Pharaoh: A Beam Search Decoder for Phrase-Based Statistical Machine Translation Models

Philipp Koehn

Computer Science and Artificial Intelligence Laboratory
Massachusetts Institute of Technology
koehn@csail.mit.edu

Abstract. We describe Pharaoh, a freely available decoder for phrase-based statistical machine translation models. The decoder is the implementation of an efficient dynamic programming search algorithm with lattice generation and XML markup for external components.

Statistical machine translation has emerged as a very active research field, which has already spawned a number of commercial systems that define the state of the art translation performance for some language pairs.

Recently, phrase-based methods have been shown to produce the best translation performance. Systems that follow this approach have been developed at RWTH Aachen [Zens et al., 2002], CMU [Vogel et al., 2003], IBM [Tillmann, 2003], ISI [Koehn et al., 2003], and JHU [Kumar and Byrne, 2004].

The field's active research community has been nurtured by the availability of standard tools, most notably GIZA++ [Och and Ney, 2003] and the Rewrite Decoder [Germann, 2003], the training and decoding tools for IBM Model 4 word-based translation models. Here, we describe Pharaoh, a freely available[1] translation engine for phrase-based statistical machine translation models.

1 Phrase-Based Standard Model

We will now define the phrase-based statistical machine translation standard model that covers roughly all cited phrase-based approaches.

Figure 1 on the following page illustrates the process of phrase-based translation. The input is segmented into a number of sequences of consecutive words (so-called "phrases"). Each phrase is translated into an English phrase, and English phrases in the output may be reordered.

Or, more formally: During decoding, the foreign input sentence f is segmented into a sequence of I phrases $\bar{f}_1^I$. Each foreign phrase $\bar{f}_i$ in $\bar{f}_1^I$ is translated into an English phrase $\bar{e}_i$. The English phrases may be reordered. Phrase translation is modeled by a translation model $p(\bar{f}_i, \bar{e}_i)$.

[1] available at http://www.isi.edu/licensed-sw/pharaoh/

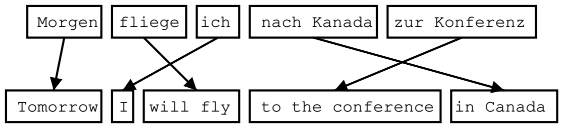

Fig. 1. Phrase-based machine translation: input is segmented in phrases, each phrase is translated and may be reordered.

Originally, the definition of machine translation models was motivated by the noisy channel model. The application of the Bayes rule $\text{argmax}_e p(e|f) = \text{argmax}_e p(f|e)p(e)$ allows the reformulation of the translation process from a source language (or "foreign", f) into a target language (or "English", e) into a translation component $p(f|e)$ and a separate language model component $p(e)$.

Recent empirical results do not always justify the mathematical inversion of the translation direction — $p(e|f)$ into $p(f|e)$ — and also suggest that a number of additional feature functions may be helpful [Och and Ney, 2002].

The standard model of phrase-based statistical machine translation is a log-linear model $\text{argmax}_e p(e|f) = exp(\sum_i \lambda_i h_i(e,f))$ that uses a number of feature functions h_i with weights λ_i. The feature functions are typically a language model, reordering model, word penalty, and various translation models (phrase translation probability, lexical translation probability, etc.).

There is currently no agreement on what is the best method to train such phrase-translation models. Typically, given a large (10-150 million words) parallel corpus of translated material, first a word alignment is established (with GIZA++), then phrase translations are extracted, and finally a scoring function is defined – which could be derived from maximum likelihood phrase translation probability estimation, lexical translation probabilities, etc. Alternatively, a parallel corpus could be directly phrase-aligned [Marcu and Wong, 2002].

2 Decoder

We will now describe the search algorithm of Pharaoh. More detailed information is given by Koehn [2003]. The decoder implements a beam search and is roughly similar to work by Tillmann [2001] and Och [2002]. In fact, by reframing Och's alignment template model as a phrase translation model, the decoder is also suitable for his model (without the use of word classes).

