UPPSALA UNIVERSITET

BLEU Decoding and Feature Weight Tuning in Docent

Aaron Smith

Uppsala University
Department of Linguistics and Philology
Master's Programme in Language Technology
Master's Thesis in Language Technology

July 2, 2015

Supervisors:
Jörg Tiedemann, Uppsala University
Christian Hardmeier, Uppsala University

Abstract

This thesis presents two sets of experiments carried out with Docent, a document-level decoder for phrase-based statistical machine translation (SMT). In the first half of the thesis, BLEU decoding is introduced, implemented and investigated, whereby proposed changes to a translation are only accepted if the score of the automatic evaluation metric BLEU increases. Rather than leading to high quality translations as might be expected, the results show that high BLEU scores can be achieved despite patently bad translations. These results complement previous criticisms of the BLEU metric, which have thrown doubt on the claim that it always correlates well with human judgements of translation quality. The argument in this thesis is however more subtle: even if BLEU correlates well with human judgements of translation quality for translations produced independently of this metric, it does not necessarily hold that manipulating systems to achieve high BLEU scores leads to better quality translations.

The second half of the thesis focuses on feature weight tuning, the process of finding optimum weights for the various feature functions employed in SMT. Several tuning paradigms already exist for standard sentence-level SMT decoders, but so far very little research has been carried out into tuning at the document level. The experiments presented here focus on adapting the PRO technique, which aims to find weights that correctly rank competing candidate translations. The ultimate goal of this work was to find a stable baseline tuning algorithm and use BLEU decoding to improve yet further on it. Despite extensive experimentation, this optimisation process fails to yield positive results in Docent: weights could not be found that improve the performance of the decoder compared to default settings. Possible reasons for these unexpected results and directions for future investigation are discussed in detail.

Contents

Acknowledgements

I would like to thank Jörg Tiedemann for introducing me to the fascinating topic of machine translation and for his generous support and patience throughout this project. I am also indebted to Christian Hardmeier who provided countless ideas for new experiments and was always on hand to help with technical difficulties. Both supervisors also provided invaluable feedback on this thesis, pointing out flaws in the logic and helping shape it into a coherent work.

I am also very grateful to all the teachers and staff who helped make the Master Programme in Language Technology such an interesting and enjoyable experience. I would particularly like to mention my fellow students on the programme: Edvin, Karin, Nina, and Yan - it was a pleasure to study with you all.

Finally my eternal gratitude to Rosalía, for bringing me to Uppsala and supporting me wholeheartedly in everything I do.

1 Introduction

This thesis presents a series of exploratory experiments related to two separate but potentially connected extensions to a document-level phrase-based statistical machine translation (SMT) decoder. The decoder, known as Docent, was developed at Uppsala University by Hardmeier et al. (2013), and allows discourse-level phenomena to be incorporated directly into SMT feature models. The decoder has at all times access to a candidate translation of the whole document, in contrast to standard SMT decoders such as Moses (Koehn et al., 2007), where texts are translated sentence-by-sentence.

The first extension to Docent described here is the implementation of what we call BLEU decoding. Given a reference translation, this allows the decoder to easily search for other translations that score highly on the automatic evaluation metric BLEU (Papineni et al., 2002). Indeed, as will be demonstrated in Chapter 3, BLEU decoding will find translations with BLEU scores close or equal to the maximum score possible, given the phrases available in the phrase table to the decoder. BLEU decoding is implemented efficiently as a feature model, and specified in a straightforward way in the Docent configuration file.

The initial intention in implementing BLEU decoding was to find high-quality translations; the results however were surprising, and show that while BLEU may correlate well with human judgements of translation quality for translations produced independently of BLEU, it is not necessarily the case that manipulating translations to have as high BLEU scores as possible leads to good quality. Given the reliance on a reference translation, BLEU decoding is obviously not appropriate in the translation of unseen test data. It might however be useful at an earlier stage, in feature weight tuning, and that is the focus of the second part of the thesis.

Feature weight tuning, equivalent to model selection in machine learning theory, is the process of finding optimal weights for the various feature models of an SMT system. The standard pipeline for creating and testing an SMT system, given a certain amount of data, is normally as follows: split the data into training, development, and test sets, train the basic probabilities of each of the feature models on the training set, tune the weights of each feature model to produce the best result possible on the development set, and finally test the resulting system with optimised weights on the test set. For sentence-level phrase-based decoders, a range of different tuning methods have been proposed, each with their own pros and cons, and these will be detailed further in Chapter 2. There has however been little attempt to adapt these techniques to work with document-level decoders, the main exception being Stymne et al. (2013). This thesis presents further experiments with feature weight tuning in Docent.

The thesis is structured as follows: Chapter 2 presents an introduction to the BLEU evaluation metric, some of the previous research into feature weight tuning in SMT, and a description of the Docent decoder, Chapter 3 describes the implementation of BLEU decoding in Docent, and experiments carried out on a German-English Europarl corpus, Chapter 4 presents experiments with feature weight tuning in Docent, and finally Chapter 5 presents conclusions and a discussion.

2 Background

This chapter provides an overview of the field of feature weight tuning in SMT, as well as an introduction to the document-level SMT decoder Docent. These two topics constitute the cornerstones of this thesis: the research presented later is related to both and is ultimately an attempt to move towards an effective tuning algorithm for Docent itself.

The chapter opens with an explanation of the automatic evaluation metric BLEU, before going in to a general description of tuning in SMT. It then covers different approaches developed by other researchers, beginning with the maximum mutual information criterion, followed by MERT, PRO, and MIRA. The way in which the Docent SMT decoder works will then briefly be outlined, as well as a summary of previous work into tuning within this framework.

2.1 BLEU

Several metrics exist to evaluate translation quality, including Multi-reference Word Error Rate (mWER), Multi-reference Position Independent Error Rate (mPER), NIST, and BLEU. The BLEU score, introduced by Papineni et al. (2002), is probably the most widely used. It works by comparing the candidate translation to one or more reference translations. For $1 \leqslant n \leqslant 4$, each candidate n-gram is checked against all of the references in order to calculate precision. To count towards precision, the candidate n-gram need only appear in one of the references; this helps to account for possible variations in style and word choice. However, the same n-gram appearing more than once in the candidate is only counted multiple times if it also appears multiple times in a single reference. For example, for the reference translation "The cat sat on the mat", the candidate translation "The the the the the the" does not have a unigram precision of 6/6, even though a naive analysis would say that all six tokens appear in the reference. Instead, as the word "the" only appears twice in the reference, the precision is 2/6. The n-gram precisions are combined by taking their geometric mean.

As multiple references are employed in calculating BLEU it is difficult to take recall into account. This could lead to short sentences scoring unfairly highly. Take for instance the candidate "the the": although it is clearly not a good translation for the previous example, it would have a perfect unigram precision of 2/2. To ensure that short sentences with high precision cannot cheat the system in this way, a brevity penalty is therefore introduced, lowering the BLEU score for cases where the length of the candidate translation c is less than the length of the reference translation r. The equation for BLEU is as follows:

$$\mathrm{BLEU} = \min\left(\exp\left(1 - r/c\right), 1\right) \cdot \exp\left(\sum_{n=1}^{N} \frac{\log p_n}{N}\right) \tag{1}$$

An obvious problem with BLEU, much like other standard evaluation metrics, is that it gives equal weighting to all words: the incorrect translation of a pronoun, for example, is penalised exactly the same as a noun that appears in the wrong form, although the effect on understanding may be much greater in one case than the other. BLEU also harshly punishes synonyms and elaborations, as well as words such as 'thus' or 'however' spliced occasionally into a text (see Callison-Burch et al. (2006) for a full discussion of these shortcomings). Despite these and other issues, however, BLEU has been shown to correlate extremely well with human judgement of translation quality in certain cases (Papineni et al., 2002). There have been a lot of recent efforts to develop more sophisticated metrics that counteract some of BLEU's weaknesses (Macháček and Bojar, 2013), but for the time being it remains ubiquitous in SMT, both in academic papers and in the evaluation of commercial translation systems.

2.2 Tuning in SMT

The standard decoding problem in phrase-based statistical machine translation (SMT) can be represented by the following equation (Koehn et al., 2003):

$$e_s^*(f_s) = \operatorname*{argmax}_{e_s} \left\{ \boldsymbol{\lambda}^T \cdot \mathbf{h}_s(f_s, e_s) \right\}. \tag{2}$$

In equation (2) we seek to find, out of all possible translations e_s of the source sentence f_s, the translation e_s^* which maximises the model score. The model score itself consists of a vector of feature scores $\mathbf{h}_s$ multiplied by a transposed weight vector $\boldsymbol{\lambda}$; each feature is thus associated with a particular weight which determines the relative importance of that feature in the overall model. The most common basic features are a phrase translation model, a distortion/rearrangement model, and a language model. Other features can be added, including similar complex features and so-called sparse features, which simply indicate the presence or absence of a particular property in the translation. This can potentially lead to models with thousands of individual features, and therefore thousands of weights.

The challenge of tuning is to find appropriate values for the weight vector $\boldsymbol{\lambda}$. As $\boldsymbol{\lambda}$ varies, so does the translation that maximises the objective function in equation (2). Generally feature weight tuning is performed on a distinct development data set, separate from both the training and test data. The reasonable assumption is made that if the weights are tuned in such a way that better results are achieved on the development data, we can also expect better results on unseen test data. This will often be the case but not always: overfitting to the development data may occur, and is something we must always be conscious of.

