Word Graphs for Statistical Machine Translation

Richard Zens and **Hermann Ney**
Chair of Computer Science VI
RWTH Aachen University
{zens,ney}@cs.rwth-aachen.de

Abstract

Word graphs have various applications in the field of machine translation. Therefore it is important for machine translation systems to produce compact word graphs of high quality. We will describe the generation of word graphs for state of the art phrase-based statistical machine translation. We will use these word graph to provide an analysis of the search process. We will evaluate the quality of the word graphs using the well-known graph word error rate. Additionally, we introduce the two novel graph-to-string criteria: the position-independent graph word error rate and the graph BLEU score.
Experimental results are presented for two Chinese–English tasks: the small IWSLT task and the NIST large data track task. For both tasks, we achieve significant reductions of the graph error rate already with compact word graphs.

1 Introduction

A statistical machine translation system usually produces the single-best translation hypotheses for a source sentence. For some applications, we are also interested in alternative translations. The simplest way to represent these alternatives is a list with the N-best translation candidates. These N-best lists have one major disadvantage: the high redundancy. The translation alternatives may differ only by a single word, but still both are listed completely. Usually, the size of the N-best list is in the range of a few hundred up to a few thousand candidate translations per source sentence. If we want to use larger N-best lists the processing time gets very soon infeasible.

Word graphs are a much more compact representation that avoid these redundancies as much as possible. The number of alternatives in a word graph is usually an order of magnitude larger than in an N-best list. The graph representation avoids the combinatorial explosion that make large N-best lists infeasible.

Word graphs are an important data structure with various applications:

- Word Filter.
  The word graph is used as a compact representation of a large number of sentences. The score information is not contained.
- Rescoring.
  We can use word graphs for rescoring with more sophisticated models, e.g. higher-order language models.
- Discriminative Training.
  The training of the model scaling factors as described in (Och and Ney, 2002) was done on N-best lists. Using word graphs instead could further improve the results. Also, the phrase translation probabilities could be trained discrimatively, rather than only the scaling factors.
- Confidence Measures.
  Word graphs can be used to derive confidence measures, such as the posterior probability (Ueffing and Ney, 2004).

- Interactive Machine Translation.
  Some interactive machine translation systems make use of word graphs, e.g. (Och et al., 2003).

State Of The Art. Although there are these many applications, there are only few publications directly devoted to word graphs. The only publication, we are aware of, is (Ueffing et al., 2002). The shortcomings of (Ueffing et al., 2002) are:

- They use single-word based models only. Current state of the art statistical machine translation systems are phrase-based.
- Their graph pruning method is suboptimal as it considers only partial scores and not full path scores.
- The N-best list extraction does not eliminate duplicates, i.e. different paths that represent the same translation candidate.
- The rest cost estimation is not efficient. It has an exponential worst-case time complexity. We will describe an algorithm with linear worst-case complexity.

Apart from (Ueffing et al., 2002), publications on weighted finite state transducer approaches to machine translation, e.g. (Bangalore and Riccardi, 2001; Kumar and Byrne, 2003), deal with word graphs. But to our knowledge, there are no publications that give a detailed analysis and evaluation of the quality of word graphs for machine translation.

We will fill this gap and give a systematic description and an assessment of the quality of word graphs for phrase-based machine translation. We will show that even for hard tasks with very large vocabulary and long sentences the graph error rate drops significantly.

The remaining part is structured as follows: first we will give a brief description of the translation system in Section 2. In Section 3, we will give a definition of word graphs and describe the generation. We will also present efficient pruning and N-best list extraction techniques. In Section 4, we will describe evaluation criteria for word graphs. We will use the graph word error rate, which is well known from speech recognition. Additionally, we introduce the novel position-independent word graph error rate and the graph BLEU score. These are generalizations of the commonly used string-to-string evaluation criteria in machine translation. We will present experimental results in Section 5 for two Chinese–English tasks: the first one, the IWSLT task, is in the domain of basic travel expression found in phrasebooks. The vocabulary is limited and the sentences are short. The second task is the NIST Chinese–English large data track task. Here, the domain is news and therefore the vocabulary is very large and the sentences are with an average of 30 words quite long.

