The Heidelberg University English-German translation system for IWSLT 2015

Laura Jehl, Patrick Simianer, Julian Hitschler, Stefan Riezler

Department of Computational Linguistics
Heidelberg University, Germany
{jehl, simianer, hitschler, riezler}@cl.uni-heidelberg.de

Abstract

We describe Heidelberg University's system for English-to-German translation of transcribed TED talks. Our system follows the hierarchical phrase-based paradigm [1]. We only used data allowed within the constrained track. Consistent gains were found using our in-house implementation of automatic source-side reordering, as well as large-scale tuning with a large, lexicalized feature set. We also confirm the success of large class-based language-models.

1. Introduction

We describe the Heidelberg University (`hdu`) submission to the IWSLT 2015 evaluation. We submitted a system for translating transcribed English TED talks into German, using only data permitted within the constrained track. We focus on improving a hierarchical phrase-based system by adding large language models and thousands of sparse, lexicalized features tuned on a large in-domain data set. We further incorporated syntactic knowledge through source-side reordering and k-best rescoring with language models based on syntactic annotations.

The paper is organized as follows: Our baseline setup is described in Section 2. Section 3 then explains our training pipeline and evaluates the contributions of each step. In Section 4, we show that scaling up the feature set and training a parallelized pairwise ranking optimizer on a larger development set further improves our system. We also conduct ablation experiments for different feature templates. Section 5 describes the integration of various external knowledge sources via k-best rescoring.

2. SMT system

All our systems use the `cdec`[1] tools for phrase extraction and decoding [2]. Our language models are estimated using KenLM [3]. For parameter tuning we use our in-house pairwise ranking optimizer `dtrain`, which is available in the `cdec` repository [4]. This section describes data preparation and the baseline system.

2.1. Data

We used all provided bilingual training data. Prior to training, we filtered out empty lines and any pairs containing sentences longer than 150 words. For the common crawl data, we applied an additional filtering step by running `langid.py` [5] on both sides to filter out sentences in the wrong source or target language. Datasets were tokenized with `cdec`'s `tokenize-anything.sh` and truecased using the truecaser available in the Moses toolkit.[2] All systems described in Sections 2 and 3 were tuned on the IWSLT `dev2010` development set with `tst2012` and `tst2013` used as progress test sets. We then added `tst2011-13` to our tuning data (Section 4), leaving `tst2010` as a held-out set for tuning our k-best reranker (Section 5). `tst2014` was treated as a blind test set.

2.2. Baseline

Our baseline model includes 21 features, namely bidirectional lexical phrase pair and word pair probabilities, seven pass-through features, three arity penalty features, a 4-gram language model built from the target side of the training data and count features for word penalty, glue rules, and language model OOVs.

3. Training Pipeline

We now describe our training pipeline and feature set and evaluate their performance of each step. The results are listed in Table 1. All tables report cased, detokenized BLEU scores obtained via the evaluation server provided by the task organizers.

3.1. Source-side reordering

To account for differences in word order, we re-arranged all source-sentences to match the syntax of the target language by applying a variation of the approach described in [6]. This approach works by permuting nodes in a dependency tree. During training, the reorderer generates all possible reordering rules within a window of three nodes governed by the same parent nodes. It then selects the rule which reduces the number of crossing alignments most on a randomly selected

[1] `https://github.com/redpony/cdec`

[2] `http://www.statmt.org/moses/`

validation set. This rule is applied to the training data and the procedure is repeated. Through this repeated permutation, the algorithm is able to generate long-range reorderings. A reordering rule stores part-of-speech and dependency label information of nodes, and a permutation order. If a matching configuration is found at test time, the permutation is applied. In order to reduce training time and to learn rules specific to spoken language, we trained the reorderer on in-domain data only. We used the Stanford parser for English[3], but our implementation can also be applied to the output of other parsers, e.g. in CoNLL format. The code will be made available.[4] We reordered and re-aligned all training data. Source-side reordering produced small but consistent improvements of 0.1 - 0.37 BLEU (experiment 1).

3.2. Domain adaptation

For domain adaptation, we added a 4-gram language model trained on the target side of the WIT3 data only to the log-linear model. In addition to that, we annotated each hierarchical phrase with binary features indicating which corpora it came from, allowing the model to learn a log-linear scaling weight for this phrase. This approach is similar to the work of [7]. Domain adaptation improved the model by 0.3 BLEU points (experiment 2).

3.3. Sparse alignment features

We included lexicalized alignment indicator features which model word alignment, deletion and insertion in source and target, as described in [8]. Even when tuned on a small development set, these features produced consistent gains of 0.16 to 0.29 BLEU points (experiment 3). More sparse features are described in Section 4.