2.2.1 MMI

The previous section talked about tuning the weights to obtain *the best result*. Exactly how that is defined is one of the factors that distinguishes different tuning methods. The first such method, known as the maximum mutual information (MMI) criterion, was suggested by Och and Ney (2002). This aims to tune the feature weights such that the model score of the reference is maximised, in other words reducing the perplexity of the model with respect to the development set as much as possible. The criterion can be stated:

$$\hat{\lambda} = \operatorname*{argmax}_{\lambda} \left\{ \sum_{s=1}^{S} \left(\lambda^{T} \cdot \mathbf{h_s}(f_s, e_s^*) \right) \right\},$$

where S is the number of sentences in the development set. Doing this has some advantages: there is a unique global optimum, there are working algorithms, and it has been shown to be relatively successful (Och, 2003). Success in this case is measured in terms of performance of the optimised model on the test data; if the final translation quality on unseen data is improved, the optimisation has done its job. A difficulty with using MMI is that the reference translation may not be reachable by the decoder; it may be impossible to make e_s^* for a particular sentence exactly equal to the reference given the translation model (for example the necessary phrases may simply not be in the phrase table). A pseudo-reference may therefore need to be defined, representing the translation most similar to the reference translation that the decoder can reach.

2.2.2 MERT

As it is so often used in the final evaluation of translations, Och (2003) proposed trying to maximise BLEU score directly on the development data; to tune the weights λ in such a way that the best translations found by the model have high BLEU scores. Och introduced what he called Minimum Error Rate Training (MERT) to do just that. The method can actually be adapted to various evaluation criteria, it is not tied to BLEU in particular. The challenge is essentially to find weights that maximise a gold scoring function $G = \sum_s g(e_s)$ over all the sentences in the development set, where g is the sentence-level variant of G. We can then write:

$$\hat{\lambda} = \operatorname*{argmax}_{\lambda} \left\{ \sum_{s} g\left(e_s^*(f_s, \lambda)\right) \right\}. \qquad (3)$$

By putting equation (3) into plain English we can begin to understand that it is an extremely difficult problem to solve: we want to find the set of weights $\hat{\lambda}$ that maximise the sum of the g-scores of the translations with the highest model scores. For every possible set of λ values we would need to decode each sentence to find e_s^*, before calculating the sum of the corresponding g-scores and searching for the highest G-value. The decoding itself of course includes an argmax function (equation (2)), with a potentially huge number of candidate translations.

Och (2003) suggested that in solving equation (3) we limit ourselves at all times to the K-best candidate translations: the K candidate translations with the highest model score given current weight values $\boldsymbol{\lambda}$. His algorithm proceeds as follows: start with a initial guess for the weights $\boldsymbol{\lambda}_0$; compute the K-best translations; re-estimate the weights based on equation (3); re-compute the K-best list and combine with the previous list; iterate until convergence. He also introduced a line optimisation algorithm for re-estimating the weights during each equation. This works by looking along a search direction $\boldsymbol{\lambda}_t + \gamma \mathbf{d}_t$ at iteration t, where γ is a scalar parameter. Along this line we have

$$e_s^*(f_s, \gamma) = \underset{k \in \{1,\ldots,K\}}{\text{argmax}} \left\{ (\boldsymbol{\lambda}_t + \gamma \mathbf{d}_t)^T \cdot \mathbf{h}_{s,k}(f_s, e_{s,k}) \right\}. \tag{4}$$

A key observation is that the best translation e_s^* of sentence f_s in equation (4) does not change with every small change in the weights along a certain direction: it stays constant until the weights change enough that a new candidate translation has the highest model score. Also note that we can simply enumerate the model score for each candidate translation on the K-best list for given weights: the full decoding process is not required. Looking exhaustively along a line for a given sentence we can then find all the boundaries - values of γ at which the candidate translation with the highest model score changes. This process can be repeated across all sentences in the development set, and the boundaries merged. The corresponding g-scores of the best translations can then be calculated and summed, and the weights updated to the point right in the middle of the line section corresponding to the highest G-score. For full details and further information on how search directions are chosen, see Och (2003).

The paper also presents results from experiments on the 2002 TIDES Chinese–English small data track task. The left-hand column in Fig. 2.1 represents the evaluation metric used during training, while the subsequent columns display the scores of various metrics when the optimised model was then applied to the test data. Fig. 2.1 shows that for each evaluation metric, the best score on

error criterion used in training	mWER [%]	mPER [%]	BLEU [%]	NIST	# words
confidence intervals	+/- 2.7	+/- 1.9	+/- 0.8	+/- 0.12	-
MMI	**68.0**	*51.0*	11.3	5.76	21933
mWER	*68.3*	*50.2*	13.5	6.28	22914
smoothed-mWER	*68.2*	*50.2*	13.2	6.27	22902
mPER	*70.2*	*49.8*	15.2	*6.71*	24399
smoothed-mPER	*70.0*	**49.7**	15.2	*6.69*	24198
BLEU	76.1	53.2	**17.2**	6.66	28002
NIST	73.3	*51.5*	*16.4*	**6.80**	26602

Figure 2.1: Results presented by Och (2003) from experiments on the 2002 TIDES Chinese–English small data track task. See text for details.

the test data was achieved by tuning to that same metric on the development data. The best BLEU score, for instance, of 17.2, is achieved when the feature weights were optimised to BLEU. This is much more effective than tuning to any other evaluation metric, or using the MMI criterion discussed earlier.

Although very effective and easy to implement, MERT has the significant drawback that it does not scale well beyond about 30 features (Hopkins and

May, 2011). This is a problem as it stymies the development of new features, particularly sparse features. Several researchers have therefore worked on developing improved tuning methods. Galley et al. (2013), for example, introduced a regularisation term to the MERT objective function, as well as presenting a new direction finding algorithm. Many have kept the basic MERT architecture, replacing just the method of optimisation, while others have looked at completely new tuning paradigms. In the following sections we look at PRO and MIRA, which belong respectively to these two groups.

2.2.3 PRO

In the paper *Tuning as Ranking* (Hopkins and May, 2011), the authors introduce Pairwise Ranking Optimisation (PRO). The tuning problem is re-cast as a ranking problem: if the gold scoring function $g(e_s)$ is higher for translation candidate e_s than translation candidate e'_s, then we want the model score of e_s to be higher than the model score of e'_s. This can be written:

$$\begin{aligned} g(e_s) > g(e'_s) &\Leftrightarrow \boldsymbol{\lambda}^T \cdot \mathbf{h}(f_s, e_s) > \boldsymbol{\lambda}^T \cdot \mathbf{h}(f_s, e'_s) \\ &\Leftrightarrow \boldsymbol{\lambda}^T \cdot (\mathbf{h}(f_s, e_s) - \mathbf{h}(f_s, e'_s)) > 0. \end{aligned} \tag{5}$$

This is then a linear binary classification problem, solved by calculating $\mathbf{h}(f_s, e_s) - \mathbf{h}(f_s, e'_s)$ for each pair of candidate translations, and labelling as either a positive or negative instance depending on whether $g(e_s) > g(e'_s)$. To ensure balance, for each candidate translation pair both possible difference vectors, $\mathbf{h}(f_s, e_s) - \mathbf{h}(f_s, e'_s)$ and $\mathbf{h}(f_s, e'_s) - \mathbf{h}(f_s, e_s)$, are included as training instances, with one labelled positive and the other negative. The data can then be fed to any off-the-shelf classification tool to obtain the final weights; Hopkins and May (2011) used MegaM (Daumé III, 2004). In reality the number of e_s, e'_s pairs may be very large if we compare every single possible combination of candidate translations, and in many cases taking a subset of these suffices. This is done by generating candidate translation pairs and keeping those with the highest differences in gold scoring function.

Conceptually, PRO differs from MERT in that it focuses on being able to judge which is best of any two candidate translations, whereas MERT is entirely directed toward finding the best candidate translations in the development set. An advantage of PRO is that it can be easily adapted on top of existing MERT architecture as the outer loop is the same: we still start with an initial guess for the weights $\boldsymbol{\lambda}$, compute the K-best translations, re-estimate the weights through the classification problem described above, re-compute the K-best list and combine with the previous list, iterating the whole process until convergence. Unlike MERT, PRO has been shown to scale well to high-dimensional feature spaces, and its success has been demonstrated on a range of different data sets (Hopkins and May, 2011).

A variation of PRO was described by Bazrafshan et al. (2012), in the paper *Tuning as Linear Regression*. They point out that by solving the linear classifier problem induced by the inequality (5), information about the magnitude of the difference in the error function is ignored: instances are simply labelled positive or negative. Instead we can solve a linear regression problem, including the difference in error scores between candidate translations. They show that their method is equally as effective as PRO, and claim that it is also easier to solve and faster to converge.

2.2.4 MIRA

The methods considered thus far are so-called *batch* methods, meaning that once an estimate for the weights has been obtained, all sentences in the development set are decoded before a subsequent update in the weight estimates. An alternative approach is *online* tuning, in which sentences are decoded one at a time, and the feature weights are updated immediately before moving on to the next sentence. Online methods are often difficult to implement and not as easily parallelisable, but they have been successfully implemented by some authors, such as Watanabe et al. (2007). They introduce a method known as Margin Infused Relaxed Algorithm (MIRA), whose underlying objective is to predict the correct output over the incorrect one by a margin at least as large as the cost incurred by predicting the incorrect output. The algorithm proceeds as follows: visit a sentence i; decode according to the current weights $\boldsymbol{\lambda}$; update $\boldsymbol{\lambda}$ so as to reduce $l_i = \max_e \left[E(e) + \boldsymbol{\lambda}^T \cdot (\mathbf{h}_i(f,e) - \mathbf{h}_i(f,e^*))\right]$. Here, e^* is an an oracle translation, while $E(e) = \text{BLEU}(e^*) - \text{BLEU}(e)$. Finding the oracle translation is a challenge in itself; further details can be found in Watanabe et al. (2007).