We start with defining the concept of translation options, describe the basic mechanism of beam search, and its necessary components. We continue with details on word lattice generation and XML markup as interface for external components.

Translation Options — Given an input string of words, a number of phrase translations could be applied. We call each such applicable phrase translation a *translation option*. This is illustrated in Figure 2, where a number of phrase

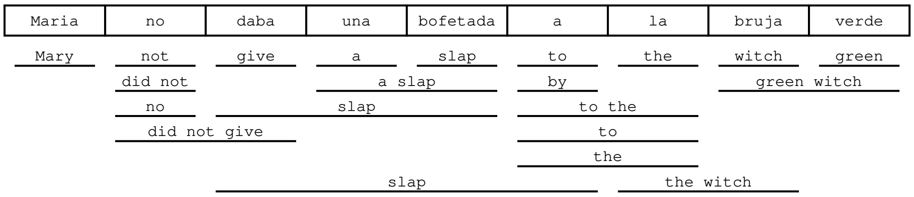

Fig. 2. Some translation options for the Spanish input sentence *Maria no daba una bofetada a la bruja verde*

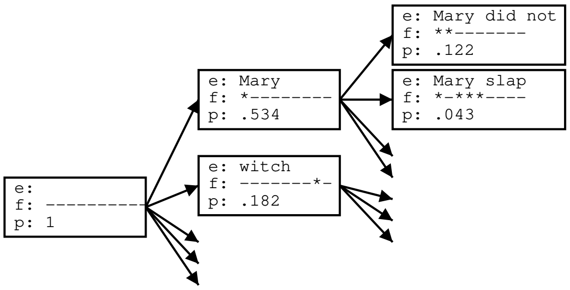

Fig. 3. State expansion in the beam decoder: in each expansion English words are generated, additional foreign words are covered (marked by *), and the probability cost so far is adjusted. In this example the input sentence is *Maria no daba una bofetada a la bruja verde.*

translations for the Spanish input sentence *Maria no daba una bofetada a la bruja verde* are given.

Translation options are collected before any decoding takes place. This allows a quicker lookup than consulting the phrase translation table during decoding.

Core Algorithm — The phrase-based decoder we developed employs a beam search algorithm, similar to the one by Jelinek [1998] for speech recognition. The English output sentence is generated left to right in form of hypotheses.

This process is illustrated in Figure 3. Starting from the initial hypothesis, the first expansion is the foreign word *Maria*, which is translated as *Mary*. The foreign word is marked as translated (marked by an asterisk). We may also expand the initial hypothesis by translating the foreign word *bruja* as *witch*.

We can generate new hypotheses from these expanded hypotheses. Given the first expanded hypothesis we generate a new hypothesis by translating *no* with *did not*. Now the first two foreign words *Maria* and *no* are marked as being covered. Following the back pointers of the hypotheses we can read of the (partial) translations of the sentence.

Given this algorithm, the size of the search space of possible translation is exponential to the length of the sentence. To overcome this, we prune out hypotheses using hypothesis recombination and by heuristic pruning.

Recombining Hypotheses — Recombining hypothesis is a risk-free way to reduce the search space. Two hypotheses can be recombined if they agree in (i) the foreign words covered so far, (ii) the last two English words generated, and (iii) the end of the last foreign phrase covered.

If there are two paths that lead to two hypotheses that agree in these properties, we keep only the cheaper hypothesis, e.g., the one with the least cost so far. The other hypothesis cannot be part of the path to the best translation, and we can safely discard it. We do keep a record of the additional arc for lattice generation (see below).

Beam Search — While the recombination of hypotheses as described above reduces the size of the search space, this is not enough for all but the shortest sentences. Let us estimate how many hypotheses (or, states) are generated during an exhaustive search. Considering the possible values for the properties of unique hypotheses, we can estimate an upper bound for the number of states by $N \simeq 2^{n_f}|V_e|^2 n_f$ where n_f is the number of foreign words, and $|V_e|$ the size of the English vocabulary. In practice, the number of possible English words for the last two words generated is much smaller than $|V_e|^2$. The main concern is the exponential explosion from the 2^{n_f} possible configurations of foreign words covered by a hypothesis. Note this causes the problem of machine translation decoding to be NP-complete [Knight, 1999] and thus dramatically harder than, for instance, speech recognition.

