Feature Weight Optimization for Discourse-Level SMT

Sara Stymne, Christian Hardmeier, Jörg Tiedemann and Joakim Nivre
Uppsala University
Department of Linguistics and Philology
Box 635, 751 26 Uppsala, Sweden
firstname.lastname@lingfil.uu.se

Abstract

We present an approach to feature weight optimization for document-level decoding. This is an essential task for enabling future development of discourse-level statistical machine translation, as it allows easy integration of discourse features in the decoding process. We extend the framework of sentence-level feature weight optimization to the document-level. We show experimentally that we can get competitive and relatively stable results when using a standard set of features, and that this framework also allows us to optimize document-level features, which can be used to model discourse phenomena.

1 Introduction

Discourse has largely been ignored in traditional machine translation (MT). Typically each sentence has been translated in isolation, essentially yielding translations that are bags of sentences. It is well known from translation studies, however, that discourse is important in order to achieve good translations of documents (Hatim and Mason, 1990). Most attempts to address discourse-level issues for statistical machine translation (SMT) have had to resort to solutions such as post-processing to address lexical cohesion (Carpuat, 2009) or two-step translation to address pronoun anaphora (Le Nagard and Koehn, 2010). Recently, however, we presented Docent (Hardmeier et al., 2012; Hardmeier et al., 2013), a decoder based on local search that translates full documents. So far this decoder has not included a feature weight optimization framework. However, feature weight optimization, or tuning, is important for any modern SMT decoder to achieve a good translation performance.

In previous research with Docent, we used grid search to find weights for document-level features while base features were optimized using standard sentence-level techniques. This approach is impractical since many values for the extra features have to be tried, and, more importantly, it might not give the same level of performance as jointly optimizing all parameters. Principled feature weight optimization is thus essential for researchers that want to use document-level features to model discourse phenomena such as anaphora, discourse connectives, and lexical consistency. In this paper, we therefore propose an approach that supports discourse-wide features in document-level decoding by adapting existing frameworks for sentence-level optimization. Furthermore, we include a thorough empirical investigation of this approach.

2 Discourse-Level SMT

Traditional SMT systems translate texts sentence by sentence, assuming independence between sentences. This assumption allows efficient algorithms based on dynamic programming for exploring a large search space (Och et al., 2001). Because of the dynamic programming assumptions it is hard to directly include discourse-level features into a traditional SMT decoder. Nevertheless, there have been several attempts to integrate intersentential and long distance models for discourse-level phenomena into standard decoders, usually as ad-hoc additions to standard models, addressing a single phenomenon.

Several studies have tried to improve pronoun anaphora by adding information about the antecedent, either by using two-step decoding (Le Nagard and Koehn, 2010; Guillou, 2012) or by extracting information from previously translated sentences (Hardmeier and Federico, 2010), unfortunately without any convincing results. To address the translation of discourse connectives, source-side pre-processing has been used to annotate surface forms either in the corpus or in the

Proceedings of the Workshop on Discourse in Machine Translation (DiscoMT), pages 60–69,
Sofia, Bulgaria, August 9, 2013.

phrase-table (Meyer and Popescu-Belis, 2012) or by using factored decoding (Meyer et al., 2012) to disambiguate connectives, with small improvements. Lexical consistency has been addressed by the use of post-processing (Carpuat, 2009), multi-pass decoding (Xiao et al., 2011; Ture et al., 2012), and cache models (Tiedemann, 2010; Gong et al., 2011). Gong et al. (2012) addressed the issue of tense selection for translation from Chinese, by the use of inter-sentential tense n-grams, exploiting information from previously translated sentences. Another way to use a larger context is by integrating word sense disambiguation and SMT. This has been done by re-initializing phrase probabilities for each sentence (Carpuat and Wu, 2007), by introducing extra features in the phrase-table (Chan et al., 2007), or as a k-best re-ranking task (Specia et al., 2008). Another type of approach is to integrate topic modeling into phrase tables (Zhao and Xing, 2010; Su et al., 2012). For a more thorough overview of discourse in SMT, see Hardmeier (2012).

Here we instead choose to work with the recent document-level SMT decoder Docent (Hardmeier et al., 2012). Unlike in traditional decoding were documents are generated sentence by sentence, feature models in Docent always have access to the complete discourse context, even before decoding is finished. It implements the phrase-based SMT approach (Koehn et al., 2003) and is based on local search, where a state consists of a full translation of a document, which is improved by applying a series of operations to improve the translation. A hill-climbing strategy is used to find a (local) maximum. The operations allow changing the translation of a phrase, changing the word order by swapping the positions of two phrases, and resegmenting phrases. The initial state can either be initialized randomly in monotonic order, or be based on an initial run from a standard sentence-based decoder. The number of iterations in the decoder is controlled by two parameters, the maximum number of iterations and a rejection limit, which stops the decoder if no change was made in a certain number of iterations. This setup is not limited by dynamic programming constraints, and enables the use of the translated target document to extract features. It is thus easy to directly integrate discourse-level features into Docent. While we use this specific decoder in our experiments, the method proposed for document-level feature weight optimization is not limited to it. It can be used with any decoder that outputs feature values at the document level.

