Scaling Phrase-Based Statistical Machine Translation to Larger Corpora and Longer Phrases

Chris Callison-Burch **Colin Bannard**
University of Edinburgh
2 Buccleuch Place
Edinburgh EH8 9LW
{chris,colin}@linearb.co.uk

Josh Schroeder
Linear B Ltd.
39 B Cumberland Street
Edinburgh EH3 6RA
josh@linearb.co.uk

Abstract

In this paper we describe a novel data structure for phrase-based statistical machine translation which allows for the retrieval of arbitrarily long phrases while simultaneously using less memory than is required by current decoder implementations. We detail the computational complexity and average retrieval times for looking up phrase translations in our suffix array-based data structure. We show how sampling can be used to reduce the retrieval time by orders of magnitude with no loss in translation quality.

1 Introduction

Statistical machine translation (SMT) has an advantage over many other statistical natural language processing applications in that training data is regularly produced by other human activity. For some language pairs very large sets of training data are now available. The publications of the European Union and United Nations provide gigbytes of data between various language pairs which can be easily mined using a web crawler. The Linguistics Data Consortium provides an excellent set of off the shelf Arabic-English and Chinese-English parallel corpora for the annual NIST machine translation evaluation exercises.

The size of the NIST training data presents a problem for phrase-based statistical machine translation. Decoders such as Pharaoh (Koehn, 2004) primarily use lookup tables for the storage of phrases and their translations. Since retrieving longer segments of human translated text generally leads to better translation quality, participants in the evaluation exercise try to maximize the length of phrases that are stored in lookup tables. The combination of large corpora and long phrases means that the table size can quickly become unwieldy.

A number of groups in the 2004 evaluation exercise indicated problems dealing with the data. Coping strategies included limiting the length of phrases to something small, not using the entire training data set, computing phrases probabilities on disk, and filtering the phrase table down to a manageable size after the testing set was distributed. We present a data structure that is easily capable of handling the largest data sets currently available, and show that it can be scaled to much larger data sets.

In this paper we:

- Motivate the problem with storing enumerated phrases in a table by examining the memory requirements of the method for the NIST data set
- Detail the advantages of using long phrases in SMT, and examine their potential coverage
- Describe a suffix array-based data structure which allows for the retrieval of translations of arbitrarily long phrases, and show that it requires far less memory than a table
- Calculate the computational complexity and average time for retrieving phrases and show how this can be sped up by orders of magnitude with no loss in translation accuracy

2 Related Work

Koehn et al. (2003) compare a number of different approaches to phrase-based statistical machine

Proceedings of the 43rd Annual Meeting of the ACL, pages 255–262,
Ann Arbor, June 2005.

length	num uniq (mil)	average # translations	avg trans length
1	.88	8.322	1.37
2	16.5	1.733	2.35
3	42.6	1.182	3.44
4	58.7	1.065	4.58
5	65.0	1.035	5.75
6	66.4	1.022	6.91
7	65.8	1.015	8.07
8	64.3	1.012	9.23
9	62.2	1.010	10.4
10	59.9	1.010	11.6

Table 1: Statistics about Arabic phrases in the NIST-2004 large data track.

length	entries (mil)	words (mil)	memory (gigs)	including alignments
1	7.3	10	.1	.11
2	36	111	.68	.82
3	86	412	2.18	2.64
4	149	933	4.59	5.59
5	216	1,645	7.74	9.46
6	284	2,513	11.48	14.07
7	351	3,513	15.70	19.30
8	416	4,628	20.34	25.05
9	479	5,841	25.33	31.26
10	539	7,140	30.62	37.85

Table 2: Estimated size of lookup tables for the NIST-2004 Arabic-English data

translation including the joint probability phrase-based model (Marcu and Wong, 2002) and a variant on the alignment template approach (Och and Ney, 2004), and contrast them to the performance of the word-based IBM Model 4 (Brown et al., 1993). Most relevant for the work presented in this paper, they compare the effect on translation quality of using various lengths of phrases, and the size of the resulting phrase probability tables.

Tillmann (2003) further examines the relationship between maximum phrase length, size of the translation table, and accuracy of translation when inducing block-based phrases from word-level alignments. Venugopal et al. (2003) and Vogel et al. (2003) present methods for achieving better translation quality by growing incrementally larger phrases by combining smaller phrases with overlapping segments.