MIRA has the advantage that it scales to many features, and is able to deal with large data sets. It also provides good performance: a significant increase in BLEU score compared to MERT. The downside is the complex implementation, however a recent paper, *Batch Tuning Strategies for Statistical Machine Translation* (Cherry and Foster, 2012), has dealt with some of these issues, introducing a best-of-both-worlds solution called BatchMIRA. As the name suggests, this is a batch version of MIRA, and the authors show it to perform at least as well as the best online methods and better than other options, while also being relatively easy to implement.

2.3 Docent

Docent (Hardmeier et al., 2013) is a decoder developed at Uppsala University for phrase-based SMT (Koehn et al., 2003). Within this framework, sentences in the source language are broken up into contiguous phrases, which are subsequently translated and reordered. Scoring functions are used to determine which is the best translation from a list of options for a particular phrase (see equation 2). The standard decoding process, implemented for example in Moses (Koehn et al., 2007), proceeds from left to right, adding phrases to the end of incomplete translations of a particular sentence. Docent, meanwhile, implements a search procedure based on local search, first proposed in Hardmeier et al. (2012).

In Docent's search algorithm, the feature models have access to a complete, if possibly poor, translation of a whole document at all stages of the search process. The algorithm is a stochastic variant of standard hill climbing; at each step, the decoder generates a successor of the current translation by randomly applying one of a set of state-changing operations at a random location in the document, and accepts the new translation only if it has a better score than the previous translation. Implemented operations include changing the translation of a phrase, changing the word order by swapping the positions of two phrases or moving a sequence of phrases, and resegmenting phrases. The initial translation can be either generated randomly from the phrase table or based on a run from Moses.

The motivation behind Docent is to facilitate the development of models with cross-sentence dependencies. According to Hardmeier et al. (2013), there has thus far been very little research into discourse-related phenomena in SMT. A classic problem is that of pronominal anaphora resolution: identifying the antecedents of pronouns in order for example to correctly translate into languages that have gendered pronouns. This type of problem is very difficult to solve in standard SMT decoders, which have hard-wired assumptions of sentence independence.

The standard tool-kit of sentence-level models, such as the phrase table, n-gram language models and distortion cost are implemented in Docent, along with document-level models including a length parity model, a semantic language model and several readability models (see Hardmeier et al. (2013) for details).

2.3.1 Tuning in Docent

Stymne et al. (2013) present experiments where they attempt to adapt existing frameworks for sentence-level feature weight tuning to Docent. They take the natural decision to compute all feature model scores and metric scores at the document rather than sentence level, and feed these scores to adapted versions of the standard Moses tuning framework. They use so-called k-lists, where k samples of the translation are taken during the decoding process. They investigate empirically the key parameters such as sample start iteration, sample interval and total number of iterations, displaying promising results on the data sets chosen for the study. Replicating these results in other test frameworks has however proved difficult, motivating the need to further investigate the tuning issue, and look at the potential of an in-built tuning method for Docent.

2.4 Summary

In this chapter we began by looking at the BLEU evaluation metric, and then defined the feature weight tuning problem in SMT. We looked at MMI as a first criteria for this optimisation, followed by several techniques which aim to tune feature model weights in such a way as to maximise BLEU, or any other evaluation metric, on development data. The document-level SMT decoder Docent was described, as well as a previous attempt to adapt sentence-level tuning techniques to this framework. In Chapter 3 we will introduce BLEU decoding in Docent, a unique feature model that allows Docent to search for translations with high BLEU scores. The connection with feature weight tuning will later be made clear in Chapter 4.

3 BLEU decoding

BLEU decoding is the name we have given to a particular mode of decoding in Docent whereby proposed changes to the document state are only accepted by the decoder if they result in an increase in the BLEU score. In this chapter we will see how BLEU decoding is implemented, and the results of various experiments carried out in BLEU-decoding mode. It will be shown how translations with high BLEU scores but low quality were the surprising product of BLEU decoding, and reasons for this will be discussed.

3.1 Implementation

A new feature model, `BleuModel`, was implemented in Docent. Feature models allow Docent to calculate a new model score for any proposed changes to the candidate translation before deciding whether to accept the change. The `BleuModel` constructor reads in a reference file in XML format, where a reference translation must be provided for each document to be decoded. The lengths of the reference translations, as well as the lengths of individual sentences within those translations and n-gram counts for $1 \leqslant n \leqslant 4$ are then processed and stored. Once Docent has created an initial candidate translation for each document, `BleuModel` calculates the BLEU score. The clipped counts for each sentence required to calculate BLEU are recorded along with the length of the candidate translation. In this way they do not need to be re-calculated every time BLEU is evaluated; instead the counts for a particular sentence only need to be updated when Docent proposes a change to that sentence. This makes `BleuModel` a particularly efficient feature model; there is no significant decrease in speed when it is included.

To specify to the decoder that `BleuModel` should be included as a feature, the following lines must be added to the `models` section of the Docent configuration file:

```
<model type="bleu-model" id="bm">
  <p name="reference-file">ref_doc.xml</p>
</model>
```

A weight must also be included in the `weights` section, as follows:

```
<weight model="bm">100</weight>
```

3.2 Exploratory experiments

In ‘pure’ BLEU-decoding mode, the weights of all standard feature functions are set to zero, and only changes to the translation that increase the BLEU score are accepted. The initial aim in running these experiments was to find good quality translations to use as data points in feature weight tuning for Docent. The results, however, show that by optimising for BLEU, translation quality actually goes down. While BLEU may be a good evaluation metric for independently produced translations, this throws into question the paradigm of directly optimising weights towards BLEU.

3.2.1 Procedure

An German-English Moses translation model was trained on just over 1.5 million sentences from Europarl v7, downloaded from `www.statmt.org/wmt13`. A 5-gram English language model (LM) was trained using KenLM on just over 2.2 million sentences from the same source. The test data was a set of 3052 sentences from the newstest2013 data. This data set contains document markup, enabling 52 separate documents (an average of 59 sentences per document) to be fed to the Docent decoder.

Two types of experiments were carried out, firstly with the candidate translation initialised by running Moses (with feature weights tuned using MERT on a development set of 2525 sentences from the newstest2009 data), and secondly by a random initialisation. Docent was then run in BLEU-decoding mode: only changes to the translation that increased BLEU were accepted. The decoder was allowed to run for 1,000,000 iterations for each document. Feature scores for 10 standard features were then monitored at iterations 2^8, 2^9, 2^{10} etc.

3.2.2 Results I: Moses-based initial translation

In the first set of experiments, with initialisation according to Moses, the BLEU score across the 52 documents after Moses decoding ranged from 7.2 to 33.1, with a mean of 19.4; after subsequently running Docent in BLEU-decoding mode, the mean had increased to 47.8, with a range from 24.1 to 70.6. Looking at the Docent output, however, it becomes clear that high BLEU scores have been achieved *in spite of* worse translation quality in many cases. Let us take a demonstrative example, considering the source (SRC), reference (REF), Moses translation (MOS), and BLEU-optimised translation (BLU1):

(Example 1)

SRC: *am wichtigsten ist es aber , mit seinem arzt zu sprechen , um zu bestimmen , ob er durchgeführt werden sollte oder nicht .*

REF: *but the important thing is to have a discussion with your doctor to determine whether or not to take it .*

MOS: *the most **important thing is** , however , with his **doctor to** speak , in order **to determine whether** it should be carried out **or not** .*

BLU1: *with **but the important thing is to** its a **doctor to** mention **to determine whether or not to** be implemented **it .** to*

We see here that the Moses translation, while not perfect, carries at least most of the meaning across from the original sentence. The BLEU-optimised translation, meanwhile, is unintelligible. The fragments in bold show n-grams for $n \geqslant 2$ where the Moses and BLEU translations match the reference. It is telling that there are no 4-gram matches at all in the Moses translation, while the long matching fragments in the BLEU translation ensure that there are as many as six such matches. The BLEU translation also has a higher unigram precision; indeed, for all $1 \leqslant n \leqslant 4$, the number of matching n-grams is much higher in the BLEU translation than the Moses translation.

Another example exposes further the internal workings when we optimise solely towards BLEU:

(Example 2)

SRC: *es ist auch ein risikofaktor für mehrere andere krebsarten .*

REF: *it is also a risk factor for a number of others .*

MOS: *there **is also a risk factor for a number of** other types of cancer .*

BLU1: ***it is also a risk factor for a number of others .** cancers*

In this example the Moses translation is actually very good; a more literal translation of the source sentence than the reference. After BLEU decoding, however, the sentence has been transformed: it now matches the whole of the reference identically, but with the word *cancers* added after the full-stop. It is straightforward to see why the BLEU translation has a higher BLEU score: the extra couple of tokens at the end of the matching fragment increase the precision for all n-grams.

In a random sample of 20 sentences from across the 52 documents, 12 were judged to be worse after BLEU decoding compared to the initial Moses translation, 6 were judged to be of equal quality and just 2 were judged to be better. This demonstrates a clear deterioration in translation quality at the document level, particularly when we bear in mind that sentences that get worse often get a lot worse, as shown in the first example, whereas sentences that improve generally only do so marginally.