3.4. Large and class-based language models

Previous work has shown the effectiveness of class-based language models (e.g. [9]). We used `brown-cluster`[5] to infer word classes from the language model training data. Since the KenLM implementation of class-based language models uses as an additional feature the probability $p(w|c)$ of a class c generating a word w, we normalized the raw frequencies returned by `brown-cluster`. We first trained a 7-gram class-based language model using 50 classes on the target side of the training data (experiment 4), but observed only a small improvement on `tst2012`, and no improvement on `tst2013`.

However, when increasing the size of the monolingual training data for word- and class-based language models to 26.8 million sentences, we were able to improve by 1.4 - 2 BLEU points (experiments 5a and 5b). We first added 300 thousand sentences from German political speeches to the language modelling data. We then applied cross-entropy based data selection using an in-domain language model to select 50% of the sentences from newscrawl, as described in [10]. To avoid the selection bias for shorter sentences, we only selected sentences with 5 words or longer. After de-duplication, we obtained 26.8 million sentences. We then built a 5-gram word-based language model, and a 7-gram class-based language model using 200 classes. We also increased the order of our in-domain language model from 4 to 5.

3.5. Comparing fast_align and GIZA++

To allow faster development, we first trained models using the re-parametrized IBM Model 2 implementation in `cdec` (`fast_align`[6]). However, our experiments confirmed that training alignments with GIZA++ [11] (we used the parallel implementation in `mgiza++` [12]) gave a significant boost in performance of 1.01 up to 1.6 BLEU (experiment 6), similar to the discrepancies observed in [13]. In particular, we observed that GIZA++-alignments substantially increased the number of extracted phrases: On `dev2010`, GIZA++-alignments produced 3.2 times as many phrases as `fast_align`.

4. Large-scale tuning with sparse features

Due to the successful results with the sparse alignment features we experimented with a wider range of sparse features (all implemented in `cdec`):

- rule identity features: one binary feature per rule
- rule shape features: generalized rules, by mapping to sequences of terminal and non-terminals
- rule bigram features: all bigrams of terminal and non-terminals inside rules, in both source and target sides

In addition to the plain rule shape features, in which terminals are replaced by a single terminal token, we also apply a variant where terminals are replaced by their word class.

While the rule identity features virtually allow to re-train the full grammar in tuning by assigning individual weights to every rule, rule shape and bigram features assess the quality of certain extraction patterns.

In total, the number of potential features is extremely high, several magnitudes larger than the total size of the grammar.

4.1. Online pairwise ranking optimization

Pairwise ranking optimization for SMT [14, 15, 4] allows tuning of a large number of features, in contrast to the traditional minimum error rate training [16]. We employ an online variant of this training scheme [4] with data sharding,

[3] `http://nlp.stanford.edu/software/lex-parser.shtml`

[4] `http://www.cl.uni-heidelberg.de/statnlpgroup/software.mhtml`.

[5] `https://github.com/percyliang/brown-cluster`

[6] `https://github.com/clab/fast_align`

exp	model	tst2012	Δ	tst2013	Δ	tst2014	Δ
0	baseline	19.78	–	21.38	–	18.86	
1	+ source-side reordering	20.15	+0.37	21.48	+0.1	19.03	+0.17
2	+ domain adaptation	20.46	+0.31	21.76	+0.28	19.32	+0.29
3	+ lexical alignment indicators	20.63	+0.17	22.05	+0.29	19.48	+0.16
4	+ class-based 7-gram (small, c=50)	20.91	+0.28	22	-0.05	19.59	+0.09
5a	+ large word-based LM (26.8M sentences)	21.34	+0.43	23.28	+1.28	20.28	+0.69
5b	+ large class-based LM (c=200)	22.05	+0.71	24.07	+0.79	20.64	+0.36
6	+ GIZA++	23.06	+1.01	25.59	+1.52	22.24	+1.6

Table 1: Components of the training pipeline.

feature selection by $\ell_1\ell_2$ regularization and randomization of the training input [17].

Sharding of the data greatly improves efficiency, as the tuning and optimization may run on several parts of the data at once. The models of different shards can then be mixed via simple averaging. Additionally, we use $\ell_1\ell_2$ regularization with a simple cut-off at 100,000 features per iteration. The input is randomized to counter-act potential effects which would depend on the order of the data. The shard size was chosen to reflect the typical tuning set size of about 1,000 segments.

