Abstract

We investigate the problem of domain adaptation for parallel data in Statistical Machine Translation (SMT). While techniques for domain adaptation of monolingual data can be borrowed for parallel data, we explore conceptual differences between translation model and language model domain adaptation and their effect on performance, such as the fact that translation models typically consist of several features that have different characteristics and can be optimized separately. We also explore adapting multiple (4–10) data sets with no a priori distinction between in-domain and out-of-domain data except for an in-domain development set.

1 Introduction

The increasing availability of parallel corpora from various sources, welcome as it may be, leads to new challenges when building a statistical machine translation system for a specific domain. The task of determining which parallel texts should be included for training, and which ones hurt translation performance, is tedious when performed through trial-and-error. Alternatively, methods for a weighted combination of in-domain and out-of-domain data exist, but there is conflicting evidence as to how well the data is, but how well-adapted it is to the task at hand.

2 Domain Adaptation for Translation Models

To motivate efforts in domain adaptation, let us review why additional training data can improve, but also decrease translation quality.

Adding more training data to a translation system is easy to motivate through the data sparseness problem. Koehn and Knight (2001) show that translation quality correlates strongly with how often a word occurs in the training corpus. Rare words or phrases pose a problem in several stages of MT modelling, from word alignment to the computation of translation probabilities through Maximum Likelihood Estimation. Unknown words are typically copied verbatim to the target text, which may be a good strategy for named entities, but is often wrong otherwise. In general, more data allows for a better word alignment, a better estimation of translation probabilities, and for the consideration of more context (in phrase-based or syntactic SMT).

A second effect of additional data is not necessarily positive. Translations are inherently ambiguous, and a strong source of ambiguity is the...
domain of a text. The German word “Wort” (engl. word) is typically translated as floor in Europarl, a corpus of Parliamentary Proceedings (Koehn, 2005), owing to the high frequency of phrases such as you have the floor, which is translated into German as Sie haben das Wort. This translation is highly idiomatic and unlikely to occur in other contexts. Still, adding Europarl as out-of-domain training data shifts the probability distribution of \( p(t|“Wort”) \) in favour of \( p(“floor”|“Wort”) \), and may thus lead to improper translations.

We will refer to the two problems as the data sparseness problem and the ambiguity problem. Adding out-of-domain data typically mitigates the data sparseness problem, but exacerbates the ambiguity problem. The net gain (or loss) of adding more data changes from case to case. Because there are (to our knowledge) no tools that predict this net effect, it is a matter of empirical investigation (or, in less suave terms, trial-and-error), to determine which corpora to use.\(^2\)

From this understanding of the reasons for and against out-of-domain data, we formulate the following hypotheses:

1. A weighted combination can control the contribution of the out-of-domain corpus on the probability distribution, and thus limit the ambiguity problem.

2. A weighted combination eliminates the need for data selection, offering a robust baseline for domain-specific machine translation.

We will discuss three mixture modelling techniques for translation models. Our aim is to adapt all four features of the standard Moses SMT translation model: the phrase translation probabilities \( p(\overline{f}|\overline{x}) \) and \( p(\overline{w}|\overline{f}) \), and the lexical weights \( lex(\overline{f}|\overline{x}) \) and \( lex(\overline{w}|\overline{f}) \).\(^3\)

2.1 Linear Interpolation

A well-established approach in language modelling is the linear interpolation of several models, i.e. computing the weighted average of the individual model probabilities. It is defined as follows:

\[
p(x|y; \lambda) = \sum_{i=1}^{n} \lambda_i p_i(x|y)
\]

with \( \lambda_i \) being the interpolation weight of each model \( i \), and with \( \sum \lambda_i = 1 \).

For SMT, linear interpolation of translation models has been used in numerous systems. The approaches diverge in how they set the interpolation weights. Some authors use uniform weights (Cohn and Lapata, 2007), others empirically test different interpolation coefficients (Finch and Sumita, 2008; Yasuda et al., 2008; Nakov and Ng, 2009; Axelrod et al., 2011), others apply monolingual metrics to set the weights for TM interpolation (Foster and Kuhn, 2007; Koehn et al., 2010).

