ILLC-UvA Adaptation System (Scorpio) at WMT'16 IT-DOMAIN Task

Hoang Cuong and **Stella Frank** and **Khalil Sima'an**
Institute for Logic, Language and Computation
University of Amsterdam
Science Park 107, 1098 XG Amsterdam, The Netherlands

Abstract

This paper describes *Scorpio*, the ILLC-UvA Adaptation System submitted to the IT-DOMAIN translation task at WMT 2016, which participated with the language pair of English-Dutch. This system consolidates the ideas in our previous work on latent variable models for adaptation, and demonstrates their effectiveness in a competitive setting.

1 Introduction

ILLC-UvA participated in the WMT 2016 Shared Task of Machine Translation for the Information Technology (IT) Domain. In this paper, we briefly describe our system, which was submitted for the language pair of English–Dutch. Our system uses simple latent domain variable models for adaptation proposed in Cuong and Sima'an (2014b) and Cuong and Sima'an (2014a). More specifically, we enhance a standard phrase-based baseline system (Koehn et al., 2007) with adapted translation models and language models.

We equip these models with a latent domain variable and adapt them to an in-domain task represented by a seed corpus. We do not adapt the reordering models as we find reordering adaptation does not help much for this language pair. Several additional adapted features proposed in Cuong and Sima'an (2015) and Cuong et al. (2016) are also deployed, including domain-specific and domain-invariant translation features.

Despite the simplicity of our adaptation models, our results show effective adaptation performance for the task. This system consolidates the ideas in our previous work of latent variable models for adaptation, and shows their effectiveness in a competitive setting.

2 Data

We use all the training data provided by the organizers. Table 1 summarizes the data.

		English-Dutch	
In-Domain	Sents	**211K**	
	Words	$1.69M$	$1.65M$
General-Domain	Sents	**1.95M**	
	Words	$52.60M$	$52.95M$
Internal Dev	Sents	**1800**	
	Words	$41.35K$	$42.06K$
Internal Test	Sents	**200**	
	Words	$6.4K$	$6.3K$

Table 1: Data Preparation.

More specifically, we use the European Parliament (Europarl) parallel corpus (Koehn, 2005) as general-domain data. We use the corpora of IT-related terms from *Wikipedia* and *Localization PO* files as the in-domain data. For training Dutch language models we use the monolingual Dutch side of Europarl, together with in-domain data.

We split the provided development data (2K sentence pairs) into two different internal datasets:

- A dev set of 1800 sentence pairs used for system optimization.
- A test set of 200 sentence pairs used for evaluation.

Preprocessing

All the data is preprocessed before training. For preprocessing we remove all sentences that have

Proceedings of the First Conference on Machine Translation, Volume 2: Shared Task Papers, pages 423–427,
Berlin, Germany, August 11-12, 2016.

more than 80 tokens. The data is tokenized and lowercased using the standard Moses toolkit (tokenizer.perl and lowercase.perl). A recaser is also built for postprocessing the system output. Finally, the standard Moses detokenizer script is used to detokenize the output.

For the submission, we also apply a few additional rules that we believe would help recasing and detokenization, such as:[1]

- " *'s*" → "*'s*" (We remove spaces before *'s*.)
- "> a" → "> A", "> b" → "> B", etc. (We uppercase the first character after > .)
- If the target sentence contains the string "> " (which has a space) but the source sentence contains only ">" (which does not have a space), we replace all "> " with ">".

Despite those additional efforts, we found there is still (1): a huge difference in final performance between BLEU case-insensitive and BLEU case-sensitive; (2): a quite big difference between BLEU scores on the final test set and our (admittedly small) validation set. This suggests that there is still lots of room for improving our final translations with better de-tokenization. However, this was not the focus of our submission.

3 System Description

We first train a baseline with standard phrase-based system, using all the parallel data, i.e. the concatenation of in-domain and general-domain data. The system includes MOSES (Koehn et al., 2007) baseline feature functions, plus eight hierarchical lexicalized reordering model feature functions (Galley and Manning, 2008). The training data is first word-aligned using GIZA++ (Och and Ney, 2003) and then symmetrized with *grow(-diag)-final-and* (Koehn et al., 2003). We limit the phrase length to a maximum of seven words.

Somewhat surprisingly, we find that increasing the maximum number of words for phrases from *three* to *seven* significantly improves the baseline on the adaptation task. We believe this is quite important. It suggests that for validating domain adaptation methods over a phrase-based system, the system itself should be built over phrases with a reasonable maximum length (e.g. seven words).

We use the phrase extraction component from Stanford Phrasal (Cer et al., 2010), instead of the phrase extraction component included in Moses. Our experience has been that this usually produces better translation accuracy, making the baseline stronger.

