Generation of Word Graphs in Statistical Machine Translation

Nicola Ueffing and Franz Josef Och and Hermann Ney
Lehrstuhl für Informatik VI, Computer Science Department
RWTH Aachen - University of Technology
D-52056 Aachen, Germany
{ueffing,och,ney}@informatik.rwth-aachen.de

Abstract

Statistical machine translation systems usually compute the single sentence that has the highest probability according to the models that are trained on data. We describe a method for constructing a word graph to represent alternative hypotheses in an efficient way. The advantage is that these hypotheses can be rescored using a refined language or translation model. Results are presented on the German-English Verbmobil corpus.

1 Introduction

The goal of the statistical machine translation process is the following: a given source string $f_1^J = f_1 \ldots f_j \ldots f_J$ is to be translated into a target string $e_1^I = e_1 \ldots e_i \ldots e_I$. Based on Bayes' decision rule, we choose the string with the highest probability:

$$\begin{aligned} \hat{e}_1^I &= \arg\max_{e_1^I} \left\{ Pr(e_1^I \mid f_1^J) \right\} \\ &= \arg\max_{e_1^I} \left\{ Pr(e_1^I) \cdot Pr(f_1^J \mid e_1^I) \right\} \end{aligned}$$

The system is divided into three parts (cf. Figure 1): the **translation model** $Pr(f_1^J|e_1^I)$ — consisting of the **lexicon** and the **alignment model** — the **language model** of the target language, $Pr(e_1^I)$, and the **search** denoted by the *arg max* operation. Among all possible target strings, we have to find the target string that maximizes the probability calculated from the models that are trained on data.

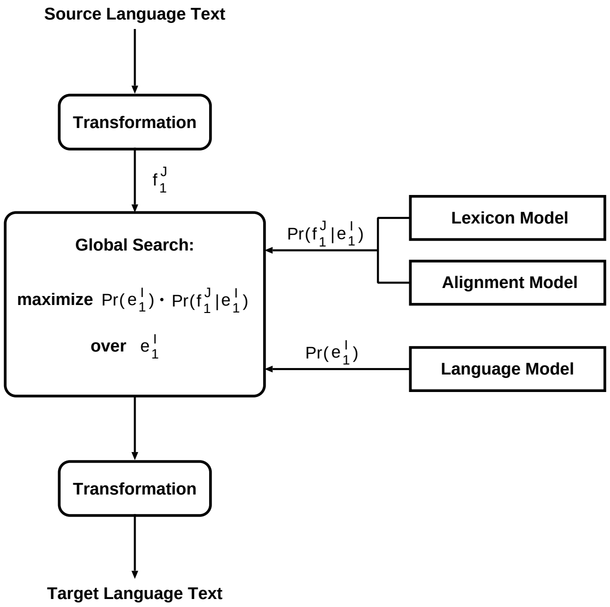

Figure 1: Architecture of a Statistical Translation System

The translation model $Pr(f_1^J|e_1^I)$ is based on dependencies between words of the source and the target language, so-called **alignments**. An alignment is a mapping $j \rightarrow i = a_j$ from source position j to target position $i = a_j$, i.e. the target word e_i is a translation of the source word f_j.

An example of an alignment is given in Figure 2. The English word 'about' is aligned to three German words, and the other English words to exactly one. The German word 'am' is aligned to the so-called **empty word** e_0, because it has no translation in the English string.

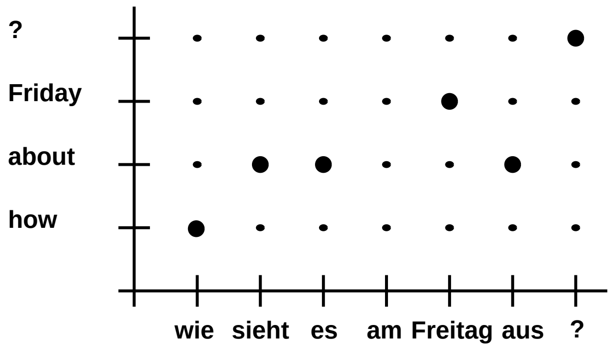

Figure 2: Example of an Alignment

In statistical alignment models $Pr(f_1^J, a_1^J | e_1^I)$, the alignment a_1^J is introduced as a hidden variable. The translation probability can then be rewritten in the following way:

