Tera-Scale Translation Models via Pattern Matching

Adam Lopez
School of Informatics, University of Edinburgh
10 Crichton Street, Edinburgh EH8 9AB, United Kingdom
alopez@inf.ed.ac.uk

Abstract

Translation model size is growing at a pace that outstrips improvements in computing power, and this hinders research on many interesting models. We show how an algorithmic scaling technique can be used to easily handle very large models. Using this technique, we explore several large model variants and show an improvement 1.4 BLEU on the NIST 2006 Chinese-English task. This opens the door for work on a variety of models that are much less constrained by computational limitations.

1 Introduction

Translation model size is growing quickly due to the use of larger training corpora and more complex models. As an example of the growth in available training data, consider the curated Europarl corpus (Koehn, 2005), which more than doubled in size from 20 to 44 million words between 2003 and 2007.[1] As an example of model complexity, consider the popular hierarchical phrase-based model of Chiang (2007), which can translate discontiguous phrases. Under the loosest interpretation of this capability, any subset of words in a sentence can be a phrase. Therefore, the number of rules that the model can learn is exponential in sentence length unless strict heuristics are used, which may limit the model's effectiveness. Many other models translate discontiguous phrases, and the size of their extracted rulesets is such a pervasive problem that it is a recurring topic in the literature (Chiang, 2007; DeNeefe et al., 2007; Simard et al., 2005).

Most decoder implementations assume that all model rules and parameters are known in advance. With very large models, computing all rules and parameters can be very slow. This is a bottleneck in experimental settings where we wish to explore many model variants, and therefore presents a real impediment to full exploration of their potential. We present a solution to this problem.

To fully motivate the discussion, we give a concrete example of a very large model, which we generate using simple techniques that are known to improve translation accuracy. The model takes 77 CPU days to compute and consumes nearly a terabyte of external storage (§2). We show how to solve the problem with a previously developed algorithmic scaling technique that we call **translation by pattern matching** (§3). The key idea behind this technique is that rules and parameters are computed only as needed. Using this technique, we explore a series of large models, giving experimental results along a variety of scaling axes (§4). Our results extend previous findings on the use of long phrases in translation, shed light on the source of improved performance in hierarchical phrase-based models, and show that our tera-scale translation model outperforms a strong baseline.

2 A Tera-Scale Translation Model

We will focus on the hierarchical phrase-based model of Chiang (2007). It compares favorably

This research was conducted while I was at the University of Maryland. I thank David Chiang, Bonnie Dorr, Doug Oard, Philip Resnik, and the anonymous reviewers for comments, and especially Chris Dyer for many helpful discussions and for running the Moses experiments. This research was supported by the GALE program of the Defense Advanced Research Projects Agency, Contract No. HR0011-06-2-001 and by the EuroMatrix project funded by the European Commission (6th Framework Programme).

© 2008. Licensed under the *Creative Commons Attribution-Noncommercial-Share Alike 3.0 Unported* license (http://creativecommons.org/licenses/by-nc-sa/3.0/). Some rights reserved.

[1] Statistics from http://www.statmt.org/europarl.

Proceedings of the 22nd International Conference on Computational Linguistics (Coling 2008), pages 505–512
Manchester, August 2008

with conventional phrase-based translation (Koehn et al., 2003) on Chinese-English news translation (Chiang, 2007). We found that a baseline system trained on 27 million words of news data is already quite strong, but we suspect that it would be possible to improve it using some simple techniques.

Add additional training data. Our baseline already uses much of the available curated news data, but there is at least three times as much curated data available in the United Nations proceedings. Adding the UN data gives us a training corpus of 107 million words per language.

Change the word alignments. Our baseline uses Giza++ alignments (Och and Ney, 2003) symmetrized with the grow-diag-final-and heuristic (Koehn et al., 2003). We replace these with the maximum entropy aligments of Ayan and Dorr (2006b). They reported improvements of 1.6 BLEU in Chinese-English translation, though with much less training data.

Change the bilingual phrase extraction heuristic. Our baseline uses a **tight** heuristic, requiring aligned words at phrase edges. However, Ayan and Dorr (2006a) showed that a **loose** heuristic, allowing unaligned words at the phrase edges, improved accuracy by 3.7 BLEU with some alignments, again with much less training data.