3 Sentence-Level Tuning

Traditionally, feature weight optimization, or tuning, for SMT is performed by an iterative process where a development set is translated to produce a k-best list. The parameters are then optimized using some procedure, generally to favor translations in the k-best list that have a high score on some MT metric. The translation step is then repeated using the new weights for decoding, and optimization is continued on a new k-best list, or on a combination of all k-best lists. This is repeated until some end condition is satisfied, for instance for a set number of iterations, until there is only very small changes in parameter weights, or until there are no new translations in the k-best lists.

SMT tuning is a hard problem in general, partly because the correct output is unreachable and also because the translation process includes latent variables, which means that many efficient standard optimization procedures cannot be used (Gimpel and Smith, 2012). Nevertheless, there are a number of techniques including MERT (Och, 2003), MIRA (Chiang et al., 2008; Cherry and Foster, 2012), PRO (Hopkins and May, 2011), and Rampion (Gimpel and Smith, 2012). All of these optimization methods can be plugged into the standard optimization loop. All of the methods work relatively well in practice, even though there are limitations, for instance that many methods are non-deterministic meaning that their results are somewhat unstable. However, there are some important differences. MERT is based on scores for the full test set, whereas the other methods are based on sentence-level scores. MERT also has the drawback that it only works well for small sets of features. In this paper we are not concerned with the actual optimization algorithm and its properties, though, but instead we focus on the integration of document-level decoding into the existing optimization frameworks.

In order to adapt sentence-level frameworks to our needs we need to address the granularity of scoring and the process of extracting k-best lists. For document-level features we do not have meaningful scores on the sentence level which are required in standard optimization frameworks. Furthermore, the extraction of k-best lists is not as

Input: inputDocs, refDocs, init weights θ_0, max decoder iters max, sample start ss, sample interval si,
Output: learned weights θ

$\theta \leftarrow \theta_0$
Initialize empty klist
run $\leftarrow 1$
repeat
  Initialize empty $\text{klist}_{\text{run}}$
  for doc $\leftarrow 1$, inputDocs.size **do** Initialize decoder state randomly for inputDocs[doc]
    for iter $\leftarrow 1$, max **do**
      Perform one hill-climbing step for inputDocs[doc]
      if iter >= ss & iter mod si == 0 **then**
        Add translation for inputDocs[doc] to $\text{klist}_{\text{run}}$
      end if
    end for
  end for
  Merge $\text{klist}_{\text{run}}$ with klist
  $\text{modelScores}_{\text{doc}} \leftarrow$ `ComputeModelScores`(klist)
  $\text{metricStats}_{\text{doc}} \leftarrow$ `ComputeMetricStats`(klist, refDocs)
  $\theta_{\text{run}} \leftarrow \theta$
  $\theta \leftarrow$ `Optimize`(θ_{run}, $\text{modelScores}_{\text{doc}}$, $\text{metricStats}_{\text{doc}}$)
  run $\leftarrow$ run + 1
until `Done`(run, θ, θ_{run})

Figure 1: Document-level feature weight optimization algorithm

straightforward in our hill-climbing decoder as in standard sentence-level decoders such as Moses (Koehn et al., 2007) where such a list can be approximated easily from the internal beam search strategy. Working on output lattices is another option in standard approaches (Cherry and Foster, 2012) which is also not applicable in our case.

In the following section we describe how we can address these issues in order to adapt sentence-level frameworks for our purposes.

4 Document-Level Tuning

To allow document-level feature weight optimization, we make some small changes to the sentence-level framework. Figure 1 shows the algorithm we use. It assumes access to an optimization algorithm, `Optimize`, and an end criterion, `Done`. The changes from standard sentence-level optimization is that we compute scores on the document level, and that we sample translations instead of using standard k-best lists.

The main challenge is that we need meaningful scores which we do not have at the sentence level in document decoding. We handle this by simply computing all scores (model scores and metric scores) exclusively at the document level. Remember that all standard MT metrics based on sentence-level comparisons with reference translations can be aggregated for a complete test set. Here we do the same for all sentences in a given document. This can actually be an advantage compared to optimization methods that use sentence-level scores, which are known to be unreliable (Callison-Burch et al., 2012). Document-level scores should thus be more stable, since they are based on more data. A potential drawback is that we get fewer data points with a test set of the same size, which might mean that we need more data to achieve as good results as with sentence-level optimization. We will see the ability of our approach to optimize weights with reasonable data sets in our experiments further down.