In the standard setting for statistical machine translation we decode to maximise the combined scores of a set of features, then use BLEU as an independent evaluation metric. We cannot in this scenario use the score of for example the language model that our system seeks to optimise as a metric to evaluate translation quality. In pure BLEU-decoding mode, however, this restriction disappears. The only input model to the decoding process is BLEU, so we are free to examine the language model score of the output as a type of evaluation metric. Generally it is assumed that BLEU correlates much better with translation quality than the language model alone; for a start it has access to the reference translation which the language model does not. We expect however the language model to correlate to some extent with translation quality, most in terms of fluency: this is of course why we normally include it in our system in the first place. With this in mind, Fig. 3.1 shows how the language model score varies as BLEU increases.

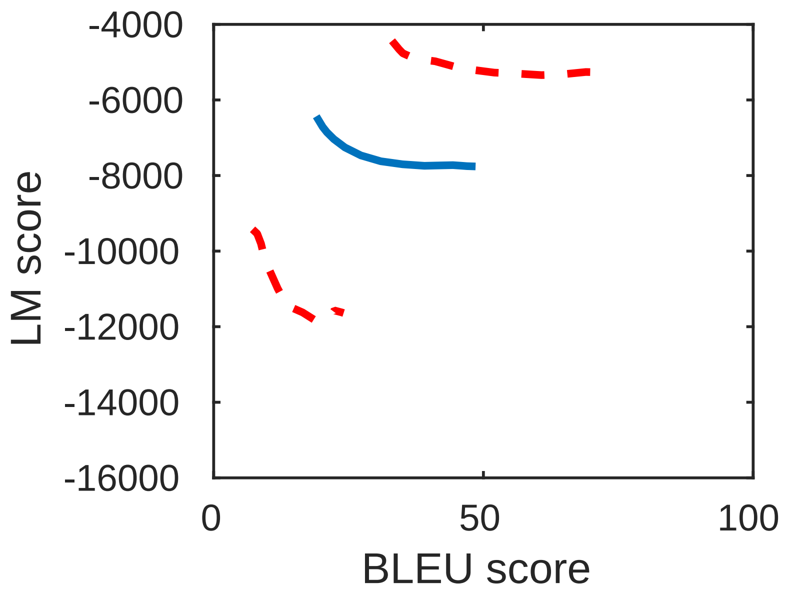

Figure 3.1: Language model score against BLEU score for BLEU decoding from a Moses-based initial translation. The three lines show, across the 52 documents, minimum LM score against minimum BLEU score, mean LM score against mean BLEU score, and max LM score against max BLEU score.

We observe that the LM scores decrease significantly as BLEU increases; Docent in BLEU-decoding mode is able to find translations with high BLEU scores that score poorly on the LM feature. The Moses-based initialisation procedure works of course to maximise the language model score, so it would be unrealistic to expect it to increase much more during BLEU decoding, unless we had reason to believe that there was significant search error in the Moses decoding process. The fact that the language model score drops so drastically however reinforces the point made earlier by the example sentences, that we have high BLEU scores but poor quality sentences. Indeed, the same trend can be observed for several other standard feature models. These results suggest that by letting BLEU run wild, we move far away from the part of the search space containing good translations.

3.2.3 Results II: Random initial translation

In the second set of experiments, with random initialisation of the candidate translation, the mean BLEU score across the 52 documents at the beginning of the decoding process was 3.7 (with range 0.0 to 7.3); after running Docent in BLEU-decoding mode it had increased to 47.4 (with range 22.9 to 67.4). This figure for the mean is very similar to that of 47.8 obtained when decoding from Moses-based initial translations, suggesting that the initial translation does not have a great effect on the final result. We can now go back to the first example sentence from the previous section and add two new translations: the random initial translation (RAND) and the revised version of this after BLEU decoding (BLU2):

(Example 1)

SRC: *am wichtigsten ist es aber , mit seinem arzt zu sprechen , um zu bestimmen , ob er durchgeführt werden sollte oder nicht .*

REF: *but the important thing is to have a discussion with your doctor to determine whether or not to take it .*

MOS: *the most **important thing is** , however , with his **doctor to** speak , in order **to determine whether** it should be carried out **or not** .*

BLU1: *with **but the important thing is to** its a **doctor to** mention **to determine whether or not to** be implemented **it .** to*

RAND: *most important of makes it while , with him doctor are talking by **to determine** be the should be conducted or no with .*

BLU2: *with his **but the important thing is** a **doctor to** talk to **determine whether or not** it take place **it . or not to***

As before, n-grams that match the reference translation for $n \geqslant 2$ have been highlighted. The similarities between BLU1 and BLU2 are striking. While not identical, they share many phrases and contiguous sets of words, as well as the property that they make no sense whatsoever. This suggests that the type of translation BLEU decoding results in is independent of the initial translation; the initial translations - MOS and RAND - of BLU1 and BLU2 are clearly very different from each other. It is also enlightening to compare BLU2 with its antecedent RAND. While neither of these translations can be said to convey much of the sense of the original German sentence, it could perhaps be argued that BLU2 is slightly more sensical than RAND. Perhaps the chunks that match the reference do actually help to bring through some trace of meaning. One way to compare the random initial translation to the BLEU decoded version is to look again at the language model.

Fig. 3.2 shows a general trend of LM score increasing with BLEU score. In this case we can therefore draw the conclusion that BLEU decoding from a random initial translation does result in translations that are slightly better, in some meaningful sense, than the initial translation. It is however clear by looking at the example sentences that the improvement in quality is nowhere

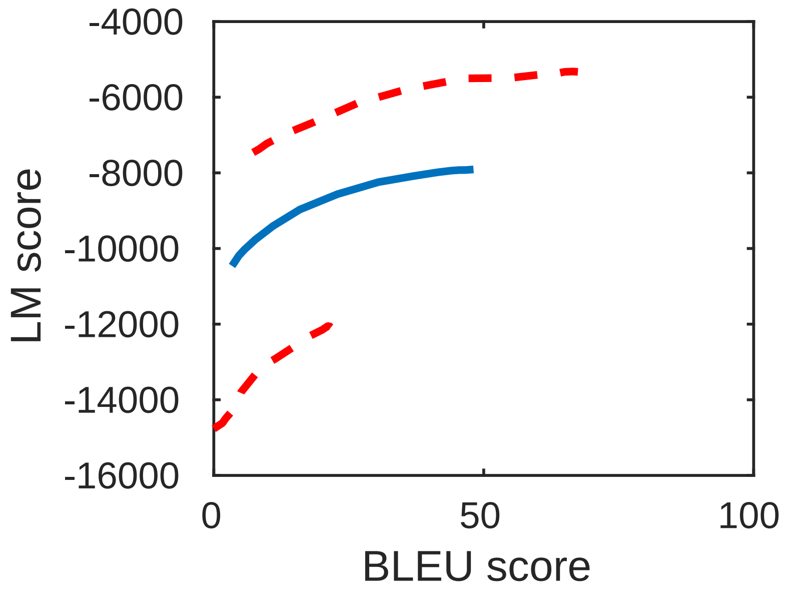

Figure 3.2: Language model score against BLEU score for BLEU decoding from a random initial translation. The three lines show, across the 52 documents, minimum LM score against minimum BLEU score, mean LM score against mean BLEU score, and max LM score against max BLEU score.

near that that would be normally be expected given the jump in BLEU score from 3.7 to 47.3.

3.3 BLEU decoding towards reachable translations

We saw in the previous sections that the mean BLEU score after 1,000,000 iterations of BLEU decoding was 47.8 when the initial translation came from Moses, and 47.3 when the initial translation was randomly chosen. While these are undoubtedly high BLEU values, they are still a long way from 100, which would represent the decoder finding the reference translation exactly. It is natural to wonder why this is the case; what is stopping the BLEU score getting much higher. Docent is of course limited by the phrases that happen to be in the phrase table, based on the training data. It is clear that not all phrases needed to construct the reference will always be found (this can be verified directly by examining the phrase table). It is not clear however whether this is the only limiting factor. Another possibility might be that the decoder's hill-climbing algorithm tends to get stuck in local maxima. The fact that the initial configuration apparently plays no role speaks against this hypothesis, but not definitively. Another test that can be carried out is to give the decoder a 'fake' reference translation, that is not really a true reference at all, but simply another random translation generated by Docent. As Docent generates this translation from phrases in the phrase table, it is guaranteed that the reference is theoretically reachable by the decoder.

3.3.1 Procedure

The experimental setup was similar to that described in Section 3.2, the only difference being the switch from the genuine reference translation to a simulated reference generated randomly by Docent. The 52 test documents were decoded from random initial translations.

3.3.2 Results

The average BLEU score before decoding was 6.7, with range from 2.8 to 11.7; after 1,000,000 iterations it was 95.1, with range 91.5 to 98.2. These results confirm that, when the necessary phrases are in the phrase table, the decoder in BLEU-decoding mode is able to get extremely close to the reference translation. This suggests that the lower BLEU scores seen earlier when BLEU decoding towards genuine reference translation are as high, or almost as high, as they can possibly get. BLEU scores near to 100 are simply impossible on this data set given a genuine reference translation and the phrases available to the decoder. The poor quality translations demonstrated by the examples in the previous section are therefore probably as good as it gets in terms of BLEU score: there are no translations that the decoder can conceivably reach with significantly higher BLEU scores.

For completeness we can also examine what happens to the score of the language model during BLEU decoding towards a reachable configuration. It is expected that it will not significantly improve: we know that the final translations are similar to translations randomly generated by Docent. The initial translation is also randomly generated, so the decoder essentially moves in a directed manner through the search space from one random translation to another. Fig. 3.3 confirms that this process does not improve the translation in terms of the LM score: as expected the mean, max and min are essentially constant as BLEU increases.

