Language Model and Grammar Extraction Variation in Machine Translation

Vladimir Eidelman[†], **Chris Dyer**[†‡], **and Philip Resnik**[†‡]
[†]UMIACS Laboratory for Computational Linguistics and Information Processing
[‡]Department of Linguistics
University of Maryland, College Park
{vlad,redpony,resnik}@umiacs.umd.edu

Abstract

This paper describes the system we developed to improve German-English translation of News text for the shared task of the Fifth Workshop on Statistical Machine Translation. Working within cdec, an open source modular framework for machine translation, we explore the benefits of several modifications to our hierarchical phrase-based model, including segmentation lattices, minimum Bayes Risk decoding, grammar extraction methods, and varying language models. Furthermore, we analyze decoder speed and memory performance across our set of models and show there is an important trade-off that needs to be made.

1 Introduction

For the shared translation task of the Fifth Workshop on Machine Translation (WMT10), we participated in German to English translation under the constraint setting. We were especially interested in translating from German due to set of challenges it poses for translation. Namely, German possesses a rich inflectional morphology, productive compounding, and significant word reordering with respect to English. Therefore, we directed our system design and experimentation toward addressing these complications and minimizing their negative impact on translation quality.

The rest of this paper is structured as follows. After a brief description of the baseline system in Section 2, we detail the steps taken to improve upon it in Section 3, followed by experimental results and analysis of decoder performance metrics.

2 Baseline system

As our baseline system, we employ a hierarchical phrase-based translation model, which is formally based on the notion of a synchronous context-free grammar (SCFG) (Chiang, 2007). These grammars contain pairs of CFG rules with aligned nonterminals, and by introducing these nonterminals into the grammar, such a system is able to utilize both word and phrase level reordering to capture the hierarchical structure of language. SCFG translation models have been shown to be well suited for German-English translation, as they are able to both exploit lexical information for and efficiently compute all possible reorderings using a CKY-based decoder (Dyer et al., 2009).

Our system is implemented within cdec, an efficient and modular open source framework for aligning, training, and decoding with a number of different translation models, including SCFGs (Dyer et al., 2010).[1] cdec's modular framework facilitates seamless integration of a translation model with different language models, pruning strategies and inference algorithms. As input, cdec expects a string, lattice, or context-free forest, and uses it to generate a hypergraph representation, which represents the full translation forest without any pruning. The forest can now be rescored, by intersecting it with a language model for instance, to obtain output translations. The above capabilities of cdec allow us to perform the experiments described below, which would otherwise be quite cumbersome to carry out in another system.

The set of features used in our model were the rule translation relative frequency $P(e|f)$, a target n-gram language model $P(e)$, a 'pass-through' penalty when passing a source language word to the target side without translating it, lexical translation probabilities $P_{lex}(\overline{e}|\overline{f})$ and $P_{lex}(\overline{f}|\overline{e})$,

[1] http://cdec-decoder.org

a count of the number of times that arity-0,1, or 2 SCFG rules were used, a count of the total number of rules used, a source word penalty, a target word penalty, the segmentation model cost, and a count of the number of times the glue rule is used. The number of non-terminals allowed in a synchronous grammar rule was restricted to two, and the non-terminal span limit was 12 for non-glue grammars. The hierarchical phrase-base translation grammar was extracted using a suffix array rule extractor (Lopez, 2007).

2.1 Data preparation

In order to extract the translation grammar necessary for our model, we used the provided Europarl and News Commentary parallel training data. The lowercased and tokenized training data was then filtered for length and aligned using the GIZA++ implementation of IBM Model 4 (Och and Ney, 2003) to obtain one-to-many alignments in both directions and symmetrized by combining both into a single alignment using the grow-diag-final-and method (Koehn et al., 2003). We constructed a 5-gram language model using the SRI language modeling toolkit (Stolcke, 2002) from the provided English monolingual training data and the non-Europarl portions of the parallel data with modified Kneser-Ney smoothing (Chen and Goodman, 1996). Since the beginnings and ends of sentences often display unique characteristics that are not easily captured within the context of the model, we explicitly annotate beginning and end of sentence markers as part of our translation process. We used the 2525 sentences in news-test2009 as our dev set on which we tuned the feature weights, and report results on the 2489 sentences of the news-test2010 test set.

