Decoding Algorithm in Statistical Machine Translation

Ye-Yi Wang and Alex Waibel
Language Technology Institute
School of Computer Science
Carnegie Mellon University
5000 Forbes Avenue
Pittsburgh, PA 15213, USA
{yyw,waibel}@cs.cmu.edu

Abstract

Decoding algorithm is a crucial part in statistical machine translation. We describe a stack decoding algorithm in this paper. We present the hypothesis scoring method and the heuristics used in our algorithm. We report several techniques deployed to improve the performance of the decoder. We also introduce a simplified model to moderate the sparse data problem and to speed up the decoding process. We evaluate and compare these techniques/models in our statistical machine translation system.

1 Introduction

1.1 Statistical Machine Translation

Statistical machine translation is based on a channel model. Given a sentence **T** in one language (German) to be translated into another language (English), it considers **T** as the target of a communication channel, and its translation **S** as the source of the channel. Hence the machine translation task becomes to recover the source from the target. Basically every English sentence is a possible source for a German target sentence. If we assign a probability $P(\mathbf{S} \mid \mathbf{T})$ to each pair of sentences (**S**, **T**), then the problem of translation is to find the source **S** for a given target **T**, such that $P(\mathbf{S} \mid \mathbf{T})$ is the maximum. According to Bayes rule,

$$P(\mathbf{S} \mid \mathbf{T}) = \frac{P(\mathbf{S})P(\mathbf{T} \mid \mathbf{S})}{P(\mathbf{T})} \quad (1)$$

Since the denominator is independent of **S**, we have

$$\hat{\mathbf{S}} = \arg\max_{\mathbf{S}} P(\mathbf{S})P(\mathbf{T} \mid \mathbf{S}) \quad (2)$$

Therefore a statistical machine translation system must deal with the following three problems:

- Modeling Problem: How to depict the process of generating a sentence in a source language, and the process used by a channel to generate a target sentence upon receiving a source sentence? The former is the problem of language modeling, and the later is the problem of translation modeling. They provide a framework for calculating $P(\mathbf{S})$ and $P(\mathbf{T} \mid \mathbf{S})$ in (2).
- Learning Problem: Given a statistical language model $P(\mathbf{S})$ and a statistical translation model $P(\mathbf{T} \mid \mathbf{S})$, how to estimate the parameters in these models from a bilingual corpus of sentences?
- Decoding Problem: With a fully specified (framework and parameters) language and translation model, given a target sentence **T**, how to efficiently search for the source sentence $\hat{\mathbf{S}}$ that satisfies (2).

The modeling and learning issues have been discussed in (Brown et al., 1993), where ngram model was used for language modeling, and five different translation models were introduced for the translation process. We briefly introduce the model 2 here, for which we built our decoder.

In model 2, upon receiving a source English sentence $\mathbf{e} = e_1, \cdots, e_l$, the channel generates a German sentence $\mathbf{g} = g_1, \cdots, g_m$ at the target end in the following way:

1. With a distribution $P(m \mid \mathbf{e})$, randomly choose the length m of the German translation **g**. In model 2, the distribution is independent of m and **e**:
$$P(m \mid \mathbf{e}) = \epsilon$$
where ϵ is a small, fixed number.
2. For each position i $(0 < i \leq m)$ in **g**, find the corresponding position a_i in **e** according to an *alignment* distribution $P(a_i \mid i, a_1^{i-1}, m, \mathbf{e})$. In model 2, the distribution only depends on i, a_i and the length of the English and German sentences:
$$P(a_i \mid i, a_1^{i-1}, m, \mathbf{e}) = a(a_i \mid i, m, l)$$
3. Generate the word g_i at the position i of the German sentence from the English word e_{a_i} at

the aligned position a_i of g_i, according to a *translation* distribution $P(g_i \mid a_1^m, g_1^{i-1}, \mathrm{e}) = t(g_i \mid e_{a_i})$. The distribution here only depends on g_i and e_{a_i}.

