Manageable Phrase-based Statistical Machine Translation Models

Badr, Ghada; Joanis, Eric; Larkin, Samuel; Kuhn, Roland

Publisher’s version / Version de l'éditeur:
https://doi.org/10.1007/978-3-540-75175-5_55
Computer Recognition Systems 2 (Advances in Intelligent and Soft Computing, vol. 45), pp. 437-444, 2007

Manageable Phrase-based Statistical Machine Translation Models

Ghada Badr[1] Eric Joanis Samuel Larkin Roland Kuhn

Institute for Information Technology
National Research Council of Canada (NRC)
{GhadaHany.Badr,Eric.Joanis,Samuel.Larkin,Roland.Kuhn}@nrc.gc.ca

Summary. Statistical Machine Translation (SMT) is an evolving field where many techniques in Syntactic Pattern Recognition (SPR) are needed and applied. A typical phrase-based SMT system for translating from a T (target) language to an S (source) language contains one or more n-gram language models (LMs) and one or more phrase translation models (TMs). These LMs and TMs have a large memory footprint (up to several gigabytes). This paper describes novel techniques for filtering these models that ensure only relevant patterns in the LMs and TMs are loaded during translation. In experiments on a large Chinese-English task, these techniques yielded significant reductions in the amount of information loaded during translation: up to 58% reduction for LMs, and up to 75% for TMs.

1 Introduction

1.1 Background

This paper focuses on efficient filtering techniques for the components that take up the largest part of SMT system memory: the LMs and TMs. Though the paper's experimental results pertain to a particular SMT system, this system is typical of a wide range of SMT systems, such as [7, 8, 10, 12, 13]. Each LM component of an SMT system returns the probability of a word sequence in the T-language; we use n-gram LMs, which estimate the conditional probability that the ith word follows the preceding $i-1$ words [5]. We often employ several different LMs, each trained on a unilingual T corpus, and several TMs, each trained on a bilingual corpus. Each TM is stored in a "phrase table": a list of triples (s,t,m) where s is a contiguous sequence of S words, t is a contiguous sequence of T words, and m measures the strength of the association between them. These triples are extracted from a parallel bilingual corpus using the "DiagAnd" method described in [9] with IBM 2 alignment. Each LM and TM receives a loglinear weight optimized by the algorithm in [11]. During decoding, the decoder matches subsequences of the input S-language sentence in the TMs. A translation hypothesis is formed by concatenating T phrases from such matching phrase pairs [8]. Every hypothesis generated by the system for an input sentence contains only words that appear in at least one t such that (s,t) is a phrase pair in a TM. Thus, given the input sentence and the TMs, one knows which T-language words in the system's vocabulary cannot possibly appear in the translation hypotheses. LMs and TMs are usually represented as *tries* [6] in memory.

Work on minimizing SMT memory requirements includes pruning of TMs for decoding based on a rudimentary analysis of the input sentences [8]; here, we show how to further prune the set of loaded phrase pairs by considering the relationship between different TMs. The work most closely related to ours for LM pruning involves a bag-of-words algorithm for the input S document [7, 10, 13]. The algorithm, "Document-Vocabulary LM Filtering" (*Doc-Voc-LM*), puts into a "bag" all and only the T words that could be used for translation, based on all sentences in the document and the phrase pairs in the TMs; an n-gram will only be loaded if all the words in it are in this "bag". A related technique has been applied to SMT rescoring [7].

This paper introduces three new techniques for "batch mode" translation of a document: "Sentence-Vocabulary LM Filtering" (*Sent-Voc-LM*) and "Sentence-Phrase LM Filtering" (*Sent-Phrase-LM*) for filtering LMs, and "Limit-TM" for filtering TMs. In the standard *Doc-Voc-LM* filtering approach, an n-gram from the LM will be loaded if all words in it come from the bag of target words for the input document. By contrast, *Sent-Voc-LM* creates a separate bag of words for each sentence in the document, and n-grams are only loaded if all words in them are in the current sentence's bag. *Sent-Phrase-LM* goes further, by prohibiting n-grams that cannot be generated from the phrases associated with the current sentence - e.g., if xyz is the only phrase with target word y associated with this sentence, then even if word w occurs in other phrases associated with the sentence, *Sent-Phrase-LM* will not load n-grams wy and yw. Initial experiments with *Sent-Voc-LM* and *Sent-Phrase-LM* yielded only a small reduction in the number of n-grams loaded from the LM. However, study of the relationship between multiple TMs used by the system yielded *Limit-TM*, which finds a much smaller set of phrases associated with each input sentence (without loss of information). Since this results in a smaller bag of target words per sentence, *Sent-Voc-LM* and *Sent-Phrase-LM* performed very well when *Limit-TM* was applied. Thus, our system can now load larger models and therefore generate better translations.

