et al., 2011]. These hyperparameters are so called to distinguish them from the parameters of the model under analysis. Equa-

tion 6.1 shows a simple Perceptron update step where W are parameters of a linear model and α is the learning rate – a hyperparameter:

$$W_i = W_{i-1} + \alpha \times f(x) \tag{6.1}$$

Hyperparameter search is then the first issue when any learning algorithm is applied for problem solving. The second problem is the selection of the optimal number of iterations of the online learning algorithm, i.e. an optimal stopping criterion. In this chapter, we will investigate techniques to optimize the hyperparameters that we have seen in previous chapters, but in principle this work could be applied to any arbitrary hyperparameter.

The main ideas of this chapter have been published in [Mathur and Cettolo, 2014]. The organization of the chapter is as follows. The following section describes the problem faced by the current *adaptive* MT systems. Sections 6.3 and 6.4 describe the approaches we use to enhance the performance of an adaptive MT system. Section 6.5 and 6.6 present experiments and results, respectively, and are followed by the conclusions.

6.2 Problem

Before jumping on to the problem of adaptive system, let us see how adaptation works for MT systems in CAT scenario. Basically, there is a two-way adaptation process (c.f. Chapter 5):

1. feature adaptation, in which adaptive features are added to the translation model;

2. weight adaptation, where the parameters (log-linear weights) of the model are adapted on the fly using online algorithm (such as MIRA [Watanabe et al., 2007b]).

Online learning in the CAT framework is performed when feedback in the form of a post-edit segment is provided by the human translator. From the implementation point of view, a particular structure where the source sentence is paired with its translation suggestion and post-edit segment (as feedback) is passed to the decoder which activates a single iteration of online learning. To perform multiple iterations, a corresponding number of copies of this structure has to be passed as input to the decoder.

In the previous chapters, we have not deeply investigated the optimization of hyperparameters involved in the online learning process. In fact, these hyperparameters are optimized by means of the Simplex algorithm (c.f. Chapter 5), and the same values are then re-used for any possible number of iterations of the online learning, disregarding the dependence between the number of iterations and the hyperparameters, which as it turns out is not an optimized solution. We will see that the values of chosen hyperparameters greatly depend on the number of iterations of the online learning.

Another problem is that these hyperparameters cannot be tuned alongside the parameters of MT models via a discriminative training procedure [Och and Ney, 2003]. To optimize the parameters of MT models, this procedure operates on a list of n-best candidate translations and re-ranks them by changing the parameters of model. Since the hyperparameters do not affect this list of n-best list once it is created, there is no direct way to optimize them via traditional tuning methods in MT. An alternative solution needs to be found.

We extend this investigation from two viewpoints:

- The selection of optimum hyperparameters;
- We then look for optimum number of iterations of online learning.

6.3 Optimization of Hyperparameters

Hyperparameters in SMT models, such as *distortion limit* and *beam size*, have been typically optimized using derivative free optimization (DFO) techniques such as Simplex [Chung and Galley, 2012] and Hill Climbing [Stymne et al., 2013]. To our knowledge there is no other work which focuses on optimizing number of iterations of online learning for SMT.

In our work, the optimization process is a cascade of the following steps:

1. The parameters of MT models are tuned using MIRA [Watanabe et al., 2007b] with the objective of minimizing the error on a development set.

2. Copies of the development set are made such that each copy represents a different number (i) of iterations of online learning ($i \in 1..10$).

3. The tuned parameters are kept fixed and hyperparameters are tuned with derivative free optimization (DFO) techniques.

This cascaded approach prevents joint optimization over all the parameters of SMT model which is not feasible using the DFO techniques because they tend not to converge with so many parameters.[1]

In this work we focus on three hyperparameters, namely the *feature learning rate* (α), the *weight learning rate* (γ) and the *slack variable* (ξ) (refer Chapter 5). α determines the rate of learning of the additional online feature; γ and ξ control respectively the learning rate and the size of the update of the online learning algorithm (MIRA) employed for updating the log-linear weights. Their optimization is performed with respect to a loss function defined over an objective MT evaluation metric, by the two DFO techniques described in the following.

[1]Here, the parameters refer to the log linear weights (14 weights in our case), plus 1 additional weight for the online learning feature (c.f. Chapter 5).

Downhill Simplex Method The Downhill Simplex method, also known as Nelder-Mead method [Nelder and Mead, 1965], is a technique for minimizing an objective multivariate function. The method is iterative and approximates a local optimum by using a simplex, that is a polytope of $N + 1$ vertices in N dimensions. At each iteration, a new test position is evaluated by extrapolating the behavior of the objective function measured at each vertex of the simplex. The algorithm then chooses to replace one of the vertices with the new test point and so the search progresses. New test positions are generated so that the simplex is stretched along promising lines (when the simplex is still far from any optimum) or shrinked towards a better point (when it is close to a local optimum).

Modified Hill Climbing Hill Climbing is generally used for a single variate function $f(x)$: it fluctuates the value of the variable x and computes the loss incurred by the function $f(x)$. Step by step, the method moves the variable towards the direction where the loss incurred is minimum. We extended the same optimization to multivariate functions $f(x_1, x_2, \ldots, x_n)$ by moving one variable at a time. Moreover, we modified the original hill climbing by initially allowing the variable to take large steps in the search space, and then constrain the variable to take smaller steps, similar to simulated annealing [Kirkpatrick et al., 1983]. This allows Hill Climbing to converge faster than in the original hill climber.

Next, we see that the optimal value of hyperparameters depends on the number of iterations used for the online learning; therefore, the optimization process is run once for each i number of iterations ($i \in 1..10$). Given that the hyperparameters are optimized with two derivative free optimizers, a total of twenty different configurations are available for each SMT system to test.

6.4 Stopping criteria

Once the optimal values of the parameters and hyperparameters have been estimated, the next step towards improving the online learning algorithm is to find the optimum number of iterations to perform. Cesa-Bianchi et al. [2008] applied two iterations of online learning and showed that systems with two iterations outperform systems with single iteration. Inspired from those outcomes, we try to find the optimal number of iterations of online learning. In theory, the model resulting from the run of the optimum number of iterations should outperform the models obtained with a random iteration number (within limited range) thereby avoiding overfitting issue on the evaluation data. We propose two solutions to find this optimum number, i.e `clustering`, which needs a pre-processing step on the development set, and `blockwise`, which works on the evaluation data directly.

The `clustering` approach stems from the work of [Sennrich, 2012c], where the authors split a large development set (coming from heterogeneous sources) into small clusters. The feature weights of MT model are then tuned on each of these small clusters, so for each cluster there is a set of tuned feature weights. Later, each sentence in evaluation set is classified into one of the clusters and the tuned weights on that cluster are applied for translation. The `blockwise` approach stems from the *file-context* effect explained in [Hardt and Elming, 2010], where the authors show that adapting on the evaluation data is useful because of the recency effect i.e. nearby data within the file has greater value than more distant data (training data).

Clustering In the clustering approach, at the pre-processing stage, the development set is partitioned into k clusters and the optimal number of iterations for each cluster is determined. At run-time, each source sentence

in the evaluation set is classified into one of the clusters of the development set and the corresponding optimal number of iterations for the assigned cluster is used for the online learning on that source and post-edit pair.

For clustering, we train two language models (LMs) on source and target sides of in-domain parallel data. For every sentence pair in the development data we calculate the entropies with the two LMs. In the end, for each pair in development set we have two features i.e. source and target LM entropy. We perform k-means clustering with random seeding to cluster the development set and use `Euclidean` distance as similarity metric.[2] Once the development set is clustered, the optimal number of iterations for each cluster is computed as follows:

1. online learning is run on each cluster for $i \in \{1 \dots 10\}$ iterations, keeping track of the error rate at each iteration;

2. each cluster is then associated with the iteration number corresponding to the minimum error rate.

The scheme in Figure 6.1 represents the clustering approach.

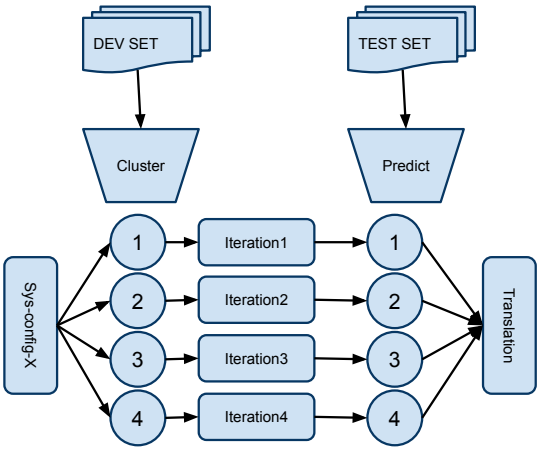

Figure 6.1: Clustering approach to find the optimal number of iterations for online learning.

[2]The `Cosine` distance was also tested: it performed similarly to the `Euclidean` distance, but `Euclidean` distance gave better quality of clusters than `Cosine`. We measure the quality of clusters with Quality of Clusters (QoC) metric.

Blockwise In the blockwise approach, the evaluation set is split into blocks of $N \in \{10, 20, 30\}$ sentences.[3] Once the X^{th} block has been post-edited by the user, the optimal number of iterations (O_X) for that block is found by minimizing the error rates (1−sBLEU) over all possible iterations of online learning, with sBLEU being the sentence level BLEU score [Lin and Och, 2004]. O_X is then used to perform the online learning on each segment (and post-edit) of the $X + 1^{th}$ block, till the whole block is processed. The blockwise method is depicted in the Figure 6.2.

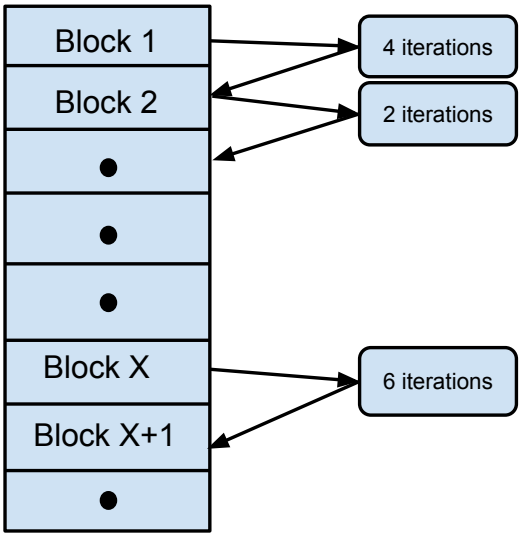

Figure 6.2: Blockwise approach to find the optimal number of iterations for online learning.

6.5 Experiments

6.5.1 Data

We evaluated our methods for optimizing the online learning systems on three translation tasks defined over two domains, namely Information Technology (IT) and Legal. The IT evaluation set involves the translation of

[3]In real texts, we can assume that bunches of some tens of segments (e.g. 10-30) are linguistically coherent such that an adaptation scheme can be effectively applied.

technical documents from English into Italian and has been used in the field test recently carried out under the MateCat project [Federico et al., 2014]. In the Legal domain, experiments involved the translation of English documents into either Spanish or French; training and evaluation sets belong to the JRC-Acquis corpus [Steinberger et al., 2006] so that the effectiveness of the proposed approaches is assessed also on publicly available data.

Since our methods regard the adaptation of MT models, the potential impact strictly depends on how much the considered text is repetitive. For measuring the same, we use the repetition rate proposed by Cettolo et al. [2014]. Statistics of the parallel data sets in both source and target languages along with their repetition rates are reported in Table 6.1.

Domain	Set	#srcTok	srcRR	#tgtTok	tgtRR
$IT_{en \rightarrow it}$	Train	57M	na	60M	na
	Development	3.7K	7.65	4K	7.61
	Evaluation	3.4K	34.33	3.7K	33.90
$Legal_{en \rightarrow es}$	Train	56M	na	62M	na
	Development	3K	24.09	3.5K	24.47
	Evaluation	11K	20.67	12.5K	20.07
$Legal_{en \rightarrow fr}$	Train	63M	na	70M	na
	Development	3K	23.52	3.7K	23.42
	Evaluation	11K	20.67	13K	20.92

Table 6.1: Statistics of the parallel data along with the corresponding repetition rate (RR).

6.5.2 Baseline Systems

We compare phrase-based systems built with the Moses toolkit [Koehn et al., 2007a]. Domain specific training data are used to create translation and lexical reordering models. 5-gram language models for each task, smoothed through the improved Kneser-Ney technique [Chen and Goodman, 1998], are estimated by means of the IRSTLM toolkit [Federico et al., 2008] on the target side of the training parallel corpora. The weights of

the log-linear interpolation of MT models (parameters) are optimized with MIRA [Watanabe et al., 2007b].[4] Performance is reported in terms of BLEU [Papineni et al., 2002c] and TER [Snover et al., 2006b]. Details of baseline SMT systems follow:

Baseline The static baseline system which does not perform any online learning and, thus, no hyperparameters are involved in the system.

Def-Param-*x Online learning systems using default values of hyperparameters, with fixed number of iterations (1, 5 and 10) of online learning. We use the default values of the hyperparameters as FLR=0.1, WLR=0.05 and SLK=0.001, which yield reasonable performance in preliminary investigations.