Therefore, $P(\mathrm{g} \mid \mathrm{e})$ is the sum of the probabilities of generating g from e over all possible alignments A, in which the position i in the target sentence g is aligned to the position a_i in the source sentence e:

$$P(\mathrm{g} \mid \mathrm{e}) = \epsilon \sum_{a_1=0}^{l} \cdots \sum_{a_m=0}^{l} \prod_{j=1}^{m} t(g_j \mid e_{a_j}) a(a_j \mid j, l, m) = \epsilon \prod_{j=1}^{m} \sum_{i=0}^{l} t(g_j \mid e_i) a(i \mid j, l, m) \qquad (3)$$

(Brown et al., 1993) also described how to use the EM algorithm to estimate the parameters $a(i \mid j, l, m)$ and $t(g \mid e)$ in the aforementioned model.

1.2 Decoding in Statistical Machine Translation

(Brown et al., 1993) and (Vogel, Ney, and Tillman, 1996) have discussed the first two of the three problems in statistical machine translation. Although the authors of (Brown et al., 1993) stated that they would discuss the search problem in a follow-up article, so far there have no publications devoted to the decoding issue for statistical machine translation. On the other side, decoding algorithm is a crucial part in statistical machine translation. Its performance directly affects the quality and efficiency of translation. Without a good and efficient decoding algorithm, a statistical machine translation system may miss the best translation of an input sentence even if it is perfectly predicted by the model.

2 Stack Decoding Algorithm

Stack decoders are widely used in speech recognition systems. The basic algorithm can be described as following:

1. Initialize the stack with a null hypothesis.
2. Pop the hypothesis with the highest score off the stack, name it as **current-hypothesis**.
3. if **current-hypothesis** is a complete sentence, output it and terminate.
4. extend **current-hypothesis** by appending a word in the lexicon to its end. Compute the score of the new hypothesis and insert it into the stack. Do this for all the words in the lexicon.
5. Go to 2.

2.1 Scoring the hypotheses

In stack search for statistical machine translation, a hypothesis H includes (a) the length l of the source sentence, and (b) the prefix words in the sentence. Thus a hypothesis can be written as $H = l : e_1 e_2 \cdots e_k$, which postulates a source sentence of length l and its first k words. The score of H, f_H, consists of two parts: the prefix score g_H for $e_1 e_2 \cdots e_k$ and the heuristic score h_H for the part $e_{k+1} e_{k+2} \cdots e_l$ that is yet to be appended to H to complete the sentence.

2.1.1 Prefix score g_H

(3) can be used to assess a hypothesis. Although it was obtained from the alignment model, it would be easier for us to describe the scoring method if we interpret the last expression in the equation in the following way: each word e_i in the hypothesis contributes the amount $\epsilon\, t(g_j \mid e_i) a(i \mid j, l, m)$ to the probability of the target sentence word g_j. For each hypothesis $H = l : e_1, e_2, \cdots, e_k$, we use $S_H(j)$ to denote the probability mass for the target word g_i contributed by the words in the hypothesis:

$$S_H(j) = \epsilon \sum_{i=0}^{k} t(g_j \mid e_i) a(i \mid j, l, m) \qquad (4)$$

Extending H with a new word will increase $S_H(j), 1 \le j \le m$.

To make the score additive, the logarithm of the probability in (3) was used. So the prefix score contributed by the translation model is $\sum_{j=0}^{m} \log S_H(j)$.

Because our objective is to maximize $P(\mathrm{e}, \mathrm{g})$, we have to include as well the logarithm of the language model probability of the hypothesis in the score, therefore we have

$$\begin{aligned} g_H &= \sum_{j=0}^{m} \log S_H(j) + \\ &\quad \sum_{i=0}^{k} \log P(e_i \mid e_{i-N+1} \cdots e_{i-1}). \end{aligned}$$

here N is the order of the ngram language model.