3.4 Discussion

This chapter covered the implementation of a BLEU feature model in Docent, whereby proposed changes to the translation are evaluated by the effect they have on the BLEU score. This feature model would not be available in a normal translation setting: it is dependent on the presence of a reference and thus only suitable during model development. The intention in implementing BLEU as a feature model was to enable the decoder to explore regions of the search space with high BLEU scores. It was assumed that the translations in this region would be of high quality: BLEU has been shown to correlate well with human judgements of translation quality. Experiments presented here with BLEU decoding, where only changes to the translation that increase BLEU are accepted, show this however not to be the case. Despite an increase in mean BLEU score from 19.4 to 47.8 across 52 documents from Europarl translated in BLEU-decoding mode from German to English from an initial translation generated by Moses, there was a clear drop in translation quality (12 out of 20 randomly chosen sentences were judged to be worse, and only 2 judged

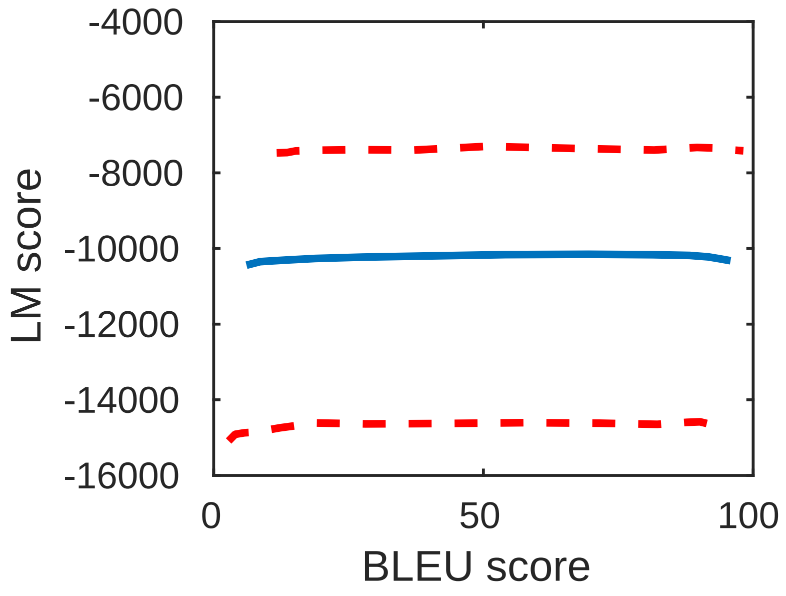

Figure 3.3: Language model score against BLEU score for BLEU decoding from a random initial translation towards a pseudo reference translation generated randomly from the phrase table by Docent. The three lines show, across the 52 documents, minimum LM score against minimum BLEU score, mean LM score against mean BLEU score, and max LM score against max BLEU score.

better). An even larger increase in BLEU, from 3.7 to 47.3, was observed when decoding from a random initial translation, but results were similar in terms of translation quality. These results have been confirmed on other data sets with different text types and language pairs.

What do these results say about BLEU as an evaluation metric? Initial impressions might suggest that the evidence presented here is damning for BLEU: it has been clearly shown that it can be 'cheated': very bad translations can get high BLEU scores. This is not the first time problems with BLEU have been highlighted (Callison-Burch et al., 2006), and research into better metrics is a very active field (Macháček and Bojar, 2013). However, it must be remembered that the experiments presented here used BLEU in a very different fashion from that for which it was designed. The original paper on BLEU demonstrated clearly that when translation quality is manipulated as the independent variable in experiments, there is a strong correlation with BLEU as the dependent variable. This does not imply, and indeed the opposite has been shown in this chapter, that manipulating BLEU as an independent variable will necessarily result in high quality translations. Another way of saying this is as follows: if translations are produced independently of BLEU, then BLEU is often a good metric to distinguish their quality; however this does not imply that actively looking for translations with high BLEU score will result in high quality. There is clearly an area of the search space with high BLEU scores and low quality translations, and while standard decoding procedures might not normally find themselves in these regions, it is important to remember that BLEU is not infallible: optimising blindly towards BLEU during feature weight tuning in SMT might not always be the right thing to do.

The implications for tuning at the document level remain to be seen. It is unclear whether pure BLEU decoding is going to give useful translations to use as data points. It will be interesting to see if it is possible to tune a system using this data that is able to get high BLEU scores on unseen data, and whether the quality of these translations will be high. In Chapter 4 we will begin to answer some of these questions.

4 Tuning in Docent

This chapter describes some experiments related to tuning in Docent. We begin by revisiting tuning as a ranking problem, and discuss how this framework can be adapted to the document level. We then look for optimised weights by decoding a development set of documents, and use these weights on an unseen test set to evaluate how well they do. Some variations in key parameters for the decoding and classification steps are investigated. We will ultimately see that it is difficult, given the data set and other variables at play, to find weights that perform better than the default weights.

4.1 Tuning as ranking

In Section 2.2.3 we saw how the PRO tuning algorithm re-casts feature weight tuning for SMT as a ranking problem. Equation (5) stated

$$g(e_s) > g(e'_s) \Leftrightarrow \boldsymbol{\lambda}^{\mathsf{T}} \cdot (\mathbf{h}(f_s, e_s) - \mathbf{h}(f_s, e'_s)) > 0,$$

where e_s and e'_s are candidate translations of the foreign sentence f_s. This equation simply states that for a given sentence, the model score of a candidate translation should be higher if the translation scores higher on a gold scoring function g. Note that while equation (5) explicitly considers the sentence level, it can equally be applied at the document level. It is still true that for a *document* the model score of a candidate translation should be higher if the translation scores higher on g. In fact, when BLEU is used as g, as is often the case, the document level is in fact more natural as BLEU is designed to work on whole documents, not on single sentences. Equation (5) can thus be written more generally:

$$g(e) > g(e') \Leftrightarrow \boldsymbol{\lambda}^{\mathsf{T}} \cdot (\mathbf{h}(f, e) - \mathbf{h}(f, e')) > 0, \tag{6}$$

where e and e' are candidate translations of the foreign text or document f.

Sentence-level decoders based on beam search such as Moses are able to output lists of candidate translations for a given source sentence along with their model scores. Training data for the pairwise ranking problem can thus be created by simply calculating difference vectors between the model scores of these candidate translations, and defining a positive and negative instance for the classifier depending on which translation has the higher g score. Some kind of sampling is in fact often necessary to avoid having *too much* training data (Hopkins and May, 2011).

Docent's hill climbing algorithm, on the other hand, produces a single final candidate translation of the whole document. There is in other words no ranked list of candidate translations from which to calculate difference vectors. There is however the history of the translation: the different stages it went through to get to the final version. By sampling from this history, we might be able to extract sufficient quantities of good data to calculate similar difference vectors to feed to the classifier. Data sparsity might well be an issue: the number of data points will be orders of magnitude lower when working at the document rather than sentence level.

The tuning algorithm proposed here thus first determines whether a particular example is positive or negative by looking at the difference in BLEU score between the translations of a document at two separate stored iterations during decoding. If the BLEU score has increased for a given document between two recorded states, the instance is considered positive: if the score has decreased, it is negative. If the BLEU score is unchanged then no instance is created. For each of eight standard Docent feature models, the difference in model scores between the two iterations is calculated and used to create an instance for the classifier. The eight Docent model scores come from the geometric distortion model (labelled F1), word penalty (F3), n-gram language model (F5), and phrase translation models (F6–F10). Neither the second score from the geometric distortion model (the weight of which is set to 10^{30} by default), nor the OOV penalty, are included at this stage (hence there is no F2 or F4).

As in the original PRO paper, a 'mirror image' instance is also created, which is negative if the original is positive and vice-versa, and where the features of the instance are the difference between the earlier recorded state and the later one (i.e. the features of the original instance multiplied by -1). A sample of the data supplied to the classifier looks as follows:

```
0 F1 0 F3 10 F5 263.68 F6 116.27 F7 169.59 F8 194.64
1 F1 0 F3 -10 F5 -263.68 F6 -116.27 F7 -169.59 F8 -194.64
1 F1 0 F3 28 F5 304.39 F6 0.77 F7 147.29 F8 131.83
0 F1 0 F3 -28 F5 -304.39 F6 -0.77 F7 -147.29 F8 -131.83
```

Each line is an instance; the first column represents whether it is positive or negative, while the remaining columns represent feature names and values. Note that the first and second lines are mirror instances of each other, as are the third and fourth. This data is fed into the MegaM classifier as follows:

```
./megam_i686.opt -fvals binary data.txt > weights.txt
```

MegaM outputs weights for the eight models in question, which can be copied directly into the Docent configuration file.

Given a series of stored translations for a set of documents, we now have a method for finding new weights. The questions of which translations to store and use in this process, and of what parameters to use when carrying out the decoding that provides the stored translations, remain to be answered. These questions will be examined in the following sections. These experiments make use of the same German-English translation model as in Section 3.2.1. In contrast to the experiments described in Chapter 3, the experiments here are carried out on the development set, consisting of 111 documents (2525 total sentences) from the newstest 2009 data set.

4.2 BLEU decoding with logarithmic sampling

By default, Docent stores the initial candidate translation and the translations at iterations 2^8, 2^9, 2^{10}, and so on up to the total number of iterations. The motivation for this logarithmic sampling is that many more proposed changes to the translation are accepted in the beginning: as decoding progresses and the translation improves, there are simply more iterations between each interesting event. When running for 1,000,000 iterations, we end up with 14 recorded translations for each document. It seems reasonable therefore to begin by creating data for the classifier based on difference vectors between the model scores at these recorded translations.