$$Pr(f_1^J \mid e_1^I) = \sum_{a_1^J} Pr(f_1^J, a_1^J \mid e_1^I) \quad .$$

Instead of summing over all possible alignments, the search is performed using the so-called maximum approximation:

$$\begin{aligned} \hat{e}_1^I &= \arg\max_{e_1^I} \left\{ Pr(e_1^I) \cdot \sum_{a_1^J} Pr(f_1^J, a_1^J \mid e_1^I) \right\} \\ &= \arg\max_{e_1^I} \left\{ Pr(e_1^I) \cdot \max_{a_1^J} Pr(f_1^J, a_1^J \mid e_1^I) \right\} \end{aligned}$$

The search space consists of the set of all possible target language strings e_1^I and all possible alignments a_1^J.

In this paper, we use the so-called Model 4 introduced by IBM in (Brown et al., 1993). Yet, many of the results presented in the following are also applicable to other alignment models.

2 Single Best Search

2.1 Search Space

In (Och et al., 2001), a possible way to structure the search space as a lattice is described. We use the same structure here and thus will shortly review the basic concepts of this search space structure: every partial hypothesis consists of a prefix of the target sentence and a corresponding alignment. A partial hypothesis is extended by accounting for exactly one additional word of the source sentence. The possible extensions are the following:

1. the source word f is translated by a word e in the target language,
2. the source word f is translated by the empty word e_0,
3. the source word f is translated by a word e in the target language, and a target language word e' without an equivalence in the source sentence is inserted before e.

Every extension yields an extension score that is computed by taking into account the probabilities of the translation model. If a new word in the target string is produced, the language model score is considered as well.

We use a standard trigram language model, so the relevant language model state of a node in the search lattice consists of the current word and the previous word. The translation model state depends on the specific model dependencies of Model 4 for the node, such as the source sentence position translated last.

We perform a **recombination** of search hypotheses in order to keep the search space manageable: every two hypotheses that can be distinguished by neither the language model state nor the translation model state can be recombined, i.e. only the hypothesis with the higher probability of the two needs to be considered in the subsequent search process.

2.2 Search Algorithm

The search is performed using a beam search algorithm that focuses the search to the most promising partial hypotheses and discards the others. A description of this algorithm can be found in (Tillmann and Ney, 2000) and (Tillmann and Ney, 2002).

The discarding of hypotheses with low probability is called **pruning**. Two different types of pruning are applied during search: **threshold** and **histogram pruning**. For the threshold pruning, the probability of the most likely partial hypothesis is computed and all hypotheses whose probability is below this value multiplied with a threshold (lower than one) will not be regarded for further expansion. Histogram pruning means that all but the M best hypotheses are pruned for a fixed M.

For finding the most likely partial hypotheses, first all hypotheses with the same *set* of covered source sentence positions are compared. After threshold and histogram pruning have been applied, we also compare all hypotheses with the same *number* of covered source sentence positions and apply both pruning types again. Those hypotheses that survive the pruning are called the **active** hypotheses.

The word graph structure and the results presented here can easily be transferred to other search algorithms, such as A* search.

3 Word Graphs

3.1 Motivation

It is widely accepted in the community that a significant improvement in translation quality will come from more sophisticated translation and language models. For example, a language model that goes beyond m-gram dependencies could be used, but this would be difficult to integrate into the search process. As a step towards the solution of this problem, we determine not only the single best sentence hypothesis, but also other complete sentences that the search algorithm found but that were judged worse. We can then apply rescoring with a refined model to those hypotheses. One efficient way to store the different alternatives is a word graph.

Word graphs have been successfully applied in speech recognition, for the search process (Ortmanns et al., 1997) and as an interface to other systems (Oerder and Ney, 1993). (Knight and Hatzivassiloglou, 1995) and (Langkilde and Knight, 1998) propose the use of word graphs for natural language generation. In this paper, we are going to present a concept for the generation of word graphs in a machine translation system.