3 Scaling to Long Phrases

Table 1 gives statistics about the Arabic-English parallel corpus used in the NIST large data track. The corpus contains 3.75 million sentence pairs, and has 127 million words in English, and 106 million words in Arabic. The table shows the number of unique Arabic phrases, and gives the average number of translations into English and their average length.

Table 2 gives estimates of the size of the lookup tables needed to store phrases of various lengths, based on the statistics in Table 1. The number of unique entries is calculated as the number unique phrases times the average number of translations. The number of words in the table is calculated as the number of unique phrases times the phrase length plus the number of entries times the average translation length. The memory is calculated assuming that each word is represented with a 4 byte integer, that each entry stores its probability as an 8 byte double and that each word alignment is stored as a 2 byte short. Note that the size of the table will vary depending on the phrase extraction technique.

length	coverage	length	coverage
1	93.5%	6	4.70%
2	73.3%	7	2.95%
3	37.1%	8	2.14%
4	15.5%	9	1.99%
5	8.05%	10	1.49%

Table 3: Lengths of phrases from the training data that occur in the NIST-2004 test set

Table 3 gives the percent of the 35,313 word long test set which can be covered using only phrases of the specified length or greater. The table shows the efficacy of using phrases of different lengths. The table shows that while the rate of falloff is rapid, there are still multiple matches of phrases of length 10. The longest matching phrase was one of length 18. There is little generalization in current SMT implementations, and consequently longer phrases generally lead to better translation quality.

3.1 Why use phrases?

Statistical machine translation made considerable advances in translation quality with the introduction of phrase-based translation. By increasing the size of the basic unit of translation, phrase-based machine translation does away with many of the problems associated with the original word-based formulation of statistical machine translation (Brown et al., 1993), in particular:

- The Brown et al. (1993) formulation doesn't have a direct way of translating phrases; instead they specify a *fertility* parameter which is used to replicate words and translate them individually.

- With units as small as words, a lot of reordering has to happen between languages with different word orders. But the *distortion* parameter is a poor explanation of word order.

Phrase-based SMT overcomes the first of these problems by eliminating the fertility parameter and directly handling word-to-phrase and phrase-to-phrase mappings. The second problem is alleviated through the use of multi-word units which reduce the dependency on the distortion parameter. Less word re-ordering need occur since local dependencies are frequently captured. For example, common adjective-noun alternations are memorized. However, since this linguistic information is not encoded in the model, unseen adjective noun pairs may still be handled incorrectly.

By increasing the length of phrases beyond a few words, we might hope to capture additional non-local linguistic phenomena. For example, by memorizing longer phrases we may correctly learn case information for nouns commonly selected by frequently occurring verbs; we may properly handle discontinuous phrases (such as French negation, some German verb forms, and English verb particle constructions) that are neglected by current phrase-based models; and we may by chance capture some agreement information in coordinated structures.

3.2 Deciding what length of phrase to store

Despite the potential gains from memorizing longer phrases, the fact remains that as phrases get longer there is a decreasing likelihood that they will be repeated. Because of the amount of memory required to store a phrase table, in current implementations a choice is made as to the maximum length of phrase to store.

length	coverage	length	coverage
1	96.3%	6	21.9%
2	94.9%	7	11.2%
3	86.1%	8	6.16%
4	65.6%	9	3.95%
5	40.9%	10	2.90%

Table 4: Coverage using only repeated phrases of the specified length

Based on their analysis of the relationship between translation quality and phrase length, Koehn et al. (2003) suggest limiting phrase length to three words or less. This is entirely a practical suggestion for keeping the phrase table to a reasonable size, since they measure minor but incremental improvement in translation quality up to their maximum tested phrase length of seven words.[1]

Table 4 gives statistics about phrases which occur more than once in the English section of the Europarl corpus (Koehn, 2002) which was used in the Koehn et al. (2003) experiments. It shows that the percentage of words in the corpus that can be covered by repeated phrases falls off rapidly at length 6, but that even phrases up to length 10 are able to cover a non-trivial portion of the corpus. This draws into question the desirability of limiting phrase retrieval to length three.

The decision concerning what length of phrases to store in the phrase table seems to boil down to a practical consideration: one must weigh the likelihood of retrieval against the memory needed to store longer phrases. We present a data structure where this is not a consideration. Our suffix array-based data structure allows the retrieval of arbitrarily long phrases, while simultaneously requiring far less memory than the standard table-based representation.

