A Multi-Domain Translation Model Framework for Statistical Machine Translation

Rico Sennrich
Institute of Computational Linguistics
University of Zurich
Binzmühlestr. 14
CH-8050 Zürich
sennrich@cl.uzh.ch

Holger Schwenk and **Walid Aransa**
LIUM, University of Le Mans
72085 Le Mans cedex 9, France
lastname@lium.univ-lemans.fr

Abstract

While domain adaptation techniques for SMT have proven to be effective at improving translation quality, their practicality for a multi-domain environment is often limited because of the computational and human costs of developing and maintaining multiple systems adapted to different domains. We present an architecture that delays the computation of translation model features until decoding, allowing for the application of mixture-modeling techniques at decoding time. We also describe a method for unsupervised adaptation with development and test data from multiple domains. Experimental results on two language pairs demonstrate the effectiveness of both our translation model architecture and automatic clustering, with gains of up to 1 BLEU over unadapted systems and single-domain adaptation.

1 Introduction

The effectiveness of domain adaptation approaches such as mixture-modeling (Foster and Kuhn, 2007) has been established, and has led to research on a wide array of adaptation techniques in SMT, for instance (Matsoukas et al., 2009; Shah et al., 2012). In all these approaches, adaptation is performed during model training, with respect to a representative development corpus, and the models are kept unchanged when then system is deployed. Therefore, when working with multiple and/or unlabelled domains, domain adaptation is often impractical for a number of reasons. Firstly, maintaining multiple systems for each language pair, each adapted to a different domain, is costly in terms of computational and human resources: the full system development pipeline needs to be performed for all identified domains, all the models are separately stored and need to be switched at runtime. This is impractical in many real applications, in particular a web translation service which is faced with texts coming from many different domains. Secondly, domain adaptation bears a risk of performance loss. If there is a mismatch between the domain of the development set and the test set, domain adaptation can potentially harm performance compared to an unadapted baseline.

We introduce a translation model architecture that delays the computation of features to the decoding phase. The calculation is based on a vector of component models, with each component providing the sufficient statistics necessary for the computation of the features. With this framework, adaptation to a new domain simply consists of updating a weight vector, and multiple domains can be supported by the same system.

We also present a clustering approach for unsupervised adaptation in a multi-domain environment. In the development phase, a set of development data is clustered, and the models are adapted to each cluster. For each sentence that is being decoded, we choose the weight vector that is optimized on the closest cluster, allowing for adaptation even with unlabelled and heterogeneous test data.

2 Related Work

(Ortiz-Martínez et al., 2010) delay the computation of translation model features for the purpose of interactive machine translation with online training. The main difference to our approach is that we store sufficient statistics not for a single model, but a vector of models, which allows us to

Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics, pages 832–840,
Sofia, Bulgaria, August 4-9 2013. ©2013 Association for Computational Linguistics

weight the contribution of each component model to the feature calculation. The similarity suggests that our framework could also be used for interactive learning, with the ability to learn a model incrementally from user feedback, and weight it differently than the static models, opening new research opportunities.

(Sennrich, 2012b) perform instance weighting of translation models, based on the sufficient statistics. Our framework implements this idea, with the main difference that the actual combination is delayed until decoding, to support adaptation to multiple domains in a single system.

(Razmara et al., 2012) describe an ensemble decoding framework which combines several translation models in the decoding step. Our work is similar to theirs in that the combination is done at runtime, but we also delay the computation of translation model probabilities, and thus have access to richer sufficient statistics. In principle, our architecture can support all mixture operations that (Razmara et al., 2012) describe, plus additional ones such as forms of instance weighting, which are not possible after the translation probabilities have been computed.

(Banerjee et al., 2010) focus on the problem of domain identification in a multi-domain setting. They use separate translation systems for each domain, and a supervised setting, whereas we aim for a system that integrates support for multiple domains, with or without supervision.