2 Language Model Filtering

We now explain *Sent-Voc-LM* and *Sent-Phrase-LM* in more detail.

2.1 Sent-Voc-LM

As mentioned above, one way of filtering the LM is to apply *Doc-Voc-LM*. However, as the number of S sentences in the input document increases, the

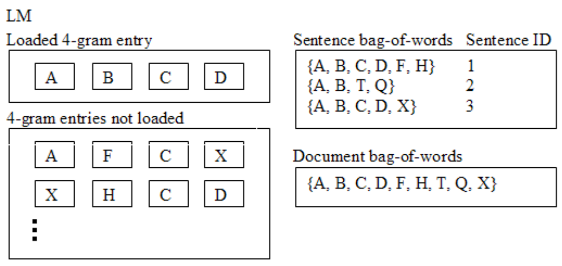

Fig. 1. Example of n-grams loaded by Doc-Voc-LM but not by Sent-Voc-LM.

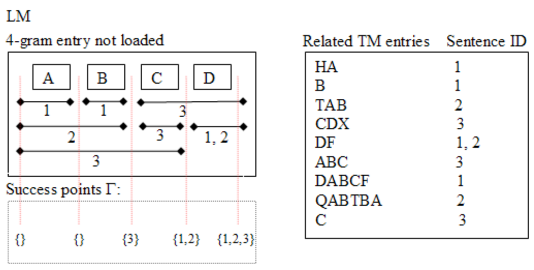

Fig. 2. Example of an n-gram loaded by Sent-Voc-LM but not by Sent-Phrase-LM.

size of the bag of words increases, and the ability to filter LMs decreases. One could load separate LMs for each S-sentence, but that would take too long; it is better to load LMs once for a group of sentences.

The *Sent-Voc-LM* filter uses per-sentence bags of words: each bag contains only the T words that might be needed to translate one S sentence in the input[1]. Only n-grams that consist of words that are all found together in at least one bag are loaded. For unigrams, this is no different from *Doc-Voc-LM*, but for bigrams and higher order n-grams, it makes a big difference. *E.g.*, we no longer load a bigram (x, y) when x is found only in the bag for sentence i, and y is only in the bag for a different sentence j. Figure 1 shows how *Sent-Voc-LM* can filter out n-grams that would be loaded by *Doc-Voc-LM*. Here, 4-grams "AFCX" and "XHCD" will be allowed by the latter (since they are made of words from the document bag-of-words) but not by the former (since they do not come from a per-sentence bag-of-words); "ABCD" will be loaded by both.

2.2 Sent-Phrase-LM

(*Sent-Phrase-LM*) filters out even more out n-grams from the LM by prohibiting those that cannot be generated by joining phrases from the TM (even

[1] The out-of-vocabulary words in S are part of each per-sentence bag of words to ensure that the corresponding n-gram entries will be loaded.

Table 1. *Limit-TM* Example with two TMs.

Phrase	TM_1 score	TM_2 score	*Limit-TM* decision
a	-1	-1	keep
b	-2	-10	keep
c	-3	-3	keep
d	-10	-2	keep
e	-2.5	-9	keep
f	-3.5	-4	reject

when all the words in the n-gram are in the same per-sentence bag of words). In Figure 2, the n-gram "ABCD" would be loaded by *Doc-Voc-LM* and *Sent-Voc-LM*, but is filtered out by *Sent-Phrase-LM* because it cannot be generated by combining phrases for the same sentence.