[1] While the improvements to translation quality reported in Koehn et al. (2003) are minor, their evaluation metric may not have been especially sensitive to adding longer phrases. They used the Bleu evaluation metric (Papineni et al., 2002), but capped the n-gram precision at 4-grams.

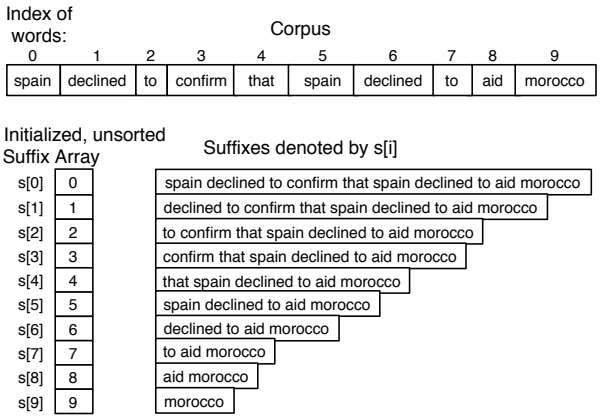

Figure 1: An initialized, unsorted suffix array for a very small corpus

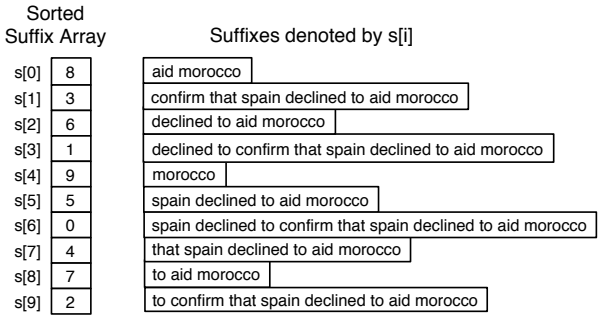

Figure 2: A sorted suffix array and its corresponding suffixes

4 Suffix Arrays

The suffix array data structure (Manber and Myers, 1990) was introduced as a space-economical way of creating an index for string searches. The suffix array data structure makes it convenient to compute the frequency and location of any substring or n-gram in a large corpus. Abstractly, a suffix array is an alphabetically-sorted list of all suffixes in a corpus, where a suffix is a substring running from each position in the text to the end. However, rather than actually storing all suffixes, a suffix array can be constructed by creating a list of references to each of the suffixes in a corpus. Figure 1 shows how a suffix array is initialized for a corpus with one sentence. Each index of a word in the corpus has a corresponding place in the suffix array, which is identical in length to the corpus. Figure 2 shows the final state of the suffix array, which is as a list of the indices of words in the corpus that corresponds to an alphabetically sorted list of the suffixes.

The advantages of this representation are that it is compact and easily searchable. The total size of the suffix array is a constant amount of memory. Typically it is stored as an array of integers where the array is the same length as the corpus. Because it is organized alphabetically, any phrase can be quickly located within it using a binary search algorithm.

Yamamoto and Church (2001) show how to use suffix arrays to calculate a number of statistics that are interesting in natural language processing applications. They demonstrate how to calculate term frequency / inverse document frequency (*tf / idf*) for all n-grams in very large corpora, as well as how to use these frequencies to calculate n-grams with high mutual information and residual inverse document frequency. Here we show how to apply suffix arrays to parallel corpora to calculate phrase translation probabilities.

4.1 Applied to parallel corpora

In order to adapt suffix arrays to be useful for statistical machine translation we need a data structure with the following elements:

- A suffix array created from the source language portion of the corpus, and another created from the target language portion of the corpus,
- An index that tells us the correspondence between sentence numbers and positions in the source and target language corpora,
- An alignment $\mathbf{a}$ for each sentence pair in the parallel corpus, where $\mathbf{a}$ is defined as a subset of the Cartesian product of the word positions in a sentence $\mathbf{e}$ of length I and a sentence $\mathbf{f}$ of length J:

$$\mathbf{a} \subseteq \{(i, j) : i = 1...I; j = 1...J\}$$

- A method for extracting the translationally equivalent phrase for a subphrase given an aligned sentence pair containing that subphrase.