The second problem, the extraction of k-best lists can be addressed in several ways. It is possible to get a k-best list from Docent by extracting the results from the last k iterations. However, since Docent operates on the document-level and does not accept updates in each iteration, there will be many identical and/or very similar hypotheses with such an approach. Another option would be to extract the translations from the k last different iterations, which would require some small changes to the decoder. Instead, we opt to use k-lists, lists of translations sampled with some interval, which contains k translations, but not necessarily all the k best translations that could be found by the decoder. A k-best list is of course a k-list, which we get with a sample interval of 1.

We also choose to restart Docent randomly in each optimization iteration, since it allows us to explore a larger part of the search space. We empirically found that this strategy worked better than restarting the decoder from the previous best state.

	German–English			English–Swedish		
	Type	Sentences	Documents	Type	Sentences	Documents
Training	Europarl	1.9M	–	Europarl	1.5M	–
	News Commentary	178K	–	–	–	–
Tuning	News2009	2525	111	Europarl (Moses)	2000	–
	News2008-2010	7567	345	Europarl (Docent)	1338	100
Test	News2012	3003	99	Europarl	690	20

Table 1: Domain and number of sentences and documents for the corpora

As seen in Figure 1, there are some additional parameters in our procedure: the sample start iteration and the sample interval. We also need to set the number of decoder iterations to run. In Section 5 we empirically investigate the effect of these parameters.

Compared to sentence-level optimization, we also have a smaller number of units to get scores from, since we use documents as units, and not sentences. The importance of this depends on the optimization algorithm. MERT calculates metric scores over the full tuning set, not for individual sentences, and should not be affected too much by the change in granularity. Many other optimization algorithms, like PRO, work on the sentence level, and will likely be more affected by the reduction of units. In this work we focus on MERT, which is the most commonly used optimization procedure in the SMT community, and which tends to work quite well with relatively few features. However, we also show contrastive results for PRO (Hopkins and May, 2011). A further issue is that Docent is non-deterministic, i.e., it can give different results with the same parameter weights. Since the optimization process is already somewhat unstable this is a potential issue that needs to be explored further, which we do in Section 5.

Implementation-wise we adapted Docent to output k-lists and adapted the infrastructure available for tuning in the Moses decoder (Koehn et al., 2007) to work with document-level scores. This setup allows us to use the variety of optimization procedures implemented there.

5 Experiments

In this section we report experimental results where we investigate several issues in connection with document-level feature weight optimization for SMT. We first describe the experimental setup, followed by baseline results using sentence-level optimization. We then present validation experiments with standard sentence-level features, which can be compared to standard optimization. Finally, we report results with a set of document-level features that have been proposed for joint translation and text simplification (Stymne et al., 2013).

5.1 Experimental Setup

Most of our experiments are for German-to-English news translation using data from the WMT13 workshop.[1] We also show results with document-level features for English-to-Swedish Europarl (Koehn, 2005). The size of the training, tuning, and test sets are shown in Table 1. First of all, we need to extract documents for tuning and testing with Docent. Fortunately, the news data already contain document markup, corresponding to individual news articles. For Europarl we define a document as a consecutive sequence of utterances from a single speaker. To investigate the effect of the size of the tuning set, we used different subsets of the available tuning data.

All our document-level experiments are carried out with Docent but we also contrast with the Moses decoder (Koehn et al., 2007). For the purpose of comparison, we use a standard set of sentence-level features used in Moses in most of our experiments: five translation model features, one language model feature, a distance-based reordering penalty, and a word count feature. For feature weight optimization we also apply the standard settings in the Moses toolkit. We optimize towards the Bleu metric, and optimization ends either when no weights are changed by more than 0.00001, or after 25 iterations. MERT is used unless otherwise noted.

Except for one of our baselines, we always run Docent with random initialization. For test we run the document decoder for a maximum of 2^{27} iterations with a rejection limit of 100,000. In our experiments, the decoder always stopped when reaching the rejection limit, usually between 1–5

[1] http://www.statmt.org/wmt13/translation-task.html

million iterations.

We show results on the Bleu (Papineni et al., 2002) and NIST (Doddington, 2002) metrics. For German–English we show the average result and standard deviation of three optimization runs, to control for optimizer instability as proposed by Clark et al. (2011). For English–Swedish we report results on single optimization runs, due to time constraints.