2 Translation System

In this section, we give a brief description of the translation system. We use a phrase-based translation approach as described in (Zens and Ney, 2004). The posterior probability $Pr(e_1^I|f_1^J)$ is modeled directly using a weighted log-linear combination of a trigram language model and various translation models: a phrase translation model and a word-based lexicon model. These translation models are used for both directions: $p(f|e)$ and $p(e|f)$. Additionally, we use a word penalty and a phrase penalty. With the exception of the language model, all models can be considered as within-phrase models as they depend only on a single phrase pair, but not on the context outside of the phrase. The model scaling factors are optimized with respect to some evaluation criterion (Och, 2003).

We extended the monotone search algorithm from (Zens and Ney, 2004) such that reorderings are possible. In our case, we assume that local reorderings are sufficient. Within a certain window, all possible permutations of the source positions are allowed. These permutations are represented as a reordering graph, similar to (Zens et al., 2002). Once we have this reordering graph, we perform a monotone phrase-based translation of this graph. More details of this reordering approach are described in (Kanthak et al., 2005).

3 Word Graphs

3.1 Definition

A word graph is a directed acyclic graph $G = (V, E)$ with one designated root node $n_0 \in V$. The edges are labeled with words and optionally with scores. We will use (n, n', w) to denote an edge from node

n to node n' with word label w. Each path through the word graph represents a translation candidate. If the word graph contains scores, we accumulate the edge scores along a path to get the sentence or string score.

The score information the word graph has to contain depends on the application.

If we want to use the word graph as a word filter, we do not need any score information at all. If we want to extract the single- or N-best hypotheses, we have to retain the string or sentence score information. The information about the hidden variables of the search, e.g. the phrase segmentation, is not needed for this purpose. For discriminative training of the phrase translation probabilities, we need all the information, even about the hidden variables.

3.2 Generation

In this section, we analyze the search process in detail. Later, in Section 5, we will show the (experimental) complexity of each step. We start with the source language sentence that is represented as a linear graph. Then, we introduce reorderings into this graph as described in (Kanthak et al., 2005). The type of reordering should depend on the language pair. In our case, we assume that only local reorderings are required. Within a certain window, all possible reorderings of the source positions are allowed. These permutations are represented as a reordering graph, similar to (Knight and Al-Onaizan, 1998) and (Zens et al., 2002).

Once we have this reordering graph, we perform a monotone phrase-based translation of this graph. This translation process consists of the following steps that will be described afterward:

1. segment into phrase
2. translate the individual phrases
3. split the phrases into words
4. apply the language model

Now, we will describe each step. The first step is the segmentation into phrases. This can be imagined as introducing “short-cuts” into the graph. The phrase segmentation does not affect the number of nodes, because only additional edges are added to the graph.

In the segmented graph, each edge represents a source phrase. Now, we replace each edge with one edge for each possible phrase translation. The edge scores are the combination of the different translation probabilities, namely the within-phrase models mentioned in Section 2. Again, this step does not increase the number of nodes, but only the number of edges.

So far, the edge labels of our graph are phrases. In the final word graph, we want to have words as edge labels. Therefore, we replace each edge representing a multi-word target phrase with a sequence of edges that represent the target word sequence. Obviously, edges representing a single-word phrase do not have to be changed.

As we will show in the results section, the word graphs up to this point are rather compact. The score information in the word graph so far consists of the reordering model scores and the phrase translation model scores. To obtain the sentence posterior probability $p(e_1^I|f_1^J)$, we apply the target language model. To do this, we have to separate paths according to the language model history. This increases the word graph size by an order of magnitude.