There are reasons against all these approaches. Uniform weights are easy to implement, but give little control. Empirically, it has been shown that they often do not perform optimally (Finch and Sumita, 2008; Yasuda et al., 2008). An optimization of BLEU scores on a development set is promising, but slow and impractical. There is no easy way to integrate linear interpolation into log-linear SMT frameworks and perform optimization through MERT. Monolingual optimization objectives such as language model perplexity have the advantage of being well-known and readily available, but their relation to the ambiguity problem is indirect at best.

Linear interpolation is seemingly well-defined in equation 1. Still, there are a few implementation details worth pointing out. If we directly interpolate each feature in the translation model, and define the feature values of non-occurring phrase pairs as 0, this disregards the meaning of each feature. If we estimate \( p(x|y) \) via MLE as in equation 2, and \( c(y) = 0 \), then \( p(x|y) \) is strictly speaking undefined. Alternatively to a naive algorithm, which treats unknown phrase pairs as having a probability of 0, which results in a deficient probability distribution, we propose and implement the following algorithm. For each value pair \( (x, y) \) for which we compute \( p(x|y) \), we replace \( \lambda_i \) with 0 for all models \( i \) with \( p(y) = 0 \), then renormalize the weight vector \( \lambda \) to 1. We do this for \( p(\overline{f}|\overline{x}) \) and \( lex(\overline{f}|\overline{x}) \), but not for \( p(\overline{w}|\overline{f}) \) and \( lex(\overline{w}|\overline{f}) \), the reasoning being the con-

\(^2\)A frustrating side-effect is that these findings rarely generalize. For instance, we were unable to reproduce the finding by Ceauşu et al. (2011) that patent translation systems are highly domain-sensitive and suffer from the inclusion of parallel training data from other patent subdomains.

\(^3\)We can ignore the fifth feature, the phrase penalty, which is a constant.
sequences for perplexity minimization (see section 2.4). Namely, we do not want to penalize a small in-domain model for having a high out-of-vocabulary rate on the source side, but we do want to penalize models that know the source phrase, but not its correct translation. A second modification pertains to the lexical weights \( lex(\pi|T) \) and \( lex(T|\pi) \), which form no true probability distribution, but are derived from the individual word translation probabilities of a phrase pair (see (Koehn et al., 2003)). We propose to not interpolate the features directly, but the word translation probabilities which are the basis of the lexical weight computation. The reason for this is that word pairs are less sparse than phrase pairs, so that we can even compute lexical weights for phrase pairs which are unknown in a model.\(^4\)

### 2.2 Weighted Counts

Weighting of different corpora can also be implemented through a modified Maximum Likelihood Estimation. The traditional equation for MLE is:

\[
p(x|y) = \frac{c(x, y)}{c(y)} = \frac{\sum_{x'} c(x', y)}{\sum_{x'} c(x', y)} \tag{2}
\]

where \( c \) denotes the count of an observation, and \( p \) the model probability. If we generalize the formula to compute a probability from \( n \) corpora, and assign a weight \( \lambda_i \) to each, we get:\(^5\)

\[
p(x|y; \lambda) = \frac{\sum_{i=1}^{n} \lambda_i c_i(x, y)}{\sum_{i=1}^{n} \sum_{x'} \lambda_i c_i(x', y)} \tag{3}
\]

The main difference to linear interpolation is that this equation takes into account how well-evidenced a phrase pair is. This includes the distinction between lack of evidence and negative evidence, which is missing in a naive implementation of linear interpolation.

Translation models trained with weighted counts have been discussed before, and have been shown to outperform uniform ones in some settings. However, researchers who demonstrated this fact did so with arbitrary weights (e.g. (Koehn, 2002), or by empirically testing different weights (e.g. (Nakov and Ng, 2009)). We do not know of any research on automatically determining weights for this method, or which is not limited to two corpora.

---

\(^4\)For instance if the word pairs (the,der) and (man,Mann) are known, but the phrase pair (the man, der Mann) is not.

\(^5\)Unlike equation 1, equation 3 does not require that \( \sum \lambda_i = 1 \).