(Yamamoto and Sumita, 2007) propose unsupervised clustering at both training and decoding time. The training text is divided into a number of clusters, a model is trained on each, and during decoding, each sentence is assigned to the closest cluster-specific model. Our approach bears resemblance to this clustering, but is different in that Yamamoto and Sumita assign each sentence to the closest model, and use this model for decoding, whereas in our approach, each cluster is associated with a mixture of models that is optimized to the cluster, and the number of clusters need not be equal to the number of component models.

3 Translation Model Architecture

This section covers the architecture of the multi-domain translation model framework. Our translation model is embedded in a log-linear model as is common for SMT, and treated as a single translation model in this log-linear combination. We implemented this architecture for phrase-based models, and will use this terminology to describe it, but in principle, it can be extended to hierarchical or syntactic models.

The architecture has two goals: move the calculation of translation model features to the decoding phase, and allow for multiple knowledge sources (e.g. bitexts or user-provided data) to contribute to their calculation. Our immediate purpose for this paper is domain adaptation in a multi-domain environment, but the delay of the feature computation has other potential applications, e.g. in interactive MT.

We are concerned with calculating four features during decoding, henceforth just referred to as the translation model features: $p(\bar{s}|\bar{t})$, $lex(\bar{s}|\bar{t})$, $p(\bar{t}|\bar{s})$ and $lex(\bar{t}|\bar{s})$. $\bar{s}$ and $\bar{t}$ denote the source and target phrase. We follow the definitions in (Koehn et al., 2003).

Traditionally, the phrase translation probabilities $p(\bar{s}|\bar{t})$ and $p(\bar{t}|\bar{s})$ are estimated through unsmoothed maximum likelihood estimation (MLE).

$$p(x|y) = \frac{c(x,y)}{c(y)} = \frac{c(x,y)}{\sum_{x'} c(x',y)} \qquad (1)$$

where c denotes the count of an observation, and p the model probability.

The lexical weights $lex(\bar{s}|\bar{t})$ and $lex(\bar{t}|\bar{s})$ are calculated as follows, using a set of word alignments a between $\bar{s}$ and $\bar{t}$:[1]

$$lex(\bar{s}|\bar{t},a) = \prod_{i=1}^{n} \frac{1}{|\{j|(i,j) \in a\}|} \sum_{\forall(i,j)\in a} w(s_i|t_j) \qquad (2)$$

A special NULL token is added to $\bar{t}$ and aligned to each unaligned word in $\bar{s}$. $w(s_i|t_j)$ is calculated through MLE, as in equation 1, but based on the word (pair) frequencies.

To combine statistics from a vector of n component corpora, we can use a weighted version of equation 1, which adds a weight vector λ of length n (Sennrich, 2012b):

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

The word translation probabilities $w(t_i|s_j)$ are defined analogously, and used in equation 2 for a weighted version.

[1] The equation shows $lex(\bar{s}|\bar{t})$; $lex(\bar{t}|\bar{s})$ is computed analogously.

In order to compute the translation model features online, a number of sufficient statistics need to be accessible at decoding time. For $p(\overline{s}|\overline{t})$ and $p(\overline{t}|\overline{s})$, we require the statistics $c(\overline{s})$, $c(\overline{t})$ and $c(\overline{s},\overline{t})$. For accessing them during decoding, we simply store them in the decoder's data structure, rather than storing pre-computed translation model features. This means that we can use existing, compact data formats for storing and accessing them.[2]

The statistics are accessed when the decoder collects all translation options for a phrase $\overline{s}$ in the source sentence. We then access all translation options for each component table, obtaining a vector of statistics $c(\overline{s})$ for the source phrase, and $c(\overline{t})$ and $c(\overline{s},\overline{t})$ for each potential target phrase. For phrase pairs which are not found, $c(\overline{s},\overline{t})$ and $c(\overline{t})$ are initially set to 0.