In this section the initial translations are generated randomly by Docent, and BLEU decoding is carried out: that is to say only changes to the document state that increase the BLEU score are accepted by the decoder. The mean BLEU score across the 111 documents before decoding was 3.1, after decoding it was 43.0 (note that this is similar to the results we obtained on a different data set in Section 3.2.3). Weights are then determined for eight standard Docent models as described in the previous section, by calculating difference vectors for all pairs of consecutive stored translations for all documents. The resultant weights are displayed in Table 4.1.

Feature	MegaM weight
Distortion 1	–0.140
Word Penalty	–0.075
Language Model	0.0831
Translation Model 0	–0.0123
Translation Model 1	0.0262
Translation Model 2	0.0814
Translation Model 3	–0.0153
Translation Model 4	1.01

Table 4.1: Weights calculated by the MegaM classifier from the development data when decoded from a random initial configuration with BLEU decoding.

In order to test how well these newly calculated weights perform on unseen test data, a small test set of six documents from the newstest 2013 test set was decoded using the weights from Table 4.1 (keeping the default values of 10^{30} for the second weight of the geometric distortion model and 100 for the OOV penalty). Documents were again initialised randomly and the experiment ran for 1,000,000 iterations. BLEU decoding was of course switched off in this case: the reference translation was unavailable to the decoder and only standard models could be used by the decoder to determine whether proposed changes to the translation should be accepted.

The mean BLEU score across the six documents at the start of the experiment, before decoding, was 3.1, and after decoding it remained at 3.1. Fig. 4.1 shows how the mean BLEU score evolves as the decoding process progresses. We observe that BLEU varies somewhat during decoding, peaking at 3.5, but ultimately there is no upward trend whatsoever. The results show that the BLEU scores after decoding are no better on average than for the

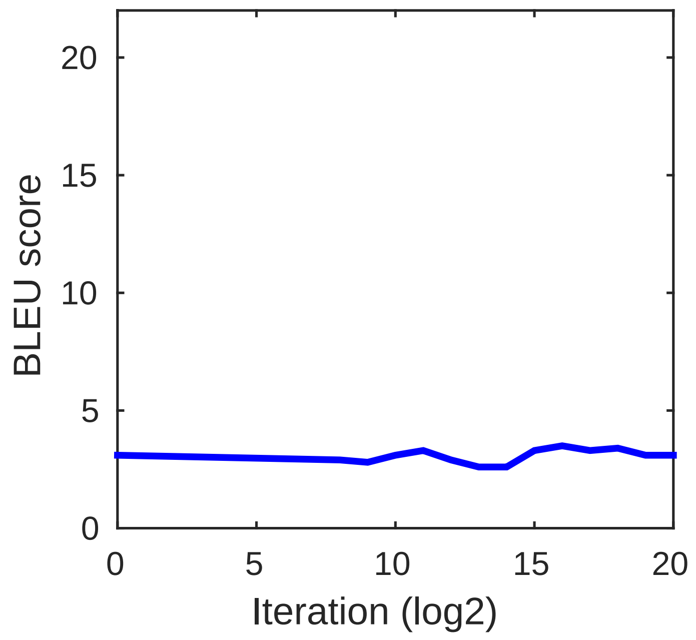

Figure 4.1: BLEU score against iteration when decoding small test set with weights generated from decoding development set with BLEU decoding.

initial configurations, suggesting that the weights in Table 4.1 are completely unreasonable. We expect of course the BLEU scores to increase significantly for all documents: this is what would happen when decoding with default weights. The new weights have thus destroyed the ability of the model to get anywhere near a reasonable translation.

The fact that several of the weights were negative should be a clue here: for most models it is implausible that having a negative weight would consistently improve translation quality, as this would imply that a lower score on the model makes for a better translation. This does not apply to the word penalty or final translation model feature weights, however, which can genuinely take negative values. Taking the first sentence of the first document as an example, we can compare the reference translation REF with the initial random translation INT and the translation after decoding with the new weights DEC:

REF: *a republican strategy to counter the re-election of obama*

INT: *is republican strategy to in re-election of , obama to counter*

DEC: *the republican , one of strategy to with obama re-election*

It is difficult to argue that the translation after decoding is any better than the initial translation: the conclusion is therefore that the model defined with the weights in Table 4.1 does no better than a random configuration.

There are several possible reasons why this experiment did not work, and several options for how to proceed. We know, from the experiments carried out in Chapter 3, that BLEU decoding on its own arrives at unreasonable translations; it is therefore perhaps unsurprising that the weights obtained in this way are not useful. A possible extension here would be to use a combination of BLEU decoding and other more standard models, however some initial experiments decoding the developments set with a combination of the BLEU feature function and the language model gave weights with a similar pattern to those in Table 4.1, and the final results were equally poor when the test set was decoded with those weights. In the next section BLEU decoding will instead be switched off as we revert to decoding with standard feature functions to see if this helps the tuning process.

4.3 Decoding with default weights

The decoding process in this section is similar to that in Section 4.2 with one key difference: the weights of the standard feature models are set to their default values (Table 4.2) instead of zero, and in contrast the weight of the BLEU feature function is set to 0. On this occasion the mean BLEU score across the

Feature	Default weight
Distortion 1	0.6
Distortion 2	1e30
Word Penalty	−1
OOV Penalty	100
Language Model	0.5
Translation Model 0	0.2
Translation Model 1	0.2
Translation Model 2	0.2
Translation Model 3	0.2
Translation Model 4	0.2

Table 4.2: Default weights for the standard feature models.

111 documents before decoding was 2.9, and after decoding it was 16.0. Note the huge difference between the final BLEU score when in BLEU-decoding mode (43.0) and in a more standard setting (16.0).

Calculating new weights using the same strategy as before yields the weights displayed in Table 4.3. When decoding with these weights on the same small test set of 6 documents as in the previous section, the mean BLEU score at the start of the experiment was 3.3, while at the end of the experiment it was 2.0. Fig. 4.2 shows the evolution of the BLEU score as decoding proceeds.

This is a somewhat incredible result: the mean BLEU score goes down when decoding with these weights! In other words, a completely random Docent translation is better than the final result. The weights in Table 4.3, like those in Table 4.1, are thus obviously completely inappropriate.

Feature	MegaM weight
Distortion 1	–0.026
Word Penalty	0.025
Language Model	0.0013
Translation Model 0	0.0044
Translation Model 1	–0.0140
Translation Model 2	0.0213
Translation Model 3	0.0074
Translation Model 4	0.0494

Table 4.3: Weights calculated by the MegaM classifier from the development data when decoded from a random initial configuration with default weights.

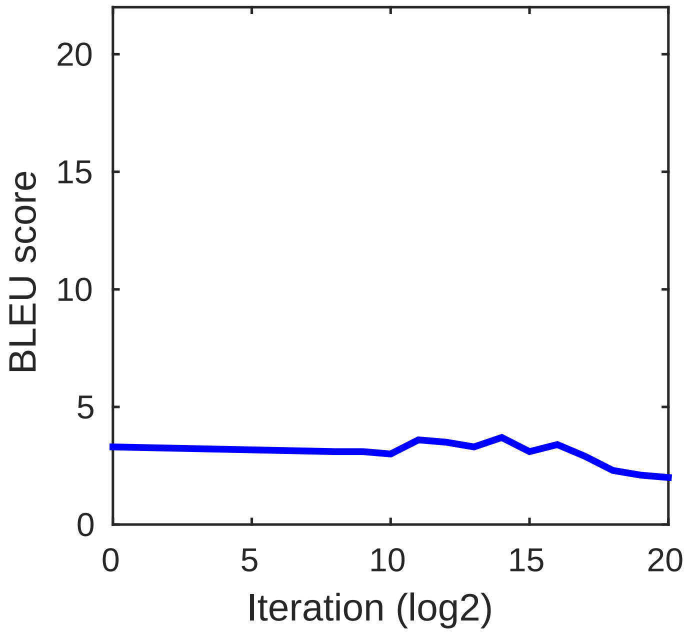

Figure 4.2: BLEU score against iteration when decoding small test set with weights generated from decoding development set with default weights.

4.4 Setting negative weights to zero

One obvious issue when looking at the weights in Table 4.3 is that certain models that should always have positive weights have in fact been given negative weights. We can manually set these unwanted negative weights to 0; in this way the models simply no longer affect the translation. Fig. 4.3 shows the result of decoding the small test set with the weights from Table 4.3, except that the unwanted negative weights are instead set to zero. The difference between Fig. 4.2 and Fig. 4.3 is striking. Setting the negative weights to zero clearly helps a great deal: BLEU score now increases as we decode the test data. In this case the mean BLEU score across the six documents before decoding was 3.3, after decoding it was 12.5.