Finally, we have generated a word graph with full sentence scores. Note that the word graph may contain a word sequence multiple times with different hidden variables. For instance, two different segmentations into source phrases may result in the same target sentence translation.

The described steps can be implemented using weighted finite state transducer, similar to (Kumar and Byrne, 2003).

3.3 Pruning

To adjust the size of the word graph to the desired density, we can reduce the word graph size using forward-backward pruning, which is well-known in the speech recognition community, e.g. see (Mangu et al., 2000). This pruning method guarantees that the good strings (with respect to the model scores) remain in the word graph, whereas the bad ones are removed. The important point is that we compare the full path scores and not only partial scores as, for instance, in the beam pruning method in (Ueffing et al., 2002).

The forward probabilities $F(n)$ and backward probabilities $B(n)$ of a node n are defined by the following recursive equations:

$$F(n) = \sum_{(n',n,w)\in E} F(n') \cdot p(n',n,w)$$

$$B(n) = \sum_{(n,n',w)\in E} B(n') \cdot p(n,n',w)$$

The forward probability of the root node and the backward probabilities of the final nodes are initialized with one. Using a topological sorting of the nodes, the forward and backward probabilities can be computed with linear time complexity. The posterior probability $q(n, n', w)$ of an edge is defined as:

$$q(n,n',w) = \frac{F(n) \cdot p(n,n',w) \cdot B(n')}{B(n_0)}$$

The posterior probability of an edge is identical to the sum over the probabilities of all full paths that contain this edge. Note that the backward probability of the root node $B(n_0)$ is identical to the sum over all sentence probabilities in the word graph. Let q^* denoted the maximum posterior probability of all edges and let τ be a pruning threshold, then we prune an edge (n, n', w) if:

$$q(n,n',w) < q^* \cdot \tau$$

3.4 N-Best List Extraction

In this section, we describe the extraction of the N-best translation candidates from a word graph.

(Ueffing et al., 2002) and (Mohri and Riley, 2002) both present an algorithm based on the same idea: use a modified A* algorithm with an optimal rest cost estimation. As rest cost estimation, the negated logarithm of the backward probabilities is used. The algorithm in (Ueffing et al., 2002) has two disadvantages: it does not care about duplicates and the rest cost computation is suboptimal as the described algorithm has an exponential worst-case complexity. As mentioned in the previous section, the backward probabilities can be computed in linear time.

In (Mohri and Riley, 2002) the word graph is represented as a weighted finite state automaton. The word graph is first determinized, i.e. the nondeterministic automaton is transformed in an equivalent deterministic automaton. This process removes the duplicates from the word graph. Out of this determinized word graph, the N best candidates are extracted. In (Mohri and Riley, 2002), ϵ-transitions are ignored, i.e. transitions that do not produce a word. These ϵ-transitions usually occur in the backing-off case of language models. The ϵ-transitions have to be removed *before* using the algorithm of (Mohri and Riley, 2002). In the presence of ϵ-transitions, two path representing the same string are considered equal only if the ϵ-transitions are identical as well.

4 Evaluation Criteria

4.1 String-To-String Criteria

To evaluate the single-best translation hypotheses, we use the following string-to-string criteria: word error rate (WER), position-independent word error rate (PER) and the BLEU score. More details on these standard criteria can be found for instance in (Och, 2003).

4.2 Graph-To-String Criteria

To evaluate the quality of the word graphs, we generalize the string-to-string criteria to work on word graphs. We will use the well-known graph word error rate (GWER), see also (Ueffing et al., 2002). Additionally, we introduce two novel graph-to-string criteria, namely the position-independent graph word error rate (GPER) and the graph BLEU score (GBLEU). The idea of these graph-to-string criteria is to choose a sequence from the word graph and compute the corresponding string-to-string criterion for this specific sequence. The choice of the sequence is such that the criterion is the optimum over all possible sequences in the word graph, i.e. the minimum for GWER/GPER and the maximum for GBLEU.