These systems provide a reference for assessing the usefulness of estimation of the optimal hyperparameters vs. the use of pre-defined values.

6.5.3 Optimization of the Hyperparameters

Having tuned the log-linear weights, we compare the above systems with the following systems:

Opt-Param-*x Online learning systems with hyperparameters optimized by means of either Simplex or Hill Climbing techniques of Section 6.3.

Figures 6.3 and 6.4 shows the convergence of the DFO methods over their number of iterations. These are the iterations required by Simplex/HillClimbing to converge while keeping a fixed the number of iterations of the online learning (in this case 1× and 10× respectively).

[4]Note that the optimization here is done on the log-linear weights of translation, reordering, language models and not for the hyperparameters.

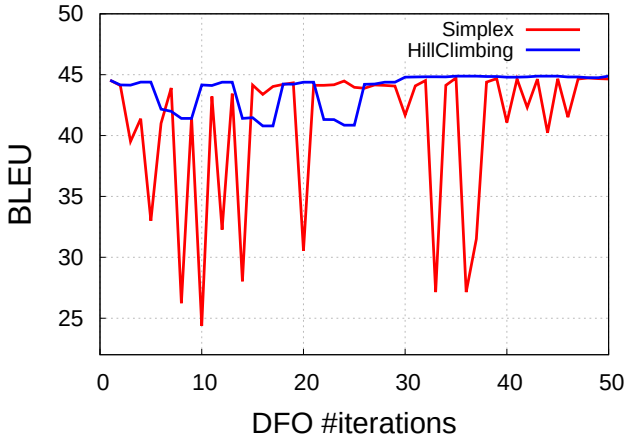

Figure 6.3: Simplex vs. Modified Hill Climber on the Legal/en→fr development set, with 1× iteration of the online learning algorithm.

From Figure 6.3, we can argue that the Simplex attempts many hyperparameter values performing quite differently, and finally converges to an optimum. Hill Climbing converges to the optimum faster, but on the other side it seems to explore a smaller portion of the search space. In Figure 6.4, optimization of hyperparameters is conducted with fixed 10 iterations of online learning. Here, Hill Climbing fails to converge to an optimum, but Simplex while attempting many hyperparameter values, finally converges to an optimum. These figures show that:

1. Learning curve of Simplex is more stable and steadily increases over time as compared to Hill Climbing;

2. Hill Climber is susceptible to initial hyperparameter values while Simplex is not;

3. For different number of online learning iterations the DFOs behave differently.

Figure 6.5 shows the optimal value of hyperparameters as a function of

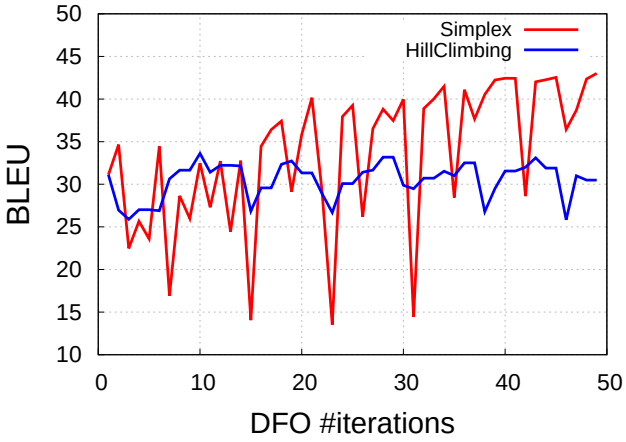

Figure 6.4: Simplex vs. Modified Hill Climber on the Legal/en→es development set, with 10× iteration of the online learning algorithm.

the number of iterations of the online learning algorithm. Apart a general and reasonable tendency to define smaller updates when more iterations are performed, it is worth to note that the optimal hyperparameter values **do** change with the number of iterations, again confirming our hypothesis.

As described in Section 6.3, tuning of hyperparameters was performed for different numbers of iterations of online learning, resulting in 10 different configurations for each DFO algorithm.

6.5.4 Online Learning Stopping Criteria

At the end of the optimization stage, 10 optimal configurations for each of the two DFO techniques are available for testing. In both DFOs, a TER-based loss function is employed, since clusters/blocks can be too small to allow the reliable computation of BLEU values. A total of 20 optimized systems are then run to look for the optimal number of iterations of the online learning to be used on the evaluation sets: this optimal number is found on the development set with the clustering technique, directly on

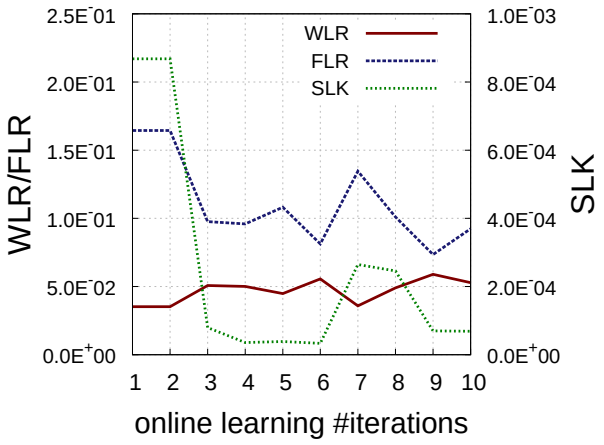

Figure 6.5: The optimal values of different hyperparameters changes with different number of online learning iterations (IT/en→it task).

the evaluation set with the blockwise technique.

Clustering As already mentioned, we first partition the development set using k-means clustering, where k takes values in $\{2, 4, 6, 8, 10, 12\}$. In principle, we can increase k but that would decrease the size of the clusters with the risk of data overfitting. The development set of the IT task consists of 300 sentences, i.e. 300 data points, hence for consistency purposes we set the maximum value of k to $\left\lfloor \sqrt[2]{300/2} \right\rfloor = 12$ for all tasks.[5]

For each cluster, we pick the configuration which performs best on the development set;[6] the sentences in the evaluation set that are assigned to that cluster are then translated with the chosen optimal configuration.

Figure 6.6 reports the average number of iterations of online learning required by the clustering technique to converge for different values of k. It shows that the larger the number of clusters, the faster the convergence of

[5]According to Mardia et al. [1980], a rule of thumb for choosing the maximum value number of cluster is to take k as $\left\lfloor \sqrt[2]{N/2} \right\rfloor$.

[6]These configurations are not compared all together at once; instead, we separately compare the 10 configurations for each DFO method.

the online learning algorithm; this is because the online learner has less to learn from small clusters, even after performing more and more iterations.

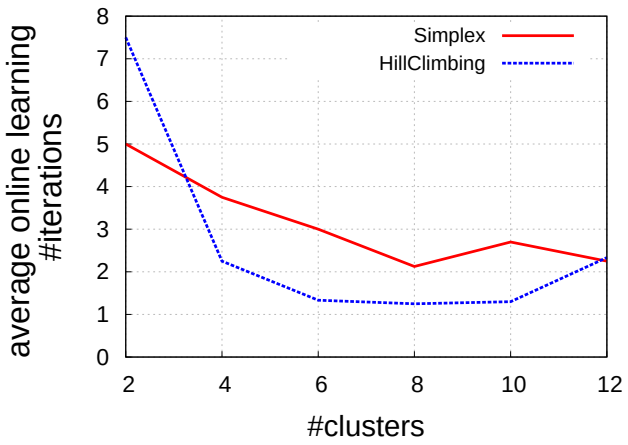

Figure 6.6: Average number of iterations of the online learning algorithm per number of clusters for the two optimizers (Legal/en→fr task).

Blockwise In the blockwise approach the number of iterations is optimized directly (but fairly) on the evaluation set. The optimal number for a given block is found once its post-edits are available and is used for the translation of the following block; this step is iterated for all blocks. In other words, the number of iterations of online learning on the current block is decided by optimizing the number of iterations on the previous block.

Nonetheless, there are two main issues with this approach:

1. what is the number of iterations for the first block?
2. what should be the size of the block?

We decided on the following. First, for the first block of the evaluation set, the optimal configuration (the optimal number of iterations) on the development set is considered. This is analogous to what we do in the

clustering approach. Second, to test the effect, different sizes of the block ($10, 20, 30$ sentences) are tried. Figure 6.7 shows how the block size affects the optimal number of online learning iterations for translation from English to Italian on IT domain.

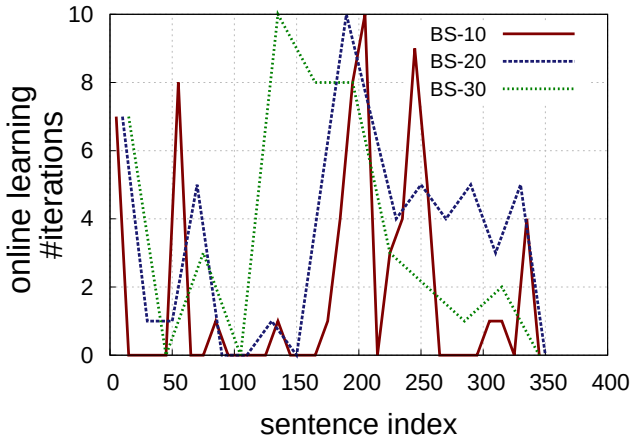

Figure 6.7: Effect of varying the block size on the number of iterations of the online learning algorithm (IT/en→it).

6.6 Results

Baselines Table 6.2 collects results from the baseline and the optimized systems (Def-Param-*, Opt-Param-*) mentioned in Sections 6.5.

The online learning system with 10× iterations and optimized hyperparameters outperforms the baseline by 0.5 to 1 BLEU/TER points on both IT/en→it and Legal/en→fr tasks. On the Legal/en→es task, the best performance is obtained by performing 1 iteration of the online learning, that allows to clearly beat the baseline by 1.70 TER points (48.34 vs. 50.04) even if no gain is observed in terms of BLEU score.

In two out of three tasks (en→it and en→es), the performance of the systems with default hyperparameters decreases rapidly as the number of

System	IT en→it		Legal en→fr		Legal en→es	
	BLEU	TER	BLEU	TER	BLEU	TER
Baseline	46.73	31.97	33.69	51.49	35.65	50.04
Def-Param-1x	46.27	31.23	34.28	50.31	35.28	48.07
Def-Param-5x	42.61	34.90	33.04	51.51	32.13	51.12
Def-Param-10x	39.18	37.66	34.34	50.25	31.08	52.74
Opt-Param-1x	46.56	31.41	34.24	50.34	35.38	48.34
Opt-Param-5x	44.48	33.28	33.32	50.87	32.68	50.82
Opt-Param-10x	47.11	31.41	34.25	50.47	34.61	48.78

Table 6.2: Metric scores for all systems: Baseline; online learning with default values of hyperparameters (Def-Param-*); online learning with optimized values of hyperparameters by means of Simplex (Opt-Param-*). Online learning is performed for fixed numbers of iterations (1,5,10).

iterations increases, because the hyperparameters are not tuned properly on the held-out dev set. This *confirms* our assumption that the value of hyperparameters plays an important role on the system's performance. In the Legal domain (en→fr) system, incidentally the default value of hyperparameters is close to the optimized one and hence the performance of both systems is similar in the two setups ([Def|Opt]-Param-*x), better than the baseline by around 0.5 BLEU and 1 TER with iterations (1× and 10×).

Table 6.3 and Table 6.4 collect results of systems with hyperparameters optimized on the basis of the optimal number of online learning iterations, as determined by means of the two investigated techniques (blockwise and clustering). As a first general consideration, apart few exceptions, Simplex is more effective than Hill Climbing in optimizing hyperparameters; therefore, in the following discussion, we focus on it although we report results with both optimizers.

Blockwise Table 6.3 report results employing the blockwise technique. On the IT/en→it task, the blockwise system (block size 10) outperforms the baseline in terms of TER (31.43 vs 31.97) and gives comparable performance to the optimized Opt-Param-10x system (31.43 vs 31.41). Note

block size	IT en→it				Legal en→fr				Legal en→es			
	simplex		hill climbing		simplex		hill climbing		simplex		hill climbing	
	BLEU	TER	BLEU	TER	BLEU	TER	BLEU	TER	BLEU	TER	BLEU	TER
10	**46.80**	**31.43**	46.31	31.48	34.64	50.18	33.68	49.95	34.99	48.87	34.16	49.31
20	46.30	31.81	46.11	31.96	**34.98**	**49.63**	34.47	49.66	**35.64**	**48.33**	34.44	48.70
30	45.42	32.42	46.40	31.71	34.78	50.68	34.30	51.13	35.68	48.67	34.63	48.64

Table 6.3: Results of blockwise technique on the three considered tasks, by varying the block size. Best systems are marked in bold.

that this same quality is obtained more efficiently in Blockwise: in fact, since for each block no more than 10 iterations are performed (likely less, according to Figure 6.7), on an average it is expected that the global cost on the whole evaluation set is lower than the cost of the Opt-Param-10x system, where 10 iterations are performed on each sentence of the evaluation set.