Note that $c(\overline{t})$ is potentially incorrect at this point, since a phrase pair not being found does not entail that $c(\overline{t})$ is 0. After all tables have been accessed, and we thus know the full set of possible translation options $(\overline{s},\overline{t})$, we perform a second round of lookups for all $c(\overline{t})$ in the vector which are still set to 0. We introduce a second table for accessing $c(\overline{t})$ efficiently, again storing it in the decoder's data structure. We can easily create such a table by inverting the source and target phrases, deduplicating it for compactness (we only need one entry per target phrase), and storing $c(\overline{t})$ as only feature.

For $lex(\overline{s}|\overline{t})$, we require an alignment a, plus $c(t_j)$ and $c(s_i,t_j)$ for all pairs (i,j) in a. $lex(\overline{t}|\overline{s})$ can be based on the same alignment a (with the exception of NULL alignments, which can be added online), but uses statistics $c(s_j)$ and $c(t_i,s_j)$. For estimating the lexical probabilities, we load the frequencies into a vector of four hash tables.[3]

Both space and time complexity of the lookup is linear to the number of component tables. We deem it is still practical because the collection of translation options is typically only a small fraction of total decoding time, with search making up the largest part. For storing and accessing the sufficient statistics (except for the word (pair) frequencies), we use an on-disk data structure provided by Moses, which reduces the memory requirements. Still, the number of components may need to be reduced, for instance through clustering of training data (Sennrich, 2012a).

phrase (pair)	$c_1(x)$	$c_2(x)$
row	300	80
(row, Zeile)	240	20
(row, Reihe)	60	60

λ	p(Zeile|row)	p(Reihe|row)
(1, 1)	0.68	0.32
(1, 10)	0.40	0.60
(10, 1)	0.79	0.21

Table 1: Illustration of instance weighting with weight vectors for two corpora.

With a small modification, our framework could be changed to use a single table that stores a vector of n statistics instead of a vector of n tables. While this would be more compact in terms of memory, and keep the number of table lookups independent of the number of components, we chose a vector of n tables for its flexibility. With a vector of tables, tables can be quickly added to or removed from the system (conceivable even at runtime), and can be polymorph. One applications where this could be desirable is interactive machine translation, where one could work with a mix of compact, static tables, and tables designed to be incrementally trainable.

In the unweighted variant, the resulting features are equivalent to training on the concatenation of all training data, excepting differences in word alignment, pruning[4] and rounding. The architecture can thus be used as a drop-in replacement for a baseline system that is trained on concatenated training data, with non-uniform weights only being used for texts for which better weights have been established. This can be done either using domain labels or unsupervised methods as described in the next section.

As a weighted combination method, we implemented instance weighting as described in equation 3. Table 1 shows the effect of weighting two corpora on the probability estimates for the translation of *row*. German *Zeile* (row in a table) is predominant in a bitext from the domain IT, whereas

[2] We have released an implementation of the architecture as part of the Moses decoder.

[3] $c(s,t)$ and $c(t,s)$ are not identical since the lexical probabilities are based on the unsymmetrized word alignment frequencies (in the Moses implementation which we reimplement).

[4] We prune the tables to the most frequent 50 phrase pairs per source phrase before combining them, since calculating the features for all phrase pairs of very common source phrases causes a significant slow-down. We found that this had no significant effects on BLEU.