Note that we do not filter any phrases. All phrases generated from the word alignments are kept. In this way, instead of discarding phrases with small translation probabilities, we keep all of them and assign a fixed and small translation probability of 0.0001 in such cases.

To tune the system, we use the k-best batch MIRA (Cherry and Foster, 2012). Finally, we use MOSES as a decoder (Koehn et al., 2007).

Our Dutch language models are interpolated 4-grams with Kneser-Ney smoothing, estimated using KenLM (Heafield et al., 2013).

In the following, we denote features from the baseline system as **Concatenation**. To improve the baseline, we enhance the system with additional adaptation models that are trained by utilizing the in-domain data. The following sections will describe our methods in detail.

3.1 Biasing translation models

Given the general-domain corpus and the small in-domain corpus, we first bias the learning of translation models over the general-domain corpus, with guidance from the in-domain data that directly represents the task. We use simple latent domain variable model for adaptation proposed in (Cuong and Sima'an, 2014b; Cuong and Sima'an, 2014a). There are four translation models we aim to learn here, specifically two translation models and two lexical weightings. More technical detail can be found in (Cuong and Sima'an, 2014b; Cuong and Sima'an, 2014a).

Along with our biased translation models (**Weighted**), we also train translation models directly on the provided in-domain data (**In-domain**). Note that our biased translation models are sharp in terms of having low entropies in translation distributions. Meanwhile, the translation statistics we induce from the in-domain are even sharper. Meanwhile, the translation statistics we induce from the in-domain are even sharper. Our

[1] However, we are not sure whether these "heuristics" rules are correct or not, as there is no way to verify them.

experience suggests the statistics induced from in-domain data still incrementally contributes to the adaptation.

We combine all three different types of translation models together. The combination is optimized over the (internal) development set using linear combination (Sennrich, 2012).

To have an idea what the combining weights look like, Table 2 presents results for four translation features, i.e. the translation models (TM) and lexical weights (LEX) in both directions (en-nl and nl-en).

Combining weights for translation features			
	Concat.	**Weighted**	**In-Domain**
TM en-nl	0.002	0.724	0.274
Lex en-nl	0.001	0.594	0.405
TM nl-en	0.002	0.755	0.243
Lex nl-en	0.001	0.573	0.426

Table 2: Combining weights

We see that most of the adaptation is credited to the models trained with biased weighting. The models trained on the in-domain data still partially contributes to the adaptation. On the other hand, the model trained on the simple concatenation of the data does not contribute much.

3.2 Biasing language models

Along with biasing the translation models, we find it useful to bias the language models as well. With similar simple latent domain variable models (but in this case, trained on target side data only), we learn the relevance of each sentence with respect to the target domain. We train 3-gram language models with relevance weights. To avoid overfitting, we find that it is necessary to apply an expected smoothing approach in training. We choose *expected Kneser-Ney smoothing* technique (Zhang and Chiang, 2014) as it is simple and achieves state-of-the-art performance on the language modeling problem.

Note that we also train a 3-gram language model directly on the provided in-domain data, as well as another one trained on the concatenation of in- and general- domain data. This results in three different language models, similar to the three translation models we trained above. They are treated as separate dense features for our system.

We provide the combining weights (after tuning) in Table 3, in order to demonstrate the relative importance of the different language models.

Tuning weights for language modeling features		
Concatenation	**Weighted**	**In-Domain**
0.0336	0.0397	0.009

Table 3: Optimized weights for language models

All language models incrementally contribute to the adaptation performance. The model that trains with biased weighting contribute most. Meanwhile, the model trained on the concatenation of all data also contributes significantly to the adaptation performance. The model trained on the in-domain data, however, contributes least, probably because its size is relatively small.

3.3 Biasing reordering models

We also try adapting reordering models with the same technique. This, however, does not lead to much improvement, at least for the language pair we deployed. We thus drop this direction.

3.4 Additional adaptation features

Following (Cuong and Sima'an, 2015), we find it useful to exploit the word-level feature derived from IBM model 1 score (Och et al., 2004). Note that adding word-level features from both translation sides does not help much, as observed by (Och et al., 2004). We thus add only one from a translation side. More technical detail can be found in Cuong and Sima'an (2015).

Finally, we found it useful to add domain-invariant translation features for SMT. Specifically, we push the system to make safer choices, preferring domain-invariant translations which work well across latent domains, over risky domain-specific alternatives. More technical detail can be found in Cuong et al. (2016). The improvement we achieve, however, is quite modest compared to what we achieve by utilizing the in-domain data. Nonetheless, we believe this is very natural, as the most effective adaptation method always comes from providing more in-domain data.