The above g-score g_H of a hypothesis $H = l : e_1 e_2 \cdots e_k$ can be calculated from the g-score of its parent hypothesis $P = l : e_1 e_2 \cdots e_{k-1}$:

$$\begin{aligned} g_H &= g_P + \log P(e_k \mid e_{k-N+1} \cdots e_{k-1}) \\ &\quad + \sum_{j=0}^{m} \log[1 + \frac{\epsilon\, t(g_j \mid e_k) a(k \mid j, l, m)}{S_P(j)}] \\ S_H(j) &= S_P(j) + \epsilon\, t(g_j \mid e_k) a(k \mid j, l, m) \end{aligned} \qquad (5)$$

A practical problem arises here. For a many early stage hypothesis P, $S_P(j)$ is close to 0. This causes problems because it appears as a denominator in (5) and the argument of the log function when calculating g_P. We dealt with this by either limiting the translation probability from the null word (Brown et al., 1993) at the hypothetical 0-position(Brown et al., 1993) over a threshold during the EM training, or setting $S_{H_0}(j)$ to a small probability π instead of 0 for the initial null hypothesis H_0. Our experiments show that $\pi = 10^{-4}$ gives the best result.

2.1.2 Heuristics

To guarantee an optimal search result, the heuristic function must be an upper-bound of the score for all possible extensions $e_{k+1}e_{k+2}\cdots e_l$(Nilsson, 1971) of a hypothesis. In other words, the benefit of extending a hypothesis should never be underestimated. Otherwise the search algorithm will conclude prematurely with a non-optimal hypothesis. On the other hand, if the heuristic function overestimates the merit of extending a hypothesis too much, the search algorithm will waste a huge amount of time after it hits a correct result to safeguard the optimality.

To estimate the language model score h_H^{LM} of the unrealized part of a hypothesis, we used the negative of the language model perplexity PP_{train} on the training data as the logarithm of the average probability of predicting a new word in the extension from a history. So we have

$$h_H^{LM} = -(l-k)PP_{train} + C. \qquad (6)$$

Here is the motivation behind this. We assume that the perplexity on training data overestimates the likelihood of the forthcoming word string **on average**. However, when there are only a few words to be extended (k is close to l), the language model probability for the words to be extended may be much higher than the average. This is why the constant term C was introduced in (6). When $k \ll l$, $-(l-k)PP_{train}$ is the dominating term in (6), so the heuristic language model score is close to the average. This can avoid overestimating the score too much. As k is getting closer to l, the constant term C plays a more important role in (6) to avoid underestimating the language model score. In our experiments, we used $C = PP_{train} + log(P_{max})$, where P_{max} is the maximum ngram probability in the language model.

To estimate the translation model score, we introduce a variable $v_{il}(j)$, the maximum contribution to the probability of the target sentence word g_j from any possible source language words at any position between i and l:

$$v_{il}(j) = \max_{i \le k \le l, e \in L_E} t(g_j \mid e)a(k \mid j, l, m). \qquad (7)$$

here L_E is the English lexicon.

Since $v_{il}(j)$ is independent of hypotheses, it only needs to be calculated once for a given target sentence.

When $k < l$, the heuristic function for the hypothesis $H = l : e_1e_2\cdots e_k$, is

$$\begin{aligned} h_H &= \sum_{j=1}^{m} \max\{0, \log(v_{(k+1)l}(j)) - \log S_H(j)\} \\ &\quad -(l-k)PP_{train} + C \end{aligned} \qquad (8)$$

where $\log(v_{(k+1)l}(j)) - \log S_H(j))$ is the maximum increasement that a new word can bring to the likelihood of the j-th target word.

When $k = l$, since no words can be appended to the hypothesis, it is obvious that $h_H = 0$.

This heuristic function over-estimates the score of the upcoming words. Because of the constraints from language model and from the fact that a position in a source sentence cannot be occupied by two different words, normally the placement of words in those unfilled positions cannot maximize the likelihood of all the target words simultaneously.

2.2 Pruning and aborting search

Due to physical space limitation, we cannot keep all hypotheses alive. We set a constant M, and whenever the number of hypotheses exceeds M, the algorithm will prune the hypotheses with the lowest scores. In our experiments, M was set to 20,000.

There is time limitation too. It is of little practical interest to keep a seemingly endless search alive too long. So we set a constant T, whenever the decoder extends more than T hypotheses, it will abort the search and register a failure. In our experiments, T was set to 6000, which roughly corresponded to 2 and half hours of search effort.