The total memory usage of the data structure is thus the size of the source and target corpora, plus the size of the suffix arrays (identical in length to the

corpora), plus the size of the two indexes that correlate sentence positions with word positions, plus the size of the alignments. Assuming we use *ints* to represent words and indices, and *shorts* to represent word alignments, we get the following memory usage:

$$2 * \text{num words in source corpus} * sizeof(int) +$$

$$2 * \text{num words in target corpus} * sizeof(int) +$$

$$2 * \text{number sentence pairs} * sizeof(int) +$$

$$\text{number of word alignments} * sizeof(short)$$

The total amount of memory required to store the NIST Arabic-English data using this data structure is

$$2 * 105{,}994{,}774 * sizeof(int) +$$

$$2 * 127{,}450{,}473 * sizeof(int) +$$

$$2 * 3{,}758{,}904 * sizeof(int) +$$

$$92{,}975{,}229 * sizeof(short)$$

Or just over 2 Gigabytes.

4.2 Calculating phrase translation probabilities

In order to produce a set of phrase translation probabilities, we need to examine the ways in which they are calculated. We consider two common ways of calculating the translation probability: using the maximum likelihood estimator (MLE) and smoothing the MLE using lexical weighting.

The maximum likelihood estimator for the probability of a phrase is defined as

$$p(\bar{f}|\bar{e}) = \frac{count(\bar{f}, \bar{e})}{\sum_{\bar{f}} count(\bar{f}, \bar{e})} \quad (1)$$

Where $count(\bar{f}, \bar{e})$ gives the total number of times the phrase $\bar{f}$ was aligned with the phrase $\bar{e}$ in the parallel corpus. We define phrase alignments as follows. A substring $\bar{e}$ consisting of the words at positions $l...m$ is aligned with the phrase $\bar{f}$ by way of the subalignment

$$\mathbf{s} = \mathbf{a} \cap \{(i,j) : i = l...m, j = 1...J\}$$

The aligned phrase $\bar{f}$ is the subphrase in $\mathbf{f}$ which spans from $min(j)$ to $max(j)$ for $j|(i,j) \in \mathbf{s}$.

The procedure for generating the counts that are used to calculate the MLE probability using our suffix array-based data structures is:

1. Locate all the suffixes in the English suffix array which begin with the phrase $\bar{e}$. Since the suffix array is sorted alphabetically we can easily find the first occurrence $s[k]$ and the last occurrence $s[l]$. The length of the span in the suffix array $l-k+1$ indicates the number of occurrences of $\bar{e}$ in the corpus. Thus the denominator $\sum_{\bar{f}} count(\bar{f}, \bar{e})$ can be calculated as $l - k + 1$.

2. For each of the matching phrases $s[i]$ in the span $s[k]...s[l]$, look up the value of $s[i]$ which is the word index w of the suffix in the English corpus. Look up the sentence number that includes w, and retrieve the corresponding sentences $\mathbf{e}$ and $\mathbf{f}$, and their alignment $\mathbf{a}$.

3. Use $\mathbf{a}$ to extract the target phrase $\bar{f}$ that aligns with the phrase $\bar{e}$ that we are searching for. Increment the count for $< \bar{f}, \bar{e} >$.

4. Calculate the probability for each unique matching phrase $\bar{f}$ using the formula in Equation 1.

A common alternative formulation of the phrase translation probability is to lexically weight it as follows:

$$p_{lw}(\bar{f}|\bar{e}, \mathbf{s}) = \prod_{i=1}^{n} \frac{1}{|\{i|(i,j) \in \mathbf{s}\}|} \sum_{\forall (i,j) \in \mathbf{s}} p(f_j|e_i) \quad (2)$$

Where n is the length of $\bar{e}$.

In order to use lexical weighting we would need to repeat steps 1-4 above for each word e_i in $\bar{e}$. This would give us the values for $p(f_j|e_i)$. We would further need to retain the subphrase alignment $\mathbf{s}$ in order to know the correspondence between the words $(i,j) \in \mathbf{s}$ in the aligned phrases, and the total number of foreign words that each e_i is aligned with ($|\{i|(i,j) \in \mathbf{s}\}|$). Since a phrase alignment $< \bar{f}, \bar{e} >$ may have multiple possible word-level alignments, we retain a set of alignments S and take the maximum:

$$p(\bar{f}|\bar{e}, S) = p(\bar{f}|\bar{e}) * \arg\max_{\mathbf{s}\in S} p_{lw}(\bar{f}|\bar{e}, \mathbf{s}) \quad (3)$$

Thus our suffix array-based data structure can be used straightforwardly to look up all aligned translations for a given phrase and calculate the probabilities on-the-fly. In the next section we turn to the computational complexity of constructing phrase translation probabilities in this way.