In our beam search we compare the hypotheses that cover the same *number* of foreign words and prune out the inferior hypotheses. We could base the judgment of what inferior hypotheses are on the cost of each hypothesis so far. However, this is generally a very bad criterion, since it biases the search to first translating the easy part of the sentence. For instance, if there is a three word foreign phrase that easily translates into a common English phrase, this may carry much less cost than translating three words separately into uncommon English words. The search will prefer to start the sentence with the easy part and discount alternatives too early.

So, our measure for pruning out hypotheses in our beam search does not only include the cost so far, but also an estimate of the future cost. This future cost estimation should favor hypotheses that already covered difficult parts of the sentence and have only easy parts left, and discount hypotheses that covered the easy parts first. For details of our future cost estimation, see [Koehn, 2003].

Given the cost so far and the future cost estimation, we can prune out hypotheses that fall outside the beam. The beam size can be defined by threshold and histogram pruning. A relative threshold cuts out a hypothesis with a probability less than a factor α of the best hypotheses (e.g., $\alpha = 0.001$). Histogram pruning keeps a certain number n of hypotheses (e.g., $n = 1000$).

Note that this type of pruning is not risk-free (opposed to the recombination). If the future cost estimates are inadequate, we may prune out hypotheses on the

```
initialize hypothesisStack[0 .. nf];
create initial hypothesis hyp_init;
add to stack hypothesisStack[0];
for i=0 to nf-1:
  for each hyp in hypothesisStack[i]:
    for each new_hyp that can be derived from hyp:
      nf[new_hyp] = number of foreign words covered by new_hyp;
      add new_hyp to hypothesisStack[nf[new_hyp]];
      prune hypothesisStack[nf[new_hyp]];
find best hypothesis best_hyp in hypothesisStack[nf];
output best path that leads to best_hyp;
```

Fig. 4. Pseudo code for the beam search algorithm

path to the best scoring translation. In a particular version of beam search, A* search, the future cost estimate is required to be *admissible*, which means that it never overestimates the future cost. Using best-first search and an admissible heuristic allows pruning that is risk-free. In practice, however, this type of pruning does not sufficiently reduce the search space. See more on search in any good Artificial Intelligence text book, such as the one by Russel and Norvig [1995].

Figure 4 describes the algorithm we used for our beam search. For each number of foreign words covered, a hypothesis stack in created. The initial hypothesis is placed in the stack for hypotheses with no foreign words covered. Starting with this hypothesis, new hypotheses are generated by committing to phrasal translations that covered previously unused foreign words. Each derived hypothesis is placed in a stack based on the number of foreign words it covers.

We proceed through these hypothesis stacks, going through each hypothesis in the stack, deriving new hypotheses for this hypothesis and placing them into the appropriate stack (see Figure 5 for an illustration). After a new hypothesis is placed into a stack, the stack may have to be pruned by threshold or histogram pruning, if it has become too large. In the end, the best hypothesis of the ones that cover all foreign words is the final state of the best translation. We can read off the English words of the translation by following the back links in each hypothesis.

Generating Word Lattices — Usually, we expect the decoder to give us the best translation for a given input according to the model. But for some applications, we might be interested in the top n best translations.

A common method in speech recognition, that has also emerged in machine translation [Koehn and Knight, 2003; Och et al., 2003], is: First, use a machine translation system as a base model to generate a set of candidate translations for each input sentence. Then, use additional features to rescore these translations.