Quadrupling the amount of training data predictably increases model size. The interaction of the alignment and phrase extraction heuristic increases the model size much more. This is because the maximum entropy alignments are sparse—fewer than 70% of the words are aligned. Consider a contiguous phrase chosen at random from a training sentence. With our sparse alignments, the chance that both of its edge words are aligned is less than half. The tight heuristic discards many possible phrases on this basis alone. The situation worsens with discontiguous phrases. However, with the loose heuristic, we see the opposite effect. Not only is a randomly chosen source phrase with unaligned edge words legal, but it may have many translations, since its minimal alignment in the target is likely to have one or more adjacent unaligned words, and any combination of these can be part of a valid target phrase.

To make matters concrete, we estimated the size of the translation model that would be produced using these modifications. We did not actually compute the full model, for reasons that will become apparent. Instead, we modified Chiang's extractor to simply report the number of rules that it would normally extract. We then computed the ratio of extracted rules to that of the baseline system. Under the rough assumption that the number of unique rules and representation size grows linearly in the number of extracted rules, we were then able to estimate the size of the large model. The results show that it would be impractical to compute the model (Table 1). Merely counting all of the rule occurrences took nearly 5 days on our 17-node research cluster. This does not even include the time required for sorting and scoring the rules, which we did not attempt. The resulting model would be nearly two orders of magnitude larger than the largest one we could find in the literature (Table 2).

3 Translation by Pattern Matching

Clearly, substantial experimentation with models this large is impossible unless we have considerable resources at our disposal. To get around this problem, we use an algorithmic scaling technique that we call translation by pattern matching. In this approach, the training text and its word alignment reside in memory. We then translate as follows.

for each input sentence **do**
  for each possible phrase in the sentence **do**
    Find its occurrences in the source text
    for each occurrence **do**
      Extract its aligned target phrase (if any)
    for each extracted phrase pair **do**
      Score using maximum likelihood
  Decode as usual using the scored rules

A similar method is used in example-based translation (Brown, 2004). It was applied to phrase-based translation by Callison-Burch et al. (2005) and Zhang and Vogel (2005). The key point is that the complete translation model is never actually computed—rules and associated parameters are computed only as needed. Obviously, the on-demand computation must be very fast. If we can achieve this, then the model can in principle be arbitrarily large. Callison-Burch et al. (2005) and Zhang and Vogel (2005) give very similar recipes for application to phrase-based models.

Fast lookup using pattern matching algorithms. The complexity of the naïve algorithm to find all occurrences of a source phrase in a training text T is linear in the length of the text, $O(|T|)$. This is much too slow for large texts. They solve this using an index data structure called a suffix

	Baseline	Large
Rules extracted (millions)	195	19,300
Extract time (CPU hours)	10.8	1,840
Unique rules (millions)	67	6,600*
Extract file size (GB)	9.3	917*
Model size (GB)	6.1	604*

Table 1: Extraction time and model sizes. The model size reported is the size of the files containing an external prefix tree representation (Zens and Ney, 2007). *Denotes estimated quantities.

Citation	Millions of rules
Simard et al. (2005)	(filtered) 4
Chiang (2007)	(filtered) 6
DeNeefe et al. (2007)	57
Zens and Ney (2007)	225
this paper	6,600

Table 2: Model sizes in the literature.

array (Manber and Myers, 1993). Its size is $4|T|$ bytes and it enables lookup of any length-m substring of T in $O(m + \log |T|)$ time.

Fast extraction using sampling. The complexity of extracting target phrases is linear in the number of source phrase occurrences. For very frequent source phrases, this is expensive. They solve this problem by extracting only from a sample of the found source phrases, capping the sample size at 100. For less frequent source phrases, all possible targets are extracted.

Fast scoring using maximum likelihood. Scoring the phrase pairs is linear in the number of pairs if we use the maximum likelihood estimate $p(e|f)$ for source phrase f and target phrase e. Since each source phrase only has small number of targets (up to the sample limit), this step is fast. However, notice that we cannot easily compute the target-to-source probability $p(f|e)$ as is commonly done. We address this in §4.1.