### 2.3 Alternative Paths

A third method is using multiple translation models as alternative decoding paths (Birch et al., 2007), an idea which Koehn and Schroeder (2007) first used for domain adaptation. This approach has the attractive theoretical property that adding new models is guaranteed to lead to equal or better performance, given the right weights. At best, a model is beneficial with appropriate weights. At worst, we can set the feature weights so that the decoding paths of one model are never picked for the final translation. In practice, each translation model adds 5 features and thus 5 more dimensions to the weight space, which leads to longer search, search errors, and/or overfitting. The expectation is that, at least with MERT, using alternative decoding paths does not scale well to a high number of models.

A suboptimal choice of weights is not the only weakness of alternative paths, however. Let us assume that all models have the same weights. Note that, if a phrase pair occurs in several models, combining models through alternative paths means that the decoder selects the path with the highest probability, whereas with linear interpolation, the probability of the phrase pair would be the (weighted) average of all models. Selecting the highest-scoring phrase pair favours statistical outliers and hence is the less robust decision, prone to data noise and data sparseness.

### 2.4 Perplexity Minimization

In language modelling, perplexity is frequently used as a quality measure for language models (Chen and Goodman, 1998). Among other applications, language model perplexity has been used for domain adaptation (Foster and Kuhn, 2007). For translation models, perplexity is most closely associated with EM word alignment (Brown et al., 1993) and has been used to evaluate different alignment algorithms (Al-Onaizan et al., 1999).

We investigate translation model perplexity minimization as a method to set model weights in mixture modelling. For the purpose of optimization, the cross-entropy \( H(p) \), the perplexity \( 2^{H(p)} \), and other derived measures are equivalent. The cross-entropy \( H(p) \) is defined as:\(^6\)

---

\(^6\)See (Chen and Goodman, 1998) for a short discussion of the equation. In short, a lower cross-entropy indicates that the model is better able to predict the development set.
\[ H(p) = - \sum_{x,y} \tilde{p}(x,y) \log_2 p(x|y) \quad (4) \]

The phrase pairs \((x, y)\) whose probability we measure, and their empirical probability \(\tilde{p}\) need to be extracted from a development set, whereas \(p\) is the model probability. To obtain the phrase pairs, we process the development set with the same word alignment and phrase extraction tools that we use for training, i.e. GIZA++ and heuristics for phrase extraction (Och and Ney, 2003). The objective function is the minimization of the cross-entropy, with the weight vector \(\lambda\) as argument:

\[ \hat{\lambda} = \arg \min_{\lambda} - \sum_{x,y} \tilde{p}(x,y) \log_2 p(x|y; \lambda) \quad (5) \]

We can fill in equations 1 or 3 for \(p(x|y; \lambda)\). The optimization itself is convex and can be done with off-the-shelf software.\(^7\) We use L-BFGS with numerically approximated gradients (Byrd et al., 1995).

Perplexity minimization has the advantage that it is well-defined for both weighted counts and linear interpolation, and can be quickly computed. Other than in language modelling, where \(p(x|y)\) is the probability of a word given a \(n\)-gram history, conditional probabilities in translation models express the probability of a target phrase given a source phrase (or vice versa), which connects the perplexity to the ambiguity problem. The higher the probability of “correct” phrase pairs, the lower the perplexity, and the more likely the model is to successfully resolve the ambiguity. The question is in how far perplexity minimization coincides with empirically good mixture weights.\(^8\) This depends, among others, on the other model components in the SMT framework, for instance the language model. We will not evaluate perplexity minimization against empirically optimized mixture weights, but apply it in situations where the latter is infeasible, e.g. because of the number of models.

Our main technical contributions are as follows: Additionally to perplexity optimization for linear interpolation, which was first applied by Foster et al. (2010), we propose perplexity optimization for weighted counts (equation 3), and a modified implementation of linear interpolation. Also, we independently perform perplexity minimization for all four features of the standard SMT translation model: the phrase translation probabilities \(p(\tilde{t}|\tilde{s})\) and \(p(\tilde{s}|\tilde{t})\), and the lexical weights \(lex(\tilde{t}|\tilde{s})\) and \(lex(\tilde{s}|\tilde{t})\).