There are three ways that an n-gram can appear in a translation for a given sentence: the n-gram appears inside a phrase, the n-gram is formed by a suffix[2] of one phrase and a prefix of the phrase that follows it in the translation, or the n-gram is formed by a suffix of one phrase, then one or more complete phrases, and then a prefix of the following phrase. To implement *Sent-Phrase-LM* efficiently, we used suffix-trees [1, 4]. The suffix-tree stores all possible T phrases including pre-/in-/suffixes of these phrases. Each node in the tree is marked to indicate whether it is a complete phrase, a prefix, a suffix, or an infix of a phrase.

We implemented the *Sent-Phrase-LM* criterion using dynamic programming [3]. The algorithm uses a Γ vector to tabulate the success points in each n-gram that can be generated, and uses the suffix-tree to retrieve the complete and pre-/in-/suffix sets. The algorithm requires at most $O(n)$ lookups in the suffix-tree to generate all success point sets. n is usually small $(3-6)$, so it requires only a few lookups and some set operations to decide if an n-gram should be kept. For details, see [2].

3 Translation Model Filtering

There are two reasons for filtering TMs: to reduce the memory taken up by the TMs themselves, and to reduce the vocabulary to make LM filtering more effective. In this section, we propose an efficient algorithm for filtering TMs. We begin with a reasonable baseline that keeps only phrase pairs whose S-language phrases occur in at least one of the sentences to be translated. This can be done efficiently by storing all S-language phrases from the source sentences in a *trie* [6] data structure. We call this technique, already used by most SMT systems, *Grep-TM*.

[2] we assume that a phrase is a prefix, a suffix, and an infix of itself.

3.1 *Limit-TM*

Limit-TM goes beyond *Grep-TM* by considering relationships between phrase pairs in different TMs. In a TM, we may have many T phrases for a given S phrase. During decoding, most SMT systems limit the number of T phrases, considering only the L best ones for a given S phrase. With only one TM, for each S-language phrase, one thus keeps only the L T phrases with the highest score. However, one usually has more than one TM. *Limit-TM* chooses the L T-language phrases to be kept by analyzing their scores in all TMs and the weights with which they may be combined.

When optimizing decoder weights, we decode the same sentences many times with different weights. Therefore, we want *Limit-TM* to filter TMs ahead of time in a way that will be correct regardless of the TM weights (for fixed L). Assume the weights are non-negative. Given a T phrase, tp_i, if there are L other T phrases that are "better" than tp_i according to all TMs, then no matter what the weights, tp_i will remain behind those L other T phrases and will never be used during decoding. In this case, tp_i could be filtered out from all TMs. For example, in Table 1 where we have two TMs and $L = 2$, a will have the top score among those phrases. Which phrase comes second, however, depends on the weights: it will be b in TM$_1$, c, or d in TM$_2$, and therefore it must always be kept. But e will never be among the top two phrases: a will always have the highest score, and then one of b or c will always be better than e. However, e will not be filtered by our criterion because detecting that can require expensive comparisons involving multiple phrase table entries. Only f can be filtered out, since a and c are better than f in both tables; typically, many more entries like f can be filtered out. This example shows that *Limit-TM* is sufficient but not necessary. The proof can be found in [2].

Limit-TM can be implemented efficiently using dynamic programming [3]. With L defined as before, let n be the number of TMs, I be the number of unique target phrases across all TMs for a given S-language phrase. In the worst case the algorithm can take up to $O(I \log(I))$ for sorting plus $O(I^2N)$ for *Limit-TM* itself, but on average $O(ILN)$. Empirically, we found that this algorithm runs reasonably fast even for large values of I. For details, see [2].

4 Experimental Results

4.1 Data Set and Models

These techniques were applied to our Chinese-English SMT system. The TM training corpora were those distributed for the NIST MT06 Chinese evaluation, large-data track (`http://www.nist.gov/speech/tests/mt`). For LMs, we additionally used the English Gigaword corpus (LDC2005T12). The test corpus consists of the 919 variable-length Chinese sentences from the Multiple Translation Chinese Corpus, Part 4 (LDC2006T04).

Table 2. TM sizes in MBs, total time in minutes (loading, filtering and writing), total vocabulary size (919 sentences), and average vocabulary size per sentence. $L = 30$.