5.2 Baselines

Most importantly, we would like to show the effectiveness of the document-level tuning procedure described above. In order to do this, we created a baseline using sentence-level optimization with a tuning set of 2525 sentences and the News2009 corpus for evaluation. Increasing the tuning set is known to give only modest improvements (Turchi et al., 2012; Koehn and Haddow, 2012).

The feature weights optimized with the standard Moses decoder can then directly be used in our document-level decoder as we only include sentence-level features in our baseline model. As expected, these optimized weights also lead to a better performance in document-level decoding compared to an untuned model as shown in Table 2. Note, that Docent can be initialized in two ways, by Moses and randomly. Not surprisingly, the result for the runs initialized with Moses are identical with the pure sentence-level decoder. Initializing randomly gives a slightly lower Bleu score but with a larger variation than with Moses initialization, which is also expected. Docent is non-deterministic, and can give somewhat varying results with the same weights. However, this variation has been shown experimentally to be very small (Hardmeier et al., 2012).

Our goal now is to show that document-level tuning can perform equally well in order to verify our approach. For this, we set up a series of experiments looking at varying tuning sets and different parameters of the decoding and optimization procedure. With this we like to demonstrate the stability of the document-level feature weight optimization approach presented above. Note that the most important baselines for comparison with the results in the next sections are the ones with Docent and random initialization.

5.3 Sentence-Level Features

In this section we present validation results where we investigate different aspects of document-level feature weight optimization with standard sentence-level features. In this way we can compare the results directly to standard sentence-level optimization, and to the results of Moses.

System	Tuning	Bleu	NIST
Moses	None	17.7	6.25
Docent-M	None	17.7	6.25
Docent-R	None	15.2 (0.05)	5.88 (0.00)
Moses	Moses	18.3 (0.04)	6.22 (0.01)
Docent-M	Moses	18.3 (0.04)	6.22 (0.01)
Docent-R	Moses	18.1 (0.13)	6.23 (0.01)

Table 2: Baseline results, where Docent-M is initialized with Moses and Docent-R randomly

Docs	Sent.	Min	Max	Bleu	NIST
111	2525	3	127	18.0 (0.11)	6.19 (0.04)
345	7567	3	127	18.1 (0.14)	6.25 (0.02)
100	1921	8	40	18.0 (0.05)	6.25 (0.10)
200	3990	8	40	17.9 (0.25)	6.20 (0.09)
100	2394	8	100	18.0 (0.12)	6.27 (0.07)
200	4600	8	100	18.1 (0.29)	6.26 (0.10)
300	6852	8	100	18.2 (0.13)	6.27 (0.03)

Table 3: Results for German–English with varying sizes of tuning set, where the number of sentences and documents are varied, as well as the minimum and maximum number of sentences per document

Corpus size We investigate how tuning is affected by corpus size. The corpus size was varied in two ways, by changing the number of documents in the tuning set, and by changing the length of documents in the tuning sets. In this experiment we run 20000 decoder iterations per optimization iteration, and use a k-list of size 101, with sample interval 100. Table 3 shows the results with varying tuning set sizes for German–English. There is very little variation between the scores, and no clear tendencies. All results are of similar quality to the baseline with random initialization and sentence-level tuning, and better than not using any tuning. The top line in Table 3 is News2009, the same tuning set as for the baselines. The scores are somewhat more unstable than the baseline scores, but stability is not related to corpus size. In the following sections we will use the tuning set with 200 documents, size 8-40.

Number of decoder iterations and k-list sampling Two issues that are relevant for feature weight optimization with the document-level decoder is the number of decoder hill-climbing iterations in each optimization iteration, and the settings for k-list sampling. These choices affect the

Iterations	K-list	UTK	Bleu	NIST
20000	101	55.6	17.9 (0.25)	6.20 (0.09)
30000	201	67.2	17.9 (0.06)	6.21 (0.01)
40000	301	79.9	18.2 (0.11)	6.28 (0.09)
50000	401	86.9	18.1 (0.20)	6.22 (0.05)
75000	651	99.2	17.8 (0.15)	6.13 (0.03)
100000	901	106.8	17.9 (0.17)	6.16 (0.03)
30000	101	21.6	18.0 (0.15)	6.21 (0.02)
40000	101	12.6	17.7 (0.53)	6.12 (0.15)
50000	101	8.2	17.9 (0.24)	6.18 (0.06)

Table 4: Results for German–English with a varying number of iterations and k-list size (UTK is the average number of unique translations per document in the k-lists)

quality of the translations in each optimization iteration, and the spread in the k-list. We will report the average number of unique translations per document in the k-lists, *UTK*, during feature weight optimization, in this section.