### 3 Other Domain Adaptation Techniques

So far, we discussed mixture modelling for translation models, which is only a subset of domain adaptation techniques in SMT.

Mixture-modelling for language models is well established (Foster and Kuhn, 2007). Language model adaptation serves the same purpose as translation model adaptation, i.e. skewing the probability distribution in favour of in-domain translations. This means that LM adaptation may have similar effects as TM adaptation, and that the two are to some extent redundant. Foster and Kuhn (2007) find that “both TM and LM adaptation are effective”, but that “combined LM and TM adaptation is not better than LM adaptation on its own”.

A second strand of research in domain adaptation is data selection, i.e. choosing a subset of the training data that is considered more relevant for the task at hand. This has been done for language models using techniques from information retrieval (Zhao et al., 2004), or perplexity (Lin et al., 1997; Moore and Lewis, 2010). Data selection has also been proposed for translation models (Axelrod et al., 2011). Note that for translation models, data selection offers an unattractive trade-off between the data sparseness and the ambiguity problem, and that the optimal amount of data to select is hard to determine.

Our discussion of mixture-modelling is relatively coarse-grained, with 2-10 models being combined. Matsoukas et al. (2009) propose an approach where each sentence is weighted according to a classifier, and Foster et al. (2010) extend this approach by weighting individual phrase pairs. These more fine-grained methods need not be seen as alternatives to coarse-grained ones. Foster et al. (2010) combine the two, applying linear interpolation to combine the instance-
weighted out-of-domain model with an in-domain model.

4 Evaluation

Apart from measuring the performance of the approaches introduced in section 2, we want to investigate the following open research questions.

1. Does an implementation of linear interpolation that is more closely tailored to translation modelling outperform a naive implementation?

2. How do the approaches perform outside a binary setting, i.e. when we do not work with one in-domain and one out-of-domain model, but with a higher number of models?

3. Can we apply perplexity minimization to other translation model features such as the lexical weights, and if yes, does a separate optimization of each translation model feature improve performance?

4.1 Data and Methods

In terms of tools and techniques used, we mostly adhere to the work flow described for the WMT 2011 baseline system\(^9\). The main tools are Moses (Koehn et al., 2007), SRILM (Stolcke, 2002), and GIZA++ (Och and Ney, 2003), with settings as described in the WMT 2011 guide. We report two translation measures: BLEU (Papineni et al., 2002) and METEOR 1.3 (Denkowski and Lavie, 2011). All results are lowercased and tokenized, measured with five independent runs of MERT (Och and Ney, 2003) and MultiEval (Clark et al., 2011) for resampling and significance testing.

We compare three baselines and four translation model mixture techniques. The three baselines are a purely in-domain model, a purely out-of-domain model, and a model trained on the concatenation of the two, which corresponds to equation 3 with uniform weights. Additionally, we evaluate perplexity optimization with weighted counts and the two implementations of linear interpolation contrasted in section 2.1. The two linear interpolations that are contrasted are a naive one, i.e. a direct, unnormalized interpolation of all translation model features, and a modified one that normalizes \(\lambda\) for each phrase pair \((s, t)\) for \(p(t|s)\) and recomputes the lexical weights based on interpolated word translation probabilities. The fourth weighted combination is using alternative decoding paths with weights set through MERT. The four weighted combinations are evaluated twice: once applied to the original four or ten parallel data sets, once in a binary setting in which all out-of-domain data sets are first concatenated.

Since we want to concentrate on translation model domain adaptation, we keep other model components, namely word alignment and the lexical reordering model, constant throughout the experiments. We contrast two language models. An unadapted, out-of-domain language model trained on data sets provided for the WMT 2011 translation task, and an adapted language model which is the linear interpolation of all data sets, optimized for minimal perplexity on the in-domain development set.

While unadapted language models are becoming more rare in domain adaptation research, they allow us to contrast different TM mixtures without the effect on performance being (partially) hidden by language model adaptation with the same effect.