Callison-Burch et al. (2005) and Zhang and Vogel (2005) found that these steps added only tenths of a second to per-sentence decoding time, which we independently confirmed (Lopez, 2008, Chapter 3). Their techniques also apply to discontiguous phrases, except for the pattern matching algorithm, which only works for contiguous phrases. Since our model (and many others) uses discontiguous phrases, we must use a different algorithm. The most straightforward way to accomplish this is to use the fast suffix array lookup for the contiguous subphrases of a discontiguous phrase, and then to combine the results. Suppose that we have substrings u and v, and a gap character X which can match any arbitrary sequence of words. Then to look up the phrase uXv, we first find all occurrences of u, and all occurrences of v. We can then compute all cases where an occurrence of u precedes and occurrence of v in the same sentence.

The complexity of this last step is linear in the number of occurrences of u and v. If either u or v is very frequent, this is too slow. Lopez (2007; 2008) solves this with a series of empirically fast exact algorithms. We briefly sketch the solution here; for details see Lopez (2008, Chapter 4).

Lossless pruning. For each phrase, we only search if we have already successfully found both its longest suffix and longest prefix. For example, if a, b, c, and d are all words, then we only search for phrase $abXcd$ if we have already found occurrences of phrases $abXc$ and $bXcd$.

Precomputation of expensive searches. For phrases containing multiple very frequent subphrases, we precompute the list of occurrences into an inverted index. That is, if both u and v are frequent, we simply precompute all locations of uXv and vXu.

Fast merge algorithm. For phrases pairing a frequent subphrase with infrequent subphrases, we use a merge algorithm whose upper bound complexity is logarithmic in the number of occurrences of the frequent subphrase. That is, if $count(u)$ is small, and $count(v)$ is big, then we can find uXv in at most $O(count(u) \cdot \log(count(v)))$ time.

Our implementation is a fast extension to the Hiero decoder (Chiang, 2007), written in Pyrex.[2] It is an order of magnitude faster than the Python implementation of Lopez (2007). Pattern matching, extraction, and scoring steps add approximately 2 seconds to per-sentence decoding time, slowing decoding by about 50% compared with a conventional exact model representation using external prefix trees (Zens and Ney, 2007). See Lopez (2008, Chapter 4) for analysis.

4 Experiments

Although the algorithmic issues of translation by pattern matching are largely solved, none of the previous work has reported any improvements in

[2]Pyrex combines Python and C code for performance. http://www.cosc.canterbury.ac.nz/greg.ewing/python/Pyrex/

state of the art with very large models.[3] In the remainder of this work, we scratch the surface of possible uses.

We experimented on Chinese-English newswire translation. Except where noted, each system was trained on 27 million words of newswire data, aligned with GIZA++ (Och and Ney, 2003) and symmetrized with the grow-diag-final-and heuristic (Koehn et al., 2003). In all experiments that follow, each system configuration was independently optimized on the NIST 2003 Chinese-English test set (919 sentences) using minimum error rate training (Och, 2003) and tested on the NIST 2005 Chinese-English task (1082 sentences). Optimization and measurement were done with the NIST implementation of case-insensitive BLEU 4n4r (Papineni et al., 2002).[4]

4.1 Baseline

We compared translation by pattern matching with a conventional exact model representation using external prefix trees (Zens and Ney, 2007). To make model computation efficient for the latter case, we followed the heuristic limits on phrase extraction used by Chiang (2007).

- Phrases were restricted to five words. Each gap character counts as a single word regardless of how many actual words it spans. Thus phrase aXb consisting of words a and b separated by a gap is three words.
- Phrases were restricted to a span of ten words in the training data.
- Phrases were restricted to two gaps.
- Gaps were required to span at least two words in the training data.
- Phrases were extracted using a tight heuristic.

Chiang (2007) uses eight features, so we incorporate these into the conventional baseline. However, as discussed previously in §3, the pattern matching architecture makes it difficult to compute the target-to-source translation probability, so this feature is not included in the pattern matching system. This may not be a problem—Och and Ney (2002) observed that this feature could be replaced by the source-to-target probability without loss of accuracy. Preliminary experiments suggested that two other features in Chiang's model based on rule counts were not informative, so we considered a model containing only five features.