The **GWER** is a generalization of the word error rate. It is a lower bound for the WER. It can be computed using a dynamic programming algorithm which is quite similar to the usual edit distance computation. Visiting the nodes of the word graph in topological order helps to avoid repeated computations.

The **GPER** is a generalization of the position-independent word error rate. It is a lower bound for the PER. The computation is not as straightforward as for the GWER.

In (Ueffing and Ney, 2004), a method for computing the string-to-string PER is presented. This method cannot be generalized for the graph-to-string computation in a straightforward way. Therefore, we will first describe an alternative computation for the string-to-string PER and then use this idea for the graph-to-string PER.

Now, we want to compute the number of position-independent errors for two strings. As the word order of the strings does not matter, we represent them as multisets[1] A and B. To do this, it is sufficient to know how many words are in A but not in B, i.e. $a := |A - B|$, and how many words are in B but not in A, i.e. $b := |B - A|$. The number of substitutions, insertions and deletions are then:

$$\begin{aligned} sub &= \min\{a, b\} \\ ins &= a - sub \\ del &= b - sub \\ error &= sub + ins + del \\ &= a + b - \min\{a, b\} \\ &= \max\{a, b\} \end{aligned}$$

It is obvious that there are either no insertions or no deletions. The PER is then computed as the number of errors divided by the length of the reference string.

Now, back to the graph-to-string PER computation. The information we need at each node of the word graph are the following: the remaining multiset of words of the reference string that are not yet produced. We denote this multiset C. The cardinality of this multiset will become the value a in the preceding notation. In addition to this multiset, we also need to count the number of words that we have produced on the way to this node but which are not in the reference string. The identity of these words is not important, we simply have to count them. This count will become the value b in the preceding notation.

If we make a transition to a successor node along an edge labeled w, we remove that word w from the set of remaining reference words C or, if the word w is not in this set, we increase the count of words that are in the hypothesis but not in the reference.

To compute the number of errors on a graph, we use the auxiliary quantity $Q(n, C)$, which is the count of the produced words that are not in the reference. We use the following dynamic programming recursion equations:

[1]A multiset is a set that may contain elements multiple times.

$$\begin{aligned} Q(n_0, C_0) &= 0 \\ Q(n, C) &= \min_{n', w : (n', n, w) \in E} \Big\{ Q(n', C \cup \{w\}), \\ &\qquad\qquad Q(n', C) + 1 \Big\} \end{aligned}$$

Here, n_0 denote the root node of the word graph, C_0 denotes the multiset representation of the reference string. As already mentioned in Section 3.1, (n', n, w) denotes an edge from node n' to node n with word label w.

In the implementation, we use a bit vector to represent the set C for efficiency reasons. Note that in the worst-case the size of the Q-table is exponential in the length of the reference string. However, in practice we found that in most cases the computation is quite fast.

The **GBLEU** score is a generalization of the BLEU score. It is an upper bound for the BLEU score. The computation is similar to the GPER computation. We traverse the word graph in topological order and store the following information: the counts of the matching n-grams and the length of the hypothesis, i.e. the depth in the word graph. Additionally, we need the multiset of reference n-grams that are not yet produced.

To compute the BLEU score, the n-gram counts are collected over the whole test set. This results in a combinatorial problem for the computation of the GBLEU score. We process the test set sentence-wise and accumulate the n-gram counts. After each sentence, we take a greedy decision and choose the n-gram counts that, if combined with the accumulated n-gram counts, result is the largest BLEU score. This gives a conservative approximation of the true GBLEU score.

4.3 Word Graph Size

To measure the word graph size we use the word graph density, which we define as the number of edges in the graph divided by the source sentence length.

5 Experimental Results

5.1 Tasks

We will show experimental results for two Chinese–English translation tasks.

Table 1: IWSLT Chinese–English Task: corpus statistics of the bilingual training data.