To enable this type of work, our decoder outputs a word lattice of possible translations for each input sentence. This lattice is taken from the search graph of the heuristic beam search. Recall the process of state expansions, illustrated in Figure 3. The generated hypotheses and the expansions that link them form

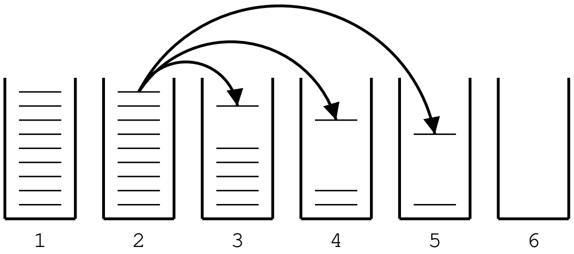

Fig. 5. Hypothesis expansion: Hypotheses are placed in stacks according to the number of foreign words translated so far. If a hypothesis is expanded into new hypotheses, these are placed in new stacks.

a graph. Paths branch out when there are multiple translation options for a hypothesis from which multiple new hypotheses can be derived. Paths join when hypotheses are recombined.

The graph of the hypothesis space (See Figure 3) can be viewed as a probabilistic finite state automaton. The hypotheses are states, and the records of back-links and the additionally stored arcs are state transitions. The added probability scores when expanding a hypothesis are the costs of the state transitions.

Finding the n-best path in such a probabilistic finite state automaton is a well-studied problem. In our implementation, we store the information about hypotheses, hypothesis transitions, and additional arcs in a file that can be processed by the finite state toolkit Carmel[2], which we use to generate the n-best lists. This toolkit uses the n shortest paths algorithm by Eppstein [1994]. Our method is related to work by Ueffing et al. [2002] for generating n-best lists for IBM Model 4.

3 XML-Markup

While statistical machine translation methods cope well with many aspects of translation of languages, there are a few special problems for which better solutions exist. One example is the translation of named entities, such as proper names, dates, quantities, and numbers.

Consider the task of translating numbers, such as *17.55*. In order for a statistical machine translation system to be able to translate this number, it has to observed it in the training data. But even if it has been seen a few times, it is possible that the translation table learned for this “word” is very noisy.

Translating numbers is not a hard problem. It is therefore desirable to be able to tell the decoder up front how to translate such numbers. To continue our example of the number 17.55, this may take the following form:

```
Er erzielte <NUMBER english='17.55'> 17,55 </NUMBER> Punkte .
```

[2] available at `http://www.isi.edu/licensed-sw/carmel/`

This modified input passes to the decoder not only the German words, but also that the third word, the number *17,55*, should be translated as *17.55*.

Phrase-Based Translation with XML Markup — Marking a sequence of words and specifying a translation for them fits neatly into the framework of phrase-based translation. In a way, for a given phrase in the sentence, a translation is provided, which is in essence a phrase translation with translation probability 1. Only for the other parts of the sentence, translation options are generated from the phrase translation table.

Since the first step of the implementation of the beam search decoder is to collect all possible translation options, only this step has to be altered to be sensitive to specifications via XML markup. The core algorithm remains unchanged.

Passing a Probability Distribution of Translations — Making the hard decision of specifying the translation for parts of the sentence has some draw-backs. For instance, the number *1* may be translated as *1, one, a,* and so on into English. So we may want to provide a set of possible translations.

Instead of passing one best translation choice to the decoder, we may want to pass along a probability distribution over a set of possible translations. Given several choices, the decoder is aided by the language model to sort out which translation to use. We extend the XML markup scheme by allowing the specification of multiple English translation options along with translation probabilities.

Example: Es ist <NPPP english='a small house|a little house' prob='0.6|0.4'> ein kleines Haus </NPPP> .

Here, both *a small house* and *a little house* are passed along as possible translations, with the translation probabilities 0.6 and 0.4, respectively.

The scores that are passed along with the translations do not have to be probabilities in a strict sense, meaning, they do not have to add up to 1. They also do not include language model probabilities, but only the phrase translation probability for the marked part of the sentence.

Multi-Path Integration — By specifying a set of possible translations, we can deal with uncertainty which of the translations is the right one in a given context. But there is also the uncertainty, when to use specified translations at all. Maybe the original model has a better way to deal with the targeted words.