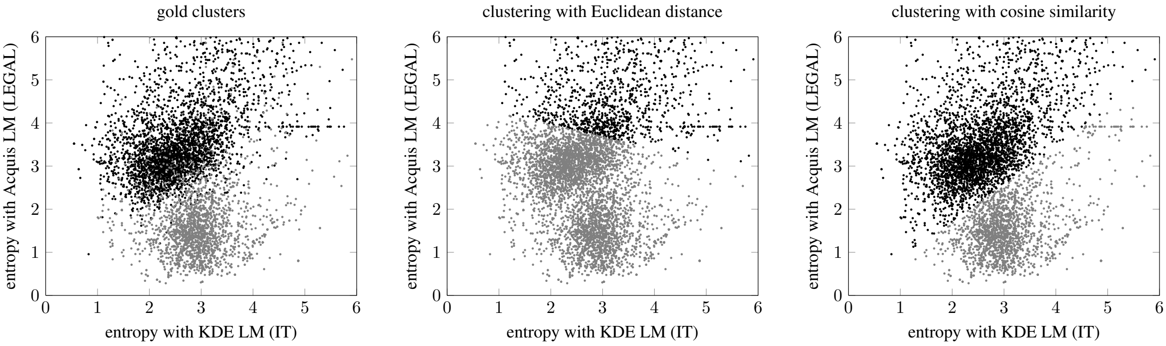

Figure 1: Clustering of data set which contains sentences from two domains: LEGAL and IT. Comparison between gold segmentation, and clustering with two alternative distance/similarity measures. Black: IT; grey: LEGAL.

Reihe (line of objects) occurs more often in a legal corpus. Note that the larger corpus (or more precisely, the one in which *row* occurs more often) has a stronger impact on the probability distribution with uniform weights (or in a concatenation of data sets). Instance weighting allows us to modify the contribution of each corpus. In our implementation, the weight vector is set globally, but can be overridden on a per-sentence basis. In principle, using different weight vectors for different phrase pairs in a sentence is conceivable. The framework can also be extended to support other combination methods, such as a linear interpolation of models.

4 Unsupervised Clustering for Online Translation Model Adaptation

The framework supports decoding each sentence with a separate weight vector of size $4n$, 4 being the number of translation model features whose computation can be weighted, and n the number of model components. We now address the question of how to automatically select good weights in a multi-domain task. As a way of optimizing instance weights, (Sennrich, 2012b) minimize translation model perplexity on a set of phrase pairs, automatically extracted from a parallel development set. We follow this technique, but want to have multiple weight vectors, adapted to different texts, between which the system switches at decoding time. The goal is to perform domain adaptation without requiring domain labels or user input, neither for development nor decoding.

The basic idea consists of three steps:

1. Cluster a development set into k clusters.
2. Optimize translation model weights for each cluster.
3. For each sentence in the test set, assign it to the nearest cluster and use the translation model weights associated with the cluster.

For step 2, we use the algorithm by (Sennrich, 2012b), implemented in the decoder to allow for a quick optimization of a running system. We will here discuss steps 1 and 3 in more detail.

4.1 Clustering the Development Set

We use k-means clustering to cluster the sentences of the development set. We train a language model on the source language side of each of the n component bitexts, and compute an n-dimensional vector for each sentence by computing its entropy with each language model. Our aim is not to discriminate between sentences that are more likely and unlikely in general, but to cluster on the basis of relative differences between the language model entropies. For this purpose, we choose the cosine as our similarity measure. Figure 1 illustrates clustering in a two-dimensional vector space, and demonstrates that Euclidean distance is unsuitable because it may perform a clustering that is irrelevant to our purposes.

As a result of development set clustering, we obtain a bitext for each cluster, which we use to optimize the model weights, and a centroid per cluster. At decoding time, we need only perform an assignment step. Each test set sentence is assigned to the centroid that is closest to it in the vector space.

4.2 Scalability Considerations

Our theoretical expectation is that domain adaptation will fail to perform well if the test data is from a different domain than the development data, or if the development data is a heterogeneous mix of domains. A multi-domain setup can mitigate this risk, but only if the relevant domain is represented in the development data, and if the development data is adequately segmented for the optimization. We thus suggest that the development data should contain enough data from all domains that one wants to adapt to, and a high number of clusters.

While the resource requirements increase with the number of component models, increasing the number of clusters is computationally cheap at runtime. Only the clustering of the development set and optimization of the translation model weights for each clusters is affected by k. This means that the approach can in principle be scaled to a high number of clusters, and support a high number of domains.[5]

The biggest risk of increasing the number of clusters is that if the clusters become too small, perplexity minimization may overfit these small clusters. We will experiment with different numbers of clusters, but since we expect the optimal number of clusters to depend on the amount of development data, and the number of domains, we cannot make generalized statements about the ideal number of k.