		Chinese	English
Train	Sentences	20 000	
	Running Words	182 904	160 523
	Vocabulary	7 643	6 982
Test	Sentences	506	
	Running Words	3 515	3 595
	avg. SentLen	6.9	7.1

Table 2: NIST Chinese English task: corpus statistics of the bilingual training data.

		Chinese	English
Train	Sentences	3.2M	
	Running Words	51.4M	55.5M
	Vocabulary	80 010	170 758
Lexicon	Entries	81 968	
Test	Sentences	878	
	Running Words	26 431	23 694
	avg. SentLen	30.1	27.0

IWSLT Chinese–English Task. The first task is the Chinese–English supplied data track task of the International Workshop on Spoken Language Translation (IWSLT 2004) (Akiba et al., 2004). The domain is travel expressions from phrase-books. This is a small task with a clean training and test corpus. The vocabulary is limited and the sentences are relatively short. The corpus statistics are shown in Table 1. The Chinese part of this corpus is already segmented into words.

NIST Chinese–English Task. The second task is the NIST Chinese–English large data track task. For this task, there are many bilingual corpora available. The domain is news, the vocabulary is very large and the sentences have an average length of 30 words. We train our statistical models on various corpora provided by LDC. The Chinese part is segmented using the LDC segmentation tool. After the preprocessing, our training corpus consists of about three million sentences with somewhat more than 50 million running words. The corpus statistics of the preprocessed training corpus are shown in Table 2. We use the NIST 2002 evaluation data as test set.

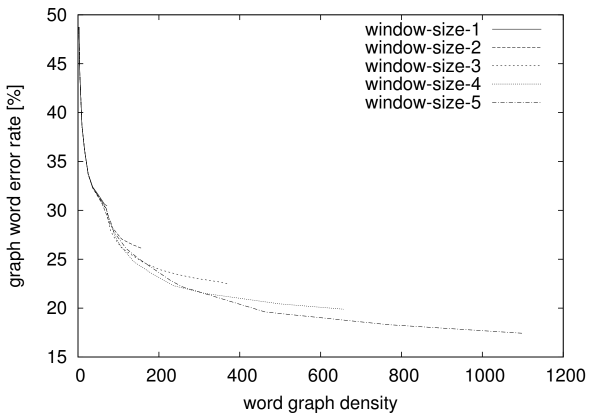

Figure 1: IWSLT Chinese–English: Graph error rate as a function of the word graph density for different window sizes.

5.2 Search Space Analysis

In Table 3, we show the search space statistics of the IWSLT task for different reordering window sizes. Each line shows the resulting graph densities after the corresponding step in our search as described in Section 3.2. Our search process starts with the reordering graph. The segmentation into phrases increases the graph densities by a factor of two. Doing the phrase translation results in an increase of the densities by a factor of twenty. Unsegmenting the phrases, i.e. replacing the phrase edges with a sequence of word edges doubles the graph sizes. Applying the language model results in a significant increase of the word graphs.

Another interesting aspect is that increasing the window size by one roughly doubles the search space.

5.3 Word Graph Error Rates

In Figure 1, we show the graph word error rate for the IWSLT task as a function of the word graph density. This is done for different window sizes for the reordering. We see that the curves start with a single-best word error rate of about 50%. For the monotone search, the graph word error rate goes down to about 31%. Using local reordering during the search, we can further decrease the graph word error rate down to less than 17% for a window size of 5. This is almost one third of the single-best word error rate. If we aim at halving the single-best word error rate, word graphs with a density of less than

Table 3: IWSLT Chinese–English: Word graph densities for different window sizes and different stages of the search process.

language	level	graph type	window size				
			1	2	3	4	5
source	word	reordering	1.0	2.7	6.2	12.8	24.4
	phrase	segmented	2.0	5.0	12.1	26.8	55.6
target		translated	40.8	99.3	229.0	479.9	932.8
	word	TM scores	78.6	184.6	419.2	869.1	1 670.4
		+ LM scores	958.2	2874.2	7649.7	18 029.7	39 030.1