The top half of Table 4 shows results with a different number of iterations, when we sample k-lists from iteration 10000 with interval 100 for German–English, which means that the size of the k-lists also changes. The differences on MT metrics are very small. The number of new unique translations in the k-lists decrease with the number of decoder iterations. With 20K iterations, 55% of the k-lists entries are unique, which could be compared to only 12% with 100K iterations. The majority of the unique translations are thus found in the beginning of the decoding, which is not surprising.

The bottom half of Table 4 shows results with a different number of decoder iterations, but a set k-list size. In this setting the number of unique hypotheses in the k-lists obviously decreases with the number of decoder iterations. Despite this, there are mostly small result differences, except for 40K iterations, which has more unstable results than the other settings. It does not seem useful to increase the number of decoder iterations without also increasing the size of the k-list. An even better strategy might be to only include unique entries in the k-lists. We will explore this in future work.

We also ran experiments where we did not restart the decoder with a random state in each iteration, but instead saved the previous state and continued decoding with the new weights from there. This, however, was largely unsuccessful, and gave very low scores. We believe that the reason for this is mainly that a much smaller part of the search space is explored when the decoder is not restarted with a new seed repeatedly. The fact that a higher overall quality can be achieved with a higher number of iterations (see Figure 2) can apparently not compensate for this drawback.

Interval	Start	UTK	Bleu	NIST
1	19900	1.4	18.2 (0.07)	6.25 (0.04)
10	19000	5.2	18.1 (0.08)	6.22 (0.03)
100	10000	55.6	17.9 (0.25)	6.20 (0.09)
200	0	82.2	17.9 (0.19)	6.15 (0.05)

Table 5: Results with different k-list-sample intervals for k-lists size 101 (UTK is the average number of unique translations per document in the k-lists)

Finally, we investigate the effect of the sample interval for the k-lists. To get k-lists of equal size, 101, we start the sampling at different iterations. Table 5 shows the results, and we can see that with a small sample interval, the number of unique translations decreases drastically. Despite this, there are no large result differences. There is actually a slight trend that a smaller sample interval is better. This does not confirm our intuition that it is important with many different translations in the k-list. Especially for interval 1 it is surprising, since there is often only 1 unique translation for a single document. We believe that the fact that k-lists from different iterations are joined, can be part of the explanation for these results. We think more work is needed in the future, to further explore these settings, and the interaction with the total number of decoder iteration, and the k-list sampling.

To further shed some light to these results, we show learning curves from the optimization. Figure 2 shows Bleu scores for the system optimized with 100K decoder iterations after different numbers of iterations, for the last three iterations in each of the three optimization runs. As shown in Hardmeier et al. (2012), the translation quality increases fast at first, but start to level out at around 40K iterations. Despite this, the optimization results are good even with 20K iterations, which is somewhat surprising. Figures 3 and 4 show the Bleu scores after each tuning iteration for the systems in Tables 4 and 5. As is normal for SMT tuning, the convergence is slow, and there are some oscillations even late in the optimization. Overall systems with many iterations seem somewhat more stable.

Overall, the results are better than the untuned

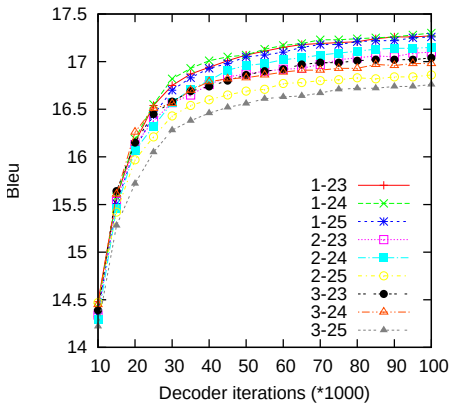

Figure 2: Bleu scores during 100000 Docent iterations during feature weight optimization

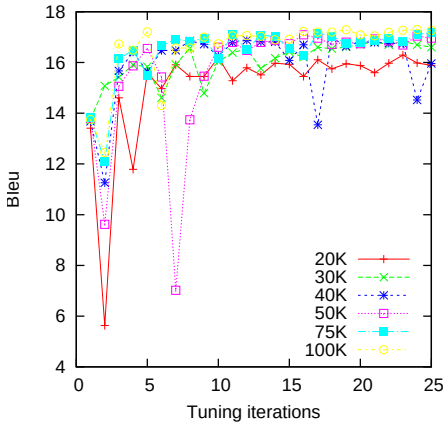

Figure 3: Bleu scores during feature weight optimization for systems with different number of decoder iterations and k-list sizes.

baseline and on par with the sentence-level tuning baselines in all settings, with a relatively modest variation, even across settings. In fact, if we calculate the total scores of all 36 systems in Tables 4 and 5, we get a Bleu score of 18.0 (0.23) and a NIST score of 6.19 (0.07), with a variation that is not higher than for many of the different settings.