3.2 Bookkeeping

During search, we keep a bookkeeping tree. It is not necessary to keep all the information that we need for the expansion of hypotheses during search in this structure, thus we store only the following:

- the produced target word e,
- the covered source sentence position j,
- a backpointer to the preceding bookkeeping entry.

After the search has finished, i.e. when all source sentence positions have been translated, we trace back the best sentence in the bookkeeping tree.

To generate the N best hypotheses after search, it is not sufficient to simply trace back the complete hypotheses with the highest probabilities in the bookkeeping, because those hypotheses have been recombined. Thus, many hypotheses with a high probability have not been stored.

To overcome this problem, we enhance the bookkeeping concept and generate a word graph as described in Section 3.3.

3.3 Word Graph Structure

If we want to generate a word graph, we have to store both alternatives in the bookkeeping when two hypotheses are recombined. Thus, an entry in the bookkeeping structure may have several backpointers to different preceding entries. The bookkeeping structure is no longer a tree but a network where the source is the bookkeeping entry with zero covered source sentence positions and the sink is a node accounting for complete hypotheses (see Figure 3). This leads us to the concept of word graph nodes and edges containing the following information:

- node
  - the last covered source sentence position j,
- edge
  - the target word e,

– the probabilities according to the different models: the language model and the translation submodels,
– the backpointer to the preceding bookkeeping entry.

After the pruning in beam search, all hypotheses that are no longer active do not have to be kept in the bookkeeping structure. Thus, we can perform **garbage collection** and remove all those bookkeeping entries that cannot be reached from the backpointers of the active hypotheses. This reduces the size of the bookkeeping structure significantly.

An example of a word graph can be seen in Figure 3. To keep the presentation simple, we chose an example without reordering of sentence positions. The words on the edges are the produced target words, and the bitvectors in the nodes show the covered source sentence positions. If an edge is labeled with two words, this means that the first English word has no equivalence in the source sentence, like 'just' and 'have' in Figure 3. The reference translation 'what did you say ?' is contained in the graph, but it has a slightly lower probability than the sentence 'what do you say ?', which is then chosen by the single best search.

The recombination of hypotheses can be seen in the nodes with two or more incoming edges: those hypotheses have been recombined, because they were indistinguishable by translation and language model state.

4 Pruning and Rescoring

4.1 Word Graph Pruning

To study the effect of the word graph size on the translation quality, we produce a conservatively large word graph. Then we apply word graph pruning with a threshold $t < 1$ and study the change of graph error rate (see Section 5). The pruning is based on the beam search concept also used in the single best search: we determine the probability of the best sentence hypothesis in the word graph. All hypotheses in the graph which probability is lower than this maximum probability multiplied with the pruning threshold are discarded.

If the pruning threshold t is zero, the word graph is not pruned at all, and if $t = 1$, we retain only the sentence with maximum probability.

4.2 Word Graph Rescoring

In single best search, a standard trigram language model is used. Search with a bigram language model is much faster, but it yields a lower translation quality. Therefore, we apply a two-pass approach as it was widely used in speech recognition in the past (Ortmanns et al., 1997). This method combines both advantages in the following way: a word graph is constructed using a bigram language model and is then rescored with a trigram language model. The rescoring algorithm is based on dynamic programming; a description can be found in (Ortmanns et al., 1997).

The results of the comparison of the one-pass and the two-pass search are given in Section 5.

4.3 Extraction of N-best Lists

We use A* search for finding the N best sentences in a word graph: starting in the root of the graph, we successively expand the sentence hypotheses. The probability of the partial hypothesis is obtained by multiplying the probabilities of the edges expanded for this sentence. As rest cost estimation, we use the probabilities determined in a backward pass as follows: for each node in the graph, we calculate the probability of a best path from this node to the goal node, i.e. the highest probability for completing a partial hypothesis. This rest cost estimation is perfect because it takes the exact probability as heuristic, i.e. the probability of the partial hypothesis multiplied with the rest cost estimation yields the actual probability of the complete hypothesis. Thus, the N best hypothesis are extracted from the graph without additional overhead of finding sentences with a lower probability.