1. Sum of logarithms of source-to-target phrase translation probabilities.
2. Sum of logarithms of source-to-target lexical weighting (Koehn et al., 2003).
3. Sum of logarithms of target-to-source lexical weighting.
4. Sum of logarithms of a trigram language model.
5. A word count feature.

The sample size (see §3) for the pattern matching system was 300.[5] Results show that both translation by pattern matching and reduced feature set are harmless to translation accuracy (Table 3).

4.2 Rapid Prototyping via Pattern Matching

Even in this limited experiment, translation by pattern matching improved experimental turnaround. For the conventional system, it took 10.8 hours to compute the full model, which required 6.1GB of space. For the pattern matching system, it took only 8 minutes to compute all data structures and indexes. These required only 852MB of space.

There are two tradeoffs. First, memory use is higher in the pattern matching system, since the conventional representation resides on disk. Second, per-sentence decoding time with the pattern matching system is slower by about 2 seconds due to the expense of computing rules and parameters on demand. Even so, the experimental turnaround time with the pattern matching system was still faster. We would need to decode nearly 20,000 sentences with it to equal the computation time of conventional model construction. We might need to decode this many times during MERT, but only because it decodes the same test set many times. However, it is straightforward to extract all rules for the development set using pattern matching, and use them in a conventional system for MERT.

[3] Zhang and Vogel (2005) report improvements, but all of their results are far below state of the art for the reported task. This may be because their system was not tuned using minimum error rate training (Och, 2003).

[4] ftp://jaguar.ncsl.nist.gov/mt/resources/mteval-v11b.pl

[5] We use deterministic sampling, which is useful for reproducibility and for minimum error rate training (Och, 2003). See Lopez (2008, Chapter 3) for details.

System	BLEU
Conventional (eight features)	30.7
Conventional (five features)	30.6
Pattern matching (five features)	30.9

Table 3: Baseline system results.

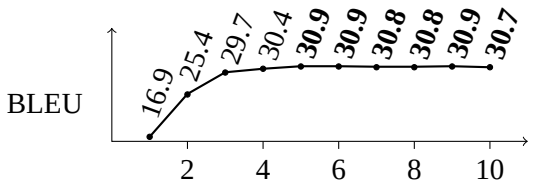

Figure 1: Effect of the maximum phrase length.

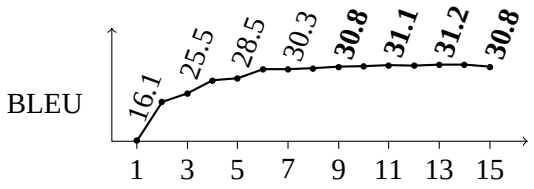

Figure 2: Effect of the maximum phrase span.

Taking our five-feature pattern matching system as a starting point, we next considered several ways in which we might scale up the translation model. Here the benefits of prototyping become more apparent. If we were to run the following experiments in a conventional system, we would need to compute a new model for each condition. With translation by pattern matching, nearly every variant uses the same underlying representation, so it was rarely even necessary to recompute data structures and indexes.

4.3 Relaxing Length Restrictions

Increasing the maximum phrase length in standard phrase-based translation does not improve BLEU (Koehn et al., 2003; Zens and Ney, 2007). However, this effect has not yet been evaluated in hierarchical phrase-based translation.

We experimented with two analogues to the maximum phrase length. First, we varied the limit on source phrase length (counting each gap as a single word), the closest direct analogue. Chiang (2007) used a limit of five. We found that accuracy plateaus at this baseline setting (Figure 1). Second, we varied the limit on the span of phrases extracted from the training text. Suppose we are interested in a source phrase uXv. If u and v are collocated in a training sentence, but within a span longer than the limit, then the model is prevented from learning a rule that translates this discontiguous phrase as a single unit. Chiang (2007) fixes this limit at ten. Again, accuracy plateaus near the baseline setting (Figure 2).

Our results are similar to those for conventional phrase-based models (Koehn et al., 2003; Zens and Ney, 2007). Though scaling along these axes is unhelpful, there is still a large space for exploration.

4.4 Interlude: Hierarchical Phrase-Based Translation versus Lexical Reordering

In a related line of inquiry, we considered the effect of increasing the number of gaps in phrases. Chiang (2007) limits them to two. Although we consider more than this, we also considered fewer, to gain insight into the hierarchical model.