On the Legal/en→fr task, the blockwise system (with block size 20) significantly outperforms (p-value < 0.05) the baseline as well as the best performing online learning system with fixed number of iterations (Def-Param-10x) by 1.86 and 0.62 TER points, respectively. One important note here is that the size of the block yielding the highest performance differs between the IT and Legal domains because the IT domain data is a collection of different technical documents which makes it rather heterogeneous. Legal domain data, on the other hand, being from single documents, are much more homogeneous in nature, allowing the use of larger blocks.

On the Legal/en→es task, we observe no BLEU gains in comparison to the baseline system but TER improves by up to 1.71 points (48.33 vs. 50.04, with the block size of 20). This is attributed to the fact that we use TER as the error metric instead of BLEU. The improvements here are similar to the Opt-Param-1x system which means that blockwise approach does not have a large positive impact on this particular dataset but at the same time it does not worsen the performance of the system either.

	IT en→it				Legal en→fr				Legal en→es			
	simplex		hill climbing		simplex		hill climbing		simplex		hill climbing	
Cluster	BLEU	TER	BLEU	TER	BLEU	TER	BLEU	TER	BLEU	TER	BLEU	TER
2	46.80	30.95	46.33	31.28	34.90	50.75	35.02	50.80	36.13	47.78	36.30	47.82
4	45.99	32.09	46.07	31.33	35.09	50.24	35.24	50.71	36.05	47.89	36.05	47.74
6	46.41	31.08	46.06	31.48	34.66	50.52	34.40	50.59	35.71	48.07	35.86	47.77
8	46.03	31.81	45.76	31.68	35.07	50.58	35.06	50.78	35.75	48.23	36.11	47.76
10	44.98	32.67	46.32	31.31	35.12	50.62	35.23	50.94	36.08	47.56	35.69	47.87
12	46.23	31.74	46.79	30.77	35.15	50.62	35.07	50.74	36.34	47.74	35.58	47.93

Table 6.4: BLEU/TER scores for varying cluster sizes; performance of the two DFO methods is reported.

Clustering Concerning the clustering technique, results are shown in Table 6.4. Similarly to the blockwise method, for the IT/en→it we do not see any BLEU gain, while TER improves the baseline by even more than 1 point (30.95 vs. 31.97).

For the Legal/en→fr task, an increase of almost 1.5 BLEU (from 33.69 up to 35.15) is observed for most of the cluster sizes. Even in terms of TER, the number of cluster (and then the cluster size) seems to not affect the score much, which always improves by around 1 point, with a peak of 1.25 (51.49 vs. 50.24) when 4 clusters are employed.

A behavior similar to IT/en→it is observed in the Legal/en→es task: minimal impact of the cluster size, small BLEU improvements (no more than 0.7 points), larger TER gains (even more than 2 points).

Summarizing, large gains of up to 2.3 TER are observed on Legal sets while we see improvements of 1 TER on IT set. Clustering approach has an advantage over the blockwise approach when the data is homogeneous in nature as the data can be divided naturally in clusters. As we know that legal domain data is in fact more homogeneous than IT domain, clustering approach results in better performance.

We also see consistent improvements in TER, but not in BLEU. As said before, it is due to the use of TER as the error metric for finding the

optimal iterations of online learning for the clusters. In fact, the size of clusters can sometimes be too small to allow the reliable computation of the BLEU. On the other hand, optimizing on TER favors short sentences, which ultimately lowers the BLEU score via brevity penalty.

6.7 Conclusion

We have shown that *optimized* online learning can be effectively integrated into MT for CAT by following a cascaded framework where one first optimizes the extra parameters involved with the learning algorithm, and then finds the optimal number of iterations of online learning required on the test set. We experimented with two derivative free optimization techniques, namely Simplex and Hill Climbing, and showed their convergence. Simplex optimizer showed a more stable learning curve over time than the Hill Climber optimizer. Two techniques, Blockwise and Clustering methods, are proposed to find the optimal number of iterations. After an extensive set of experiments we can conclude that the clustering technique performs better than the blockwise approach when the evaluation set is homogeneous in nature, otherwise the blockwise with small blocks is preferable.

Chapter 7

Online Multi-User Adaptive MT

In the previous chapters we have talked about adapting a statistical machine translation system on the feedback provided by a single translator. In this chapter we go one step further and investigate the problem of adapting a MT system to the feedback provided by multiple translators. It is well known that translators might have a very different post-editing style and that this variability hinders the application of online learning methods, which indeed assume a homogeneous source of adaptation data. We propose a *multi-task learning* approach to leverage the bias information from each post-editor in order to constrain the evolution of the MT system. Then a new framework for significance testing with sentence level metrics is described which shows that multi-task learning approaches outperform existing online learning approaches, with significant reductions in TER score over a strong online adaptive baseline. Our experiments were run on a test set of post-edits produced by four translators and on a popular benchmark with multiple references.

7.1 Introduction

In the previous chapters (see. Chapter 5) and in Denkowski et al. [2014], MT is fed with post edited sentences, allowing the system to adapt to the

corrections made by the translators. These kind of systems work well if the document is being post edited by a single translator because models can adapt to the style of that translator. Problems arise when a document is being post edited by a group of translators which is usually the case with large documents. In fact, if the MT system adapts to the corrections of all translators together, it will likely mix or overlap stylistic features of the post-editors and thus not learn to mimic well any of them. On the other side, if the system adapts to each individual post-editor, then useful feedback from other post-editors clearly gets wasted.

The main motivation for adapting MT systems in the backend of CAT tools is to improve the quality of translation suggestions and make less mistakes over time by learning from the feedback. For example, *translator A* does not post-edit phrase *Fibre Channel* because she thinks that the phrase is a named entity, while *translator B* post-edits the same as *Canale a fibre* because he does not recognize the phrase as a named entity. Meanwhile in the backend, the MT system first adapts the model such that it keeps the named entity intact but after second post-edition the system adapts the model to translate *Fibre Channel* $\rightarrow$ *Canale a fibre*. Now, if the system receives again the input *Fibre Channel*, it will prefer to translate the phrase as *Canale a fibre*. *Translator A* upon receiving the incorrect translation loses trust in the MT system and may not rely on it in the future. This repetition of translation error slows down the process of post-editing for the translator which is in fact completely opposite to the idea of considering first the MT suggestion than any other suggestion.

In this chapter, we aim at building an adaptive MT system which can solve this dilemma of contrasting updates. To do so, we propose using multi-task learning (henceforth MTL) [Caruana, 1993] in machine translation systems. Here, we can consider the translators as different *tasks* and their post-edits as an incoming stream of data the system wants to learn

from. Our system also maintains a prior relationship between the translators (a.k.a. *tasks*), according to the framework specified in Cavallanti et al. [2010].

In this chapter, we first describe the generic online multi-task learning algorithm developed by Cavallanti et al. [2010] for classification problems. Then, Section 7.3 describes the online multi-task learning algorithm for statistical machine translation. Experiments and results are shown in Section 7.4. We conclude the chapter with a few words on related work.

The main ideas of the presented approach have been published in [Mathur et al., 2014a].

7.2 Online Multi-Task Learning

Multi-task learning framework allows adaptation of an automated system over multiple translators, here called tasks. In the online multi-task learning (OMTL) [Cavallanti et al., 2010] system, training is done jointly on k tasks in an online fashion so as to improve generalization capability over all tasks simultaneously. We extend the use of OMTL from classification and regression to machine translation.

The protocol for OMTL at each iteration i is as follows:

1. Receive an input pair (x_i, s), where x_i is the sample and s is the task id,
2. Output a prediction $\hat{y} = w_{s,i}\ h_s(x_i)$, using the current *weights* $w_{s,i}$ for the current task s and the *feature vector* $h_s(x_i)$,
3. Receive the correct label y,
4. Calculate the loss for the task s and
5. Update the weight vectors $w_{s,i+1}$ for all tasks $s \in 1, \ldots, k$ based on the loss incurred by task s.

If we use a single task learning algorithm like Perceptron [Rosenblatt, 1958] for multiple tasks at the same time, weights will be updated one task at a time with a product of a learning rate (α) and a feature vector ($h(x)$) in the direction of the signed value ($sgn(\hat{y}, y)$) determined by comparing the prediction and the label as shown in Equation 7.1.

$$w_i = w_{i-1} + sgn(\hat{y}, y)\alpha \cdot \Delta h(x) \tag{7.1}$$

The update step in OMTL is similar to that of the Perceptron, but updates are done for all tasks at once. The learning rates in OMTL are defined in an *interaction matrix* (shown in Equation 7.2) which encodes the *relatedness* measure among the different tasks. Relatedness measures the closeness or the similarity between two tasks in terms of their predefined reference vectors. In case of translation, translators being the task, we can check how similar are two translators based on their past post-edition efforts to the same translation or some other metric. Post edition effort on the same translation would give us an idea of how related the translators are. If both translators post-edit the same amount of words for a given number of translations then the two are highly related otherwise not. α_{12} element in the interaction matrix defines the learning rate for task s_1 when task s_2 is being executed.

$$A^{-1} = \frac{1}{k+1} \begin{bmatrix} \alpha_{11} & \alpha_{12} & \cdots & \alpha_{1k} \\ \alpha_{21} & \alpha_{22} & \cdots & \alpha_{2k} \\ \cdots & \cdots & \cdots & \cdots \\ \alpha_{k1} & \alpha_{k2} & \cdots & \alpha_{kk} \end{bmatrix} \tag{7.2}$$

The update rule at iteration i for OMTL is defined as:

$$W_i = W_{i-1} + sgn(\hat{y}, y)(A \otimes I_d)^{-1} H(x_{s,i}) \tag{7.3}$$

where W represents a compound weight vector for all tasks, basically a concatenation of weight vectors for all tasks as shown in Equation 7.4. The size of the compound vector is thus $k \times d$:

$$W = [w_{1,1}, \ldots, w_{1,d}, \ldots, \ldots, w_{s,1}, \ldots, w_{s,d}, \ldots, \ldots, w_{k,1}, \ldots, w_{k,d}] \quad (7.4)$$

Once the sample is received for a task s, features $h_s(x)$ are computed and stored as a compound vector $H_s(x)$, where the features are activated for the current task and the rest are zeroed out as shown in Equation 7.5.

$$H_s(x) = (\underbrace{0, \ldots, 0}_{(s-1)d \text{ times}}, h_s(x), \underbrace{0, \ldots, 0}_{(k-s)d \text{ times}}) \in \mathbb{R}^{kd} \quad (7.5)$$

$\otimes$ denotes the Kronecker product[1] of the interaction matrix (A^{-1}) of dimension $k \times k$ and identity matrix (I_d) of dimensions $d \times d$, making a $kd \times kd$ matrix. The $kd \times kd$ matrix in the update rule *regularizes* the weight vector (W_{i-1}) by forcing the learner to account for the relatedness between the tasks. The term $(A \otimes I_d)^{-1} H(x_{s,i})$ produces an *update vector* of size $k \times d$ with which the compound weight vector is updated.

7.3 MIRA with multitasking

MIRA has been successfully applied to tune the log linear weights of MT models in post editing scenario as seen in the previous chapters. Before diving into details of multi-task learning with MIRA, let's recap how MIRA works for MT. MIRA is an ultra conservative online algorithm, i.e. the updates to the weight vector are minimal so as to satisfy a constraint in

[1] $\otimes$ shows mixed-product property, so one can calculate A^{-1} and then compute the Kronecker product of $A^{-1} \otimes I_d^{-1}$.

the following equation:

$$l_j \leq w^T \Delta h_j; j \in 1 \ldots N \tag{7.6}$$

The constraint basically says that the loss (l_j) of the external metric (e.g sentence BLEU) should not be greater than the loss of the model score ($w^T \Delta h_j$) for the translation candidate at position j in the N-best list.

$$w_i = w_{i-1} + \sum_j \alpha_j \cdot \Delta h_j. \tag{7.7}$$

Here, we extend this online algorithm to fit the scenario where input comes from k different translators (a.k.a tasks[2]) and the system predicts the weights for all the tasks simultaneously.

We modify Equation 7.7 by adding the matrix co-regularization factor of $(A \otimes I_d)^{-1}$ (from Equation 7.3), such that the difference of feature vector from j^{th} candidate translation (i.e. Δh_j) affecting the change in weights for current task s takes into account the bias from each task. After substitution, the update rule becomes:

$$\begin{aligned} w_s &= w'_s + \sum_j \alpha_j \cdot \langle \Delta h_{s,j} \rangle && \text{where} \\ \langle \Delta h_{s,j} \rangle &= (A \otimes I_d)^{-1} \cdot \Delta H_{s,j} && \text{and} \\ \Delta H_{s,j} &= (\ \underbrace{0, \ldots, 0}_{(s-1)d \ times}\ , \Delta h_{s,j}, \ \underbrace{0, \ldots, 0}_{(k-s)d \ times}\ ) \end{aligned} \tag{7.8}$$

Here, $\Delta H_{s,j}$ is a compound row vector for candidate translation j of size kd with d being the size of the standard log linear features used in MT[3]. A^{-1} as seen from Equation 7.2 defines the task relatedness. In CAT scenario we can see the interaction matrix as the matrix which defines relatedness

[2]We are using the terms tasks and translators interchangeably as the tasks are translators in the CAT scenario.

