Are Very Large N-best Lists Useful for SMT?

Saša Hasan, Richard Zens, Hermann Ney
Human Language Technology and Pattern Recognition
Lehrstuhl für Informatik 6 – Computer Science Department
RWTH Aachen University, D-52056 Aachen, Germany
{hasan,zens,ney}@cs.rwth-aachen.de

Abstract

This paper describes an efficient method to extract large n-best lists from a word graph produced by a statistical machine translation system. The extraction is based on the k shortest paths algorithm which is efficient even for very large k. We show that, although we can generate large amounts of distinct translation hypotheses, these numerous candidates are not able to significantly improve overall system performance. We conclude that large n-best lists would benefit from better discriminating models.

1 Introduction

This paper investigates the properties of large n-best lists in the context of statistical machine translation (SMT). We present a method that allows for fast extraction of very large n-best lists based on the k shortest paths algorithm by (Eppstein, 1998). We will argue that, despite being able to generate a much larger amount of hypotheses than previously reported in the literature, there is no significant gain of such a method in terms of translation quality.

In recent years, phrase-based approaches evolved as the dominating method for feasible machine translation systems. Many research groups use a decoder based on a log-linear approach incorporating phrases as main paradigm (Koehn et al., 2003). As a by-product of the decoding process, one can extract n-best translations from a word graph and use these fully generated hypotheses for additional reranking.

In the past, several groups report on using n-best lists with n ranging from 1 000 to 10 000. The advantage of n-best reranking is clear: we can apply complex reranking techniques, based e.g. on syntactic analyses of the candidates or using huge additional language models, since the whole sentence is already generated. During the generation process, these models would either need hard-to-implement algorithms or large memory requirements.

1.1 Related work

The idea of n-best list extraction from a word graph for SMT was presented in (Ueffing et al., 2002). In (Zens and Ney, 2005), an improved method is reported that overcomes some shortcomings, such as duplicate removal by determinization of the word graph (represented as a weighted finite state automaton) and efficient rest-cost estimation with linear time complexity.

There are several research groups that use a two-pass approach in their MT systems. First, they generate n-best translation hypotheses with the decoder. Second, they apply additional models to the output and rerank the candidates (see e.g. (Chen et al., 2006)).

Syntactic features were investigated in (Och et al., 2004) with moderate success. Although complex models, such as features based on shallow parsing or treebank-based syntactic analyses, were applied to the n-best candidates, the "simpler" ones were more promising (e.g. IBM model 1 on sentence-level).

In the following section 2, we describe our SMT system and explain how an improved n-best extraction method is capable of generating a very large number of distinct candidates from the word graph. In section 3, we show our experiments related to n-best list reranking with various sizes and the corresponding performance in terms of MT evaluation measures. Finally, we discuss the results in section 4 and give some conclusive remarks.

Proceedings of NAACL HLT 2007, Companion Volume, pages 57–60,
Rochester, NY, April 2007.

2 Generating N-best lists

We use a phrase-based SMT system (Mauser et al., 2006) and enhance the n-best list extraction with Eppstein's k shortest path algorithm which allows for generating a very large number of translation candidates in an efficient way.

2.1 Baseline SMT system

The baseline system uses phrases automatically extracted from a word-aligned corpus (trained with GIZA++) and generates the best translations using weighted log-linear model combination with several features, such as word lexicon, phrase translation and language models. This direct approach is currently used by most state-of-the-art decoders. The model scaling factors are trained discriminatively on some evaluation measure, e.g. BLEU or WER, using the simplex method.

2.2 N-best list extraction

We incorporated an efficient extraction of n best translations using the k shortest path algorithm (Eppstein, 1998) into a state-of-the-art SMT system. The implementation is partly based on code that is publicly available.[1]

Starting point for the extraction is a word graph, generated separately by the decoder for each sentence. Since these word graphs are directed and acyclic, it is possible to construct a shortest path tree spanning from the sentence begin node to the end node. The efficiency of finding the k shortest paths in this tree lies in the book-keeping of edges through a binary heap that allows for an implicit representation of paths. The overall performance of the algorithm is efficient even for large k. Thus, it is feasible to use in situations where we want to generate a large number of paths, i.e. translation hypotheses in this context.