2.2 Viterbi envelope semiring training

To optimize the feature weights for our model, we use Viterbi envelope semiring training (VEST), which is an implementation of the minimum error rate training (MERT) algorithm (Dyer et al., 2010; Och, 2003) for training with an arbitrary loss function. VEST reinterprets MERT within a semiring framework, which is a useful mathematical abstraction for defining two general operations, addition ($\oplus$) and multiplication ($\otimes$) over a set of values. Formally, a semiring is a 5-tuple $(\mathbb{K}, \oplus, \otimes, \overline{0}, \overline{1})$, where addition must be communicative and associative, multiplication must be associative and must distribute over addition, and an identity element exists for both. For VEST, having $\mathbb{K}$ be the set of line segments, $\oplus$ be the union of them, and $\otimes$ be Minkowski addition of the lines represented as points in the dual plane, allows us to compute the necessary MERT line search with the INSIDE algorithm.[2] The error function we use is BLEU (Papineni et al., 2002), and the decoder is configured to use cube pruning (Huang and Chiang, 2007) with a limit of 100 candidates at each node. During decoding of the test set, we raise the cube pruning limit to 1000 candidates at each node.

2.3 Compound segmentation lattices

To deal with the aforementioned problem in German of productive compounding, where words are formed by the concatenation of several morphemes and the orthography does not delineate the morpheme boundaries, we utilize word segmentation lattices. These lattices serve to encode alternative ways of segmenting compound words, and as such, when presented as the input to the system allow the decoder to automatically choose which segmentation is best for translation, leading to markedly improved results (Chris Dyer, 2009).

In order to construct diverse and accurate segmentation lattices, we built a maximum entropy model of compound word splitting which makes use of a small number of dense features, such as frequency of hypothesized morphemes as separate units in a monolingual corpus, number of predicted morphemes, and number of letters in a predicted morpheme. The feature weights are tuned to maximize conditional log-likelihood using a small amount of manually created reference lattices which encode linguistically plausible segmentations for a selected set of compound words.[3]

To create lattices for the dev and test sets, a lattice consisting of all possible segmentations for every word consisting of more than 6 letters was created, and the paths were weighted by the posterior probability assigned by the segmentation model. Then, max-marginals were computed using the forward-backward algorithm and used to prune out paths that were greater than a factor of 2.3 from the best path, as recommended by Chris Dyer (2009). To create the translation model for lattice input, we segmented the training data us-

[2] This algorithm is equivalent to the hypergraph MERT algorithm described by Kumar et al. (2009).

[3] The reference segmentation lattices used for training are available in the cdec distribution.

ing the 1-best segmentation predicted by the segmentation model, and word aligned this with the English side. This version of the parallel corpus was concatenated with the original training parallel corpus.

3 Experimental variation

This section describes the experiments we performed in attempting to assess the challenges posed by current methods and our exploration of new ones.

3.1 Bloom filter language model

Language models play a crucial role in translation performance, both in terms of quality, and in terms of practical aspects such as decoder memory usage and speed. Unfortunately, these two concerns tend to trade-off one another, as increasing to a higher-order more complex language model improves performance, but comes at the cost of increased size and difficulty in deployment. Ideally, the language model will be loaded into memory locally by the decoder, but given memory constraints, it is entirely possible that the only option is to resort to a remote language model server that needs to be queried, thus introducing significant decoding speed delays.

One possible alternative is a randomized language model (RandLM) (Talbot and Osborne, 2007). Using Bloom filters, which are a randomized data structure for set representation, we can construct language models which significantly decrease space requirements, thus becoming amenable to being stored locally in memory, while only introducing a quantifiable number of false positives. In order to assess what the impact on translation quality would be, we trained a system identical to the one described above, except using a RandLM. Conveniently, it is possible to construct a RandLM directly from an existing SRILM, which is the route we followed in using the SRILM described in Section 2.1 to create our RandLM. Table 1 shows the comparison of SRILM and RandLM with respect to performance on BLEU and TER (Snover et al., 2006) on the test set.