2.3 Multi-Stack Search

The above decoder has one problem: since the heuristic function overestimates the merit of extending a hypothesis, the decoder always prefers hypotheses of a long sentence, which have a better chance to maximize the likelihood of the target words. The decoder will extend the hypothesis with large l first, and their children will soon occupy the stack and push the hypotheses of a shorter source sentence out of the stack. If the source sentence is a short one, the decoder will never be able to find it, for the hypotheses leading to it have been pruned permanently.

This "incomparable" problem was solved with multi-stack search(Magerman, 1994). A separate stack was used for each hypothesized source sentence length l. We do compare hypotheses in different stacks in the following cases. First, we compare a complete sentence in a stack with the hypotheses in other stacks to safeguard the optimality of search result; Second, the top hypothesis in a stack is compared with that of another stack. If the difference is greater than a constant δ, then the less probable one will not be extended. This is called soft-pruning, since whenever the scores of the hypotheses in other stacks go down, this hypothesis may revive.

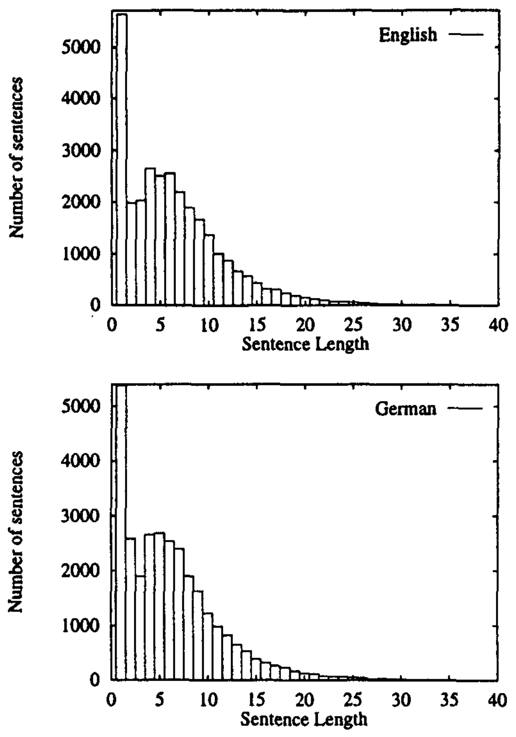

Figure 1: Sentence Length Distribution

3 Stack Search with a Simplified Model

In the IBM translation model 2, the alignment parameters depend on the source and target sentence length l and m. While this is an accurate model, it causes the following difficulties:

1. there are too many parameters and therefore too few training data per parameter. This may not be a problem when massive training data are available. However, in our application, this is a severe problem. Figure 1 plots the length distribution for the English and German sentences. When sentences get longer, there are fewer training data available.
2. the search algorithm has to make multiple hypotheses of different source sentence length. For each source sentence length, it searches through almost the same prefix words and finally settles on a sentence length. This is a very time consuming process and makes the decoder very inefficient.

To solve the first problem, we adjusted the count for the parameter $a(i \mid j,l,m)$ in the EM parameter estimation by adding to it the counts for the parameters $a(i \mid j,l',m')$, assuming (l,m) and (l',m') are close enough. The closeness were measured in

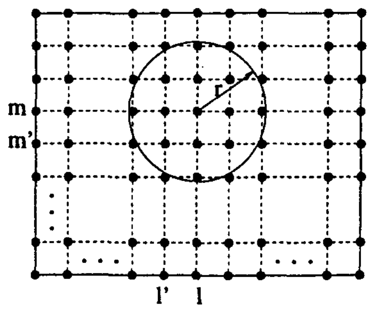

Figure 2: Each x/y position represents a different source/target sentence length. The dark dot at the intersection (l,m) corresponds to the set of counts for the alignment parameters $a(\bullet \mid \bullet, l, m)$ in the EM estimation. The adjusted counts are the sum of the counts in the neighboring sets residing inside the circle centered at (l,m) with radius r. We took $r=3$ in our experiment.