Optimization method In this section we compare the performance of the MERT optimization algorithm with that of PRO, and a combination that starts MERT with weights initialized with PRO (MERT+PRO), suggested by Koehn and Haddow (2012). Here we run 30000 decoder iterations. Table 6 shows the results. Initializing MERT with PRO did not affect the scores much. The scores with only PRO, however, are slightly lower than for MERT, and have a much larger score variation. This could be because PRO is

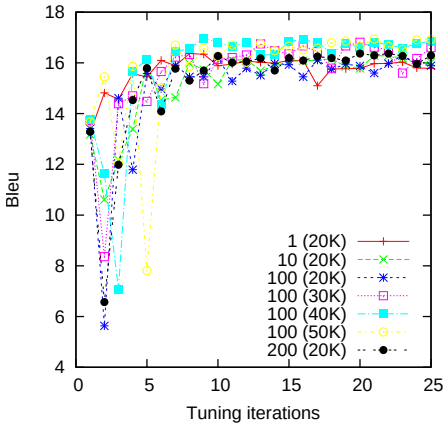

Figure 4: Bleu scores during feature weight optimization for systems with different k-list sample interval and number of decoder iterations.

	Bleu	NIST
MERT	17.9 (0.06)	6.21 (0.01)
PRO	17.5 (0.41)	6.15 (0.20)
MERT+PRO	18.0 (0.12)	6.18 (0.06)

Table 6: Results with different optimization algorithms for German–English

likely to need more data, since it calculates metric scores on individual units, sentences or documents, not across the full tuning set, like MERT. This likely means that 200 documents are too few for stable results with optimization methods that depend on unit-level metric scores.

5.4 Document-Level Features

In this section we investigate the effect of optimization with a number of document-level features. We use a set of features proposed in Stymne et al. (2013), in order to promote the readability of texts. In this scenario, however, we use these features in a standard SMT setting, where they can potentially improve the lexical consistency of translations. The features are:

- Type token ratio (TTR) – the ratio of types, unique words, to tokens, total number of words
- OVIX – a reformulation of TTR that has traditionally been used for Swedish and that is less sensible to text length than TTR, see Eq. 1
- Q-value, phrase level (QP) - The Q-value was developed as a measure for bilingual term quality (Deléger et al., 2006), to promote common and consistently translated terms. See Eq. 2, where $f(st)$ is the frequency of

System	Optimization	German–English Bleu	German–English NIST	English–Swedish Bleu	English–Swedish NIST
Moses	Sentence	18.3 (0.04)	6.22 (0.01)	24.3	6.12
Docent	Sentence	18.1 (0.13)	6.23 (0.01)	24.1	6.06
Docent	Document	17.9 (0.25)	6.20 (0.09)	23.4	6.01
TTR	Document	18.3 (0.16)	6.33 (0.04)	23.6	6.15
OVIX	Document	18.3 (0.13)	6.30 (0.03)	23.4	5.99
QW	Document	18.1 (0.14)	6.22 (0.03)	24.2	6.11
QP	Document	18.0 (0.10)	6.23 (0.05)	21.2	5.70

Table 7: Results when using document-level features

the phrase pair, $n(s)$ is the number of unique target phrases which the source phrase is aligned to in the document, and $n(t)$ is the same for the target phrase. Here the Q-value is applied on the phrase level.

- Q-value, word level (QW) - Same as above, but here we apply the Q-value for source words and their alignments on the target side.

$$\text{OVIX} = \frac{\log(\text{count(tokens)})}{\log\left(2 - \frac{\log(\text{count(types)})}{\log(\text{count(tokens)})}\right)} \quad (1)$$

$$\text{Q-value} = \frac{f(st)}{n(s) + n(t)} \quad (2)$$

We added these features one at a time to the standard feature set. Optimization was performed with 20000 decoder iterations, and a k-list of size 101. As shown in the previous sections, there are slightly better settings, which could have been used to boost the results somewhat.

The results are shown in Table 7. For German–English, the results are generally on par with the baselines for Bleu and slightly higher on NIST for OVIX and TTR. For English–Swedish, we used a smaller tuning set on the document level than on the sentence level, see Table 1, due to time constraints. This is reflected in the scores, which are generally lower than for sentence-level decoding. Using the QW feature, however, we receive competitive scores to the sentence-based baselines, which indicates that it can be meaningful to use document-level features with the suggested tuning approach.

While the results do not improve much over the baselines, these experiments still show that we can optimize discourse-level features with our approach. We need to identify more useful document-level features in future work, however.