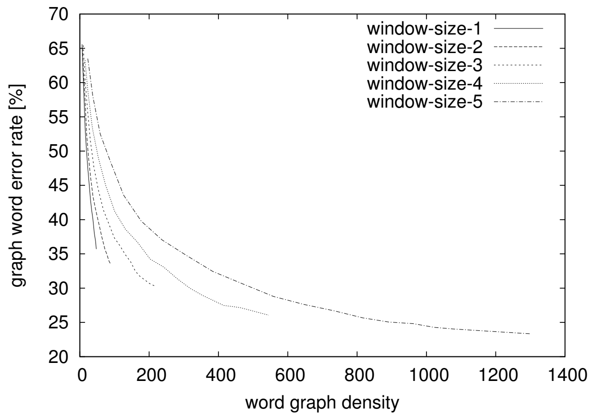

Figure 2: NIST Chinese–English: Graph error rate as a function of the word graph density for different window sizes.

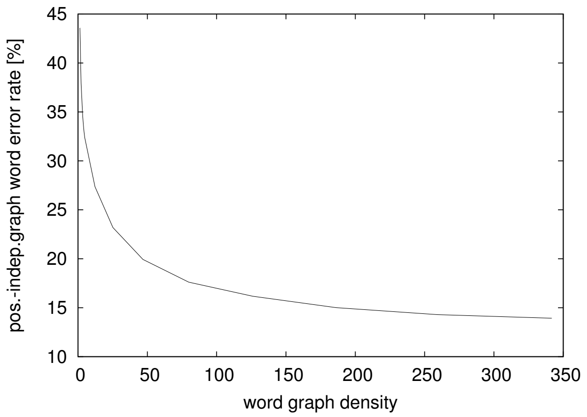

Figure 3: IWSLT Chinese–English: Graph position-independent word error rate as a function of the word graph density.

200 would already be sufficient.

In Figure 2, we show the same curves for the NIST task. Here, the curves start from a single-best word error rate of about 64%. Again, dependent on the amount of reordering the graph word error rate goes down to about 36% for the monotone search and even down to 23% for the search with a window of size 5. Again, the reduction of the graph word error rate compare to the single-best error rate is dramatic. For comparison we produced an N-best list of size 10 000. The N-best list error rate (or oracle-best WER) is still 50.8%. A word graph with a density of only 8 has about the same GWER.

In Figure 3, we show the graph position-independent word error rate for the IWSLT task. As this error criterion ignores the word order it is not affected by reordering and we show only one curve. We see that already for small word graph densities the GPER drops significantly from about 42% down to less than 14%.

In Figure 4, we show the graph BLEU scores for the IWSLT task. We observe that, similar to the GPER, the GBLEU score increases significantly already for small word graph densities. We attribute this to the fact that the BLEU score and especially the PER are less affected by errors of the word order than the WER. This also indicates that producing translations with correct word order, i.e. syntactically well-formed sentences, is one of the major problems of current statistical machine translation systems.

6 Conclusion

We have described word graphs for statistical machine translation. The generation of word graphs during the search process has been described in detail. We have shown detailed statistics of the individual steps of the translation process and have given insight in the experimental complexity of each step. We have described an efficient and optimal

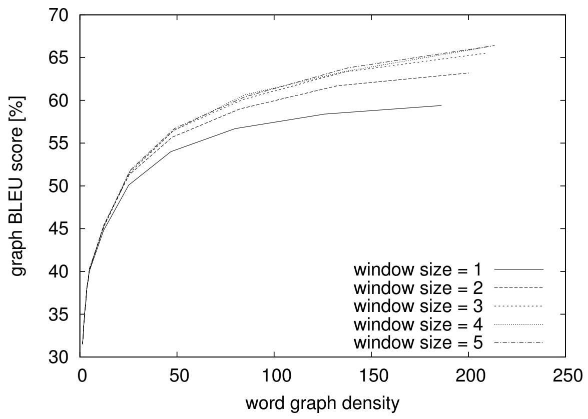

Figure 4: IWSLT Chinese–English: Graph BLEU score as a function of the word graph density.

pruning method for word graphs. Using these technique, we have generated compact word graphs for two Chinese–English tasks. For the IWSLT task, the graph error rate drops from about 50% for the single-best hypotheses to 17% of the word graph. Even for the NIST task, with its very large vocabulary and long sentences, we were able to reduce the graph error rate significantly from about 64% down to 23%.