	Multi-Prob-TM	Grep-TM	Limit-TM
TM	958M	122M	29M
Time	57m	14m	68m
Total vocabulary	1,783,641	268,375	69,644
Average vocabulary	-	11,891	1,581

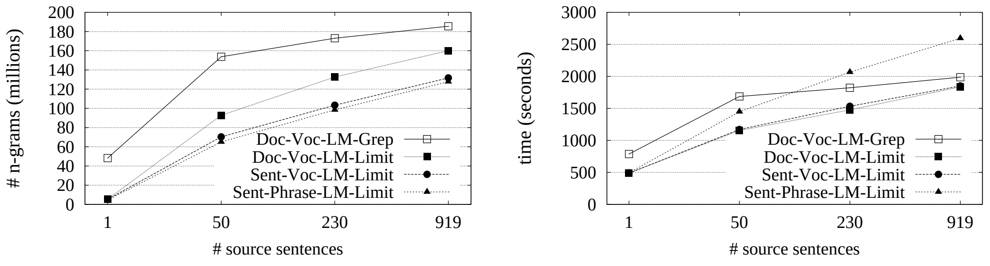

Fig. 3. n-gram counts (left) and total loading times (right) for *Giga-4g-LM* as function of no. of source Chinese sentences for different techniques.

We tested the proposed LM and TM filters using the two different LMs and five different TMs included in our highest-scoring NIST06 Chinese-English system. There were two training corpora for LMs: one has 3.2M sentences and the other (Gigaword) has 13M Sentences. For each LM training corpus, we generated 3-, 4-, and 5-gram models. For instance, the Giga corpus yields a 4-gram LM of size 2,000MB and a 5-gram LM of size 3,200MB. We trained the five TMs on a total of 8.2M sentences. We then combined all of them in one table with multiple probabilities: *Multi-Prob-TM* (958MB). In this table, each phrase pair is associated with its probabilities from various TMs. Phrases missing from a TM are given a small, arbitrary probability value.

4.2 Experimental Setup and Results

As a baseline for TM filtering, the *Multi-Prob-TM* was filtered using the *Grep-TM* technique. We then applied the *Limit-TM* technique to *Grep-TM* output. For each LM, we created two baselines using *Doc-Voc-LM*. The first baseline, *Doc-Voc-LM-Grep*, was created by using the vocabulary collected from *Grep-TM*, and the second one, *Doc-Voc-LM-Limit*, was created using vocabulary collected from *Limit-TM*.

Experiments were conducted to compare *Sent-Voc-LM* and *Sent-Phrase-LM* with *Doc-Voc-LM-Grep*, in the case where the only TM filtering is *Grep-TM*. Results did not show a big reduction in LM sizes in this case. These disappointing results were not unexpected: Table 2 shows that the average size of vocabulary per sentence was $11,891$! Note (below) that we get much better results when combining LM filtering with *Limit-TM*. In fact, these LM filtering techniques can be effectively combined with any TM filtering technique that shrinks average per-sentence vocabulary.

Our TM filtering experiments compared *Limit-TM* with *Grep-TM*: Table 2 shows that *Limit-TM* reduces the combined TM size by more than 75% when the limit, L, is set to 30[3]. The table also shows that *Limit-TM* reduces the average per-sentence vocabulary by nearly an order of magnitude, greatly improving the effectiveness of LM filtering. The table also shows processing time.

Finally, we carried out experiments to measure the impact of LM filtering for the two kinds of TM filtering. Figure 3 (left) shows the number of n-grams in millions after applying the techniques on *Giga-4g-LM*, as a function of number of sentences. The result shown here is typical of most n-gram LMs. Except for the "Doc-Voc-LM-Grep" line, results were based on TMs that were filtered using *Limit-TM*. The top two lines show that *Limit-TM* significantly reduces LM requirements (*Doc-Voc-LM-Limit* uses up to 40% fewer 4-grams than *Doc-Voc-LM-Grep*). The figure also shows that *Sent-Voc-LM-Limit* performs much better with *Limit-TM*. For example, for Gigaword 4-gram LM and 50 source sentences, *Sent-Voc-LM-Limit* uses about 54% fewer 4-grams than *Doc-Voc-LM-Grep*, and about 24% fewer than *Doc-Voc-LM-Limit*. Under the same conditions, *Sent-Phrase-LM-Limit*[4] uses about 58% fewer 4-grams than *Doc-Voc-LM-Grep*, and about 30% less 4-grams than *Doc-Voc-LM-Limit*. The time requirement is shown in Figure 3 (right). *Sent-Voc-LM-Limit* combines effective filtering and speed. It can be done on the fly while loading models for decoding, and slightly speeds up decoding (evidence not shown here).