The first data set is a DE–FR translation scenario in the domain of mountaineering. The in-domain corpus is a collection of Alpine Club pub-

---

9http://www.statmt.org/wmt11/baseline.html
lications (Volk et al., 2010). As parallel out-of-domain dataset, we use Europarl, a collection of parliamentary proceedings (Koehn, 2005), JRC-Acquis, a collection of legislative texts (Steinberger et al., 2006), and OpenSubtitles v2, a parallel corpus extracted from film subtitles10 (Tiedemann, 2009). For language modelling, we use in-domain data and data from the 2011 Workshop on Statistical Machine Translation. The respective sizes of the data sets are listed in tables 1 and 2.

As the second data set, we use the Haitian Creole – English data from the WMT 2011 featured translation task. It consists of emergency SMS sent in the wake of the 2010 Haiti earthquake. Originally, Microsoft Research and CMU operated under severe time constraints to build a translation system for this language pair. This limits the ability to empirically verify how much each data set contributes to translation quality, and increases the importance of automated and quick domain adaptation methods.

Note that both data sets have a relatively high ratio of in-domain to out-of-domain parallel training data (1:20 for DE–EN and 1:5 for HT–EN) Previous research has been performed with ratios of 1:100 (Foster et al., 2010) or 1:400 (Axelrod et al., 2011). Since domain adaptation becomes more important when the ratio of IN to OUT is low, and since such low ratios are also realistic11, we also include results for which the amount of in-domain parallel data has been restricted to 10% of the available data set.

We used the same development set for language/translation model adaptation and setting the global model weights with MERT. While it is theoretically possible that MERT will give too high weights to models that are optimized on the same development set, we found no empirical evidence for this in experiments with separate development sets.

4.2 Results

The results are shown in tables 5 and 6. In the DE–FR translation task, results vary between 13.5 and 18.9 BLEU points; in the HT–EN task, between 24.3 and 33.8. Unsurprisingly, an adapted

<table>
<thead>
<tr>
<th>Data set</th>
<th>units</th>
<th>words (en)</th>
</tr>
</thead>
<tbody>
<tr>
<td>SMS (in-domain)</td>
<td>16 500</td>
<td>380 000</td>
</tr>
<tr>
<td>Medical</td>
<td>1 600</td>
<td>10 000</td>
</tr>
<tr>
<td>Newswire</td>
<td>13 500</td>
<td>330 000</td>
</tr>
<tr>
<td>Glossary</td>
<td>35 700</td>
<td>90 000</td>
</tr>
<tr>
<td>Wikipedia</td>
<td>8 500</td>
<td>110 000</td>
</tr>
<tr>
<td>Wikipedia NE</td>
<td>10 500</td>
<td>34 000</td>
</tr>
<tr>
<td>Bible</td>
<td>30 000</td>
<td>920 000</td>
</tr>
<tr>
<td>Haitisurf dict</td>
<td>3 700</td>
<td>4000</td>
</tr>
<tr>
<td>Krengle dict</td>
<td>1 600</td>
<td>2 600</td>
</tr>
<tr>
<td>Krengle</td>
<td>650</td>
<td>4 200</td>
</tr>
<tr>
<td>Total train</td>
<td>120 000</td>
<td>1 900 000</td>
</tr>
<tr>
<td>Dev</td>
<td>900</td>
<td>22 000</td>
</tr>
<tr>
<td>Test</td>
<td>1274</td>
<td>25 000</td>
</tr>
</tbody>
</table>

Table 3: Parallel data sets for Haiti Creole – English translation task.

<table>
<thead>
<tr>
<th>Data set</th>
<th>sentences</th>
<th>words</th>
</tr>
</thead>
<tbody>
<tr>
<td>SMS (in-domain)</td>
<td>16k</td>
<td>380k</td>
</tr>
<tr>
<td>News</td>
<td>113 000k</td>
<td>2 650 000k</td>
</tr>
</tbody>
</table>

Table 4: Monolingual English data sets for Haiti Creole – English translation task.