Language Model	BLEU	TER
RandLM	22.4	69
SRILM	**23.1**	**68**

Table 1: Impact of language model on translation

3.2 Minimum Bayes risk decoding

During minimum error rate training, the decoder employs a maximum derivation decision rule. However, upon exploration of alternative strategies, we have found benefits to using a minimum risk decision rule (Kumar and Byrne, 2004), wherein we want the translation E of the input F that has the least expected loss, again as measured by some loss function L:

$$\begin{aligned}\widehat{E} &= \arg\min_{E'} \mathbb{E}_{P(E|F)}[L(E,E')] \\ &= \arg\min_{E'} \sum_{E} P(E|F)L(E,E')\end{aligned}$$

Using our system, we generate a unique 500-best list of translations to approximate the posterior distribution $P(E|F)$ and the set of possible translations. Assuming $H(E,F)$ is the weight of the decoder's current path, this can be written as:

$$P(E|F) \propto \exp \alpha H(E,F)$$

where α is a free parameter which depends on the models feature functions and weights as well as pruning method employed, and thus needs to be separately empirically optimized on a held out development set. For this submission, we used $\alpha = 0.5$ and BLEU as the loss function. Table 2 shows the results on the test set for MBR decoding.

Language Model	Decoder	BLEU	TER
RandLM	Max-D	22.4	69
	MBR	**22.7**	**68.8**
SRILM	Max-D	23.1	68
	MBR	**23.4**	**67.7**

Table 2: Comparison of maximum derivation versus MBR decoding

3.3 Grammar extraction

Although the grammars employed in a SCFG model allow increased expressivity and translation quality, they do so at the cost of having a large number of rules, thus efficiently storing and accessing grammar rules can become a major problem. Since a grammar consists of the set of rules extracted from a parallel corpus containing tens of

Language Model	Grammar	Decoder Memory (GB)	Decoder time (Sec/Sentence)
Local SRILM	corpus	14.293 ± 1.228	**5.254 ± 3.768**
Local SRILM	sentence	10.964 ± .964	5.517 ± 3.884
Remote SRILM	corpus	3.771 ± .235	15.252 ± 10.878
Remote SRILM	sentence	**.443 ± .235**	14.751 ± 10.370
RandLM	corpus	7.901 ± .721	9.398 ± 6.965
RandLM	sentence	4.612 ± .699	9.561 ± 7.149

Table 3: Decoding memory and speed requirements for language model and grammar extraction variations

millions of words, the resulting number of rules can be in the millions. Besides storing the whole grammar locally in memory, other approaches have been developed, such as suffix arrays, which lookup and extract rules on the fly from the phrase table (Lopez, 2007). Thus, the memory requirements for decoding have either been for the grammar, when extracted beforehand, or the corpus, for suffix arrays. In cdec, however, loading grammars for single sentences from a disk is very fast relative to decoding time, thus we explore the additional possibility of having sentence-specific grammars extracted and loaded on an as-needed basis by the decoder. This strategy is shown to massively reduce the memory footprint of the decoder, while having no observable impact on decoding speed, introducing the possibility of more computational resources for translation. Thus, in addition to the large corpus grammar extracted in Section 2.1, we extract sentence-specific grammars for each of the test sentences. We measure the performance across using both grammar extraction mechanisms and the three different language model configurations: local SRILM, remote SRILM, and RandLM.

As Table 3 shows, there is a marked trade-off between memory usage and decoding speed. Using a local SRILM regardless of grammar increases decoding speed by a factor of 3 compared to the remote SRILM, and approximately a factor of 2 against the RandLM. However, this speed comes at the cost of its memory footprint. With a corpus grammar, the memory footprint of the local SRILM is twice as large as the RandLM, and almost 4 times as large as the remote SRILM. Using sentence-specific grammars, the difference becomes increasingly glaring, as the remote SRILM memory footprint drops to ≈450MB, a factor of nearly 24 compared to the local SRILM and a factor of 10 compared to the process size with the RandLM. Thus, using the remote SRILM reduces the memory footprint substantially but at the cost of significantly slower decoding speed, and conversely, using the local SRILM produces increased decoder speed but introduces a substantial memory overhead. The RandLM provides a median between the two extremes: reduced memory and (relatively) fast decoding at the price of somewhat decreased translation quality.