[3]To keep the notation light we again drop the dependency of h from x.

between different translators. This relatedness can be captured by finding a correlation between the translators on their previous post-editions of a given dataset. The more similar their post-editions on a particular dataset (including suggestion coming from one MT system), the higher is the relatedness value between the two translators. In Section 7.4.2, a detailed explanation on how to compute the correlation between the translators is provided.

7.4 Experiments and Results

7.4.1 Data

We evaluated our method on three translation tasks defined over three different domains, namely Information Technology (IT), Travel domain (BTEC) and Legal domain.

The IT data involves the translation of technical documents from English into Italian. The test set has been used in the field test carried out under the MateCat project [Federico et al., 2014]. It has been translated by four professional translators, i.e. four different translations of the source document are available.

BTEC is a publicly available corpus in the travel domain, and have been proposed in the translation task at the IWSLT evaluation campaigns up to 2010. In addition to its availability, BTEC is of interest for us because the test set contains six human references, allowing us to simulate the multi-task scenario.

Legal domain data has been released as a part of JRC-acquis corpus [Steinberger et al., 2006]. The dataset contains translation of legal documents from English to Italian. This dataset was also a part of the field test carried out under the same MateCat project, so essentially we have post-edited data from 4 different translators on a test set of 90 sentences.

Similar to previous chapters we compute the repetition rate [Bertoldi et al., 2013] to forecast the potential impact of adaptation in SMT systems. Statistics of the parallel sets on source and target sides along with the repetition rates are reported in Table 7.1.

Domain	Set	#srcTok	SrcRR	#tgtTok	TgtRR
$IT_{en \rightarrow it}$	Train	57M	na	60M	na
	Dev	3.3K	19.08	3.6K	18.01
	Test	3K	31.32	3.3K	22.18
$BTEC_{en \rightarrow it}$	Train	0.14M	na	0.13M	na
	Dev	2K	9.47	1.9K	6.73
	Test	1.9K	12.5	1.8K	7.76
$Legal_{en \rightarrow it}$	Train	63M	na	65M	na
	Dev	2.9K	14.37	3.2K	11.25
	Test	2.7K	13.59	2.85K	12.00

Table 7.1: Statistics of parallel data. SrcRR and TgtRR are the repetition rate of the source and the target sides.

Preparing Data for MTL Since we have k translations for a source document, we shuffle the references/post-editions such that we have one source document and one target document with the sentences containing meta information for the translators who produced these translations. Table 7.2 shows a sample of source and target document from IT dataset. The figure reads: sentence #1 is translated by translator #0, then feedback (sentence #2) goes to the system with its post-edited translation, system performs multi-task learning and so on. If one removes the meta-information about the translator's ID, the resulting development set is used for single-task online learning. If one also removes the feedback, then the development set is used for baseline system (refer Section 7.4.2).

This shuffling of data also impacts the repetition rate. In fact, the repetition rates on the target side of IT test set for each translator varied from 26.95 to 28.70, while the repetition rate on the shuffled target side is 22.18, as reported in Table 7.1; this could be due to the fact that translators

tend to be not consistent among themselves, yielding less repetitions in each post-edited test set than in the shuffled test set.

#Sentence	Sentence	OnlineLearning	Translator ID
1	Input Date_#_0	Not Activated	0
2	Input Date_#_Data di input_#_0	Activated	0
3	Evaluates conditionally_#_1	Not Activated	1
4	Evaluates conditionally_#_ Valuta in modo condizionale_#_1	Activated	1

Table 7.2: Excerpt from IT development set tagged with meta data.

7.4.2 Experiments

The SMT systems were built using the Moses toolkit [Koehn et al., 2007a]. Domain specific training data was used to create translation and lexical reordering models. 5-gram language models for each task were estimated by means of the IRSTLM toolkit [Federico et al., 2008], with improved Kneser-Ney smoothing [Chen and Goodman, 1998], on the target side of the training parallel corpora. After the training of MT models, the log linear weights were optimized with MERT [Och, 2003a] provided in the Moses toolkit. Performance is computed not with corpus level metrics but with sentence level metrics. We decided to do this to avoid a metric mismatch between the evaluation and actual optimization where the margin is calculated by the sentence level BLEU scores [Lin and Och, 2004]. Therefore, we computed sBLEU scores and sentence level TER [Snover et al., 2006b] scores and reported their average over the whole documents. We call them avg-sBLEU and avg-sTER.[4]

Calculating A^{-1} matrix: Interaction matrix can be computed in different ways. It basically conveys the relatedness/correlation between the trans-

[4]In principle both TER and avg-sTER calculations are the same. In order to be consistent with the naming convention of avg-sBLEU we chose to write avg-sTER and not TER.

lators who are post-editing a particular document. Usually a localization company keeps a ranking of the hired translators; either we can use the ranking to exploit the relatedness between the translators or we can calculate their correlation based on a known previous post-edited data set. Here, we assume that the relatedness between the translators can be seen as the similarity between their post-edited segments given that the MT suggestions originated from the same system for all translators. This assumption is quite intuitive.

To compute the similarity, we calculate sentence level TER scores between the MT suggestions and the post-edited segments. In the cases where we do not have post-edited MT suggestions, for example BTEC where only multiple references are available, we simulate the conditions of post-editing by using the SMT translations provided by our own baseline system as MT suggestions. Now, the relatedness can be seen as the correlation between the sentence wise TER scores. Thus, we used the Pearson correlation coefficient (henceforth r).

Once it is calculated, we rescale these coefficients so that the values are between [0,1], instead of [-1,1] as given by r. We do this rescaling of correlations because matrix-based regularization is not able to handle the negative relatedness between the tasks. These values are computed on the corresponding development sets (which also contain post-edited segments from same translators) and are used to construct the A^{-1} matrix. Since the r is bi-directional, the interaction matrix is symmetric in nature. r values between the translators for IT and BTEC datasets are shown in Tables 7.3 and 7.4 respectively.

Now, we give a brief description of the various SMT systems involved in the experiments:

Translators	**T1**	**T2**	**T3**	**T4**
T1	1	0.82	0.83	0.70
T2	0.82	1	0.86	0.79
T3	0.83	0.86	1	0.77
T4	0.70	0.79	0.77	1

Table 7.3: Pearson correlation amongst translators on IT dataset.

Translators	**T1**	**T2**	**T3**	**T4**	**T5**	**T6**
T1	1	0.69	0.68	0.92	0.96	0.97
T2	0.69	1	0.57	0.64	0.64	0.66
T3	0.68	0.57	1	0.71	0.66	0.67
T4	0.92	0.64	0.71	1	0.90	0.91
T5	0.96	0.64	0.66	0.90	1	0.98
T6	0.97	0.66	0.67	0.91	0.98	1

Table 7.4: Pearson correlation amongst translators on BTEC dataset. These correlations are computed on a simulated environment.

Baseline: SMT models are trained on the domain specific training data; log linear weights are tuned on shuffled development set without any feedback and meta data about translator's ID.

Online: Feedback is added to the development set without the translator's ID. First, log linear weights are tuned on this development data by means of MERT; then, keeping them fixed to the optimal values, additional hyper parameters (used in *Online* system) are tuned again on the development set by means of the Simplex algorithm [Nelder and Mead, 1965]. This system contains a single weight vector for all the translators and is the same as explained in [Mathur et al., 2013].

MTL-pearson: Meta-information is added to the development set, and log linear weights are tuned on the dev set. There is an additional bias feature while using multi-task learning which is tuned using Simplex algorithm on the dev set. The elements of the interaction matrix are the rescaled Pearson coefficients. This system keeps track of k different weight vectors

for each translator.

MTL-halfupdate: The diagonal elements of the interaction matrix are set to 1, the off-diagonal elements to 0.5. This means that for every update in the current task $j \in 1 \dots k$ we do a half-update to the rest of the tasks. Note that this system does not need a development set to calculate the interaction matrix unlike MTL-pearson.

K-independent: The interaction matrix is set to be the identity matrix; it means that the tasks are independent of each other because no correlation is assumed between the translators. This system differs from *Online* system because here there is a separate instance of online learning for every translator, while in *Online* system there is a single instance of online learning for all the translators.

7.4.3 Results

Table 7.5 shows the avg-sTER[5] and avg-sBLEU scores over whole test set for all the systems. On the IT test set MTL-pearson shows gain of 1 avg-sBLEU points and 3.3 avg-sTER points over the Baseline system and 1.24 avg-sTER points over the strong *Online* system.

However, MTL-pearson does not perform well on BTEC test set, that is we are not able to capture well the task-relatedness in this scenario. Since the actual post-edit translations for BTEC are not available, we simulated them by generating MT suggestions from baseline system, which likely affects the effectiveness of the method. Nevertheless, MTL-halfupdate being a default system is able to capture quite well the correlation between the translators and significantly outperforms all the other systems. We can

[5]It has been shown in the past by Snover et al. [2006b] that in post-edit scenario TER has higher correlation than BLEU against the post-editing effort, and so we set our primary metric to be avg-sTER.

then conclude that if one does not have access to prior information about the translators for calculating the relatedness amongst them it is a good idea to back-off to use the default half-updates option.

On the Legal domain test set Multi-Task learning is not able to significantly improve over the online learning system. One reason for this could be the total number of sentences in the test set (90), that is each post-editor post edits only 22-25 sentences which is quite less in number as compared to other dataset where total number of sentences are 176 (IT) and 250 (BTEC) and hence each post-editor edits 44 and 42 sentences respectively. The other reason could be the relatively low repetition rate observed on the Legal test set.

System	IT		BTEC		Legal	
	avg-sTER	avg-sBLEU	avg-sTER	avg-sBLEU	avg-sTER	avg-sBLEU
Baseline	46.91	38.28	42.76	46.69	39.44	41.09
Online	44.86	39.21	42.64	46.72	38.96	41.56
MTL-pearson	**43.62**	**39.27**	41.76	47.17	**38.93**	**41.58**
MTL-halfupdate	44.63	38.94	**40.76**	**47.71**	38.93	41.58
K-independent	46.55	38.04	42.25	47.05	38.93	41.55

Table 7.5: BLEU scores achieved by using different techniques of online learning. Best BLEU and TER scores are marked in bold fonts.

Significance Testing: Here, we employ a non-parametric multiple hypothesis testing framework such as Friedman tests [Friedman, 1937]. The strategy for significance testing is as follows:

1. We mark epochs at every 10% of test set i.e. t epochs at 10%, 20% .. 100%.

2. At every epoch we measure the average performance of the system in question i.e. calculate avg-sTER.

3. In the end we have avg-sTER scores of five different systems at t different epochs.

4. The average performance of the aforementioned methods on the epochs can be seen as multiple systems trying to solve multiple problems. To calculate the p-values of these multiple systems, we use Friedman test [Friedman, 1937].

5. Once p-values are calculated, we use a post-hoc Holm's procedure [Holm, 1979] to check for the significance.

Results are reported in Table 7.6.

	P-Value		
Algorithm	IT	BTEC	Legal
MTL-pearson vs. Online	$0.022^{\diamond}$	$0.028^{\diamond}$	$0.066^{\diamond}$
MTL-halfupdate vs. Online	$0.003^{\diamond}$	$0.000^{\diamond}$	$0.066^{\diamond}$
K-Independent vs. Online	0.311	0.479	0.160
MTL-pearson vs. K-Independent	0.200	0.137	0.670
MTL-halfupdate vs. K-Independent	0.050	$0.000^{\diamond}$	0.670
MTL-pearson vs. MTL-halfupdate	0.500	$0.007^{\diamond}$	1.000

Table 7.6: p-values given by Friedman test. $\diamond$ depicts a significant difference between the systems that are being compared.

We plotted the incremental avg-sTER scores over t different epochs on all test sets in Figure 7.1.