Recognizing that the hard decision of breaking out certain words in the input sentence and providing translations to them may be occasionally harmful, we now want to relax this decision. We allow the decoder to use the specified translations, but also to bypass them and use its own translations.

We call this multi-path integration, since we allow two pathways when translating. The path may go through the specified translations, or through translation options from the regular translation model.

4 Experiments

In this section, we will report on a few experiments that illustrate the speed and accuracy of the decoder at different beam sizes.

Threshold	0.0001	0.001	0.01	0.05	0.08	0.1	0.15	0.2	0.3
Time per Sentence	149 sec	119 sec	70 sec	27 sec	18 sec	15 sec	13 sec	10 sec	7 sec
Search Errors	-	+0%	+0%	+0%	+0%	+1%	+3%	+6%	+12%

Table 1. Threshold pruning: hypothesis that score worse by a threshold factor than the best hypothesis in the same stack (see Figure 5) are discarded. A threshold of 0.1 gives a good trade-off between speed (15 seconds) and search errors (+1% over baseline).

Beam Size	1000	200	100	50	20	10	5
Speed	15 sec	15 sec	14 sec	10 sec	9 sec	9 sec	7 sec
Search Errors	+1%	+1%	+2%	+4%	+8%	+20%	+35%

Table 2. Histogram pruning: maximum number of hypothesis kept in a stack (see Figure 5). A beam size of 100 gives a good trade-off between speed (14 seconds) and search errors (+2% over baseline). Threshold pruning with factor 0.1 is applied.)

The translation mode has been trained on a 30 million word German-English parallel corpus extracted from European Parliament proceedings. The model uses various phrase scoring methods, e.g. maximum likelihood phrase translation, lexical translation, and word penalty. The test set used consists of 1500 sentence of average length 28.9. Translation direction is German to English.

Threshold Pruning — If we do not limit the beam size, the number of hypotheses in each stack grows exponentially (or polynomially when a fixed reordering limit is used).

Recall that there are two criteria to limit the number of hypotheses in each stack. One is threshold pruning, where hypotheses that are worse by a certain ratio in respect to the best hypothesis in the stack are discarded. Table 1 displays the effect of different thresholds on the speed and the accuracy of the decoder. Speed is given in translation time per sentence (excluding the start-up time for the decoder). Accuracy is given in the percentage of search errors.

Since we are dealing with some very long sentences, computing the best translation according to the model without any restrictions on the search is prohibitively expensive. We therefore report on relative search errors in respect to the translations generated by the decoder with the threshold 0.0001 and maximum beam size 1000. Table 1 reveals that search time is almost linear with the threshold. With a threshold of 0.1 we can achieve a ten-fold increase in speed with only 1% more search errors.

Histogram Pruning — The second pruning method, histogram pruning, limits the maximum number of hypothesis per stack. Results are given in Table 2. Using now a threshold of 0.1, noticeable changes to speed and accuracy can only be observed with very small beam size: a beam size of 10 increases the number of search errors by +20%, while barely doubling the speed.

Limits on Translation Table — A final strategy to increase the speed of the decoder is to reduce the number of translation options. By limiting the num-

T-Table Limit	1000	500	200	100	50	20	10	5
Speed	15.0 sec	7.6 sec	3.8 sec	1.9 sec	0.9 sec	0.4 sec	0.2 sec	0.1 sec
Search Errors	+1%	+1%	+1%	+1%	+1%	+2%	+7%	+18%

Table 3. Size of translation table per input phrase: Fewer translation options speeds up the search. A beam size of 100 and a pruning threshold of 0.1 is used.

ber of translation table entries that are used for each foreign phrase, the number of generated hypotheses can be reduced – not just the number of hypotheses that are entered into the stack.

Table 3 shows the linear increase in speed in respect to the translation table limit. If we limit the number of translation options to, say, 50, we can drop the translation time per sentence to under a second, with the same 1% relative search error.