Euclidean distance (Figure 2). So we have

$$\hat{c}(i \mid j,l,m) = \sum_{(l-l')^2+(m-m')^2<r^2;\mathrm{e},\mathrm{g}} c(i \mid j,l',m';\mathrm{e},\mathrm{g}) \quad (9)$$

where $\hat{c}(i \mid j,l,m)$ is the adjusted count for the parameter $a(i \mid j,l,m)$, $c(i \mid j,l,m;\mathrm{e},\mathrm{g})$ is the expected count for $a(i \mid j,l,m)$ from a paired sentence (e g), and $c(i \mid j,l,m;\mathrm{e},\mathrm{g}) = 0$ when $|\mathrm{e}| \neq l$, or $|\mathrm{g}| \neq m$, or $i > l$, or $j > m$.

Although (9) can moderate the severity of the first data sparse problem, it does not ease the second inefficiency problem at all. We thus made a radical change to (9) by removing the precondition that (l,m) and (l',m') must be close enough. This results in a simplified translation model, in which the alignment parameters are independent of the sentence length l and m:

$$\begin{aligned} P(i \mid j,m,\mathrm{e}) &= P(i \mid j,l,m) \\ &= a(i \mid j) \end{aligned}$$

here $i,j < L_m$, and L_m is the maximum sentence length allowed in the translation system. A slight change to the EM algorithm was made to estimate the parameters.

There is a problem with this model: given a sentence pair g and e, when the length of e is smaller than L_m, then the alignment parameters do not sum to 1:

$$\sum_{i=0}^{|\mathrm{e}|} a(i \mid j) < 1. \quad (10)$$

We deal with this problem by padding e to length L_m with dummy words that never gives rise to any word in the target of the channel.

Since the parameters are independent of the source sentence length, we do not have to make an assumption about the length in a hypothesis. Whenever a hypothesis ends with the sentence end symbol </s> and its score is the highest, the decoder reports it as the search result. In this case, a hypothesis can be expressed as $H = e_1, e_2, \cdots, e_k$, and $|H|$ is used to denote the length of the sentence prefix of the hypothesis H, in this case, k.

3.1 Heuristics

Since we do not make assumption of the source sentence length, the heuristics described above can no longer be applied. Instead, we used the following heuristic function:

$$\begin{aligned} h_H^n &= \sum_{j=1}^{m} \max\{0, \log(\frac{v_{(|H|+1)(|H|+n)}(j)}{S_H(j)})\} \\ &\quad -n * PP_{train} + C \qquad (11) \\ h_H &= \sum_{n=1}^{L_m-|H|} P_p(|H|+n \,|\, m) * h_H^n \qquad (12) \end{aligned}$$

here h_H^n is the heuristics for the hypothesis that extend H with n more words to complete the source sentence (thus the final source sentence length is $|H|+n$.) $P_p(x \,|\, y)$ is the Poisson distribution of the source sentence length conditioned on the target sentence length. It is used to calculate the mean of the heuristics over all possible source sentence length. m is the target sentence length. The parameters of the Poisson distributions can be estimated from training data.

4 Implementation

Due to historical reasons, stack search got its current name. Unfortunately, the requirement for search states organization is far beyond what a stack and its *push pop* operations can handle. What we really need is a dynamic set which supports the following operations:

1. *INSERT:* to insert a new hypothesis into the set.
2. *DELETE:* to delete a state in hard pruning.
3. *MAXIMUM:* to find the state with the best score to extend.
4. *MINIMUM:* to find the state to be pruned.

We used the Red-Black tree data structure (Cormen, Leiserson, and Rivest, 1990) to implement the dynamic set, which guarantees that the above operations take $O(\log n)$ time in the worst case, where n is the number of search states in the set.

5 Performance

We tested the performance of the decoders with the scheduling corpus(Suhm et al., 1995). Around 30,000 parallel sentences (400,000 words altogether for both languages) were used to train the IBM model 2 and the simplified model with the EM algorithm. A larger English monolingual corpus with around 0.5 million words was used to train a bigram for language modelling. The lexicon contains 2,800 English and 4,800 German words in morphologically inflected form. We did not do any preprocessing/analysis of the data as reported in (Brown et al., 1992).