There is another issue that has to be addressed. In phrase-based SMT, we have to deal with different phrase segmentations for each sentence. Due to the large number of phrases, it is possible that we have paths through the word graph representing the same sentence but internally having different phrase boundaries. In n-best list generation, we want to get rid of these duplicates. Due to the efficiency of the k shortest paths algorithm, we allow for generating a very large number of hypotheses (e.g. $100 \cdot n$) and then filter the output via a prefix tree (also called trie) until we get n distinct translations.

With this method, it is feasible to generate 100 000-best lists without much hassle. In general, the file input/output operations are more time-consuming than the actual n-best list extraction. The average generation time of n-best candidates for each of the sentences of the development list is approximately 30 seconds on a 2.2GHz Opteron machine, whereas 7.4 million hypotheses are computed per sentence on average. The overall extraction time including filtering and writing to hard-disk takes around 100 seconds per sentence. Note that this value could be optimized drastically if checking for how many duplicates are generated on average beforehand and adjusting the initial number of hypotheses before applying the filtering. We only use the $k = 100 \cdot n$ as a proof of concept.

2.3 Rescoring models

After having generated the 100 000-best lists, we have to apply additional rescoring models to all hypotheses. We select the models that have shown to improve overall translation performance as used for recent NIST MT evaluations. In addition to the main decoder score (which is already a combination of several models and constitutes a strong baseline), these include several large language models trained on up to 2.5 billion running words, a sentence-level IBM model 1 score, m-gram posterior probabilities and an additional sentence length model.

3 Experiments

The experiments in this section are carried out on n-best lists with n going up to 100 000. We will show that, although we are capable of generating this large amount of hypotheses, the overall performance does not seem to improve significantly beyond a certain threshold. Or to put it simple: although we generate lots of hypotheses, most of them are not very useful.

As experimental background, we choose the large data track of the Chinese-to-English NIST task, since the length of the sentences and the large vocabulary of the task allow for large n-best lists. For smaller tasks, e.g. the IWSLT campaign, the domain is rather limited such that it does not make sense to generate lists reaching beyond several thousand hypotheses. As development data, we use the 2002 eval set, whereas for test, the 2005 eval set is chosen. The corpus statistics are shown in Table 1.

[1] `http://www.ics.uci.edu/~eppstein/pubs/graehl.zip`

		Chinese	English
Train	Sentence Pairs	7M	
	Running Words	199M	213M
	Vocabulary Size	222K	351K
Dev	Sentence Pairs	878	3 512
	Running Words	25K	105K
Test	Sentence Pairs	1 082	4 328
	Running Words	33K	148K

Table 1: Corpus statistics for the Chinese-English NIST MT task.

	Oracle-WER [%]		Oracle-PER [%]	
N	BLEU	abs. imp.	BLEU	abs. imp.
1	36.1		36.1	
10	38.8	+2.7	38.0	+1.9
100	41.3	+2.5	39.8	+1.8
1000	43.3	+2.0	41.0	+1.2
10000	44.4	+1.1	42.0	+1.0
100000	45.3	+0.9	42.0	+0.0

Table 2: Dev BLEU scores of oracle-best hypotheses based on minimum WER/PER.

3.1 Oracle-best hypotheses

In the first experiment, we examined the oracle-best hypotheses in the n-best lists for several list sizes. For an efficient calculation of the true BLEU oracle (the hypothesis which has a maximum BLEU score when compared to the reference translations), we use approximations based on WER/PER-oracles, i.e. we extract the hypotheses that have the lowest edit distance (WER, word error rate) to the references. The same is applied by disregarding the word order (leading to PER, position-independent word error rate).