4 Results

Our baseline, as described earlier, is created from the concatenation of all parallel data provided

by the organizer. The language models are also trained by concatenating all monolingual data provided by the organizer. The baseline has 17 translation and language modeling features in total. Meanwhile, our system has 23 features (17 + 6 adapted features).

Table 4 and 5 present translation results on the internal dev and test sets respectively, with BLEU (Papineni et al., 2002), METEOR (Denkowski and Lavie, 2011), TER (Snover et al., 2006) and finally BEER (Stanojević and Sima'an, 2014).

English-Dutch				
System	BLEU	METEOR	TER	BEER
Baseline	28.1	28.7	53.3	18.4
Scorpio	30.1	29.9	51.6	20.9

Table 4: Results on Dev set

English-Dutch				
System	BLEU	METEOR	TER	BEER
Baseline	34.5	32.9	45.3	24.7
Scorpio	36.8	34.5	43.1	28.7

Table 5: Results on Test set

Note that these results are case-insensitive, without the post-processing steps for detokenizing/recasing sentences as described above.

Despite the simplicity of the adaptation models, our experiments suggest efficient adaptation performance for the task. The adaptations consistently improve all measures by more than 1 point, occasionally much more.

5 Conclusion

We have described our ILLC-UvA adaptation system (Scorpio) at WMT'16 IT-DOMAIN Task. Relying on simple latent domain variable models proposed in our previous work (Cuong and Sima'an, 2014b; Cuong and Sima'an, 2014a), the system shows promising performance for the adaptation task.

Acknowledgements

We thank two anonymous reviewers for their constructive comments on earlier versions. The first author is supported by the EXPERT (EXPloiting Empirical appRoaches to Translation) Initial Training Network (ITN) of the European Union's Seventh Framework Programme. The second author is supported by funding from the European Union's Horizon 2020 research and innovation programme under grant agreement Nr. 645452. The third author is supported by VICI grant nr. 277-89-002 from the Netherlands Organization for Scientific Research (NWO).

References

Daniel Cer, Michel Galley, Daniel Jurafsky, and Christopher D. Manning. 2010. Phrasal: A toolkit for statistical machine translation with facilities for extraction and incorporation of arbitrary model features. In *NAACL HLT 2010 Demonstration Session*.

Colin Cherry and George Foster. 2012. Batch tuning strategies for statistical machine translation. In *NAACL HLT*.

Hoang Cuong and Khalil Sima'an. 2014a. Latent domain phrase-based models for adaptation. In *EMNLP*.

Hoang Cuong and Khalil Sima'an. 2014b. Latent domain translation models in mix-of-domains haystack. In *COLING*.

Hoang Cuong and Khalil Sima'an. 2015. Latent domain word alignment for heterogeneous corpora. In *Proceedings of NAACL-HLT*.

Hoang Cuong, Khalil Sima'an, and Ivan Titov. 2016. Adapting to all domains at once: Rewarding domain invariance in SMT. In *TACL*.

Michael Denkowski and Alon Lavie. 2011. Meteor 1.3: Automatic metric for reliable optimization and evaluation of machine translation systems. In *Proceedings of the Sixth Workshop on Statistical Machine Translation*, WMT '11. Association for Computational Linguistics.

Michel Galley and Christopher D. Manning. 2008. A simple and effective hierarchical phrase reordering model. In *EMNLP*.

Kenneth Heafield, Ivan Pouzyrevsky, Jonathan H. Clark, and Philipp Koehn. 2013. Scalable modified kneser-ney language model estimation. In *ACL (Short Papers)*.

Philipp Koehn, Franz Josef Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *NAACL*.

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 (Interactive Poster and Demonstration Sessions)*.

Philipp Koehn. 2005. Europarl: A Parallel Corpus for Statistical Machine Translation. In *MTSummit*.

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

Franz Josef Och, Daniel Gildea, Sanjeev Khudanpur, Anoop Sarkar, Kenji Yamada, Alex Fraser, Shankar Kumar, Libin Shen, David Smith, Katherine Eng, Viren Jain, Zhen Jin, and Dragomir Radev. 2004. A smorgasbord of features for statistical machine translation. In *HLT-NAACL*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: A method for automatic evaluation of machine translation. In *ACL*.

Rico Sennrich. 2012. Perplexity minimization for translation model domain adaptation in statistical machine translation. In *EACL*.

Matthew Snover, Bonnie Dorr, R. Schwartz, L. Micciulla, and J. Makhoul. 2006. A study of translation edit rate with targeted human annotation. In *AMTA*.

Miloš Stanojević and Khalil Sima'an. 2014. Beer: Better evaluation as ranking. In *WMT*.

Hui Zhang and David Chiang. 2014. Kneser-Ney smoothing on expected counts. In *ACL*.