5 Conclusions

In this paper, we described three techniques for reducing the size of SMT models in memory by loading only entries that are needed for translation. This will make it possible to use larger models, yielding improved translations. Two LM filtering techniques, *Sent-Voc-LM* and *Sent-Phrase-LM*, were proposed; they work best with a small per-sentence vocabulary. A TM filtering technique, *Limit-TM*, was presented to reduce the size of the TMs. *Limit-TM* greatly reduces the T-language vocabulary size, which in turn improves LM filtering. The combined techniques showed reductions in LM sizes by up to 58%. *Limit-TM* showed a remarkable reduction in the joint TM size by up to 75%. These techniques enabled us for the first time to find room in memory for 5-gram LMs, which measurably improved the quality of the systems output.

[3] We tested our decoder with several values for L; $L = 30$ was optimal.

[4] *Sent-Phrase-LM* performance strongly depends on the training data - it should be tried on other data.

References

1. A. Andersson and S. Nilsson. Efficient implementation of suffix trees. *SOFT-PREX: Software–Practice and Experience*, 25(2):129–141, 1995.
2. G. Badr, E. Joanis, and S. Larkin. Manageable phrase-based statistical machine translation models. Technical report nrc 49304, 2007.
3. T. H. Cormen, C.E. Leiserson, and R.L. Rivest. *Introduction to Algorithms*. The MIT Press, Cambridge, MA, 1990.
4. M. Fuketa, T. Sumitomo, M. Shishibori, and J. Aoe. A suffix compression algorithm of tries. *ICCPOL'99: 18th International Conference on Computer Processing of Original Languages*, 18:345–348, 1999.
5. J. T. Goodman. A bit of progress in language modeling. Technical Report MSR-TR-2001-72, 2001.
6. P. Jacquet and W. Szpankowski. Analysis of digital tries with Markovian dependency. *IEEE Trans. Information Theory, IT-*, 37(5):1470–1475, 1991.
7. K. Kirchhoff, K. Duh, and C. Lim. The University of Washington machine translation system for IWSLT 2006. In *Proc. of the International Workshop on Spoken Language Translation*, pages 145–152, Kyoto, Japan, 2006.
8. P. Koehn. Pharaoh: a beam search decoder for phrase-based statistical machine translation models. Washington DC, 2004.
9. P. Koehn, F. J. Och, and D. Marcu. Statistical phrase-based translation. In *Proceedings of the Human Language Technology Conf. (HLT-NAACL)*, pages 127–133, Edmonton, Canada, June 2003.
10. A. Mauser, R. Zens, E. Matusov, S. Hasan, and H. Ney. The RWTH statistical machine translation system for the IWSLT 2006 evaluation. In *Proc. of the International Workshop on Spoken Language Translation*, pages 103–110, Kyoto, Japan, November 2006.
11. F. J. Och and H. Ney. Discriminative training and maximum entropy models for statistical machine translation. In *Proceeding of the 40th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 295–302, Philadelphia, PA, July 2002.
12. F. Sadat, H. Johnson, A. Agbago, G. Foster, R. Kuhn, J. Martin, and A. Tikuisis. Portage: A phrase-based machine translation system. In *The Association for Computational Linguistics (ACL) 2005 Workshop on Building and Using Parallel Texts: Data-Driven Machine Translation and Beyond*, pages 133–136, Ann Arbor, Michigan, USA, 2005.
13. Wade Shen, Richard Zens, Nicola Bertoldi, and Marcello Federico. The JHU workshop 2006 IWSLT system. In *Proc. of the International Workshop on Spoken Language Translation*, pages 59–63, Kyoto, Japan, 2006.