LM performs better than an out-of-domain one, and using all available in-domain parallel data is better than using only part of it. The same is not true for out-of-domain data, which highlights the problem discussed in the introduction. For the DE–FR task, adding 86 million words of out-of-domain parallel data to the 5 million in-domain data set does not lead to consistent performance gains. We observe a decrease of 0.3 BLEU points with an out-of-domain LM, and an increase of 0.4 BLEU points with an adapted LM. The out-of-domain training data has a larger positive effect if less in-domain data is available, with a gain of 1.4 BLEU points. The results in the HT–EN translation task (table 6) paint a similar picture. An interesting side note is that even tiny amounts of in-domain parallel data can have strong effects on performance. A training set of 1600 emergency SMS (38 000 tokens) yields a comparable performance to an out-of-domain data set of 1.5 million tokens.

As to the domain adaptation experiments, weights optimized through perplexity minimization are significantly better in the majority of cases, and never significantly worse, than uniform
Table 5: Domain adaptation results DE–FR. Domain: Alpine texts. Full IN TM: Using the full in-domain parallel corpus; small IN TM: using 10% of available in-domain parallel data.

<table>
<thead>
<tr>
<th>System</th>
<th>out-of-domain LM</th>
<th>adapted LM</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>full IN TM</td>
<td>full IN TM</td>
</tr>
<tr>
<td></td>
<td>BLEU</td>
<td>METEOR</td>
</tr>
<tr>
<td>in-domain</td>
<td>16.8</td>
<td>35.9</td>
</tr>
<tr>
<td>out-of-domain</td>
<td>13.5</td>
<td>31.3</td>
</tr>
<tr>
<td>counts (concatenation)</td>
<td>16.5</td>
<td>35.7</td>
</tr>
<tr>
<td>binary in/out</td>
<td></td>
<td></td>
</tr>
<tr>
<td>weighted counts</td>
<td>17.4</td>
<td>36.6</td>
</tr>
<tr>
<td>linear interpolation</td>
<td>17.4</td>
<td>36.7</td>
</tr>
<tr>
<td>linear interpolation</td>
<td>17.2</td>
<td>36.5</td>
</tr>
<tr>
<td>alternative paths</td>
<td>17.2</td>
<td>36.5</td>
</tr>
<tr>
<td>4 models</td>
<td></td>
<td></td>
</tr>
<tr>
<td>weighted counts</td>
<td>17.3</td>
<td>36.6</td>
</tr>
<tr>
<td>linear interpolation</td>
<td>17.1</td>
<td>36.5</td>
</tr>
<tr>
<td>linear interpolation</td>
<td>17.2</td>
<td>36.5</td>
</tr>
<tr>
<td>alternative paths</td>
<td>17.0</td>
<td>36.2</td>
</tr>
</tbody>
</table>

weights. However, the difference is smaller for the experiments with an adapted language model than for those with an out-of-domain one, which confirms that the benefit of language model adaptation and translation model adaptation are not fully cumulative. Performance-wise, there seems to be no clear winner between weighted counts and the two alternative implementations of linear interpolation. We can still argue for weighted counts on theoretical grounds. A weighted MLE (equation 3) returns a true probability distribution, whereas a naive implementation of linear interpolation results in a deficient model. Consequently, probabilities are typically lower in the naively interpolated model, which results in higher (worse) perplexities. While the deficiency did not affect MERT or decoding negatively, it might become problematic in other applications, for instance if we want to use an interpolated model as a component in a second perplexity-based combination of models.

When moving from a binary setting with one in-domain and one out-of-domain translation model (trained on all available out-of-domain data) to 4–10 translation models, we observe a serious performance degradation for alternative paths, while performance of the perplexity optimization methods does not change significantly. This is positive for perplexity optimization because it demonstrates that it requires less a priori information, and opens up new research possibilities, i.e. experiments with different clusterings of parallel data. The performance degradation for alternative paths is partially due to optimization problems in MERT, but also due to a higher susceptibility to statistical outliers, as discussed in section 2.3.