The final model is an average of the weight vectors of all (15) training iterations. Longer training time neither lead to further improvements, nor did the model overfit. As the algorithm is a (margin) perceptron at its core, it has a single meta-parameter η which can be interpreted as a learning rate. Its optimal value 10^{-4} was found by a simple grid search. Note, that starting from $\mathbf{0}$, a fixed learning rate has no effect on the final model. With the margin perceptron however, it serves as a scaling factor which implicitly controls the number of pairs considered for each k-best list.

An ablation test, concatenating `dev2010`, `tst2011` and `tst2012` for tuning and validating on `tst2013` is given in Table 2. The baseline (experiment 8a) uses 27 features, including a single language model and the domain features. Isolating features shows some notable results (experiments 8b, 8c): While rule identifiers slightly degrade below the baseline (8b) and bigram and shape features show only little improvement (8c), the combination of bigram and shape features can be improved using ids by about 0.4 points (8d). A similar behavior can be observed with the lexical alignment indicators. When combining more sparse feature templates (experiment 8d), the final model sizes are very similar, as are the results on the validation set, which implies no or just a small additivity of lexical and rule id features. Improvements however are best at 0.74 points combining all features (experiment 8e). For the baseline system three runs were carried out to test the effect of the randomization – the standard derivation of the final score is quite low at 0.06 points.

When applying large-scale tuning with all features to our best setup from Section 3, we obtained a further improvement of 0.5 - 0.92 BLEU points (Table 3, experiment 9).

Exp.	feature set	tst2013 Δ	model size
8a	baseline	23.14±0.06	27
8b	bigram	23.44 Δ+0.30	150,514
	lexical	23.49 Δ+0.35	69,105
	id	22.99 Δ−0.15	224,685
	shape	23.30 Δ+0.16	202,777
8c	lex., id	23.15 Δ+0.01	227,743
	bigram, shape	23.37 Δ+0.23	204.537
8d	lex., id, shape	23.56 Δ+0.42	272,061
	bigram, id, shape	23.73 Δ+0.59	265,316
	bigram, lex., shape	23.77 Δ+0.63	228,929
	bigram, lex., id	23.81 Δ+0.67	280,830
8e	all	23.88 Δ+0.74	260,697

Table 2: Ablation test for sparse features (the baseline used GIZA++ alignments, but only one target-side 4-gram language model).

5. k-best rescoring with syntactic and neural network language models

We incorporated more knowledge sources via k-best rescoring. We used three in-domain language models built from target side syntactic annotation, namely part-of-speech, morphology and lemma. The annotations were obtained by running the German dependency parser `parzu`[7]. We also trained an in-domain and a target-side feed-forward neural language model using the NPLM toolkit [18]. All experiments used $k = 100$.[8]

Weights for the different language models were learned using a pairwise ranking approach as described in [19], with an SGD classifier from `scikit-learn` [20]. We did not re-tune the SMT model features, but instead used the model score as a single feature to be tuned.

Results for rescoring are given in Table 3. The first two entries (experiment 4 and 7) show results for the best small-scale system described in Section 3 (no large language models, word alignments from `fast_align`). For this system, we observed gains on `tst2012` and `tst2013`, but a small

[7] `https://github.com/rsennrich/parzu`

[8] We experimented with $k = 1000$, but did not see an improvement.

loss on `tst2014`. The two bottom entries (experiment 9 and 10) show the effect of k-best rescoring on our best system, including large language models, GIZA++ alignments and large-scale tuning as described in the previous section. With this seutp, rescoring did improve BLEU. However, we conducted a small-scale human evaluation by having four raters express pairwise preferences for 30 randomly chosen sentences. The translation pairs were permuted and presented in different order to each rater. In total, we observed a preference for the rescored system in 61.67 percent of the cases, with an average pairwise agreement of 0.36 between annotators. This lead us to still submit the rescored system as our primary submission with the system without rescoring as contrastive submission.

6. Conclusions

We built a hierarchical phrase-based translation system for English-German translation using source and target side syntactic information, large-scale class- and word-based language models, and large-scale tuning with sparse features. On the small scale, combining source-side reordering, domain adaptation, sparse lexicalized alignment features, and a class-based language model, yielded 0.62 - 1.13 BLEU over our baseline. We were able to gain 1.14 - 2.07 BLEU points by adding large language models. Using slower, but more reliable, GIZA++ training, another 1.01 - 1.52 BLEU points were gained. Large-scale tuning with sparse features gave a further 0.5 - 0.92 BLEU points. For k-best reranking we observed gains on the held-out sets for the smaller model, but no additional gains in BLEU over the large model. However, human evaluation indicated a preference for the reranked outputs. Our final results are stated in Table 3 (experiment 9 and 10). They exceed the official baseline by 4.73 - 5.88 BLEU.