Hierarchical phrase-based translation is often reported to be better than conventional phrase-based translation, but the actual reason for this is unknown. It is often argued that the ability to translate discontiguous phrases is important to modeling translation (Chiang, 2007; Simard et al., 2005; Quirk and Menezes, 2006), and it may be that this explains the results. However, there is another hypothesis. The model can also translate phrases in the form uX or Xu (a single contiguous unit and a gap). If it learns that uX often translates as Xu', then in addition to learning that u translates as u', it has also learned that u switches places with a neighboring phrase during translation. This is similar to **lexicalized reordering** in conventional phrase-based models (Tillman, 2004; Al-Onaizan and Papineni, 2006).[6] If this is the real benefit of the hierarchical model, then the ability to translate discontiguous phrases may be irrelevant.

To tease apart these claims, we make the following distinction. Rules in which both source and target phrases contain a single contiguous element—that is, in the form u, Xu, uX, or XuX—encode lexicalized reordering in hierarchical form. Rules representing the translation of discontiguous units—minimally uXv—encode translation knowledge that is strictly outside the purview of lexical reordering.

We ran experiments varying both the number of contiguous subphrases and the number of gaps (Ta-

[6] This hypothesis was suggested independently in personal communications with several researchers, including Chris Callison-Burch, Chris Dyer, Alex Fraser, and Franz Och.

ble 4). For comparison, we also include results of the phrase-based system Moses (Koehn et al., 2007) with and without lexicalized reordering.

Our results are consistent with those found elsewhere in the literature. The strictest setting allowing no gaps replicates a result in Chiang (2007, Table 7), with significantly worse accuracy than all others. The most striking result is that the accuracy of Moses with lexicalized reordering is indistinguishable from the accuracy of the full hierarchical system. Both improve over non-lexicalized Moses by about 1.4 BLEU. The hierarchical emulation owes its performance only partially to lexicalized reordering. Additional improvement is seen when we add discontiguous phrases. That the effect of lexicalized reordering is weaker in the hierarchical model is unsurprising, since its parameterization is much simpler than the one used by the Moses, which includes several specialized features for this purpose. This suggests that the hierarchical model could be improved through better parameterization, and still benefit from the translation of discontiguous phrases.

Finally, we observe that using more gaps does not improve the hierarchical model.

Subphrases	Gaps	Example	BLEU
1	0	u	26.3
1	1	uX, Xu	30.2*
1	2	XuX	30.0*
2	1	uXv	30.5
2	2	$uXvX, XuXv$	**30.8**
3	2	$uXvXw$	**30.9**
4	3	$uXvXwXy$	**30.9**
5	4	$uXvXwXyXz$	**30.8**
Moses without lexicalized reordering			29.4
Moses with lexicalized reordering			**30.7**

Table 4: Comparison with Moses and effect of the maximum number of subphrases and gaps. *Denotes emulation of lexicalized reordering.

4.5 The Tera-Scale Model

These are interesting scientific findings, but we have so far failed to show an improvement over the baseline. For this, we return to the tera-scale model of §2. Recall that in this model, we modify the baseline by adding 80 million words of UN data and using sparse maximum entropy alignments with a loose phrase extraction heuristic.

To avoid conflating rules learned from in-domain newswire and out-of-domain UN data, we treat each corpus independently. We sample from up to 300 source phrase occurrences from each, and compute lexical weighting and the source-to-target phrase translation probabilities separately for both samples. For the UN corpus, the resulting probabilities are incorporated into three new features. These features receive a value of zero for any rule computed from the newswire data. Likewise, the baseline source-to-target phrase translation probability and lexical weighting features receive a value of zero for rules computed from the UN data.

We make one more modification to the model that is quite easy with pattern matching. We notice that it is not always possible to extract a translation for a source phrase occurrence, even under the loose heuristic. This is because there may be no consistently aligned target phrase according to the alignment. If a phrase occurs frequently but we can only rarely extract a translation for it, then our confidence that it represents a natural unit of translation should diminish. Conversely, if we usually extract a translation, then the phrase is probably a good unit of translation. We call this property **coherence**. Conventional offline extraction methods usually ignore coherence. If a phrase occurs many times but we can only extract a translation for it a few times, then those translations tend to receive very high probabilities, even though they might simply be the result of noisy alignments.