5.1 Decoder Success Rate

Table 1 shows the success rate of three models/decoders. As we mentioned before, the comparison between hypotheses of different sentence length made the single stack search for the IBM model 2 fail (return without a result) on a majority of the test sentences. While the multi-stack decoder improved this, the simplified model/decoder produced an output for all the 120 test sentences.

5.2 Translation Accuracy

Unlike the case in speech recognition, it is quite arguable what "accurate translations" means. In speech recognition an output can be compared with the sample transcript of the test data. In machine translation, a sentence may have several legitimate translations. It is difficult to compare an output from a decoder with a designated translation. Instead, we used human subjects to judge the machine-made translations. The translations are classified into three categories[1].

1. Correct translations: translations that are grammatical and convey the same meaning as the inputs.
2. Okay translations: translations that convey the same meaning but with small grammatical mistakes or translations that convey most but not the entire meaning of the input.
3. Incorrect translations: Translations that are ungrammatical or convey little meaningful information or the information is different from the input.

Examples of correct, okay, and incorrect translations are shown in Table 2.

Table 3 shows the statistics of the translation results. The accuracy was calculate by crediting a correct translation 1 point and an okay translation 1/2 point.

There are two different kinds of errors in statistical machine translation. A *modeling error* occurs when the model assigns a higher score to an incorrect translation than a correct one. We cannot do anything about this with the decoder. A *decoding*

[1]This is roughly the same as the classification in IBM statistical translation, except we do not have "legitimate translation that conveys different meaning from the input" — we did not observed this case in our outputs.

	Total Test Sentences	Decoded Sentenced	Failed sentences
Model 2, Single Stack	120	32	88
Model 2, Multi-Stack	120	83	37
Simplified Model	120	120	0

Table 1: Decoder Success Rate

Correct	German	ich habe ein Meeting von halb zehn bis um zwölf
	English (target)	I have a meeting from nine thirty to twelve
	English (output)	I have a meeting from nine thirty to twelve
	German	versuchen wir sollten es vielleicht mit einem anderen Termin
	English (target)	we might want to try for some other time
	English (output)	we should try another time
Okay	German	ich glaube nicht däs ich noch irgend etwas im Januar frei habe
	English (target)	I do not think I have got anything open in January
	English (output)	I think I will not free in January
	German	ich glaube wir sollten ein weiteres Meeting vereinbaren
	English (target)	I think we have to have another meeting
	English (output)	I think we should fix a meeting
Incorrect	German	schlagen Sie doch einen Termin vor
	English (target)	why don't you suggest a time
	English (output)	why you an appointment
	German	ich habe Zeit für den Rest des Tages
	English (target)	I am free the rest of it
	English (output)	I have time for the rest of July

Table 2: Examples of Correct, Okay, and Incorrect Translations: for each translation, the first line is an input German sentence, the second line is the human made (target) translation for that input sentence, and the third line is the output from the decoder.

error or *search error* happens when the search algorithm misses a correct translation with a higher score.

When evaluating a decoding algorithm, it would be attractive if we can tell how many errors are caused by the decoder. Unfortunately, this is not attainable. Suppose that we are going to translate a German sentence g, and we know from the sample that e is one of its possible English translations. The decoder outputs an incorrect e′ as the translation of g. If the score of e′ is lower than that of e, we know that a search error has occurred. On the other hand, if the score of e′ is higher, we cannot decide if it is a modeling error or not, since there may still be other legitimate translations with a score higher than e′ — we just do not know what they are.

Although we cannot distinguish a modeling error from a search error, the comparison between the decoder output's score and that of a sample translation can still reveal some information about the performance of the decoder. If we know that the decoder can find a sentence with a better score than a "correct" translation, we will be more confident that the decoder is less prone to cause errors. Table 4 shows the comparison between the score of the outputs from the decoder and the score of the sample translations when the outputs are incorrect. In most cases, the incorrect outputs have a higher score than the sample translations. Again, we consider a "okay" translation a half error here.

This result hints that model deficiencies may be a major source of errors. The models we used here are very simple. With a more sophisticated model, more training data, and possibly some preprocessing, the total error rate is expected to decrease.