While it is not the focus of this paper, we also evaluate language model adaptation. We perform a linear interpolation of models for each cluster, with interpolation coefficients optimized using perplexity minimization on the development set. The cost of moving language model interpolation into the decoding phase is far greater than for translation models, since the number of hypotheses that need to be evaluated by the language model is several orders of magnitudes higher than the number of phrase pairs used during the translation. For the experiments with language model adaptation, we have chosen to perform linear interpolation offline, and perform language model switching during decoding. While model switching is a fast operation, it also makes the space complexity of storing the language models linear to the number of clusters. For scaling the approach to a high number of clusters, we envision that multi-pass decoding, with an unadapted language model in the first phase, and rescoring with a language model adapted online, could perform adequately, and keep the complexity independent of the number of clusters.

data set	sentences	words (de)
kde	216 000	1 990 000
kdedoc	2880	41 000
kdegb	51 300	450 000
oo	41 000	434 000
oo3	56 800	432 000
php	38 500	301 000
tm	146 000	2 740 000
acquis	2 660 000	58 900 000
dgt	372 000	8 770 000
ecb	110 000	2 850 000
ep7	1 920 000	50 500 000
nc7	159 000	3 950 000
total (train)	5 780 000	131 000 000
dev (IT)	3500	47 000
dev (LEGAL)	2000	46 800
test (IT)	5520	51 800
test (LEGAL)	9780	250 000

Table 2: Parallel data sets English–German.

data set	sentences	words (en)
eu	1 270 000	25 600 000
fiction	830 000	13 700 000
navajo	30 000	490 000
news	110 000	2 550 000
paraweb	370 000	3 930 000
subtitles	2 840 000	21 200 000
techdoc	970 000	7 270 000
total (train)	6 420 000	74 700 000
dev	3500	50 700
test	3500	49 600

Table 3: Parallel data sets Czech–English.

5 Evaluation

5.1 Data and Methods

We conduct all experiments with Moses (Koehn et al., 2007), SRILM (Stolcke, 2002), and GIZA++ (Och and Ney, 2003). Log-linear weights are optimized using MERT (Och and Ney, 2003). We keep the word alignment and lexical reordering models constant through the experiments to minimize the number of confounding factors. We report translation quality using BLEU (Papineni et

[5] If the development set is labelled, one can also use a gold segmentation of development sets instead of k-means clustering. At decoding time, cluster assignment can be performed by automatically assigning each sentence to the closest centroid, or again through gold labels, if available.

system	TM adaptation		LM adaptation		TM+LM adaptation	
	IT	LEGAL	IT	LEGAL	IT	LEGAL
baseline	21.1	49.9	21.1	49.9	21.1	49.9
1 cluster (no split)	21.3*	49.9	21.8*	49.7	21.8*	49.8
2 clusters	21.6*	49.9	22.2*	50.4*	22.8*	50.2*
4 clusters	21.7*	49.9	23.1*	50.2*	22.6*	50.2*
8 clusters	22.1*	49.9	23.1*	50.1*	22.7*	50.3*
16 clusters	21.1	49.9	22.6*	50.3*	21.9*	50.1*
gold clusters	21.8*	50.1*	22.4*	50.1*	23.2*	49.9

Table 4: Translation experiments EN–DE. BLEU scores reported.

al., 2002). We account for optimizer instability by running 3 independent MERT runs per system, and performing significance testing with MultEval (Clark et al., 2011). Systems significantly better than the baseline with $p < 0.01$ are marked with (*).