We can incorporate the notion of coherence directly into the phrase translation probability. In the baseline model, the denominator of this probability is the sum of the number of rule occurrences containing the source phrase, following Koehn et al. (2003).[7] We replace this with the number of attempted extractions. This parameterization may interact nicely with the loose extraction heuristic, reducing the probability of many greedily extracted but otherwise noisy phrases.

We compared the baseline with our tera-scale model. Since we had already performed substantial experimentation with the NIST 2005 set, we also included the NIST 2006 task as a new held-out test set. Results including variants produced by ablating a single modification on the development set are given in Table 5.

We also compared our modified system with an augmented baseline using a 5-gram language

[7] Or the sample size, whichever is less.

System	NIST 2005 BLEU	loss	NIST 2006 BLEU
Tera-Scale Model (all modifications)	32.6	–	28.4
with grow-diag-final-and instead of maximum entropy alignment	32.1	-0.5	
with tight extraction heuristic instead of loose	31.6	-1.0	
without UN data	31.6	-1.0	
without separate UN features	32.2	-0.4	
with standard $p(f\|e)$ instead of coherent $p(f\|e)$	31.7	-0.9	
Baseline (conventional)	30.7	-1.9	
Baseline (pattern matching)	30.9	-1.7	27.0

Table 5: Results of scaling modifications and ablation experiments.

model and rule-based number translation. The objective of this experiment is to ensure that our improvements are complementary to better language modeling, which often subsumes other improvements. The new baseline achieves a score of 31.9 on the NIST 2005 set, making it nearly the same as the state-of-the-art results reported by Chiang (2007). Our modifications increase this to 34.5, a substantial improvement of 2.6 BLEU.

5 Related Work and Open Problems

There are several other useful approaches to scaling translation models. Zens and Ney (2007) remove constraints imposed by the size of main memory by using an external data structure. Johnson et al. (2007) substantially reduce model size with a filtering method. However, neither of these approaches addresses the preprocessing bottleneck. To our knowledge, the strand of research initiated by Callison-Burch et al. (2005) and Zhang and Vogel (2005) and extended here is the first to do so. Dyer et al. (2008) address this bottleneck with a promising approach based on parallel processing, showing reductions in real time that are linear in the number of CPUs. However, they do not reduce the overall CPU time. Our techniques also benefit from parallel processing, but they reduce overall CPU time, thus comparing favorably even in this scenario.[8] Moreover, our method works even with limited parallel processing.

Although we saw success with this approach, there are some interesting open problems. As discussed in §4.2, there are tradeoffs in the form of slower decoding and increased memory usage. Decoding speed might be partially addressed using a mixture of online and offline computation as in Zhang and Vogel (2005), but faster algorithms are still needed. Memory use is important in non-distributed systems since our data structures will compete with the language model for memory. It may be possible to address this problem with a novel data structure known as a compressed self-index (Navarro and Mäkinen, 2007), which supports fast pattern matching on a representation that is close in size to the information-theoretic minimum required by the data.

Our approach is currently limited by the requirement for very fast parameter estimation. As we saw, this appears to prevent us from computing the target-to-source probabilities. It would also appear to limit our ability to use discriminative training methods, since these tend to be much slower than the analytical maximum likelihood estimate. Discriminative methods are desirable for feature-rich models that we would like to explore with pattern matching. For example, Chan et al. (2007) and Carpuat and Wu (2007) improve translation accuracy using discriminatively trained models with contextual features of source phrases. Their features are easy to obtain at runtime using our approach, which finds source phrases in context. However, to make their experiments tractable, they trained their discriminative models offline only for the specific phrases of the test set. Combining discriminative learning with our approach is an open problem.

6 Conclusion

We showed that very large translation models present an interesting engineering challenge, and illustrated a solution to this challenge using pattern matching algorithms. This enables practical, rapid exploration of vastly larger models than those currently in use. We believe that many other improvements are possible when the size of our models is

[8] All of our reported decoding runs were done in parallel.

unconstrained by resource limitations.