We also tried one other grammar extraction configuration, which was with so-called 'loose' phrase extraction heuristics, which permit unaligned words at the edges of phrases (Ayan and Dorr, 2006). When decoded using the SRILM and MBR, this achieved the best performance for our system, with a BLEU score of 23.6 and TER of 67.7.

4 Conclusion

We presented the University of Maryland hierarchical phrase-based system for the WMT2010 shared translation task. Using cdec, we experimented with a number of methods that are shown above to lead to improved German-to-English translation quality over our baseline according to BLEU and TER evaluation. These include methods to directly address German morphological complexity, such as appropriate feature functions, segementation lattices, and a model for automatically construcing the lattices, as well as alternative decoding strategies, such as MBR. We also presented several language model configuration alternatives, as well as grammar extraction methods, and emphasized the trade-off that must be made between decoding time, memory overhead, and translation quality in current statistical machine translation systems.

References

Necip Fazil Ayan and Bonnie J. Dorr. 2006. Going beyond AER: An extensive analysis of word alignments and their impact on MT. In *Proceedings of the Joint Conference of the International Committee on Computational Linguistics and the Association for Computational Linguistics (COLING-ACL'2006)*, pages 9–16, Sydney.

Stanley F. Chen and Joshua Goodman. 1996. An empirical study of smoothing techniques for language modeling. In *Proceedings of the 34th Annual Meeting of the Association for Computational Linguistics*, pages 310–318.

David Chiang. 2007. Hierarchical phrase-based translation. In *Computational Linguistics*, volume 33(2), pages 201–228.

Chris Dyer. 2009. Using a maximum entropy model to build segmentation lattices for MT. In *Proceedings of NAACL-HLT*.

Chris Dyer, Hendra Setiawan, Yuval Marton, and P. Resnik. 2009. The University of Maryland statistical machine translation system for the Fourth Workshop on Machine Translation. In *Proceedings of the EACL-2009 Workshop on Statistical Machine Translation*.

Chris Dyer, Adam Lopez, Juri Ganitkevitch, Jonathan Weese, Ferhan Ture, Phil Blunsom, Hendra Setiawan, Vladimir Eidelman, and Philip Resnik. 2010. cdec: A decoder, alignment, and learning framework for finite-state and context-free translation models. In *Proceedings of ACL System Demonstrations*.

Liang Huang and David Chiang. 2007. Forest rescoring: Faster decoding with integrated language models. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 144–151.

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.

Shankar Kumar and William Byrne. 2004. Minimum bayes-risk decoding for statistical machine translation. In *HLT-NAACL 2004: Main Proceedings*.

Shankar Kumar, Wolfgang Macherey, Chris Dyer, and Franz Och. 2009. Efficient minimum error rate training and minimum bayes-risk decoding for translation hypergraphs and lattices. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 163–171.

Adam Lopez. 2007. Hierarchical phrase-based translation with suffix arrays. In *Proceedings of EMNLP*, pages 976–985.

Franz Och and Hermann Ney. 2003. A systematic comparison of various statistical alignment models. In *Computational Linguistics*, volume 29(21), pages 19–51.

Franz Josef Och. 2003. Minimum error rate training in statistical machine translation. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, pages 160–167.

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

Matthew Snover, Bonnie Dorr, Richard Schwartz, Linnea Micciulla, and John Makhoul. 2006. A study of translation edit rate with targeted human annotation. In *In Proceedings of Association for Machine Translation in the Americas*, pages 223–231.

Andreas Stolcke. 2002. Srilm - an extensible language modeling toolkit. In *Intl. Conf. on Spoken Language Processing*.

David Talbot and Miles Osborne. 2007. Randomised language modelling for statistical machine translation. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, June.