We conduct experiments on two data sets. The first is an English–German translation task with two domains, texts related to information technology (IT) and legal documents (LEGAL). We use data sets from both domains, plus out-of-domain corpora, as shown in table 2. 7 data sets come from the domain IT: 6 from OPUS (Tiedemann, 2009) and a translation memory (tm) provided by our industry partner. 3 data sets are from the legal domain: the ECB corpus from OPUS, plus the JRC-Acquis (Steinberger et al., 2006) and DGT-TM (Steinberger et al., 2012). 2 data sets are out-of-domain, made available by the 2012 Workshop on Statistical Machine Translation (Callison-Burch et al., 2012). The development sets are random samples from the respective in-domain bitexts (held-out from training). The test sets have been provided by Translated, our industry partner in the MATECAT project.

Our second data set is CzEng 0.9, a Czech–English parallel corpus (Bojar and Zabokrtský, 2009). It contains text from 7 different sources, on which we train separate component models. The size of the corpora is shown in table 3. As development and test sets, we use 500 sentences of held-out data per source.

For both data sets, language models are trained on the target side of the bitexts. In all experiments, we keep the number of component models constant: 12 for EN–DE, 7 for CZ–EN. We vary the number of clusters k from 1, which corresponds to adapting the models to the full development set, to 16. The baseline is the concatenation of all training data, with no adaptation performed. We also evaluate the labelled setting, where instead of unsupervised clustering, we use gold labels to split the development and test sets, and adapt the models to each labelled domain.

Data set	λ_{IT}	λ_{LEGAL}	$\lambda_{\text{cluster 1}}$	$\lambda_{\text{cluster 2}}$
kde	1.0	1.0	1.0	1.0
kdedoc	0.64	12.0	86.0	6.4
kdegb	1.6	2.3	1.7	2.7
oo	0.76	1.6	0.73	1.7
oo3	1.8	4.7	2.4	2.7
php	0.79	6.3	0.69	3.5
tm	1.3	1.3	1.5	1.1
acquis	0.024	3.5	0.018	1.9
dgt	0.053	4.5	0.033	2.4
ecb	0.071	2.3	0.039	1.2
ep7	0.037	0.53	0.024	0.29
nc7	0.1	1.1	0.063	0.62

Table 5: Weight vectors for feature $p(\bar{t}|\bar{s})$ optimized on four development sets (from gold split and clustering with $k = 2$).

5.2 Results

Table 4 shows results for the EN–DE data set. For our clustering experiments, the development set is the concatenation of the LEGAL and IT development sets. However, we always use the gold segmentation between LEGAL and IT for MERT and testing. This allows for a detailed analysis of the effect of development data clustering for the purpose of model adaptation. In an unlabelled setting, one would have to run MERT either on the full development set (as we will do for the CZ–EN task) or separately on each cluster, or use an alternative approach to optimize log-linear weights in a multi-domain setting, such as feature augmentation as described by (Clark et al., 2012).

system	TM adaptation	LM adaptation	TM+LM adaptation
baseline	34.4	34.4	34.4
1 cluster (no split)	34.5	33.7	34.1
2 clusters	34.6	34.0	34.4
4 clusters	34.7*	34.3	34.6
8 clusters	34.7*	34.5	34.9*
16 clusters	34.7*	34.7*	35.0*
gold clusters	35.0*	35.0*	35.4*

Table 6: Translation experiments CZ–EN. BLEU scores reported.

We find that an adaptation of the TM and LM to the full development set (system "1 cluster") yields the smallest improvements over the unadapted baseline. The reason for this is that the mixed-domain development set is not representative for the respective test sets. Using multiple adapted systems yields better performance. For the IT test set, the system with gold labels and TM adaptation yields an improvement of 0.7 BLEU (21.1 → 21.8), LM adaptation yields 1.3 BLEU (21.1 → 22.4), and adapting both models outperforms the baseline by 2.1 BLEU (21.1 → 23.2). The systems that use unsupervised clusters reach a similar level of performance than those with gold clusters, with best results being achieved by the systems with 2–8 clusters. Some systems outperform both the baseline and the gold clusters, e.g. TM adaptation with 8 clusters (21.1 → 21.8 → 22.1), or LM adaptation with 4 or 8 clusters (21.1 → 22.4 → 23.1).