Acknowledgment

This work was partly funded by the European Union under the integrated project TC-Star (Technology and Corpora for Speech to Speech Translation, IST-2002-FP6-506738, http://www.tc-star.org).

References

Y. Akiba, M. Federico, N. Kando, H. Nakaiwa, M. Paul, and J. Tsujii. 2004. Overview of the IWSLT04 evaluation campaign. In *Proc. of the Int. Workshop on Spoken Language Translation (IWSLT)*, pages 1–12, Kyoto, Japan, September/October.

S. Bangalore and G. Riccardi. 2001. A finite-state approach to machine translation. In *Proc. Conf. of the North American Association of Computational Linguistics (NAACL)*, Pittsburgh, May.

S. Kanthak, D. Vilar, E. Matusov, R. Zens, and H. Ney. 2005. Novel reordering approaches in phrase-based statistical machine translation. In *43rd Annual Meeting of the Assoc. for Computational Linguistics: Proc. Workshop on Building and Using Parallel Texts: Data-Driven Machine Translation and Beyond*, Ann Arbor, MI, June.

K. Knight and Y. Al-Onaizan. 1998. Translation with finite-state devices. In D. Farwell, L. Gerber, and E. H. Hovy, editors, *AMTA*, volume 1529 of *Lecture Notes in Computer Science*, pages 421–437. Springer Verlag.

S. Kumar and W. Byrne. 2003. A weighted finite state transducer implementation of the alignment template model for statistical machine translation. In *Proc. of the Human Language Technology Conf. (HLT-NAACL)*, pages 63–70, Edmonton, Canada, May/June.

L. Mangu, E. Brill, and A. Stolcke. 2000. Finding consensus in speech recognition: Word error minimization and other applications of confusion networks. *Computer, Speech and Language*, 14(4):373–400, October.

M. Mohri and M. Riley. 2002. An efficient algorithm for the n-best-strings problem. In *Proc. of the 7th Int. Conf. on Spoken Language Processing (ICSLP'02)*, pages 1313–1316, Denver, CO, September.

F. J. Och and H. Ney. 2002. Discriminative training and maximum entropy models for statistical machine translation. In *Proc. of the 40th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 295–302, Philadelphia, PA, July.

F. J. Och, R. Zens, and H. Ney. 2003. Efficient search for interactive statistical machine translation. In *EACL03: 10th Conf. of the Europ. Chapter of the Association for Computational Linguistics*, pages 387–393, Budapest, Hungary, April.

F. J. Och. 2003. Minimum error rate training in statistical machine translation. In *Proc. of the 41th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 160–167, Sapporo, Japan, July.

N. Ueffing and H. Ney. 2004. Bayes decision rule and confidence measures for statistical machine translation. In *Proc. EsTAL - España for Natural Language Processing*, pages 70–81, Alicante, Spain, October.

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. 2004. Improvements in phrase-based statistical machine translation. In *Proc. of the Human Language Technology Conf. (HLT-NAACL)*, pages 257–264, Boston, MA, May.

R. Zens, F. J. Och, and H. Ney. 2002. Phrase-based statistical machine translation. In M. Jarke, J. Koehler, and G. Lakemeyer, editors, *25th German Conf. on Artificial Intelligence (KI2002)*, volume 2479 of *Lecture Notes in Artificial Intelligence (LNAI)*, pages 18–32, Aachen, Germany, September. Springer Verlag.