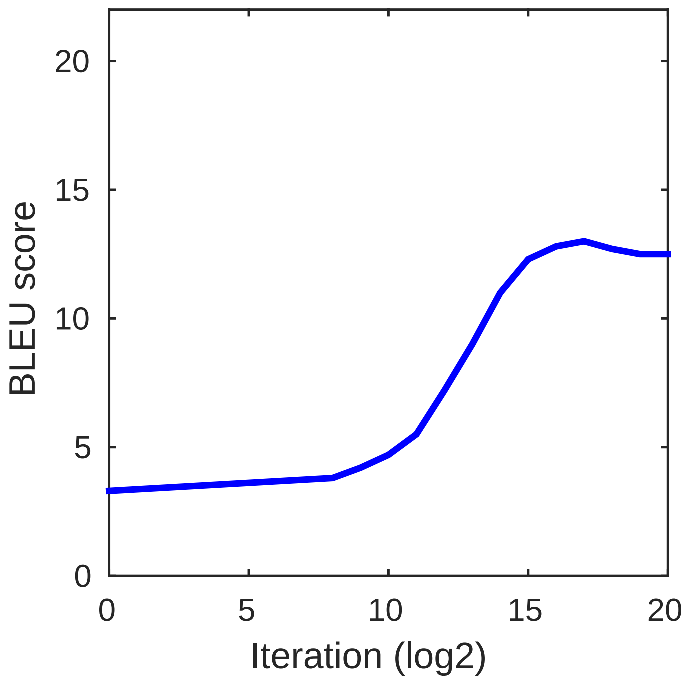

Figure 4.3: BLEU score against iteration when decoding small test set with weights generated from decoding development set with default weights and setting unwanted negative weights to zero.

Remember however that the goal is not just that the BLEU score should increase, but that it should increase more with optimised weights than it does with default weights (this is of course a statement of the obvious, but perhaps worth reiterating given the poor results observed in previous sections). In Section 4.3 the development set was decoded with the default weights presented in Table 4.2 in order to find new weights to decode the test set, but the results of decoding the test set itself with default weights have not yet been seen. They are displayed in Fig. 4.4. In this case the mean BLEU score before decoding was 3.4, and after decoding was 20.3. We see therefore that the default weights of Table 4.2 do much better on the test set than the optimised weights of Table 4.3, proving that they are still not in fact optimised at all.

4.5 Comparing to all subsequent iterations

It has already been mentioned that data sparsity is a potential problem: the classifier may simply not be getting enough instances to produce meaningful weights. Given the current set-up, the maximum number of instances that can possibly be provided to the classifier is $111 \times 13 \times 2 = 2886$ (number of documents in development set $\times$ (number of recorded translations - 1) $\times$ 2 instances per difference vector). The actual number of instances may of course be fewer, as no instance is created if the BLEU score does not change between recorded translations (for a start, if BLEU does not change it is difficult to decide whether we have a positive or negative instance). The reason we

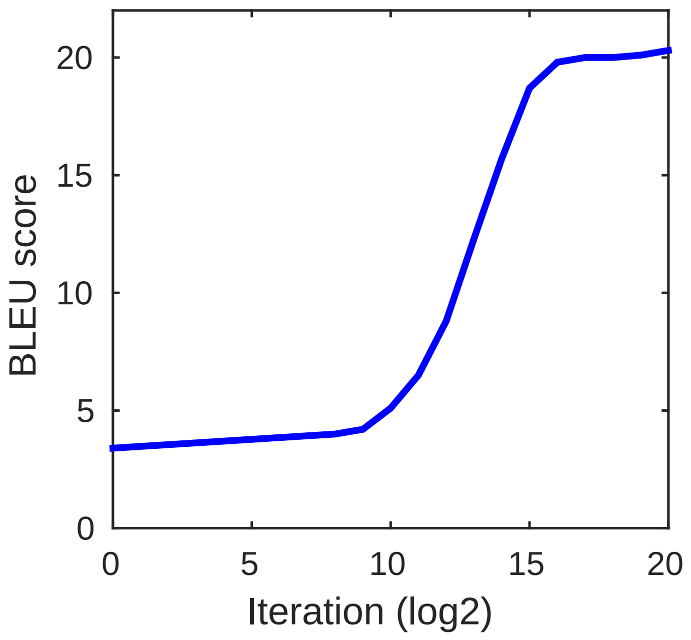

Figure 4.4: BLEU score against iteration when decoding small test set with default weights.

have (number of recorded translations - 1) in the equation is that difference vectors are created by comparing the first recorded translation to the second, the second to the third, and so on. A simple way to magnify the number of instances from the same recorded translations is to compare the first recorded translation to the second, third, fourth etc, and then compare the second to the third, fourth, fifth etc, and so on. This would lead to a maximum of $111 \times (13+12+...+2+1) \times 2 = 20202$ instances for the classifier, a seven-fold increase.

Table 4.4 shows the weights obtained when creating the data for the classifier in this way, from the recorded translations from the decoding of the development set in Section 4.3. Negative weights are again set to 0 as this

Feature	MegaM weight
Distortion 1	0
Word Penalty	0.013
Language Model	0
Translation Model 0	0.005
Translation Model 1	0
Translation Model 2	0.024
Translation Model 3	0
Translation Model 4	0.018

Table 4.4: Weights calculated by the MegaM classifier from the development data when decoded from a random initial configuration with default weights. In this case difference vectors are created for all combinations of recorded translations.

proved to effective in the previous section. Fig. 4.5 shows how the BLEU score changes as decoding progresses. The pattern displayed is similar to both Fig. 4.3

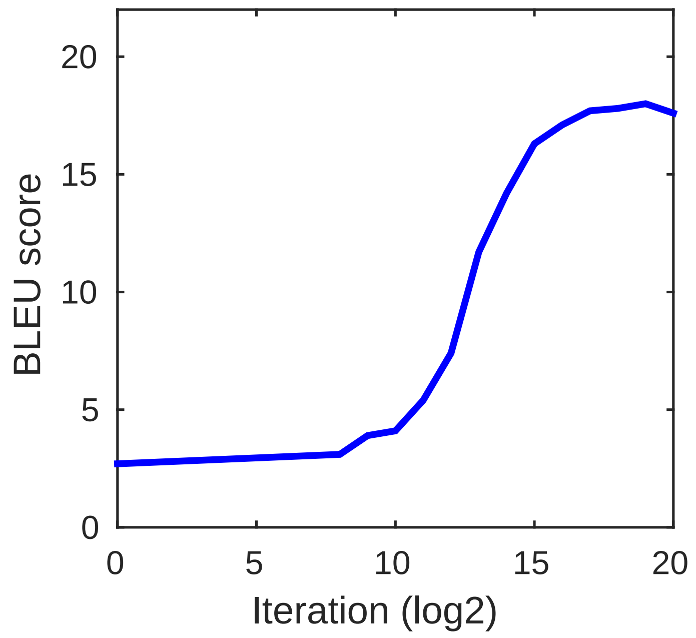

Figure 4.5: BLEU score against iteration when decoding small test set with weights generated from decoding development set with default weights, creating difference vectors by comparing to all subsequent iterations and setting unwanted negative weights to zero.

and Fig. 4.4. The mean BLEU score across the 6 documents before decoding was 2.7, and after decoding was 17.6. Therefore, although the new weights, based on a larger number of instances, do better than the weights derived from a smaller number of instances (Fig. 4.3), they are still not as good as the default weights. Remember the beating the default weights is an absolute minimum requirement; if we can not equal their performance then something is still seriously wrong. The fact that so many weights in Table 4.4 are set to zero is not a promising sign. We know that these models are generally helpful to the translation quality, and by setting their weights to zero they are being discarded entirely.

4.6 Using MERT weights in Docent

So far it has proved impossible to get a higher BLEU score on the test set than that achieved by simply using the default weights. At this stage, it is perhaps sensible to check what happens when we use weights optimised in the standard sentence-based MERT framework, to check that the problems seen so far are not some strange relic of the data set being tested. MERT (Och, 2003) is the most commonly used optimisation method in SMT, and is known to boost performance on evaluation metrics in a wide variety of circumstances. The development set of 111 parallel documents was therefore concatenated to plain text files of 2525 parallel sentences, which were fed to the `mert-moses.pl` script that comes with Moses. The resultant weights are displayed in Table 4.5.

Feature	MERT weight
Distortion 1	0.128
Word Penalty	–0.333
Language Model	0.142
Translation Model 0	0.094
Translation Model 1	0.053
Translation Model 2	0.124
Translation Model 3	0.042
Translation Model 4	–0.084

Table 4.5: Weights calculated by MERT from the development data.

The small test set of six documents was also concatenated into a plain text file of 348 sentences. Decoding in Moses with default weights yielded a BLEU score on this test set of 19.8, while decoding in Moses with the MERT-optimised weights of Table 4.5 yielded a BLEU of 20.2. We see therefore a clear, albeit small, increase in performance on the test set in Moses when optimising with MERT. We saw previously in Section 4.4 that Docent achieves a mean BLEU score across the six documents of the small test set of 20.3. Taking the weights from Table 4.5, and plugging them into the Docent configuration file, we obtain a mean BLEU score of 20.2. The MERT-optimised weights, in other words, improve the performance of the Moses decoder, but do not do so when used in Docent. This could be interpreted in one of two ways: firstly, that it demonstrated clearly the need for a document-level specific tuning paradigm; or secondly, that feature weight tuning in Docent on this particular data set is extremely difficult, explaining the negative results in previous sections.

4.7 Further experiments

Several further experiments were carried out in the attempt to make tuning work in Docent, and it is worth at least mentioning several lines of inquiry that did not result in improved performance. Firstly, a larger development set of newstest data was used, consisting of slightly over 13,500 sentences. Cutting this different ways into separate documents allowed the process to be carried out both with fewer larger documents, and more smaller documents. Results on the test set, were however disappointing, with no improvement in either case over the default weights.

In these results presented in this chapter, both the development and test sets were decoded from random initial translations. The experiments were also repeated with initial translations given by Moses. There are of course various combinations that can be employed here: decoding the development set from a random initial translation and the test set from a Moses-based initial translation, for example. No extra benefit was found in any these experiments, and indeed further problems arose from the fact that there were very few changes to the already high-quality translations during Docent decoding of the development set, leaving very few data points for the classifier to work with.