As can be seen in Table 2, the improvements are steadily decreasing, i.e. with increasing number of generated hypotheses, there are less and less useful candidates among them. For the first 10 000 candidates, we therefore have the possibility to find hypotheses that could increase the BLEU score by at least 8.3% absolute *if* our models discriminated them properly. For the next 90 000 hypotheses, there is only a small potential to improve the whole system by around 1%. This means that most of the generated hypotheses are not very useful in terms of oracle-WER and likely distracting the "search" for the needle(s) in the haystack. It has been shown in (Och et al., 2004) that true BLEU oracle scores on lists with much smaller $n \leq 4096$ are more or less linear in $\log(n)$. Our results support this claim since the oracle-WER/PER is a lower bound of the real BLEU oracle. For the PER criterion, the behavior of the oracle-best hypotheses is similar. Here we can notice that after 10,000 hypotheses, the BLEU score of the oracle-PER hypotheses stays the same.

These observations already impair the alleged usefulness of a large amount of translation hypotheses by showing that the overall possible gain with increasing n gets disproportionately small if one puts it in relation to the exponential growth of the n.

3.2 Rescoring performance

As a next step, we show the performance of tuning the model scaling factors towards best translation performance. In our experiments, we use the BLEU score as objective function of the simplex method.

Figure 1 shows the graphs for the development (on the left) and test set (on the right). The upper graphs depict the oracle-WER BLEU scores (cf. also Table 2) for comparison. As was already stated, these are a lower bound since the real oracle-BLEU hypotheses might have even higher scores. Still, it is an indicator of what could be achieved if the models discriminated good from bad hypotheses properly.

The lower two graphs show the behavior when (a) optimizing and extracting hypotheses on a subset (the first n) of the 100k-best hypotheses and (b) optimizing on a subset but extracting from the full 100k set. As can be seen, extracting from the full set does not even help for the development data on which the scaling factors were tuned. Experiments on the test list show similar results. We can also observe that the improvement declines rapidly with higher n. Note that an optimization on the full 100k list was not possible due to huge memory requirements. The highest n that fit into the 16GB machine was 60 000. Thus, this setting was used for extraction on the full 100k set.

The results so far indicate that it is not very useful to go beyond $n = 10\,000$. For the development set, the baseline of 36.1% BLEU can be improved by 1.6% absolute to 37.7% for the first 10k entries, whereas for the 60k setting, the absolute improvement is only increased by a marginal 0.1%. For the chosen setting, whose focus was on various list sizes for optimization and extraction, the improvements on the development lists do not carry over to the test list. From the baseline of 31.5%, we only get a moderate improvement of approximately 0.5% BLEU.

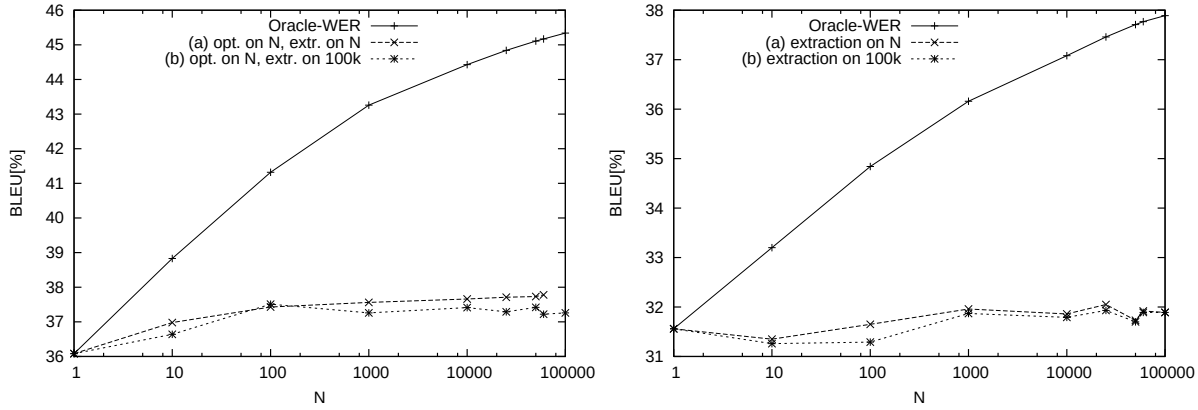

Figure 1: BLEU scores of the reranked system. Development set (left) vs. Test set (right).