Of course, the hypotheses must not be recombined during this search. We have to keep every partial hypothesis in the priority queue in order to determine the N best sentences. Otherwise, we might lose one of them by recombination.

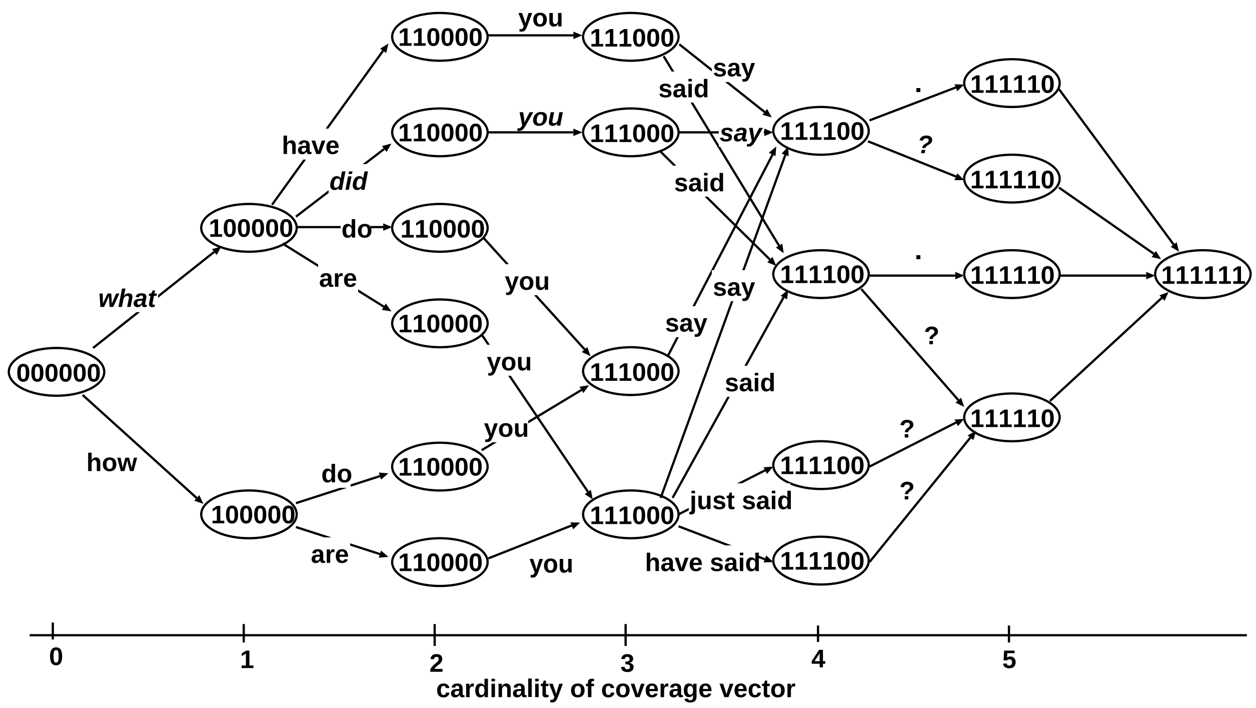

Figure 3: Example of a Word Graph for the German Input Sentence 'was hast du gesagt?' Reference Translation: 'what did you say?'

The details of the algorithm are given in Table 1.

5 Experimental Results

5.1 Corpus and Evaluation

We performed experiments on the German-English Verbmobil corpus. This is a speech translation task in the domain of appointment scheduling and travel planning. The corpus statistics are given in Table 2. The difficulty of this task is that it contains spontaneous speech and thus the syntactic structure of the sentences is far less restricted than in written language.

For evaluation, we use the following error criteria:

- WER (word error rate):
  The word error rate is based on the Levenstein distance. It is computed as the minimum number of substitution, insertion and deletion operations that have to be performed to convert the generated string into the reference string.

- mWER (multi-reference word error rate):
  Often there exist many possible correct translations of a sentence. The WER compares the produced translation only to one given reference which might be insufficient due to variance in syntax. Thus, we built a set of reference translations for each test sentence. For each translation hypothesis, the Levenstein distance to the most similar reference sentence is calculated. This yields a more reliable error measure and is a lower bound of the WER.