Results with 16 clusters are slightly worse than those with 2–8 clusters due to two effects. Firstly, for the system with adapted TM, one of the three MERT runs is an outlier, and the reported BLEU score of 21.1 is averaged from the three MERT runs achieving 22.1, 21.6, and 19.6 BLEU, respectively. Secondly, about one third of the IT test set is assigned to a cluster that is not IT-specific, which weakens the effect of domain adaptation for the systems with 16 clusters.

For the LEGAL subset, gains are smaller. This can be explained by the fact that the majority of training data is already from the legal domain, which makes it unnecessary to boost its impact on the probability distribution even further.

Table 5 shows the automatically obtained translation model weight vectors for two systems, "gold clusters" and "2 clusters", for the feature $p(\bar{t}|\bar{s})$. It illustrates that all the corpora that we consider out-of-domain for IT are penalized by a factor of 10–50 (relative to the in-domain *kde* corpus) for the computation of this feature. For the LEGAL domain, the weights are more uniform, which is congruent with our observation that BLEU changes little.

Table 6 shows results for the CZ–EN data set. For each system, MERT is performed on the full development set. As in the first task, adaptation to the full development set is least effective. The systems with unsupervised clusters significantly outperform the baseline. For the system with 16 clusters, we observe an improvement of 0.3 BLEU for TM adaptation, and 0.6 BLEU for adapting both models (34.4 → 34.7 → 35.0). The labelled system, i.e. the system with 7 clusters corresponding to the 7 data sources, both for the development and test set, performs best. We observe gains of 0.6 BLEU (34.4 → 35.0) for TM or LM adaptation, and 1 BLEU (34.4 → 35.4) when both models are adapted.

We conclude that the translation model architecture is effective in a multi-domain setting, both with unsupervised clusters and labelled domains. The fact that language model adaptation yields an additional improvement in our experiments suggests that it it would be worthwhile to also investigate a language model data structure that efficiently supports multiple domains.

6 Conclusion

We have presented a novel translation model architecture that delays the computation of translation model features to the decoding phase, and uses a vector of component models for this computation. We have also described a usage scenario for this architecture, namely its ability to quickly switch between weight vectors in order to serve as an adapted model for multiple domains. A simple, unsupervised clustering of development data is sufficient to make use of this ability and implement a multi-domain translation system. If available, one can also use the architecture in a labelled setting.

Future work could involve merging our translation model framework with the online adaptation of other models, or the log-linear weights. Our approach is orthogonal to that of (Clark et al., 2012), who perform feature augmentation to obtain multiple sets of adapted log-linear weights. While (Clark et al., 2012) use labelled data, their approach could in principle also be applied after unsupervised clustering.

The translation model framework could also serve as the basis of real-time adaptation of translation systems, e.g. by using incremental means to update the weight vector, or having an incrementally trainable component model that learns from the post-edits by the user, and is assigned a suitable weight.

Acknowledgments

This research was partially funded by the Swiss National Science Foundation under grant 105215_126999, the European Commission (MATECAT, ICT-2011.4.2 287688) and the DARPA BOLT project.

References

Pratyush Banerjee, Jinhua Du, Baoli Li, Sudip Kumar Naskar, Andy Way, and Josef Van Genabith. 2010. Combining multi-domain statistical machine translation models using automatic classifiers. In *9th Conference of the Association for Machine Translation in the Americas (AMTA 2010)*, Denver, Colorado, USA.

Ondrej Bojar and Zdenek Zabokrtský. 2009. Czeng 0.9: Large parallel treebank with rich annotation. *Prague Bull. Math. Linguistics*, 92:63–84.

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, June. Association for Computational Linguistics.