A pessimistic interpretation of the results would point out that performance gains compared to the best baseline system are modest or even inexistent in some settings. However, we want to stress two important points. First, we often do not know a priori whether adding an out-of-domain data set boosts or weakens translation performance. An automatic weighting of data sets reduces the need for trial-and-error experimentation and is worthwhile even if a performance increase is not guaranteed. Second, the potential impact of a weighted combination depends on the translation scenario and the available data sets. Generally, we expect non-uniform weighting to have a bigger impact when the models that are combined are more dissimilar (in terms of fitness for the task), and if the ratio of in-domain to out-of-domain data is low. Conversely, there are situa-
Table 6: Domain adaptation results HT–EN. Domain: emergency SMS. Full IN TM: Using the full in-domain parallel corpus; small IN TM: using 10% of available in-domain parallel data.

<table>
<thead>
<tr>
<th>System</th>
<th>out-of-domain LM</th>
<th>adapted LM</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>full IN TM</td>
<td></td>
</tr>
<tr>
<td>BLEU</td>
<td>METEOR</td>
<td>full IN TM</td>
</tr>
<tr>
<td>in-domain</td>
<td>30.4</td>
<td>30.7</td>
</tr>
<tr>
<td>out-of-domain</td>
<td>24.3</td>
<td>28.0</td>
</tr>
<tr>
<td>counts (concatenation)</td>
<td>30.3</td>
<td>31.2</td>
</tr>
</tbody>
</table>

binary in/out

| weighted counts              | 31.0 | 31.6         | 33.8 | 32.4 | 31.5 | 31.3 |
| linear interpolation (naive) | 30.8 | 31.4         | 33.7 | 32.4 | 31.9 | 31.3 |
| linear interpolation (modified) | 30.8 | 31.5         | 33.7 | 32.4 | 31.7 | 31.2 |
| alternative paths            | 30.8 | 31.3         | 33.2 | 32.4 | 29.8 | 30.7 |

10 models

| weighted counts              | 31.0 | 31.5         | 33.5 | 32.3 | 31.8 | 31.5 |
| linear interpolation (naive) | 30.9 | 31.4         | 33.8 | 32.4 | 31.9 | 31.3 |
| linear interpolation (modified) | 31.0 | 31.6         | 33.8 | 32.5 | 32.1 | 31.5 |
| alternative paths            | 25.9 | 29.2         | 24.3 | 29.1 | 29.8 | 30.9 |

Table 7: Contrast between a separate optimization of each feature and applying the weight vector optimized on one feature to the whole model. HT–EN with out-of-domain LM.
average of the other four. For linear interpolation, we also include a model whose weights have been optimized through language model perplexity optimization, with a 3-gram language model (modified Knesey-Ney smoothing) trained on the target side of each parallel data set.

Table 7 shows the results. In terms of BLEU score, a separate optimization of each feature is a winner in our experiment in that no other scheme is better, with 8 of the 11 alternative weighting schemes (excluding uniform weights) being significantly worse than a separate optimization. The differences in BLEU score are small, however, since the alternative weighting schemes are generally felicitious in that they yield both a lower perplexity and better BLEU scores than uniform weighting. While our general expectation is that lower perplexities correlate with higher translation performance, this relation is complicated by several facts. Since the interpolated models are deficient (i.e. their probabilities do not sum to 1), perplexities for weighted counts and our implementation of linear interpolation cannot be compared. Also, note that not all features are equally important for decoding. Their weights in the log-linear model are set through MERT and vary between optimization runs.

5 Conclusion

This paper contributes to SMT domain adaptation research in several ways. We expand on work by (Foster et al., 2010) in establishing translation model perplexity minimization as a robust baseline for a weighted combination of translation models.\footnote{The source code is available in the Moses repository \url{http://github.com/moses-smt/mosesdecoder}} We demonstrate perplexity optimization for weighted counts, which are a natural extension of unadapted MLE training, but are of little prominence in domain adaptation research. We also show that we can separately optimize the four variable features in the Moses translation model through perplexity optimization.

We break with prior domain adaptation research in that we do not rely on a binary clustering of in-domain and out-of-domain training data. We demonstrate that perplexity minimization scales well to a higher number of translation models. This is not only useful for domain adaptation, but for various tasks that profit from mixture modeling. We envision that a weighted combination could be useful to deal with noisy datasets, or applied after a clustering of training data.

Acknowledgements

This research was funded by the Swiss National Science Foundation under grant 105215_126999.

References