6 Conclusion

We have shown how the standard feature weight optimization workflow for SMT can be adapted to document-level decoding, which allows easy integration of discourse-level features into SMT. We modified the standard framework by calculating scores on the document-level instead of the sentence level, and by using k-lists rather than k-best lists.

Experimental results show that we can achieve relatively stable results, on par with the results for sentence-level optimization and better than without tuning, with standard features. This is despite the fact that we use the hill-climbing decoder without initialization by a standard decoder, which means that it is somewhat unstable, and is not guaranteed to find any global maximum, even according to the model. We also show that we can optimize document-level features successfully. We investigated the effect of a number of parameters relating to tuning set size, the number of decoder iterations, and k-list sampling. There were generally small differences relating to these parameters, however, indicating that the suggested approach is robust. The interaction between parameters does need to be better explored in future work, and we also want to explore better sampling, without duplicate translations.

This is the first attempt of describing and experimentally investigating feature weight optimization for direct document-level decoding. While we show the feasibility of extending sentence-level optimization to the document level, there is still much more work to be done. We would, for instance, like to investigate other optimization procedures, especially for systems with a high number of features. Most importantly, there is a large need for the development of useful discourse-level features for SMT, which can now be optimized.

Acknowledgments

This work was supported by the Swedish strategic research programme eSSENCE.

References

Chris Callison-Burch, Philipp Koehn, Christof Monz, Matt Post, Radu Soricut, and Lucia Specia. 2012. Findings of the 2012 workshop on statistical machine translation. In *Proceedings of the Seventh Workshop on Statistical Machine Translation*, pages 10–51, Montréal, Canada.

Marine Carpuat and Dekai Wu. 2007. Improving statistical machine translation using word sense disambiguation. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 61–72, Prague, Czech Republic.

Marine Carpuat. 2009. One translation per discourse. In *Proceedings of the Workshop on Semantic Evaluations: Recent Achievements and Future Directions (SEW-2009)*, pages 19–27, Boulder, Colorado.

Yee Seng Chan, Hwee Tou Ng, and David Chiang. 2007. Word sense disambiguation improves statistical machine translation. In *Proceedings of the 45th Annual Meeting of the ACL*, pages 33–40, Prague, Czech Republic.

Colin Cherry and George Foster. 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, Montréal, Canada.

David Chiang, Yuval Marton, and Philip Resnik. 2008. Online large-margin training of syntactic and structural translation features. In *Proceedings of Human Language Technologies: The 2008 Annual Conference of the NAACL*, pages 224–233, Honolulu, Hawaii.

Jonathan H. Clark, Chris Dyer, Alon Lavie, and Noah A. Smith. 2011. Better hypothesis testing for statistical machine translation: Controlling for optimizer instability. In *Proceedings of the 49th Annual Meeting of the ACL: Human Language Technologies*, pages 176–181, Portland, Oregon, USA.

Louise Deléger, Magnus Merkel, and Pierre Zweigenbaum. 2006. Enriching medical terminologies: an approach based on aligned corpora. In *International Congress of the European Federation for Medical Informatics*, pages 747–752, Maastricht, The Netherlands.

George Doddington. 2002. Automatic evaluation of machine translation quality using n-gram co-occurence statistics. In *Proceedings of the Second International Conference on Human Language Technology*, pages 228–231, San Diego, California, USA.

Kevin Gimpel and Noah A. Smith. 2012. Structured ramp loss minimization for machine translation. In *Proceedings of the 2012 Conference of the NAACL: Human Language Technologies*, pages 221–231, Montréal, Canada.

Zhengxian Gong, Min Zhang, and Guodong Zhou. 2011. 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.

Zhengxian Gong, Min Zhang, Chew Lim Tan, and Guodong Zhou. 2012. N-gram-based tense models for statistical machine translation. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 276–285, Jeju Island, Korea.

Liane Guillou. 2012. Improving pronoun translation for statistical machine translation. In *Proceedings of the EACL 2012 Student Research Workshop*, pages 1–10, Avignon, France.

Christian Hardmeier and Marcello Federico. 2010. Modelling pronominal anaphora in statistical machine translation. In *Proceedings of the International Workshop on Spoken Language Translation*, pages 283–289, Paris, France.

Christian Hardmeier, Joakim Nivre, and Jörg Tiedemann. 2012. Document-wide decoding for phrase-based statistical machine translation. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 1179–1190, Jeju Island, Korea.

Christian Hardmeier, Sara Stymne, Jörg Tiedemann, and Joakim Nivre. 2013. Docent: A document-level decoder for phrase-based statistical machine translation. In *Proceedings of the 51st Annual Meeting of the ACL, Demonstration session*, Sofia, Bulgaria.