In conclusion, using the beam threshold 0.1, maximum stack size 1000, and translation table limit 50, we can dramatically increase the speed of the decoder – by a factor of 1000 in respect to the original setting – with only limited cost in terms of search errors.

5 Conclusions

We described Pharaoh, a beam search decoder for phrase-based statistical machine translation models. Our experiments show that the decoder can achieve very fast translation performance (one sentence per second).

The decoder also allows the generation word lattices and n-best lists, which enable the exploration of reranking methods. An XML interface allows the integration of external knowledge, e.g., a dedicated name translation module.

We hope that the availability of the decoder fosters research in training phrase translation models and the integration of machine translation into wider natural language applications.

Acknowledgments — The development of the decoder was aided by many helpful hints by Franz Och. The XML markup owes much to Ulrich Germann, who developed a similar scheme for his greedy decoder for IBM Model 4 [Germann, 2003].

Bibliography

Eppstein, D. (1994). Finding the k shortest paths. In *Proc. 35th Symp. Foundations of Computer Science*, pages 154–165. IEEE.

Germann, U. (2003). Greedy decoding for statistical machine translation in almost linear time. In *Proceedings of HLT-NAACL*.

Jelinek, F. (1998). *Statistical Methods for Speech Recognition*. The MIT Press.

Knight, K. (1999). Decoding complexity in word-replacement translation models. *Computational Linguistics*, 25(4):607–615.

Koehn, P. (2003). *Noun Phrase Translation*. PhD thesis, University of Southern California, Los Angeles.

Koehn, P. and Knight, K. (2003). Feature-rich translation of noun phrases. In *41st Annual Meeting of the Association of Computational Linguistics (ACL)*.

Koehn, P., Och, F. J., and Marcu, D. (2003). Statistical phrase based translation. In *Proceedings of HLT-NAACL*.

Kumar, S. and Byrne, W. (2004). Minimum bayes-risk decoding for statistical machine translation. In *Proceedings of HLT-NAACL*.

Marcu, D. and Wong, D. (2002). A phrase-based, joint probability model for statistical machine translation. In *Proceedings of EMNLP*, pages 133–139.

Och, F. J. (2002). *Statistical Machine Translation: From Single-Word Models to Alignment Templates*. PhD thesis, RWTH Aachen, Germany.

Och, F. J., Gildea, D., Sarkar, A., Khudanpur, S., Yamada, K., Fraser, A., Shen, L., Kumar, S., Smith, D., Jain, V., Eng, K., Jin, Z., and Radev, D. (2003). Syntax for machine translation. Technical report, John Hopkins University Summer Workshop `http://www.clsp.jhu.edu/ws2003/groups/translate/`.

Och, F. J. and Ney, H. (2002). Discriminative training and maximum entropy models for statistical machine translation. In *Proceedings of ACL*.

Och, F. J. and Ney, H. (2003). A systematic comparison of various statistical alignment models. *Computational Linguistics*, 29(1):19–52.

Russel, S. and Norvig, P. (1995). *Artificial Intelligence: A Modern Approach*. Prentice Hall, New Jersey.

Tillmann, C. (2001). *Word Re-Ordering and Dynamic Programming based Search Algorithm for Statistical Machine Translation*. PhD thesis, RWTH Aachen, Germany.

Tillmann, C. (2003). A projection extension algorithm for statistical machine translation. In *Proceedings of EMNLP*, pages 1–8.

Ueffing, N., Och, F. J., and Ney, H. (2002). Generation of word graphs in statistical machine translation. In *Proceedings of EMNLP*, pages 156–163, Philadelphia. Association for Computational Linguistics.

Vogel, S., Zhang, Y., Huang, F., Tribble, A., Venugopal, A., Zhao, B., and Waibel, A. (2003). The CMU statistical machine translation system. In *Proceedings of the Ninth Machine Translation Summit*.

Zens, R., Och, F. J., and Ney, H. (2002). Phrase-based statistical machine translation. In *Proceedings of the German Conference on Artificial Intelligence (KI 2002)*.