First of all, it is worth to compare the plots of *MTL-pearson* and of *Online* systems on IT test set, for which the improvement of 1.24 avg-sTER points reported in Table 7.5 is significant (p=0.022 from Table 7.6). In fact, the gap between the *MTL-pearson* system and the *Online* system is visible in the plot only after the 6th epoch, that is for 6 out of 10 epochs differences are not large enough; nevertheless; the difference is significant. *MTL-halfupdate* performs better than any other system at least on 6 out of 10 epochs, but even on all epochs with respect to the *Online* system: this is why it outperforms the *Online* at 95% of confidence interval. Interesting to note that MTL-halfupdate is the best performing system till 6 epochs; after that, MTL-pearson becomes the best one: this basically says that for

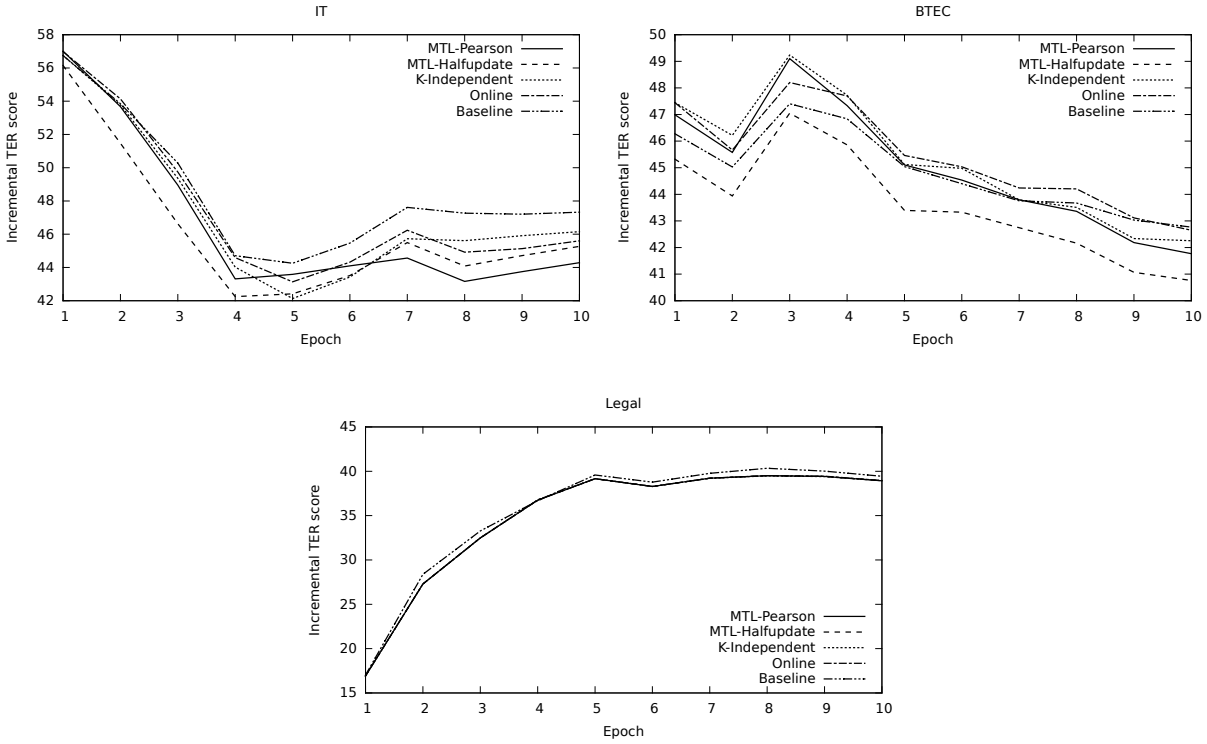

Figure 7.1: Learning curve of different systems on IT (top left), BTEC (top right) and Legal (bottom) test sets.

the starting 60% of the data the translators had a correlation of half with each other, while later they were as coherent as they were when they post-edited the development set (because MTL-pearson correlation is calculated on development set). This also means that the relatedness between the translators is evolving even while post-editing the same dataset.

On BTEC test set, MTL-halfupdate consistently outperforms all other MT systems on each epoch; this explains why it is significantly better than all other systems. On 9 epochs out of 10, MTL-pearson is better than the Online system; hence, the difference is significant. Results on BTEC put in evidence the importance of estimating a reliable interaction matrix to allow multi-task learning working at its best, but also that half-update is an effective back-off solution.

Significance tests on Legal test set[6] shows that MTL-* systems are better than the Online system with a p-value of 0.066.

So far, the evaluations were done on a shuffle of test set where the translators were assigned in sequence, i.e. first sentence to first translator, second to second and so on. This is usually not the case in a real world scenario, because a sentence can be assigned to any of the translators and not necessarily in sequence. To replicate such scenario, we developed an assigning scheme through which each translator is assigned an equal number of segments from a document to post-edit. The scheme is as follows:

1. For n translators, all possible permutations of the series $1...n$ is computed (total of $n!$).

2. The document to be post edited is divided in blocks of n sentences.

3. For each block we randomly pick a permutation series among the $n!$ choices, and assign it to the block in question.

Following this scheme, we created 100 different shuffles of the IT test set which are closer to the real life setting. Similar to the learning curve we built before, we averaged out avg-sTER scores over 100 shuffles on sequential epochs i.e. (10%, 20%...100% of data). Figure 7.2 reports the learning curves of different adaptive systems over epochal data.

Here, unlike in the previous case, for each of 176 sentences we have 100 different sentence wise TER scores using 5 different systems. Since just the IT domain is considered, data are more homogeneous and then we could apply *Approximate Randomization* [Noreen, 1989], a statistical test that is well established in the NLP community [Chinchor et al., 1993]. The test has been shown [Riezler and Maxwell, 2005] to be less prone to type-I

[6]The error curve in Legal domain shows an apparently surprising increasing trend. This is due to the nature of the test set where the starting sentences are easier to translate than the later ones.

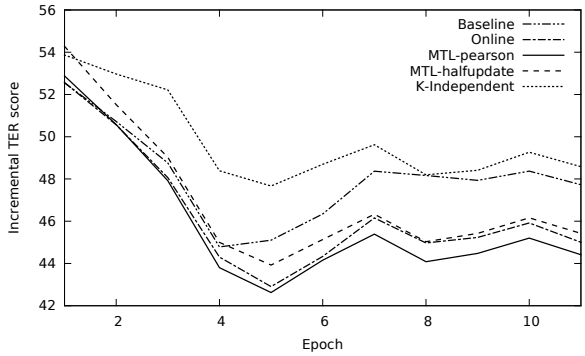

Figure 7.2: Learning curves of different systems on *shuffled* IT test set.

errors than the boostrap method [Efron and Tibshirani, 1993]. We report the significance results in Table 7.7.

Systems Compared	P-Value
Baseline vs. Online	0.001
Baseline vs. MTL-pearson	0.001
MTL-pearson vs. K-Independent	0.001
MTL-HalfUpdate vs. Online	0.04
MTL-pearson vs. MTL-HalfUpdate	0.001
Online vs. MTL-pearson	0.007

Table 7.7: p-values given by Approximate Randomization test on shuffled IT test set. All the reported results in the table are significant.

Even after the shuffling, we see that MTL-pearson system is resistant to the shuffles and still performs significantly better than any other system. However, we observe a contradictory information from the previous results; MTL-HalfUpdate system performs significantly worse than Online system over 100 shuffles, which means that MTL-HalfUpdate system is susceptible to randomly assignment of translators while Online is not. The same behaviour is observed in K-independent system where the system's performance is significantly worse than the Baseline system. All the other results remain consistent to what we observed in the previous case.

Table 7.8 shows an excerpt from the IT test set. The phrase *backup* in the source sentence (#21) is translated to *copia di riserva* by both K-

Independent and MTL-pearson systems but the translator post-edits the phrase in both translation hypotheses to *backup*. Later, in sentence #23 the phrase appears again and this time Multi-Task correctly outputs the translation of the phrase *backup* to *backup* but K-independent system is not able to correct the mistake. Reiterating, K-Independent system runs a single instance of online learning for each of the post-editors. In the example the first sentence is post-edited by translator #3 and the latter by translator#1, thus, the system is not able to recognize the mistake committed for the translator #1 and consequently cannot correct it for translator #3. While the system MTL-pearson learns jointly over the corrections by all the translators and thus is able to correct the translation hypothesis the next time.

Source - 21	with minimal copying of data from the production volume to **backup** volume ._#_3
K-Ind - 21	con un minimo la copia di dati dal volume di produzione per il volume della **copia di riserva** .
Multi-Task - 21	con un minimo la copia di dati dal volume di produzione per il volume della **copia di riserva** .
Post-Edit - 21	con una copia minima dei dati dal volume di produzione nel volume di **backup** .
Source - 23	you create a **backup** and after it completes _#_1
K-Ind - 23	possibile creare una **copia di riserva** e dopo il completamento
Multi-Task - 23	creare un **backup** e dopo il completamento
Post-Edit - 23	possibile creare un **backup** e , al suo completamento

Table 7.8: Example from the IT test set. Here *Multi-Task* refers to MTL-pearson system and *K-ind* is K-independent system.

Overall, the results show that Multi-Task learning outperforms the existing standard MT and the strong online learning systems. If we have the meta information on the post-editors apriori, i.e. their mutual correlation, we can boost the performance of the adaptive system. One can use the MTL-pearson system if the correlation matrix can be calculated accurately; if not, it is preferable to back-off to MTL-halfupdate system.

7.5 Conclusion

In a CAT framework, we addressed the problem of adapting a single MT system to multiple post-editions, i.e. to an incoming stream of feedback from different translators. In such a situation, standard online learning methods can lead to incoherent translations by the MT system. To the best of our knowledge, this kind of problem has never been addressed before. As a solution we propose to adopt a multi-task learning scheme, which relies on the correlation amongst the translators computed using prior knowledge; the online learner is then constrained to take into account the relatedness amongst the translators.

Different online systems have been compared, and online multi-task learning MT systems outperformed in most cases strong online learning MT baselines. Whenever not enough information about the correlation amongst the translators is available, our experimental outcomes suggest to use multi-task learning with half-updates, which is a good generalization of the interaction between the translators. We also compared the Multi-Task approach to the K-Independent system where each translator has been allotted an online learning MT system; evidently, multi-task also fared better against this system setup. Moreover, MTL can also be applied to tune the log-linear weights of MT models when multiple references are given.

Chapter 8

Conclusion

8.1 Summary

There has been an increasing demand for high quality translations for the localization of products, dissemination purpose, translation of patents etc. Due to the high demand, the pressure on translation industry to deliver (the translations in a short time) is more than ever. In this thesis, we target our efforts towards helping translation industry by providing solutions to build a high quality machine translation (MT) system. To build a high quality translation, we need to adapt the MT system to the target domain. Often, there is a mismatch between the domain for which training data are available and this target domain. We analyse these business use cases where:

1. There is an urgent need of domain adapted MT system using a minimalistic resource of in-domain data;

2. Consistent translation of terminologies across different topics in text is required.

Keeping this in mind, throughout this PhD, we have examined various techniques of adaptation for MT systems such as domain adaptation ap-

proaches described in Chapter 3 and topic adaptation approaches described in Chapter 4.

Our other major focus is on providing solutions for the translation industry, where there is a need to deliver high quality MT systems. So far, we are at a stage where machine translation is not as good as human translation i.e. a human is still required to produce high quality translations. At the same time, the data is increasing at a fast pace and not all the data can be translated manually. We need to find a middle ground. Our job now is to support the translators and reduce their translation effort as much as we can. We also extend this thesis to include improvements for these MT systems in their role to support the translators. We analyse various business use cases in this scenario where:

1. The MT system needs to adapt to the corrections of translator while the translator is translating a document.

2. A single MT system needs to adapt to the corrections of multiple translators at the same time while they are translating a document.

8.2 Future Works

While most of the machine translation research in the recent times have been in the area of neural machine translation (NMT); statistical MT approaches have not been completely written off (yet) by the MT community. Neural approaches are taking off because they have broken the plateau performance of the phrase-based statistical MT (over the years) and have shown a promising performance growth over time [Bojar et al., 2015, 2016]. Applying the adaptation ideas proposed in this thesis for statistical MT in current NMT approaches could be one of the directions for the future research work. This thesis will act as a good reference for the (MT) scientists who are working or will work on adaptation approaches for NMT.

Having said that, we do know that there are some areas of improvement within the thesis. For example in Chapter 3, we discussed the problem of tuning the log-linear weights of SMT model when there is no available indomain parallel data to tune on. Our strategy to treat this case as a supervised learning problem using BLEU-PT as a feature works well (c.f. Section 3.6). The issue we see in this approach is that only one feature was used which could easily lead to the problems of overfitting or underfitting the data. An obvious extension of this work is to make it robust and explore ways to add more features in the training data.

In Chapter 5, we successfully leveraged MIRA [Crammer and Singer, 2003] as our online algorithm to update the weights of the log-linear model. MIRA even successfully handled the same model when it was extended with an additional set of sparse features. However, there have been various online algorithms introduced recently for tuning the weights of SMT models with sparse features such as AdaGrad [Green et al., 2013b] with pairwise ranked optimization objective [Hopkins and May, 2011]. A straight away extension of our work here is thus, applying different online algorithms in the same settings.

In the same chapter, we also discussed the online learning framework where the feedback to the MT system is a post-edited segment. This setting is tailored for a self-adaptive machine translation system running in the background of the CAT tool. Prior to that, we discussed another problem scenario of item title translation in e-Commerce domain in Chapter 4. These titles describing the inventory in the e-Commerce websites are a major factor in a user's decision. They might decided to buy or not buy an item depending on the readability of the item title. Given that these titles are noisy and contain a lot of jargon text; translation of same could have a bad cascading effect on translation quality and lead to an undesirable translation of an item title. If the translation quality is bad, users may

skip buying that particular item; on the other hand, if its good the users may buy the item. These indicators (buy/not buy) from users can act as an implicit (although weak) feedback for the translation system which can be used to improve the translation system. Recent studies such as Sokolov et al. [2016] targeted the exact problem. Their work on bandit learning with weak feedback for machine translation have opened up an exciting new branch of research in machine translation.

In the previous chapter, we looked at ways of adapting a single MT system with feedback from multiple translators at the same time. The techniques we used took into assumption that the translators do not evolve with time i.e. the relationship (matrix) between the translators is the same throughout the post-editing phase of the document. In a real world, this may not be true. A translator whose style of correction differs a lot from that of the other translators (translating the same document), could may as well evolve over time and begin editing in the same style as other translators are. This problem where the relationship between the tasks evolve over time has been reported in the paper by Saha et al. [2011] although for different classification tasks. The authors applied various divergence rules to learn the task relationship matrix in an online learning framework. These methods could be applied in our use case to learn the relationship matrix between the translators.