- SSER (subjective sentence error rate):
  For a more detailed analysis, subjective judgments by test persons are necessary. Each translated sentence was judged by a human examiner according to an error scale from 0.0 to 1.0. A score of 0.0 means that the translation is semantically and syntactically correct, a score of 0.5 means that a sentence is semantically correct but syntactically wrong and a score of 1.0 means that the sentence is semantically wrong.

Table 1: Algorithm for the Extraction of an N-best List from a Word Graph

input: word graph G

initialization: set $\mathcal{N} = \{goal\ node\ of\ G\}$, $\mathcal{N}_{new} = \emptyset$

while $\mathcal{N}$ is non-empty do

  for each node n in $\mathcal{N}$ do

    for each incoming edge (s, n) do

      $costs(s) = costs(n) + edgecosts((s, n))$

      insert s into $\mathcal{N}_{new}$

  $\mathcal{N} = \mathcal{N}_{new}$, $\mathcal{N}_{new} = \emptyset$

perform A* search with $costs(n)$ as rest cost estimation

Table 2: Training and Test Corpus Statistics

		German	English
Training	Sentences	58 073	
	Words	519 523	549 921
	Words without Punctuation Marks	418 979	453 632
Vocabulary	Size	7 911	4 648
	Singletons	3 453	1 699
Test	Sentences	251	
	Words	2 627	2 866
	Bigram/Trigram Perplexity	-	39.3/30.7

- GER (graph error rate):
  The graph error rate is computed by determining that sentence in the word graph that has the minimum Levenstein distance to a given reference. Thus, it is a lower bound for the word error rate and gives a measurement of what can be achieved by rescoring with more complex models.
  The calculation of the graph error rate is performed by a dynamic programming based algorithm. Its space complexity is the number of graph nodes times the length of the reference translation.

5.2 Effect of Word Graph Pruning

In our experiments, we varied the word graph pruning threshold in order to obtain word graphs of different densities, i.e. different numbers of hypotheses. The word graph density is computed as the total number of word graph edges divided by the number of reference sentence words – analogously to the word graph density in speech recognition.

The effect of pruning on the graph error rate is shown in Table 3. The value of the pruning threshold is given as the negative logarithm of the probability. Thus, $t = 0$ refers to pruning everything but the best hypothesis.

Figure 4 shows the change in graph error rate in relation to the average graph density. We see that for graph densities up to 200, the graph error rate significantly changes if the graph is enlarged. The saturation point of the GER lies at 13% and is reached for an average graph density about 1000 which relates to a pruning threshold of 20.

Table 4: Length of N-best List and the Effect on the Word Error Rate

N	1	5	10	20	50	100	200	500	1000	1500	2000
WER	41.07	33.96	31.31	29.33	26.72	24.97	23.27	21.56	20.24	19.64	19.30

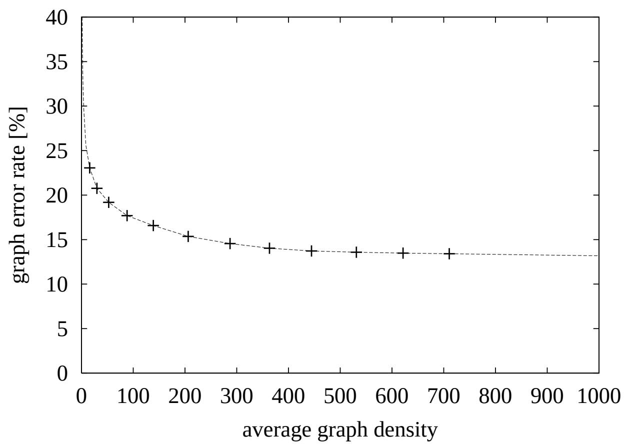

Figure 4: Effect of the Average Graph Density on the GER

Table 3: Graph Density and Graph Error Rate for Various Pruning Thresholds

Pruning Threshold t	Graph Density	GER
1.0	2.26	34.52
2.0	4.38	29.78
3.0	8.37	25.60
4.0	15.91	23.06
5.0	29.79	20.76
7.0	88.02	17.69
10.0	287.05	14.56
12.0	444.51	13.72
15.0	710.61	13.41
20.0	1060.39	13.13
50.0	1282.54	13.10
100.0	1282.57	13.10