One possible explanation for this lies in the poor performance of the rescoring models. A short test was carried out in which we added the reference translations to the n-best list and determined the corresponding scores of the additional models, such as the large LM and the IBM model 1. Interestingly, only less than 1/4 of the references was ranked as the best hypothesis. Thus, most reference translations would never have been selected as final candidates. This strongly indicates that we have to come up with better models in order to make significant improvements from large n-best lists. Furthermore, it seems that the exponential growth of n-best hypotheses for maintaining a quasilinear improvement in oracle BLEU score has a strong impact on the overall system performance. This is in contrast to a word graph, where a linear increment of its density yields disproportionately high improvements in oracle BLEU for lower densities (Zens and Ney, 2005).

4 Conclusion

We described an efficient n-best list extraction method that is based on the k shortest paths algorithm. Experiments with large 100 000-best lists indicate that the models do not have the discriminating power to separate the good from the bad candidates. The oracle-best BLEU scores stay linear in $\log(n)$, whereas the reranked system performance seems to saturate at around 10k best translations given the actual models. Using more hypotheses currently does not help to significantly improve translation quality.

Given the current results, one should balance the advantages of n-best lists, e.g. easily testing complex rescoring models, and word graphs, e.g. representation of a much larger hypotheses space. However, as long as the models are not able to correctly fire on good candidates, both approaches will stay beneath their capabilities.

Acknowledgments

This material is partly based upon work supported by the Defense Advanced Research Projects Agency (DARPA) under Contract No. HR0011-06-C-0023, and was partly funded by the Deutsche Forschungsgemeinschaft (DFG) under the project "Statistische Textübersetzung" (NE 572/5-3).

References

B. Chen, R. Cattoni, N. Bertoldi, M. Cettolo, and M. Federico. 2006. The ITC-irst SMT system for IWSLT 2006. In *Proc. of the International Workshop on Spoken Language Translation*, pages 53–58, Kyoto, Japan, November.

D. Eppstein. 1998. Finding the k shortest paths. *SIAM J. Computing*, 28(2):652–673.

P. Koehn, F. J. Och, and D. Marcu. 2003. Statistical phrase-based translation. In *Proc. of the Human Language Technology Conf. (HLT-NAACL)*, pages 127–133, Edmonton, Canada, May/June.

A. Mauser, R. Zens, E. Matusov, S. Hasan, and H. Ney. 2006. 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.

F. J. Och, D. Gildea, S. Khudanpur, A. Sarkar, K. Yamada, A. Fraser, S. Kumar, L. Shen, D. Smith, K. Eng, V. Jain, Z. Jin, and D. Radev. 2004. A smorgasbord of features for statistical machine translation. In *Proc. 2004 Meeting of the North American chapter of the Association for Computational Linguistics (HLT-NAACL)*, pages 161–168, Boston, MA, May.

N. Ueffing, F. J. Och, and H. Ney. 2002. Generation of word graphs in statistical machine translation. In *Proc. of the Conf. on Empirical Methods for Natural Language Processing (EMNLP)*, pages 156–163, Philadelphia, PA, July.

R. Zens and H. Ney. 2005. Word graphs for statistical machine translation. In *43rd Annual Meeting of the Assoc. for Computational Linguistics: Proc. Workshop on Building and Using Parallel Texts*, pages 191–198, Ann Arbor, MI, June.