Christian Hardmeier. 2012. Discourse in statistical machine translation: A survey and a case study. *Discours*, 11.

Basil Hatim and Ian Mason. 1990. *Discourse and the Translator*. Longman, London, UK.

Mark Hopkins and Jonathan May. 2011. Tuning as ranking. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 1352–1362, Edinburgh, Scotland.

Philipp Koehn and Barry Haddow. 2012. Towards effective use of training data in statistical machine translation. In *Proceedings of the Seventh Workshop on Statistical Machine Translation*, pages 317–321, Montréal, Canada.

Philipp Koehn, Franz Josef Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *Proceedings of the 2003 Human Language Technology Conference of the NAACL*, pages 48–54, Edmonton, Alberta, Canada.

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. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th Annual Meeting of the ACL, Demo and Poster Sessions*, pages 177–180, Prague, Czech Republic.

Philipp Koehn. 2005. Europarl: A parallel corpus for statistical machine translation. In *Proceedings of MT Summit X*, pages 79–86, Phuket, Thailand.

Ronan Le Nagard and Philipp Koehn. 2010. Aiding pronoun translation with co-reference resolution. In *Proceedings of the Joint Fifth Workshop on Statistical Machine Translation and MetricsMATR*, pages 252–261, Uppsala, Sweden.

Thomas Meyer and Andrei Popescu-Belis. 2012. Using sense-labeled discourse connectives for statistical machine translation. In *Proceedings of the Joint Workshop on Exploiting Synergies between Information Retrieval and Machine Translation (ESIRMT) and Hybrid Approaches to Machine Translation (HyTra)*, pages 129–138, Avignon, France.

Thomas Meyer, Andrei Popescu-Belis, Najeh Hajlaoui, and Andrea Gesmundo. 2012. Machine translation of labeled discourse connectives. In *Proceedings of the 10th Biennial Conference of the Association for Machine Translation in the Americas*, San Diego, California, USA.

Franz Josef Och, Nicola Ueffing, and Hermann Ney. 2001. An efficient A* search algorithm for Statistical Machine Translation. In *Proceedings of the ACL 2001 Workshop on Data-Driven Machine Translation*, pages 55–62, Toulouse, France.

Franz Josef Och. 2003. Minimum error rate training in statistical machine translation. In *Proceedings of the 42nd Annual Meeting of the ACL*, pages 160–167, Sapporo, Japan.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: A method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the ACL*, pages 311–318, Philadelphia, Pennsylvania, USA.

Lucia Specia, Baskaran Sankaran, and Maria das Graças Volpe Nunes. 2008. N-best reranking for the efficient integration of word sense disambiguation and statistical machine translation. In *Proceedings of the 9th International Conference on Intelligent Text Processing and Computational Linguistics (CICLING)*, pages 399–410, Haifa, Israel.

Sara Stymne, Jörg Tiedemann, Christian Hardmeier, and Joakim Nivre. 2013. Statistical machine translation with readability constraints. In *Proceedings of the 19th Nordic Conference on Computational Linguistics (NODALIDA'13)*, pages 375–386, Oslo, Norway.

Jinsong Su, Hua Wu, Haifeng Wang, Yidong Chen, Xiaodong Shi, Huailin Dong, and Qun Liu. 2012. Translation model adaptation for statistical machine translation with monolingual topic information. In *Proceedings of the 50th Annual Meeting of the ACL*, pages 459–468, Jeju Island, Korea.

Jörg Tiedemann. 2010. Context adaptation in statistical machine translation using models with exponentially decaying cache. In *Proceedings of the ACL 2010 Workshop on Domain Adaptation for Natural Language Processing (DANLP)*, pages 8–15, Uppsala, Sweden.

Marco Turchi, Tijl De Bie, Cyril Goutte, and Nello Cristianini. 2012. Learning to translate: A statistical and computational analysis. *Advances in Artificial Intelligence*, 2012. Article ID 484580.

Ferhan Ture, Douglas W. Oard, and Philip Resnik. 2012. Encouraging consistent translation choices. In *Proceedings of the 2012 Conference of the NAACL: Human Language Technologies*, pages 417–426, Montréal, Canada.

Tong Xiao, Jingbo Zhu, Shujie Yao, and Hao Zhang. 2011. Document-level consistency verification in machine translation. In *Proceedings of MT Summit XIII*, pages 131–138, Xiamen, China.

Bing Zhao and Eric P. Xing. 2010. HM-BiTAM: Bilingual topic exploration, word alignment,and translation. In *Advances in Neural Information Processing Systems 20 (NIPS)*, pages 1689–1696, Cambridge, Massachusetts, USA.