All of our work on user and multi-user adaptation is publicly available in Moses on github.[1] The work on domain adaptation (Chapter 3) and topic adaptation (Chapter 4) were sponsored by private companies and thus, we cannot release the software due to copyright issues.

[1] `https://github.com/mtresearcher`

Bibliography

E.H. Hovy. Deepening wisdom or compromised principles?-the hybridization of statistical and symbolic mt systems. *IEEE Expert*, 11(2):16–18, Apr 1996. ISSN 0885-9000. doi: 10.1109/64.491313.

Donald A. DePalma, Hèléne Pielmeier, Robert G. Stewart, and Stephen Henderson. The language services market: 2015. *Common Sense Advisory*, 2015.

Spence Green, Jeffrey Heer, and Christopher D. Manning. The efficacy of human post-editing for language translation. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 439–448. ACM, 2013a.

Marcello Federico, Alessandro Cattelan, and Marco Trombetti. Measuring user productivity in machine translation enhanced computer assisted translation. In *Proceedings of the Tenth Conference of the Association for Machine Translation in the Americas (AMTA)*, 2012. URL `http://www.mt-archive.info/AMTA-2012-Federico.pdf`.

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

Ventsislav Zhechev and Josef van Genabith. Seeding statistical machine translation with translation memory output through tree-based struc-

tural alignment. In *Proceedings of the 4th Workshop on Syntax and Structure in Statistical Translation*, pages 43–51, Beijing, China, August 2010. Coling 2010 Organizing Committee. URL `http://www.aclweb.org/anthology/W10-3806`.

Franz Josef Och. Minimum Error Rate Training in Statistical Machine Translation. In Erhard Hinrichs and Dan Roth, editors, *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, pages 160–167, 2003a. URL `http://www.aclweb.org/anthology/P03-1021.pdf`.

Prashant Mathur, Marcello Federico, Selçuk Köprü, Shahram Khadivi, and Hassan Sawaf. Topic Adaptation for Machine Translation of e-Commerce Content. In *Proceedings of 15th Machine Translation Summit*, Miami, US, October 2015. Machine Translation Summit XV.

Prashant Mathur and Mauro Cettolo. Optimized MT Online Learning in Computer Assisted Translation. In *Proceedings of Interactive and Adaptive Machine Translation Workshop*, Vancouver, Canada, October 2014. American Machine Translation Association.

Prashant Mathur, Mauro Cettolo, Marcello Federico, and José G.C. de Souza. Online Multi User Adaptive SMT. In *Proceedings of American Machine Translation Association*, Vancouver, Canada, October 2014a. American Machine Translation Association.

Prashant Mathur, Sriram Venkatapathy, and Nicola Cancedda. Fast domain adaptation of smt models without in-domain parallel data. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 1114–1123, Dublin, Ireland, August 2014b. Dublin City University and Association for

Computational Linguistics. URL http://www.aclweb.org/anthology/C14-1105.

Prashant Mathur, Mauro Cettolo, and Marcello Federico. Online Learning Approaches in Computer Assisted Translation. In *Proceedings of the Eighth Workshop on Statistical Machine Translation*, pages 301–308, Sofia, Bulgaria, August 2013. Association for Computational Linguistics.

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

S. Nirenburg, J. Carbonell, M. Tomita, and K. Goodman. *Machine Translation: A knowledge-based approach*. Morgan Kaufman, San Mateo, 1992.

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

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. Sequence to sequence learning with neural networks. In *Proceedings of the 27th International Conference on Neural Information Processing Systems*, NIPS'14, pages 3104–3112, Cambridge, MA, USA, 2014. MIT Press. URL http://dl.acm.org/citation.cfm?id=2969033.2969173.

Peter F. Brown, Stephen A. Della Pietra, Vincent J. Della Pietra, and Robert L. Mercer. The mathematics of statistical machine translation: Parameter estimation. *Computational Linguistics*, 19(2):263–312, 1993. URL http://aclweb.org/anthology-new/J/J93/J93-2003.pdf.

Philipp Koehn, Franz Josef Och, and Daniel Marcu. Statistical phrase-based translation. In *Proceedings of HLT-NAACL 2003*, pages 127–133,

Edmonton, Canada, 2003. URL http://aclweb.org/anthology-new/N/N03/N03-1017.pdf.

Kenji Yamada and Kevin Knight. A syntactic-based statistical translation model. In *Proceedings of the 39th Meeting of the Association for Computational Linguistics (ACL)*, pages 523–530, Toulouse, France, 2001.

David Chiang, Adam Lopez, Nitin Madnani, Christof Monz, Philip Resnik, and Michael Subotin. The hiero machine translation system: Extensions, evaluation, and analysis. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pages 779–786, Vancouver, British Columbia, Canada, October 2005. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/H/H05/H05-1098.

Franz Josef Och and Hermann Ney. Discriminative training and maximum entropy models for statistical machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 295–302, Philadelhpia, PA, 2002. URL http://aclweb.org/anthology-new/P/P02/P02-1038.pdf.

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

Chris Dyer, Victor Chahuneau, and Noah A. Smith. A simple, fast, and effective reparameterization of ibm model 2. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 644–648, Atlanta, Georgia, June 2013. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/N13-1073.

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*, pages 177–180, Prague, Czech Republic, 2007a. URL `http://aclweb.org/anthology-new/P/P07/P07-2045.pdf`.

Monica Rogati. *Domain adaptation of translation models for multilingual applications*. ProQuest, 2009.

Philipp Koehn and Kevin Knight. Knowledge sources for word-level translation models. In *In Proceedings of the 2001 Conference on Empirical Methods in Natural Language Processing*, pages 27–35, 2001.

Robert C. Moore and William Lewis. Intelligent selection of language model training data. In *ACL (Short Papers)*, pages 220–224, 2010.

Amittai Axelrod, Xiaodong He, and Jianfeng Gao. Domain adaptation via pseudo in-domain data selection. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '11, pages 355–362, Stroudsburg, PA, USA, 2011. Association for Computational Linguistics. ISBN 978-1-937284-11-4. URL `http://dl.acm.org/citation.cfm?id=2145432.2145474`.

Almut Silja Hildebrand, Matthias Eck, Stephan Vogel, and Alex Waibel. Adaptation of the translation model for statistical machine translation based on information retrieval. In *Proceedings of the 10th Conference of the European Association for Machine Translation (EAMT)*, pages 133–142, Budapest, May 2005.

Gerard Salton and Michael J. McGill. *Introduction to Modern Information Retrieval*. McGraw-Hill, Inc., New York, NY, USA, 1986. ISBN 0070544840.

George Foster and Roland Kuhn. Mixture-model adaptation for smt. In *Proceedings of the Second Workshop on Statistical Machine Translation*, StatMT '07, pages 128–135, Stroudsburg, PA, USA, 2007. Association for Computational Linguistics. URL `http://dl.acm.org/citation.cfm?id=1626355.1626372`.

Philipp Koehn and Josh Schroeder. Experiments in Domain Adaptation for Statistical Machine Translation. In *Proceedings of the Second Workshop on Statistical Machine Translation*, pages 224–227, Prague, Czech Republic, June 2007a. Association for Computational Linguistics. URL `http://www.aclweb.org/anthology/W/W07/W07-0233`.

Jorge Civera and Alfons Juan. Domain adaptation in statistical machine translation with mixture modelling. In *Proceedings of the Second Workshop on Statistical Machine Translation*, pages 177–180, Prague, Czech Republic, June 2007. Association for Computational Linguistics. URL `http://www.aclweb.org/anthology/W/W07/W07-0222`.

Majid Razmara, George Foster, Baskaran Sankaran, and Anoop Sarkar. Mixing multiple translation models in statistical machine translation. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*, pages 940–949. Association for Computational Linguistics, 2012.

Nan Duan, Mu Li, Dongdong Zhang, and Ming Zhou. Mixture model-based minimum bayes risk decoding using multiple machine translation systems. In *Proceedings of the 23rd International Conference on Compu-*

tational Linguistics, pages 313–321. Association for Computational Linguistics, 2010.

Arianna Bisazza, Nick Ruiz, and Marcello Federico. Fill-up versus Interpolation Methods for Phrase-based SMT Adaptation. In *International Workshop on Spoken Language Translation (IWSLT)*, pages 136–143, San Francisco, CA, 2011a.

Pratyush Banerjee, Jinhua Du, Baoli Li, Sudip Kumar Naskar, Andy Way, and Josef Van Genabith. Combining multi-domain statistical machine translation models using automatic classifiers. In *Proceedings of 9th Conference of the Association for Machine Translation in the Americas*, 2010.

Rico Sennrich. Perplexity minimization for translation model domain adaptation in statistical machine translation. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*, EACL '12, pages 539–549, Stroudsburg, PA, USA, 2012a. Association for Computational Linguistics. ISBN 978-1-937284-19-0. URL `http://dl.acm.org/citation.cfm?id=2380816.2380881`.

Rico Sennrich. Mixture-modeling with unsupervised clusters for domain adaptation in statistical machine translation. In *Proceedings of the 16th Annual Conference of the European Association of Machine Translation (EAMT)*, 2012b.

Zhengxian Gong, Min Zhang, and Guodong Zhou. Cache-based Document-level Statistical Machine Translation. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 909–919, Edinburgh, Scotland, UK., July 2011. Association for Computational Linguistics. URL `http://www.aclweb.org/anthology/D11-1084`.

Nick Ruiz and Marcello Federico. Topic Adaptation for Lecture Translation Through Bilingual Latent Semantic Models. In *Proceedings of the Sixth Workshop on Statistical Machine Translation*, WMT '11, pages 294–302, Stroudsburg, PA, USA, 2011. Association for Computational Linguistics. ISBN 978-1-937284-12-1. URL `http://dl.acm.org/citation.cfm?id=2132960.2132998`.

Vladimir Eidelman, Jordan Boyd-Graber, and Philip Resnik. Topic Models for Dynamic Translation Model Adaptation. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 115–119, Jeju Island, Korea, July 2012. Association for Computational Linguistics. URL `http://www.aclweb.org/anthology/P12-2023`.

Jinsong Su, Hua Wu, Haifeng Wang, Yidong Chen, Xiaodong Shi, Huailin Dong, and Qun Liu. Translation Model Adaptation for Statistical Machine Translation with Monolingual Topic Information. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers - Volume 1*, ACL '12, pages 459–468, Stroudsburg, PA, USA, 2012. Association for Computational Linguistics. URL `http://dl.acm.org/citation.cfm?id=2390524.2390589`.

Amit Gruber, Yair Weiss, and Michal Rosen-Zvi. Hidden topic Markov models. In *Proceedings of the International Conference on Artificial Intelligence and Statistics*, pages 163–170, 2007.

Xinyan Xiao, Deyi Xiong, Min Zhang, Qun Liu, and Shouxun Lin. A Topic Similarity Model for Hierarchical Phrase-based Translation. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers - Volume 1*, ACL '12, pages 750–758, Strouds-

burg, PA, USA, 2012. Association for Computational Linguistics. URL http://dl.acm.org/citation.cfm?id=2390524.2390630.

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

E. Hellinger. Neue begrndung der theorie quadratischer formen von unendlichvielen vernderlichen. *Journal fr die reine und angewandte Mathematik*, 136:210–271, 1909. URL http://eudml.org/doc/149313.

Sanjika Hewavitharana, Dennis Mehay, Sankaranarayanan Ananthakrishnan, and Prem Natarajan. Incremental Topic-Based Translation Model Adaptation for Conversational Spoken Language Translation. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 697–701, Sofia, Bulgaria, August 2013. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/P13-2122.

David M Blei, Andrew Y Ng, and Michael I Jordan. Latent Dirichlet Allocation. *the Journal of machine Learning research*, 3:993–1022, 2003.

J. Lin. Divergence measures based on the shannon entropy. *IEEE Trans. Inf. Theor.*, 37(1):145–151, September 2006. ISSN 0018-9448. doi: 10.1109/18.61115. URL http://dx.doi.org/10.1109/18.61115.

Eva Hasler, Barry Haddow, and Philipp Koehn. Combining domain and topic adaptation for SMT. In *Proceedings of the Eleventh Conference of the Association for Machine Translation in the Americas (AMTA)*, volume 1, pages 139–151, 2014. URL http://www.mt-archive.info/10/AMTA-2014-Hasker.pdf.

Nicola Bertoldi, Mauro Cettolo, and Marcello Federico. Cache-based Online Adaptation for Machine Translation Enhanced Computer Assisted

Translation. In *Proceedings of the MT Summit XIV*, pages 35–42, Nice, France, September 2013.

Michael Denkowski, Chris Dyer, and Alon Lavie. Learning from post-editing: Online model adaptation for statistical machine translation. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pages 395–404, Gothenburg, Sweden, April 2014. Association for Computational Linguistics. URL `http://www.aclweb.org/anthology/E14-1042`.