7. References

[1] D. Chiang, "Hierarchical phrase-based translation," *Computational Linguistics*, vol. 33, no. 2, 2007.

[2] C. Dyer, A. Lopez, J. Ganitkevitch, J. Weese, F. Ture, P. Blunsom, H. Setiawan, V. Eidelman, and P. Resnik, "cdec: A decoder, alignment, and learning framework for finite-state and context-free translation models," in *Proceedings of the ACL 2010 System Demonstrations*, Uppsala, Sweden, 2010.

[3] K. Heafield, I. Pouzyrevsky, J. H. Clark, and P. Koehn, "Scalable modified Kneser-Ney language model estimation," in *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, Sofia, Bulgaria, 2013.

[4] P. Simianer, S. Riezler, and C. Dyer, "Joint Feature Selection in Distributed Stochastic Learning for Large-Scale Discriminative Training in SMT," in *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics*, Jeju, Korea, 2012.

[5] M. Lui and T. Baldwin, "`langid.py`: An off-the-shelf language identification tool," in *Proceedings of the ACL 2012 System Demonstrations*, Jeju Island, Korea, 2012.

[6] D. Genzel, "Automatically learning source-side reordering rules for large scale machine translation," in *Proceedings of the 23rd international conference on Computational Linguistics*, Beijing, China, 2010.

[7] J. Niehues and A. Waibel, "Domain adaptation in statistical machine translation using factored translation models," in *Proceedings of EAMT*, Stroudsburg, PA, USA, 2010.

[8] F. Hieber and S. Riezler, "Bag-of-Words Forced Decoding for Cross-Lingual Information Retrieval," in *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics - Human Language Technologies*, Denver, Colorado, 2015.

[9] J. Wuebker, S. Peitz, A. Guta, and H. Ney, "The RWTH Aachen Machine Translation Systems for IWSLT 2014," in *Proceedings of the Int. Workshop on Spoken Language Translation*, South Lake Tahoe, CA, USA, 2014.

[10] A. Axelrod, X. He, and J. Gao, "Domain adaptation via pseudo in-domain data selection," in *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, Edinburgh, Scotland, 2011.

[11] F. J. Och and H. Ney, "A systematic comparison of various statistical alignment models," *Computational Linguistics*, vol. 29, no. 1, 2003.

[12] Q. Gao and S. Vogel, "Parallel implementations of word alignment tool," in *Software Engineering, Testing, and Quality Assurance for Natural Language Processing*, Columbus, Ohio, USA, 2008.

[13] C. Ding, M. Utiyama, and E. Sumita, "Improving fast_align by reordering," in *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, Lisbon, Portugal, 2015.

[14] L. Shen, A. Sarkar, and F. J. Och, "Discriminative reranking for machine translation," in *HLT-NAACL*, 2004, pp. 177–184.

[15] T. Watanabe, J. Suzuki, H. Tsukada, and H. Isozaki, "NTT statistical machine translation for IWSLT 2006," in *2006 International Workshop on Spoken Language Translation, IWSLT 2006, Keihanna Science City, Kyoto, Japan, November 27-28, 2006*, 2006, pp. 95–102.

[16] F. J. Och, "Minimum error rate training in statistical machine translation," in *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics*, Sapporo, Japan, 2003.

exp	model	tst2012	Δ	tst2013	Δ	tst2014	Δ
4	no rescoring	20.91	–	22	–	19.59	–
7	+ rescoring	21.25	+0.34	22.86	+0.86	19.4	-0.19
9	Contrastive (large-scale, no rescoring)	23.98†	–	26.09	–	23.24	–
10	Primary (large-scale + rescoring)	23.93†	-0.05	25.97	-0.12	23.22	-0.03

Table 3: Reranking experiments. † indicates different tuning sets: This experiment was tuned on dev2010, tst2011 and tst2013, leaving out tst2012.

[17] P. Simianer and S. Riezler, "Multi-task learning for improved discriminative training in SMT," in *Proceedings of the Eighth Workshop on Statistical Machine Translation*, Sofia, Bulgaria, 2013.

[18] A. Vaswani, Y. Zhao, V. Fossum, and D. Chiang, "Decoding with large-scale neural language models improves translation," in *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, Seattle, Washington, USA, 2013.

[19] M. Hopkins and J. May, "Tuning as ranking," in *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, Edinburgh, Scotland, UK., 2011.

[20] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, 2011.