Another series of experiments involved setting up an iterative process, where the weights from the classifier were used to decode the development set again, generating new weights, and so on. This was based on the idea that perhaps different parts of the search space needed to be explored to eventually find optimum weights; however, performance on the development and test sets either oscillated wildly or simply deteriorated between iterations.

The data sampling strategy was also investigated further: storing more proposed translations in regions where the BLEU score changes the most, for example, and storing at linear intervals rather than logarithmic. An attempt was also made to look specifically for examples where the BLEU score decreases between iterations, and ensure that these were represented sufficiently in the data for the classifier. Some of these methods were more or less successful than others, but ultimately nothing was able to provide weights that achieved better performance than the default weights on the test set.

5 Discussion and Conclusions

This thesis explored two different types of experiment with the document-level SMT decoder Docent. In Chapter 3, BLEU decoding was introduced; a process by which proposed changes to candidate translations are only accepted if they lead to an increase in BLEU score. As BLEU had previously been shown to correlate very well with human judgements of translation quality, it was assumed that BLEU decoding, by producing documents with very high BLEU scores, would consequently produce very high quality translations. The results showed clearly, however, that this is not the case. Instead what was found in many cases were unintelligible translations.

We saw that when starting the decoder from a Moses-based initial configuration with a mean BLEU score of 19.4, Docent in BLEU-decoding mode was able to increase the mean to 47.8 after 1,000,000 iterations. This huge jump in BLEU should have led to a parallel increase in translation quality, but a quick glance at the results showed that no such increase was forthcoming. Instead many understandable, albeit far from perfect, sentences from the Moses translation had been transformed beyond all recognition and could no longer be made sense of. We looked at the language model score of the document as decoding proceeded and saw that this confirmed our suspicions of a serious deterioration in translation quality. It was also shown that when starting from random initial translations, BLEU decoding was still able to achieve similarly high BLEU scores, but again without any corresponding translation quality.

At the end of Chapter 3 experiments with BLEU decoding towards a reachable translation were presented. This shows that almost perfect BLEU scores can be achieved when the reference translation is reachable by the decoder. This, along with the fact that experiments were repeated several times from different initial translations, supports the view that BLEU decoding produces the translations with the highest BLEU scores that the decoder can reach, given the types of feature model employed in SMT. When we optimise towards BLEU score, therefore, we are essentially optimising towards nonsense. That is not however to say that BLEU is not a useful metric in certain controlled circumstances; it is merely to question the logic of directly trying to manipulate the BLEU score produced by our models.

BLEU decoding is completely reliant on the existence of a reference translation, something which is obviously not available in normal circumstances when we want to translate a new document (if it were, we would not need to do the translation in the first place). The stage at which it might be useful, however, is in feature weight tuning; finding the optimal weights for the various feature models in an SMT system. In Chapter 4, experiments were described that attempted initially to take advantage of BLEU decoding in this context. It became clear quickly, however, that pure BLEU decoding was not going to give good enough translations to use in this process, so we switched back to simpler methods of decoding to find candidate translations from which to extract the relevant data for tuning.

The idea was to use translations stored at regular intervals during decoding to calculate difference vectors in feature model scores, and feed this information to a linear binary classifier to obtain weights, with the class label given by whether the BLEU score had increased or decreased between the iterations in question. This method mirrors that used in the PRO algorithm for sentence-level phrase-based SMT decoders. It was hoped that a successful baseline model could eventually be compared to a model incorporating BLEU decoding, but once again results were not as expected. The first attempt was to simply decode a development set with default weights, and use candidate translations stored at logarithmic intervals to calculate the necessary difference vectors. Subsequently decoding a small test set with the obtained weights, the BLEU score was found to decrease by the end of decoding. This was a very surprising result, as the aim was not just to make BLEU increase, but to make it increase by more than it would do with default weights on the test set. That is the whole point of tuning.

Several avenues for improvement were then explored, with a moderate degree of success. The first of these was to simply set unlikely negative weights to zero. This produced an immediate gain, with BLEU score increasing significantly on the test set with the new weights. The final results were, however, still significantly lower than those produced by just using the default weights. The sparseness of data available to the classifier was also proposed as a possible reason for the failure of the experiments, and a method to increase the number of data points by almost an order of magnitude was proposed and tested. This improved the final BLEU score yet further, but still not to the level of that obtained with default weights. At this point a different type of experiment was carried out, instead tuning the weights in a standard way but on the same data set with MERT. This proved that MERT works on this data set for decoding with Moses. However, even with these MERT weights, the final BLEU score at the end of Docent decoding was lower than with standard weights.

The exact reason for the failure of the tuning experiments remains unclear. It is possible that the default weights used in Docent (and Moses) happen to perform extremely well on the data set used in this thesis (of course these default weights are not random, they are specifically chosen as they are known to perform well in many circumstances). Many of these experiments have however been repeated on other data sets, with similar results. The number of data points is also a potential issue, even with the expanded set fed to the classifier in later experiments. Perhaps sampling of candidate translations needs to be carried out much more frequently, or simply a larger number of documents is required. The data sampling interval is also potentially an important variable; in the experiments presented here a logarithmic sampling method was employed. Previous experiments in feature weight tuning in Docent employed instead linear sampling, with some positive results (Stymne et al., 2013). Extra experiments, not documented in detail in this thesis, failed to find any benefit to varying the sampling frequency or interval.

Overall, the experiments presented in this thesis led to two surprising results:

firstly, that very high BLEU scores can be produced despite very low translation quality; and secondly that it was much more difficult than expected to adapt sentence-level optimisation techniques to the document level. Despite this, the potential for an effective tuning algorithm in Docent remains high. If a working baseline system can be found, the addition of BLEU decoding, not in its pure form but in combination with more standard feature models, has the potential to boost performance significantly. The presence of standard models will likely keep the decoder in a region of the search space with good translations, while the presence of the BLEU feature model will work to maximise translation quality in this region. The necessity for reliable tuning techniques is very apparent, and will only become more important as further document-level feature models are developed.

Bibliography

Bazrafshan, M., Chung, T., and Gildea, D. (2012). Tuning as Linear Regression. In *Proceedings of the 2012 Conference of the North American Chapter of the ACL: Human Language Technologies*, pages 543–547, Montreal, Canada. Association for Computational Linguistics.

Callison-Burch, C., Osborne, M., and Koehn, P. (2006). Re-evaluating the Role of BLEU in Machine Translation Research. In *11th Conference of the European Chapter of the Association for Computational Linguistics*, pages 249–256, Trento, Italy. Association for Computational Linguistics.

Cherry, C. and Foster, G. (2012). 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, Montreal, Canada. Association for Computational Linguistics.

Daumé III, H. (2004). Notes on CG and LM-BFGS Optimization of Logistic Regression.

Galley, M., Quirk, C., Cherry, C., and Toutanova, K. (2013). Regularized Minimum Error Rate Training. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, Seattle, Washington. Association for Computational Linguistics.

Hardmeier, C., Nivre, J., and Tiedemann, J. (2012). Document-Wide Decoding for Phrase-Based Statistical Machine Translation. In *Proceedings of the 2012 Joint Conference on EmpiricalMethods in Natural Language Processing and Computational Natural LanguageLearning*, pages 1179–1190, Jeju Island, Korea. Association for Computational Linguistics.

Hardmeier, C., Stymne, S., Tiedemann, J., and Nivre, J. (2013). Docent: A Document-Level Decoder for Phrase-Based Statistical Machine Translation. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 193–198, Sofia, Bulgaria. Association for Computational Linguistics.

Hopkins, M. and May, J. (2011). Tuning as Ranking. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 1352–1362, Edinburgh, Scotland, UK. Association for Computational Linguistics.

Koehn, P., Hoang, H., Birch, A., Callison-Burch, C., Federico, M., Bertoldi, N., Cowan, B., Shen, W., Moran, C., Zens, R., et al. (2007). 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.

Koehn, P., Och, F. J., and Marcu, D. (2003). Statistical Phrase-based Translation. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology - Volume 1*, NAACL '03, pages 48–54, Stroudsburg, PA, USA. Association for Computational Linguistics.

Macháček, M. and Bojar, O. (2013). Results of the WMT13 Metrics Shared Task. In *Proceedings of the Eighth Workshop on Statistical Machine Translation*, pages 45–51, Sofia, Bulgaria. Association for Computational Linguistics.

Och, F. J. (2003). Minimum Error Rate Training in Statistical Machine Translation. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics - Volume 1*, ACL '03, pages 160–167, Stroudsburg, PA, USA. Association for Computational Linguistics.

Och, F. J. and Ney, H. (2002). Discriminative Training and Maximum Entropy Models for Statistical Machine Translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, ACL '02, pages 295–302, Stroudsburg, PA, USA. Association for Computational Linguistics.

Papineni, K., Roukos, S., Ward, T., and Zhu, W. J. (2002). BLEU: a Method for Automatic Evaluation of Machine Translation. In *Proceeding of the 40th Annual Meeting of the Association for Computationial Linguistics*, pages 311–318, Philadelphia. Association for Computational Linguistics.

Stymne, S., Hardmeier, C., Tiedemann, J., and Nivre, J. (2013). Feature Weight Optimization for Discourse-Level SMT. In *Proceedings of the Workshop on Discourse in Machine Translation (DiscoMT)*, pages 60–69, Sofia, Bulgaria. Association for Computational Linguistics.

Watanabe, T., Suzuki, J., Tsukada, H., and Isozaki, H. (2007). 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. Association for Computational Linguistics.