Daniel Ortiz-Martínez, Ismal García-Varea, and Francisco Casacuberta. Online learning for interactive statistical machine translation. In *Proceedings of Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the AC (HLT-NAACL '10)*, 2010.

Pascual Martínez-Gómez, Germán Sanchis-Trilles, and Francisco Casacuberta. Online learning via dynamic reranking for computer assisted translation. In *Proceedings of the 12th international conference on Computational linguistics and intelligent text processing - Volume Part II*, CICLing'11, pages 93–105, Berlin, Heidelberg, 2011. Springer-Verlag. ISBN 978-3-642-19436-8. URL `http://dl.acm.org/citation.cfm?id=1964750.1964759`.

Mauro Cettolo, Nicola Bertoldi, and Marcello Federico. The Repetition Rate of Text as a Predictor of the Effectiveness of Machine Translation Adaptation. In *Proceedings of the 11th Biennial Conference of the Association for Machine Translation in the Americas (AMTA 2014)*, pages 166–179, Vancouver, BC, Canada, 2014.

Laurent Nepveu, Guy Lapalme, Philippe Langlais, and George Foster. Adaptive language and translation models for interactive machine translation. In Dekang Lin and Dekai Wu, editors, *Proceedings of EMNLP*

2004, pages 190–197, Barcelona, Spain, July 2004. Association for Computational Linguistics.

Jörg Tiedemann. Context adaptation in statistical machine translation using models with exponentially decaying cache. In *Proceedings of the 2010 Workshop on Domain Adaptation for Natural Language Processing*. Association for Computational Linguistics, 2010.

Daniel Hardt and Jakob Elming. Incremental re-training for post-editing SMT. In *AMTA'10*, 2010.

Katharina Wäschle, Patrick Simianer, Nicola Bertoldi, Stefan Riezler, and Marcello Federico. Generative and discriminative methods for online adaptation in smt. In *Proceedings of the MT Summit XIV*, pages 11–18, Nice, France, September 2013.

Mauro Cettolo, Marcello Federico, and Nicola Bertoldi. Mining parallel fragments from comparable texts. In *Proc. of the International Workshop on Spoken Language Translation (IWSLT)*, Paris, France, 2010.

Percy Liang and Dan Klein. Online em for unsupervised models. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, NAACL '09, pages 611–619, Stroudsburg, PA, USA, 2009. Association for Computational Linguistics. ISBN 978-1-932432-41-1. URL `http://dl.acm.org/citation.cfm?id=1620754.1620843`.

Masa-Aki Sato and Shin Ishii. On-line em algorithm for the normalized gaussian network. *Neural Comput.*, 12(2):407–432, February 2000. ISSN 0899-7667. doi: 10.1162/089976600300015853. URL `http://dx.doi.org/10.1162/089976600300015853`.

Olivier Cappé and Eric Moulines. Online em algorithm for latent data models. *CoRR*, abs/0712.4273, 2007. URL `http://dblp.uni-trier.de/db/journals/corr/corr0712.html#abs-0712-4273`.

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

Abby Levenberg, Chris Callison-Burch, and Miles Osborne. Stream-based translation models for statistical machine translation. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics (HLT-NAACL'10)*, pages 394–402, Los Angeles, California, June 2010.

Adam Lopez. Tera-scale translation models via pattern matching. In *Proceedings of the 22nd International Conference on Computational Linguistics - Volume 1*, COLING '08, pages 505–512, Stroudsburg, PA, USA, 2008. Association for Computational Linguistics. ISBN 978-1-905593-44-6. URL `http://dl.acm.org/citation.cfm?id=1599081.1599145`.

Marion Potet, Emmanuelle Esperana-rodier, Laurent Besacier, and Herv Blanchon. Preliminary Experiments on Using Users Post-Editions to Enhance a SMT System. In *EAMT (European Association for Machine Translation) Conference*, Leuven (Belgium), may 2011.

Nicolò Cesa-Bianchi, Gabriele Reverberi, and Sandor Szedmak. Online learning algorithms for computer-assisted translation. Technical report, SMART (`www.smart-project.eu`), 2008.

Pascual Martínez-Gómez, Germán Sanchis-Trilles, and Francisco Casacuberta. Online adaptation strategies for statistical machine translation in post-editing scenarios. *Pattern Recogn.*, 45(9):3193–3203, Septem-

ber 2012. ISSN 0031-3203. doi: 10.1016/j.patcog.2012.01.011. URL http://dx.doi.org/10.1016/j.patcog.2012.01.011.

Spence Green, Sida Wang, Daniel Cer, and Christopher D. Manning. Fast and adaptive online training of feature-rich translation models. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 311–321, Sofia, Bulgaria, August 2013b. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/P13-1031.

Koby Crammer and Yoram Singer. Ultraconservative online algorithms for multiclass problems. *J. Mach. Learn. Res.*, 3:951–991, March 2003. ISSN 1532-4435.

Mark Hopkins and Jonathan May. Tuning as ranking. In *Proceedings of 2011 Conference on Empirical Methods in Natural Language Processing (EMNLP'11)*, 2011.

Mauro Cettolo and Marcello Federico. Minimum error training of log-linear translation models. In *Proceedings of the International Workshop on Spoken Language Translation (IWSLT)*, pages 103–106, Kyoto, Japan, September 2004.

Percy Liang, Alexandre Bouchard-Côté, Dan Klein, and Ben Taskar. An end-to-end discriminative approach to machine translation. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 761–768, Sydney, Australia, July 2006. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/P/P06/P06-1096.

Michael Collins. Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In *Proceedings of EMNLP*, 2002.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. Research Report RC22176, IBM Research Division, Thomas J. Watson Research Center, 2001.

F. Rosenblatt. The Perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65:386–408, 1958.

Eva Hasler, Barry Haddow, and Philipp Koehn. Margin Infused Relaxed Algorithm for Moses. *The Prague Bulletin of Mathematical Linguistics*, 96:69–78, 2011.

R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification*. John Wiley & Sons, New York, NY, 2001. 2nd ed.

John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *J. Mach. Learn. Res.*, 12:2121–2159, July 2011. ISSN 1532-4435. URL `http://dl.acm.org/citation.cfm?id=1953048.2021068`.

Ulrich Germann. Dynamic phrase tables for machine translation in an interactive post-editing scenario. In *AMTA 2014 Workshop on Interactive and Adaptive Machine Translation, Vancouver, BC, Canada*, pages 20–31, 2014.

Richard Caruana. Multitask learning: A knowledge-based source of inductive bias. In *Proceedings of the Tenth International Conference on Machine Learning*, pages 41–48. Morgan Kaufmann, 1993.

Kevin Duh, Katsuhito Sudoh, Hajime Tsukada, Hideki Isozaki, and Masaaki Nagata. N-best reranking by multitask learning. In *Proceedings of the Joint Fifth Workshop on Statistical Machine Translation and MetricsMATR*, WMT '10, pages 375–383, Stroudsburg, PA, USA, 2010. Association for Computational Linguistics. ISBN 978-1-932432-71-8. URL `http://dl.acm.org/citation.cfm?id=1868850.1868907`.

Patrick Simianer, Katharina Wschle, and Stefan Riezler. Multi-task minimum error rate training for smt. *Prague Bull. Math. Linguistics*, 96: 99–108, 2011.

Patrick Simianer, Stefan Riezler, and Chris Dyer. Joint feature selection in distributed stochastic learning for large-scale discriminative training in SMT. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (ACL'12)*, 2012.

Giovanni Cavallanti, Nicolò Cesa-Bianchi, and Claudio Gentile. Linear algorithms for online multitask classification. *J. Mach. Learn. Res.*, 11: 2901–2934, December 2010. ISSN 1532-4435. URL `http://dl.acm.org/citation.cfm?id=1756006.1953026`.

Avishek Saha, Piyush Rai, Hal Daum III, and Suresh Venkatasubramanian. Online learning of multiple tasks and their relationships. In Geoffrey J. Gordon, David B. Dunson, and Miroslav Dudk, editors, *AISTATS*, volume 15 of *JMLR Proceedings*, pages 643–651. JMLR.org, 2011. URL `http://dblp.uni-trier.de/db/journals/jmlr/jmlrp15.html#SahaRDV11`.

Trevor Cohn and Lucia Specia. Modelling annotator bias with multi-task gaussian processes: An application to machine translation quality estimation. In *ACL (1)*, pages 32–42. The Association for Computer Lin-

guistics, 2013. ISBN 978-1-937284-50-3. URL http://dblp.uni-trier.de/db/conf/acl/acl2013-1.html#CohnS13.

José G. C. de Souza, Marco Turchi, and Matteo Negri. Machine translation quality estimation across domains. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 409–420, Dublin, Ireland, August 2014. Dublin City University and Association for Computational Linguistics. URL http://www.aclweb.org/anthology/C14-1040.

Robert Michael Lewis, Virginia Torczon, and Michael W. Trosset. Direct search methods: then and now. *Journal of Computational and Applied Mathematics*, 124(12):191 – 207, 2000. ISSN 0377-0427. Numerical Analysis 2000. Vol. IV: Optimization and Nonlinear Equations.

Tagyoung Chung and Michel Galley. Direct error rate minimization for statistical machine translation. In *Proceedings of the Seventh Workshop on Statistical Machine Translation*, WMT '12, pages 468–479, Stroudsburg, PA, USA, 2012. Association for Computational Linguistics. URL http://dl.acm.org/citation.cfm?id=2393015.2393081.

Sara Stymne, Christian Hardmeier, Jrg Tiedemann, and Joakim Nivre. Tunable distortion limits and corpus cleaning for smt. In *In Proceedings of the Eighth Workshop on Statistical Machine Translation*, pages 223–229, 2013.

Christian Hardmeier, Sara Stymne, Jrg Tiedemann, and Joakim Nivre. Docent: A document-level decoder for phrase-based statistical machine translation. In *In Proceedings of the 51st Annual Meeting of the ACL, Demonstration*, 2013.

James Bergstra and Yoshua Bengio. Random search for hyper-parameter optimization. *Journal of Machine Learning Research*, 13(Feb):281–305, 2012.

Jasper Snoek, Hugo Larochelle, and Ryan Prescott Adams. Practical bayesian optimization of machine learning algorithms. In *Advances in Neural Information Processing Systems 25*, 12/2012 2012.

J. Larsen, L. K. Hansen, C. Svarer, and M. Ohlsson. Design and regularization of neural networks: The optimal use of a validation set, 1996.

A. Moore and M. Lee. Efficient algorithms for minimizing cross validation error. In *Proceedings of the Eleventh International Conference on Machine Learning*, pages 190–198, 1994.

Nicola Cancedda. Private access to phrase tables for statistical machine translation. In *ACL (2)*, pages 23–27, 2012.

Rico Sennrich, Holger Schwenk, and Walid Aransa. A multi-domain translation model framework for statistical machine translation. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 832–840, Sofia, Bulgaria, August 2013. Association for Computational Linguistics. URL `http://www.aclweb.org/anthology/P13-1082`.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-jing Zhu. BLEU : a Method for Automatic Evaluation of Machine Translation. In *Computational Linguistics*, volume pages, pages 311–318, 2002a. doi: 10.3115/1073083.1073135. URL `http://portal.acm.org/citation.cfm?id=1073083.1073135`.

Mehryar Mohri. Finite-state transducers in language and speech processing. *Computational Linguistics*, 23(2):269–311, June 1997. ISSN 0891-2017. URL http://dl.acm.org/citation.cfm?id=972695.972698.

Franz Josef Och. Minimum Error Rate Training in Statistical Machine Translation. In Erhard Hinrichs and Dan Roth, editors, *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, pages 160–167, 2003b. URL http://www.aclweb.org/anthology/P03-1021.pdf.

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)*, pages 764–773, Prague, Czech Republic, June 2007a. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/D/D07/D07-1080.

Pavel Pecina, Antonio Toral, and Josef van Genabith. Simple and effective parameter tuning for domain adaptation of statistical machine translation. In *COLING*, pages 2209–2224, 2012.

Rich Caruana. Multitask learning. *Mach. Learn.*, 28(1):41–75, July 1997. ISSN 0885-6125. doi: 10.1023/A:1007379606734. URL http://dx.doi.org/10.1023/A:1007379606734.

Yi Yang, Heng Tao Shen, Zhigang Ma, Zi Huang, and Xiaofang Zhou. l 2, 1-norm regularized discriminative feature selection for unsupervised learning. In *Proceedings of the Twenty-Second international joint conference on Artificial Intelligence-Volume Volume Two*, pages 1589–1594. AAAI Press, 2011.

Jason R. Smith, Herve Saint-Amand, Magdalena Plamada, Philipp Koehn, Chris Callison-Burch, and Adam Lopez. Dirt cheap web-scale parallel text from the common crawl. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1374–1383, Sofia, Bulgaria, August 2013. Association for Computational Linguistics. URL `http://www.aclweb.org/anthology/P13-1135`.

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

Mauro Cettolo, Christian Girardi, and Marcello Federico. Wit^3: Web inventory of transcribed and translated talks. In *Proceedings of the 16th Conference of the European Association for Machine Translation (EAMT)*, pages 261–268, Trento, Italy, May 2012a.