5.3 Error Rates for N-best Lists

We studied the relation of the length of an N-best list extracted from the word graph to the translation quality of the hypotheses contained in it. For each sentence in the N-best list, we computed the word error rate and determined the minimum over the list. Table 4 shows the results. We see that the error rate decreases significantly for growing length of the list. For N larger than 1000, there is no significant change in WER even if N is doubled. From this, we conclude that rescoring on word graphs has to be preferred to rescoring on N-best lists, because a word graph leaves more room for improvement: the GER decreases down to 13%, and even graphs with a density below 100 contain better hypotheses than a 2000-best list.

5.4 Effect of Trigram Rescoring

To investigate the effect of a one-pass search versus language model rescoring, we performed a single best search with a trigram language model and constructed a word graph with a bigram language model. The graph was then rescored using a trigram language model. Table 5 shows the error rates and the computation time for the two approaches. The time given for the two-pass search is the time needed for the word graph construction plus the time needed for trigram rescoring. The two-pass search is around 3.5 times faster than the one-pass search.

In order to make the results comparable, we applied the same pruning for the integrated trigram search and the word graph construction with the bigram. The WER and mWER for the two-pass approach are slightly higher than those for the integrated search. This is due to the fact that the bigram search misses some of the good translations that are found by the trigram search. This can be compensated by constructing a larger word graph. The subjective sentence error rate even slightly decreases for the two-pass search. The degradation in WER and mWER is not significant.

Table 5: Error Rates [%] and CPU Time for One-Pass and Two-Pass Search

		Integrated Trigram Search	Bigram Search + Trigram Rescoring
Error Rates	WER	41.41	42.70
	mWER	36.46	37.26
	SSER	36.90	35.54
CPU Time	sec./sentence	10.75	2.77+0.19

6 Conclusion

We have presented a concept for constructing word graphs for statistical machine translation by extending the single best search algorithm. Experiments have shown that the graph error rate significantly decreases for rising word graph densities. The quality of the hypotheses contained in a word graph is better than of those in an N-best list. This indicates that word graph rescoring can yield a significant gain in translation quality. For the future, we plan the application of refined translation and language models for rescoring on word graphs.

References

Peter F. Brown, Stephen A. Della Pietra, Vincent J. Della Pietra, and Robert L. Mercer. 1993. The mathematics of statistical machine translation: Parameter estimation. *Computational Linguistics*, 19(2):263–311.

Kevin Knight and Vasileios Hatzivassiloglou. 1995. Two-level, many-paths generation. In *Proceedings of the Conference of the Association for Computational Linguistics*, pages 252–260, Cambridge, MA, June.

Irene Langkilde and Kevin Knight. 1998. The practical value of n-grams in generation. In *Proceedings of the International Natural Language Generation Workshop*, pages 248–255, Ontario, Canada, August.

Franz J. Och, Nicola Ueffing, and Hermann Ney. 2001. An efficient A* search algorithm for statistical machine translation. In *ACL-EACL-2001: 39th Annual Meeting of the Association for Computational Linguistics - joint with EACL 2001: Proceedings of the Workshop on Data-Driven Machine Translation*, pages 55–62, Toulouse, France, July.

Martin Oerder and Hermann Ney. 1993. Word graphs: An efficient interface between continous speech recognition and language understanding. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, volume 2, pages 119–122, Minneapolis, MN, April.

Stefan Ortmanns, Hermann Ney, and Xavier Aubert. 1997. A word graph algorithm for large vocabulary continuous speech recognition. *Computer, Speech and Language*, 11(1):43–72, January.

Christoph Tillmann and Hermann Ney. 2000. Word re-ordering and DP-based search in statistical machine translation. In *COLING '00: The 18th Int. Conf. on Computational Linguistics*, pages 850–856, Saarbrücken, Germany, July.

Christoph Tillmann and Hermann Ney. 2002. Word re-ordering and DP beam search for statistical machine translation. *to appear in Computational Linguistics.*