5 Computational Complexity

Computational complexity is relevant because there is a speed-memory tradeoff when adopting our data structure. What we gained in memory efficiency may be rendered useless if the time it takes to calculate phrase translation probabilities is unreasonably long. The computational complexity of looking up items in a hash table, as is done in current table-based data structures, is extremely fast. Looking up a single phrase can be done in unit time, $O(1)$.

The computational complexity of our method has the following components:

- The complexity of finding all occurrences of the phrase in the suffix array
- The complexity of retrieving the associated aligned sentence pairs given the positions of the phrase in the corpus
- The complexity of extracting all aligned phrases using our phrase extraction algorithm
- The complexity of calculating the probabilities given the aligned phrases

The methods we use to execute each of these, and their complexities are as follow:

- Since the array is sorted, finding all occurrences of the English phrase is extremely fast. We can do two binary searches: one to find the first occurrence of the phrase and a second to find the last. The computational complexity is therefore bounded by $O(2\log(n))$ where n is the length of the corpus.
- We use a similar method to look up the sentences $\mathbf{e_i}$ and $\mathbf{f_i}$ and word-level alignment $\mathbf{a_i}$ that are associated with the position w_i in the corpus of each phrase occurrence $\bar{e_i}$. The complexity is $O(k * 2\log(m))$ where k is the number of occurrences of $\bar{e}$ and m is the number of sentence pairs in the parallel corpus.
- The complexity of extracting the aligned phrase for a single occurrence of $\bar{e_i}$ is $O(2\log(|\mathbf{a_i}|)$ to get the subphrase alignment $\mathbf{s_i}$, since we store the alignments in a sorted array. The complexity of then getting $\bar{f_i}$ from $\mathbf{s_i}$ is $O(length(\bar{f_i}))$.
- The complexity of summing over all aligned phrases and simultaneously calculating their probabilities is $O(k)$.

Thus we have a total complexity of:

$$O(2\log(n) + k * 2\log(m) \quad (4)$$

$$+ \sum_{\mathbf{a_i}, \bar{f_i}|\bar{e_i}}^{\bar{e_1}...\bar{e_k}} (2\log(|\mathbf{a_i}|) + length(\bar{f_i})) + k) \quad (5)$$

for the MLE estimation of the translation probabilities for a single phrase. The complexity is dominated by the k terms in the equation, when the number of occurrences of the phrase in the corpus is high. Phrases with high frequency may cause excessively long retrieval time. This problem is exacerbated when we shift to a lexically weighted calculation of the phrase translation probability. The complexity will be multiplied across each of the component words in the phrase, and the component words themselves will be more frequent than the phrase.

Table 5 shows example times for calculating the translation probabilities for a number of phrases. For frequent phrases like *of the* these times get unacceptably long. While our data structure is perfect for

phrase	freq	O	time (ms)
respect for the dead	3	80	24
since the end of the cold war	19	240	136
the parliament	1291	4391	1117
of the	290921	682550	218369

Table 5: Examples of O and calculation times for phrases of different frequencies

overcoming the problems associated with storing the translations of long, infrequently occurring phrases, it in a way introduces the converse problem. It has a clear disadvantage in the amount of time it takes to retrieve commonly occurring phrases. In the next section we examine the use of sampling to speed up the calculation of translation probabilities for very frequent phrases.

6 Sampling

Rather than compute the phrase translation probabilities by examining the hundreds of thousands of occurrences of common phrases, we instead sample from a small subset of the occurrences. It is unlikely that we need to extract the translations of all occurrences of a high frequency phrase in order to get a good approximation of their probabilities. We instead cap the number of occurrences that we consider, and thus give a maximum bound on k in Equation 5.

In order to determine the effect of different levels of sampling, we compare the translation quality against cumulative retrieval time for calculating the phrase translation probabilities for all subphrases in an evaluation set. We translated a held out set of 430 German sentences with 50 words or less into English. The test sentences were drawn from the 01/17/00 proceedings of the Europarl corpus. The remainder of the corpus (1 million sentences) was used as training data to calculate the phrase translation probabilities. We calculated the translation quality using Bleu's modified n-gram precision metric (Papineni et al., 2002) for n-grams of up to length four. The framework that we used to calculate the translation probabilities was similar to that detailed in Koehn et al. (2003). That is:

$$\hat{\mathbf{e}} = \arg\max_{\mathbf{e_1^I}} p(\mathbf{e_1^I}|\mathbf{f_1^I}) \quad (6)$$

$$= \arg\max_{\mathbf{e_1^I}} p_{LM}(\mathbf{e_1^I}) * \quad (7)$$

$$\prod_{i=1}^{I} p(\bar{f}_i|\bar{e}_i) d(a_i - b_{i-1}) p_{lw}(\bar{f}_i|\bar{e}_i, \mathbf{a}) \quad (8)$$

Where p_{LM} is a language model probability and d is a distortion probability which penalizes movement.