References

Yaser Al-Onaizan and Kishore Papineni. 2006. Distortion models for statistical machine translation. In *Proc. of ACL-COLING*, pages 529–536, Jul.

Necip Fazil Ayan and Bonnie Dorr. 2006a. Going beyond AER: An extensive analysis of word alignments and their impact on MT. In *Proc. of ACL-COLING*, pages 9–16, Jul.

Necip Fazil Ayan and Bonnie J. Dorr. 2006b. A maximum entropy approach to combining word alignments. In *Proc. of HLT-NAACL*, pages 96–103, Jun.

Ralf D. Brown. 2004. A modified Burrows-Wheeler transform for highly-scalable example-based translation. In *Proc. of AMTA*, number 3265 in LNCS, pages 27–36. Springer, Sep.

Chris Callison-Burch, Colin Bannard, and Josh Schroeder. 2005. Scaling phrase-based statistical machine translation to larger corpora and longer phrases. In *Proc. of ACL*, pages 255–262, Jun.

Marine Carpuat and Dekai Wu. 2007. Improving statistical machine translation using word sense disambiguation. In *Proc. of ACL*, pages 61–72, Jun.

Yee Seng Chan, Hwee Tou Ng, and David Chiang. 2007. Word sense disambiguation improves statistical machine translation. In *Proc. of ACL*, pages 33–40, Jun.

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

Steve DeNeefe, Kevin Knight, Wei Wang, and Daniel Marcu. 2007. What can syntax-based MT learn from phrase-based MT? In *Proc. of EMNLP-CoNLL*, pages 755–763, Jun.

Chris Dyer, Aaron Cordova, Alex Mont, and Jimmy Lin. 2008. Fast, easy, and cheap: Construction of statistical machine translation models with mapreduce. In *Proc. of the Workshop on Statistical Machine Translation*, pages 199–207, Jun.

Howard Johnson, Joel Martin, George Foster, and Roland Kuhn. 2007. Improving translation quality by discarding most of the phrasetable. In *Proc. of EMNLP-CoNLL*, pages 967–975, Jun.

Philipp Koehn, Franz Josef Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *Proc. of HLT-NAACL*, pages 127–133, May.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondrej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *Proc. of ACL Demo and Poster Sessions*, pages 177–180, Jun.

Philipp Koehn. 2005. Europarl: A parallel corpus for statistical machine translation. In *Proc. of MT Summit*.

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

Adam Lopez. 2008. *Machine Translation by Pattern Matching*. Ph.D. thesis, University of Maryland, Mar.

Udi Manber and Gene Myers. 1993. Suffix arrays: A new method for on-line string searches. *SIAM Journal of Computing*, 22(5):935–948.

Gonzalo Navarro and Veli Mäkinen. 2007. Compressed full-text indexes. *ACM Computing Surveys*, 39(1), Apr.

Franz Josef Och and Hermann Ney. 2002. Discriminative training and maximum entropy models for machine translation. In *Proc. of ACL*, pages 156–163, Jul.

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

Franz Josef Och. 2003. Minimum error rate training in statistical machine translation. In *Proc. of ACL*, Jul.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: a method for automatic evaluation of machine translation. In *Proc. of ACL*, pages 311–318, Jul.

Chris Quirk and Arul Menezes. 2006. Do we need phrases? Challenging the conventional wisdom in statistical machine translation. In *Proc. of HLT-NAACL*, pages 8–16, Jun.

Michel Simard, Nicola Cancedda, Bruno Cavestro, Marc Dymetman, Eric Gaussier, Cyril Goutte, Kenji Yamada, Philippe Langlais, and Arne Mauser. 2005. Translating with non-contiguous phrases. In *Proc. of HLT-EMNLP*, pages 755–762, Oct.

Christoph Tillman. 2004. A unigram orientation model for statistical machine translation. In *Proc. of HLT-NAACL: Short Papers*, pages 101–104, May.

Richard Zens and Hermann Ney. 2007. Efficient phrase-table representation for machine translation with applications to online MT and speech translation. In *Proc. of HLT-NAACL*.

Ying Zhang and Stephan Vogel. 2005. An efficient phrase-to-phrase alignment model for arbitrarily long phrase and large corpora. In *Proc. of EAMT*, May.