Jörg Tiedemann. Parallel data, tools and interfaces in opus. In Nicoletta Calzolari (Conference Chair), Khalid Choukri, Thierry Declerck, Mehmet Uur Doan, Bente Maegaard, Joseph Mariani, Jan Odijk, and Stelios Piperidis, editors, *Proceedings of the Eight International Conference on Language Resources and Evaluation (LREC'12)*, Istanbul, Turkey, may 2012. European Language Resources Association (ELRA). ISBN 978-2-9517408-7-7.

Adam Lopez and Philipp Resnik. Word-based alignment, phrase-based translation: What's the link? In *In Proceedings of AMTA*, pages 90–99,

2006.

Qin Gao and Stephan Vogel. Parallel implementations of word alignment tool. In *Software Engineering, Testing, and Quality Assurance for Natural Language Processing*, SETQA-NLP '08, pages 49–57, Stroudsburg, PA, USA, 2008a. Association for Computational Linguistics. ISBN 978-1-932432-10-7. URL `http://dl.acm.org/citation.cfm?id=1622110.1622119`.

Andreas Stolcke. Srilm - an extensible language modeling toolkit. In *Proceedings of ICSLP*, Denver, Colorado, 2002. URL `http://www.speech.sri.com/cgi-bin/run-distill?papers/icslp2002-srilm.ps.gz`.

Stanley F. Chen and Joshua Goodman. An empirical study of smoothing techniques for language modeling. *Computer Speech and Language*, 4 (13):359–393, 1999.

Jonathan H Clark, Chris Dyer, Alon Lavie, and Noah A Smith. Better hypothesis testing for statistical machine translation: Controlling for optimizer instability. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: short papers-Volume 2*, pages 176–181. Association for Computational Linguistics, 2011a.

Barry Haddow and Philipp Koehn. Analysing the effect of out-of-domain data on smt systems. In *Proceedings of the Seventh Workshop on Statistical Machine Translation*, pages 422–432, Montreal, Canada, June 2012. Association for Computational Linguistics.

Philipp Koehn and Josh Schroeder. Experiments in domain adaptation for statistical machine translation. In *Proceedings of the Second Workshop on Statistical Machine Translation*, StatMT '07, pages 224–227, Strouds-

burg, PA, USA, 2007b. Association for Computational Linguistics. URL http://dl.acm.org/citation.cfm?id=1626355.1626388.

Nicola Bertoldi and Marcello Federico. Domain adaptation for statistical machine translation with monolingual resources. In *Proceedings of the Fourth Workshop on Statistical Machine Translation*, StatMT '09, pages 182–189, Stroudsburg, PA, USA, 2009. Association for Computational Linguistics. URL http://dl.acm.org/citation.cfm?id=1626431.1626468.

Jyotti Guha and Carmen Heger. Machine Translation for Global E-Commerce on eBay. In *Proceedings of the AMTA*, volume 2: MT Users, pages 31–37, 2014.

Jose Sanchez and Tanya Badeka. Linguistic QA for MT of user-generated content at eBay. In *Proceedings of the AMTA*, volume 2: MT Users, pages 1–24, 2014.

Silvio Picinini. Challenges of Machine Translation for User Generated Content: Queries from Brazilian users. In *Proceedings of the AMTA*, volume 2: MT Users, pages 55–65, 2014.

Thomas Hofmann. Probabilistic Latent Semantic Analysis. In *Proceedings of the Fifteenth conference on Uncertainty in artificial intelligence*, pages 289–296. Morgan Kaufmann Publishers Inc., 1999.

Daniel Ramage, Evan Rosen, Jason Chuang, Christopher D. Manning, and Daniel A. McFarland. Topic modeling for the social sciences. In *Workshop on Applications for Topic Models, NIPS*, 2009. URL http://vis.stanford.edu/papers/topic-modeling-social-sciences.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine

Moran, Richard Zens, et al. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th annual meeting of the ACL on interactive poster and demonstration sessions*, pages 177–180. Association for Computational Linguistics, 2007b.

Colin Cherry and George Foster. Batch tuning strategies for statistical machine translation. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 427–436, Montréal, Canada, June 2012. Association for Computational Linguistics. URL `http://www.aclweb.org/anthology/N12-1047`.

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

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

Nadir Durrani, Helmut Schmid, and Alexander M. Fraser. A joint sequence translation model with integrated reordering. In *The 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies, Proceedings of the Conference, 19-24 June, 2011, Portland, Oregon, USA*, pages 1045–1054, 2011. URL `http://www.aclweb.org/anthology/P11-1105`.

Kenneth Heafield. Kenlm: Faster and smaller language model queries. In *Proceedings of the Sixth Workshop on Statistical Machine Translation*, pages 187–197. Association for Computational Linguistics, 2011.

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

Arianna Bisazza, Nick Ruiz, and Marcello Federico. Fill-up versus interpolation methods for phrase-based SMT adaptation. In *IWSLT*, pages 136–143, 2011b.

Prashant Mathur. Online Adaptation of Statistical Machine Translation with Sparse Features. Technical report, University of Trento, Italy, 2015.

Chin-Yew Lin and Franz Josef Och. Orange: a method for evaluating automatic evaluation metrics for machine translation. In *Proceedings of Coling 2004*, pages 501–507, Geneva, Switzerland, Aug 23–Aug 27 2004. COLING.

Eva Hasler, Barry Haddow, and Philipp Koehn. Sparse lexicalised features and topic adaptation for SMT. In *Proceedings of IWSLT*, 2012. URL `http://www.mt-archive.info/IWSLT-2012-Hasler-2.pdf`.

Mohammad Amin Farajian, Nicola Bertoldi, and Marcello Federico. Online word alignment for online adaptive machine translation. In *EACL 2014 Workshop on Humans and Computer-assisted Translation (HaCaT)*, pages 84–92, Gothenburg, Sweden, April 2014.

Ben Taskar, Vassil Chatalbashev, Daphne Koller, and Carlos Guestrin. Learning structured prediction models: A large margin approach. In *Proceedings of the 22nd International Conference on Machine Learning*, ICML '05, pages 896–903, New York, NY, USA, 2005. ACM. ISBN

1-59593-180-5. doi: 10.1145/1102351.1102464. URL http://doi.acm.org/10.1145/1102351.1102464.

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*, pages 764–773, 2007b.

Preslav Nakov, Francisco Guzmán, and Stephan Vogel. Optimizing for sentence-level bleu+1 yields short translations. In *COLING*, pages 1979–1994, 2012.

Matthew Snover, Bonnie Dorr, Rich Schwartz, Linnea Micciulla, and John Makhoul. A study of translation edit rate with targeted human annotation. In *5th Conference of the Association for Machine Translation in the Americas (AMTA)*, Boston, Massachusetts, August 2006b.

Yair Al Censor and Stavros A. Zenios. *Parallel Optimization: Theory, Algorithms and Applications.* Oxford University Press, 1997. ISBN 019510062X.

Marcello Federico, Nicola Bertoldi, Mauro Cettolo, Matteo Negri, Marco Turchi, Marco Trombetti, Alessandro Cattelan, Antonio Farina, Domenico Lupinetti, Andrea Martines, Alberto Massidda, Holger Schwenk, Loïc Barrault, Frederic Blain, Philipp Koehn, Christian Buck, and Ulrich Germann. The MateCat Tool. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: System Demonstrations*, pages 129–132, Dublin, Ireland, August 2014. Dublin City University and Association for Computational Linguistics. URL http://www.aclweb.org/anthology/C14-2028.

Mauro Cettolo, Christian Girardi, and Marcello Federico. WIT^3: Web Inventory of Transcribed and Translated Talks. In *Proceedings of the Annual Conference of the European Association for Machine Translation (EAMT)*, Trento, Italy, May 2012b. URL `http://hltshare.fbk.eu/EAMT2012/html/Papers/59.pdf`.

Marcello Federico, Nicola Bertoldi, and Mauro Cettolo. IRSTLM: an Open Source Toolkit for Handling Large Scale Language Models. In *Proceedings of Interspeech*, pages 1618–1621, Brisbane, Australia, 2008.

Stanley F. Chen and Joshua Goodman. An empirical study of smoothing techniques for language modeling. Technical Report TR-10-98, Harvard University, 1998.

Jonathan Clark, Chris Dyer, Alon Lavie, and Noah Smith. Better hypothesis testing for statistical machine translation: Controlling for optimizer instability. In *Proceedings of the Association for Computational Linguistics*, ACL 2011, Portland, Oregon, USA, 2011b. Association for Computational Linguistics.

S. Vogel, H. Ney, and C. Tillmann. HMM-based word alignment in statistical translation. In *Proceedings of COLING*, pages 836–841, Copenhagen, Denmark, 1996.

Qin Gao and Stephan Vogel. Parallel implementations of word alignment tool. In *Software Engineering, Testing, and Quality Assurance for Natural Language Processing*, SETQA-NLP '08, pages 49–57, Stroudsburg, PA, USA, 2008b. Association for Computational Linguistics. ISBN 978-1-932432-10-7. URL `http://dl.acm.org/citation.cfm?id=1622110.1622119`.

J. A. Nelder and R. Mead. A simplex method for function minimization. *The Computer Journal*, 7(4):308–313, 1965. doi: 10.1093/comjnl/7.

4.308. URL http://comjnl.oxfordjournals.org/content/7/4/308.abstract.

Joel Martin, Rada Mihalcea, and Ted Pedersen. Word alignment for languages with scarce resources. In *Proceedings of the ACL Workshop on Building and Using Parallel Texts*, pages 65–74, Ann Arbor, Michigan, June 2005. Association for Computational Linguistics. URL http://web.eecs.umich.edu/~mihalcea/papers/martin.acl05.wpt.pdf.

Ralf Steinberger, Bruno Pouliquen, Anna Widiger, Camelia Ignat, Tomaž Erjavec, Dan Tufiş, and Dániel Varga. The JRC-Acquis: A multilingual aligned parallel corpus with 20+ languages. In *In Proceedings of the 5th International Conference on Language Resources and Evaluation (LREC'2006)*, pages 2142–2147, Genoa, Italy, 2006.

James S. Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. Algorithms for hyper-parameter optimization. pages 2546–2554, 2011. URL http://papers.nips.cc/paper/4443-algorithms-for-hyper-parameter-optimization.pdf.

S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220:671–680, 1983.

Rico Sennrich. Mixture-modeling with unsupervised clusters for domain adaptation in statistical machine translation. In *Proceedings of the Conference of the European Association for Machine Translation (EAMT)*, pages 185–192, May 2012c. URL http://dx.doi.org/10.5167/uzh-62826.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. BLEU: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association of Computa-*

tional Linguistics (ACL), pages 311–318, Philadelphia, PA, 2002c. URL `http://aclweb.org/anthology-new/P/P02/P02-1040.pdf`.

Kantilal Varichand Mardia, John T Kent, and John M Bibby. *Multivariate analysis*. Academic press, 1980.

M. Friedman. The use of ranks to avoid the assumption of normality implicit in the analysis of variance. *Journal of the American Statistical Association*, 32(200):675–701, 1937.

S. Holm. A simple sequentially rejective multiple test procedure. *Scandinavian Journal of Statistics*, 6:65–70, 1979.

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

Nancy Chinchor, Lynette Hirschman, and David D. Lewis. Evaluating message understanding systems: An analysis of the third message understanding conference (muc-3). *Computational Linguistics*, 19(3):409–449, 1993.

Stefan Riezler and John T. Maxwell. On some pitfalls in automatic evaluation and significance testing for MT. In *Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*, pages 57–64, Ann Arbor, Michigan, June 2005. Association for Computational Linguistics. URL `http://www.aclweb.org/anthology/W/W05/W05-0908`.

Bradley Efron and Robert J. Tibshirani. *An Introduction to the Bootstrap*. Chapman and Hall, 1993.

Ondřej Bojar, Rajen Chatterjee, Christian Federmann, Barry Haddow, Matthias Huck, Chris Hokamp, Philipp Koehn, Varvara Logacheva, Christof Monz, Matteo Negri, Matt Post, Carolina Scarton, Lucia Specia,

and Marco Turchi. Findings of the 2015 workshop on statistical machine translation. In *Proceedings of the Tenth Workshop on Statistical Machine Translation*, pages 1–46, Lisbon, Portugal, September 2015. Association for Computational Linguistics. URL http://aclweb.org/anthology/W15-3001.

Ondřej Bojar, Rajen Chatterjee, Christian Federmann, Yvette Graham, Barry Haddow, Matthias Huck, Antonio Jimeno Yepes, Philipp Koehn, Varvara Logacheva, Christof Monz, Matteo Negri, Aurelie Neveol, Mariana Neves, Martin Popel, Matt Post, Raphael Rubino, Carolina Scarton, Lucia Specia, Marco Turchi, Karin Verspoor, and Marcos Zampieri. Findings of the 2016 conference on machine translation. In *Proceedings of the First Conference on Machine Translation*, pages 131–198, Berlin, Germany, August 2016. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/W/W16/W16-2301.

Artem Sokolov, Stefan Riezler, and Tanguy Urvoy. Bandit structured prediction for learning from partial feedback in statistical machine translation. *CoRR*, abs/1601.04468, 2016. URL http://arxiv.org/abs/1601.04468.