Table 6 gives a comparison of the translation quality under different levels of sampling. While the accuracy fluctuates very slightly it essentially remains uniformly high for all levels of sampling. There are a number of possible reasons for the fact that the quality does not decrease:

sample size	time	quality
unlimited	6279 sec	.290
50000	1051 sec	.289
10000	336 sec	.291
5000	201 sec	.289
1000	60 sec	.288
500	35 sec	.288
100	10 sec	.288

Table 6: A comparison of retrieval times and translation quality when the number of translations is capped at various sample sizes

- The probability estimates under sampling are sufficiently good that the most probable translations remain unchanged,
- The interaction with the language model probability rules out the few misestimated probabilities, or
- The decoder tends to select longer or less frequent phrases which are not affected by the sampling.

While the translation quality remains essentially unchanged, the cumulative time that it takes to calculate the translation probabilities for all subphrases in the 430 sentence test set decreases radically. The total time drops by orders of magnitude from an hour and a half without sampling down to a mere 10 seconds with a cavalier amount of sampling. This suggests that the data structure is suitable for deployed SMT systems and that no additional caching need be done to compensate for the structure's computational complexity.

7 Discussion

The paper has presented a super-efficient data structure for phrase-based statistical machine translation. We have shown that current table-based methods are unwieldily when used in conjunction with large data sets and long phrases. We have contrasted this with our suffix array-based data structure which provides

a very compact way of storing large data sets while simultaneously allowing the retrieval of arbitrarily long phrases.

For the NIST-2004 Arabic-English data set, which is among the largest currently assembled for statistical machine translation, our representation uses a very manageable 2 gigabytes of memory. This is less than is needed to store a table containing phrases with a maximum of three words, and is ten times less than the memory required to store a table with phrases of length eight.

We have further demonstrated that while computational complexity can make the retrieval of translation of frequent phrases slow, the use of sampling is an extremely effective countermeasure to this. We demonstrated that calculating phrase translation probabilities from sets of 100 occurrences or less results in nearly no decrease in translation quality.

The implications of the data structure presented in this paper are significant. The compact representation will allow us to easily scale to parallel corpora consisting of billions of words of text, and the retrieval of arbitrarily long phrases will allow experiments with alternative decoding strategies. These facts in combination allow for an even greater exploitation of training data in statistical machine translation.

References

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

Philipp Koehn, Franz Josef Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *Proceedings of HLT/NAACL*.

Philipp Koehn. 2002. Europarl: A multilingual corpus for evaluation of machine translation. Unpublished Draft.

Philipp Koehn. 2004. Pharaoh: A beam search decoder for phrase-based statistical machine translation models. In *Proceedings of AMTA*.

Udi Manber and Gene Myers. 1990. Suffix arrays: A new method for on-line string searches. In *The First Annual ACM-SIAM Symposium on Dicrete Algorithms*, pages 319–327.

Daniel Marcu and William Wong. 2002. A phrase-based, joint probability model for statistical machine translation. In *Proceedings of EMNLP*.

Franz Josef Och and Hermann Ney. 2004. The alignment template approach to statistical machine translation. *Computational Linguistics*, 30(4):417–450, December.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: A method for automatic evaluation of machine translation. In *Proceedings of ACL*.

Christoph Tillmann. 2003. A projection extension algorithm for statistical machine translation. In *Proceedings of EMNLP*.

Ashish Venugopal, Stephan Vogel, and Alex Waibel. 2003. Effective phrase translation extraction from alignment models. In *Proceedings of ACL*.

Stephan Vogel, Ying Zhang, Fei Huang, Alicia Tribble, Ashish Venugopal, Bing Zhao, and Alex Waibel. 2003. The CMU statistical machine translation system. In *Proceedings of MT Summit 9*.

Mikio Yamamoto and Kenneth Church. 2001. Using suffix arrays to compute term frequency and document frequency for all substrings in a corpus. *Compuatational Linguistics*, 27(1):1–30.