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 Association for Computational Linguistics: Human Language Technologies*, pages 176–181, Portland, Oregon, USA, June. Association for Computational Linguistics.

Jonathan H. Clark, Alon Lavie, and Chris Dyer. 2012. One system, many domains: Open-domain statistical machine translation via feature augmentation. In *Conference of the Association for Machine Translation in the Americas 2012 (AMTA 2012)*, San Diego, California, USA.

George Foster and Roland Kuhn. 2007. Mixture-model adaptation for SMT. In *Proceedings of the Second Workshop on Statistical Machine Translation*, StatMT '07, pages 128–135, Prague, Czech Republic. Association for Computational Linguistics.

Philipp Koehn, Franz Josef Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *NAACL '03: Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology*, pages 48–54, Edmonton, Canada. Association for Computational Linguistics.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondřej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open Source Toolkit for Statistical Machine Translation. In *ACL 2007, Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics Companion Volume Proceedings of the Demo and Poster Sessions*, pages 177–180, Prague, Czech Republic, June. Association for Computational Linguistics.

Spyros Matsoukas, Antti-Veikko I. Rosti, and Bing Zhang. 2009. Discriminative corpus weight estimation for machine translation. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 2 - Volume 2*, pages 708–717, Singapore. Association for Computational Linguistics.

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

Daniel Ortiz-Martínez, Ismael García-Varea, and Francisco Casacuberta. 2010. Online learning for interactive statistical machine translation. In *HLT-NAACL*, pages 546–554. The Association for Computational Linguistics.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: A method for automatic evaluation of machine translation. In *ACL '02: Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Majid Razmara, George Foster, Baskaran Sankaran, and Anoop Sarkar. 2012. Mixing multiple translation models in statistical machine translation. In

Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics, Jeju, Republic of Korea. Association for Computational Linguistics.

Rico Sennrich. 2012a. Mixture-modeling with unsupervised clusters for domain adaptation in statistical machine translation. In *16th Annual Conference of the European Association for Machine Translation (EAMT 2012)*, pages 185–192, Trento, Italy.

Rico Sennrich. 2012b. Perplexity minimization for translation model domain adaptation in statistical machine translation. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*, pages 539–549, Avignon, France. Association for Computational Linguistics.

Kashif Shah, Loc Barrault, and Holger Schwenk. 2012. A general framework to weight heterogeneous parallel data for model adaptation in statistical machine translation. In *Conference of the Association for Machine Translation in the Americas 2012 (AMTA 2012)*, San Diego, California, USA.

Ralf Steinberger, Bruno Pouliquen, Anna Widiger, Camelia Ignat, Tomaz Erjavec, Dan Tufis, and Daniel Varga. 2006. The JRC-Acquis: A multilingual aligned parallel corpus with 20+ languages. In *Proceedings of the 5th International Conference on Language Resources and Evaluation (LREC'2006)*, Genoa, Italy.

Ralf Steinberger, Andreas Eisele, Szymon Klocek, Spyridon Pilos, and Patrick Schlüter. 2012. DGT-TM: A freely available translation memory in 22 languages. In *Proceedings of the Eight International Conference on Language Resources and Evaluation (LREC'12)*, Istanbul, Turkey. European Language Resources Association (ELRA).

Andreas Stolcke. 2002. SRILM – An Extensible Language Modeling Toolkit. In *Seventh International Conference on Spoken Language Processing*, pages 901–904, Denver, CO, USA.

Jörg Tiedemann. 2009. News from OPUS - a collection of multilingual parallel corpora with tools and interfaces. In N. Nicolov, K. Bontcheva, G. Angelova, and R. Mitkov, editors, *Recent Advances in Natural Language Processing*, volume V, pages 237–248. John Benjamins, Amsterdam/Philadelphia, Borovets, Bulgaria.

Hirofumi Yamamoto and Eiichiro Sumita. 2007. Bilingual cluster based models for statistical machine translation. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 514–523, Prague, Czech Republic.