5.3 Decoding Speed

Another important issue is the efficiency of the decoder. Figure 3 plots the average number of states being extended by the decoders. It is grouped according to the input sentence length, and evaluated on those sentences on which the decoder succeeded.

The average number of states being extended in the model 2 single stack search is not available for long sentences, since the decoder failed on most of the long sentences.

The figure shows that the simplified model/decoder works much more efficiently than the other mod-

	Total	Correct	Okay	Incorrect	Accuracy
Model 2, Multi-Stack	83	39	12	32	54.2%
Simplified Model	120	64	15	41	59.6%

Table 3: Translation Accuracy

	Total Errors	$Score_e \geq Score_{e'}$	$Score_e < Score_{e'}$
Model 2, Multi-Stack	38	3.5 (7.9%)	34.5 (92.1%)
Simplified Model	48.5	4.5 (9.3%)	44 (90.7%)

Table 4: Sample Translations versus Machine-Made Translations

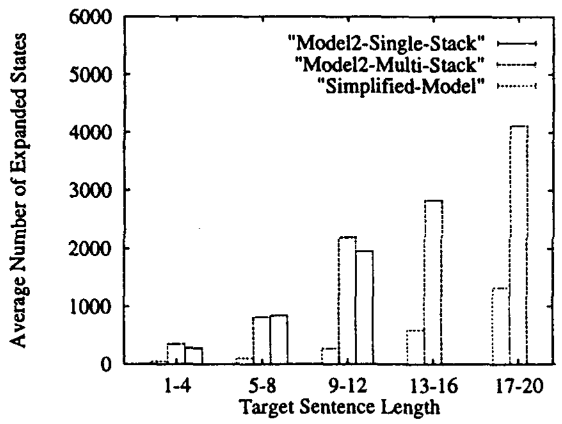

Figure 3: Extended States versus Target Sentence Length

els/decoders.

6 Conclusions

We have reported a stack decoding algorithm for the IBM statistical translation model 2 and a simplified model. Because the simplified model has fewer parameters and does not have to posit hypotheses with the same prefixes but different length, it outperformed the IBM model 2 with regard to both accuracy and efficiency, especially in our application that lacks a massive amount of training data. In most cases, the erroneous outputs from the decoder have a higher score than the human made translations. Therefore it is less likely that the decoder is a major contributor of translation errors.

7 Acknowledgements

We would like to thank John Lafferty for enlightening discussions on this work. We would also like to thank the anonymous ACL reviewers for valuable comments. This research was partly supported by ATR and the Verbmobil Project. The views and conclusions in this document are those of the authors.

References

Brown, P. F., S. A. Della-Pietra, V. J Della-Pietra, and R. L. Mercer. 1993. The Mathematics of Statistical Machine Translation: Parameter Estimation. *Computational Linguistics*, 19(2):263–311.

Brown, P. F., S. A. Della Pietra, V. J. Della Pietra, J. D. Lafferty, and R. L. Mercer. 1992. Analysis, Statistical Transfer, and Synthesis in Machine Translation. In *Proceedings of the fourth International Conference on Theoretical and Methodological Issues in Machine Translation*, pages 83–100.

Cormen, Thomas H., Charles E. Leiserson, and Ronald L. Rivest. 1990. *Introduction to Algorithms.* The MIT Press, Cambridge, Massachusetts.

Magerman, D. 1994. *Natural Language Parsing as Statistical Pattern Recognition.* Ph.D. thesis, Stanford University.

Nilsson, N. 1971. *Problem-Solving Methods in Artificial Intelligence.* McGraw Hill, New York, New York.

Suhm, B., P.Geutner, T. Kemp, A. Lavie, L. Mayfield, A. McNair, I. Rogina, T. Schultz, T. Sloboda, W. Ward, M. Woszczyna, and A. Waibel. 1995. JANUS: Towards multilingual spoken language translation. In *Proceedings of the ARPA Speech Spoken Language Technology Workshop, Austin, TX, 1995.*

Vogel, S., H. Ney, and C. Tillman. 1996. HMM-Based Word Alignment in Statistical Translation. In *Proceedings of the Seventeenth International Conference on Computational Linguistics: COLING-96*, pages 836-841, Copenhagen, Denmark.