TURKALATOR[1]
A Suite of Tools for Augmenting English-to-Turkish Statistical Machine Translation

by

Gorkem Ozbek
[gorkem@stanford.edu]

Siddharth Jonathan
[jonsid@stanford.edu]

CS224N: Natural Language Processing
Final Project
Prof. Christopher D. Manning

June 7, 2006

1 Introduction

Statistical machine translation (here on MT) research has traditionally focused on popular languages of the developed world: English, other major European languages, and some major Asian languages. The reason for this is two-fold. On one hand agencies funding MT research, guided by practical concerns that may sometimes elude researchers, demand systems tuned for major these languages and where the funding goes, research goes, as evidenced, for example, by the recent increase in Arabic and Chinese MT

[1] A note on the title: Turks believe, perhaps more than people of other cultures, that one needs to first and foremost master the customs and idiosyncrasies of a culture to truly belong in it. This is a fitting metaphor for our goals in this project: customize the standard statistical MT system building pipeline for specific features of Turkish so that the resulting product not only *translates* English text, but it also *turkalates* it.

research. On the other hand, until recently the resources required for developing statistical MT systems, especially large parallel corpora, have been only available for major languages. Overall languages for which there are scarce resources, whether financial or linguistic, have received little attention from the statistical MT research community.

Despite being spoken widely around the world in different forms, Turkish has been one of these languages: there has been minimal research in MT systems where Turkish is either the source or target language. This project attempts to fill this gap, if only slightly. We designed and implemented a suite of tools intended for aiding the statistical MT creation process by exploiting linguistic features of Turkish and how these features differ from their counterparts in English. The tools we designed were then coupled with existing software to produce an English-to-Turkish MT system that would hopefully perform better than the baseline system constructed without concern for the idiosyncrasies of the specific task at hand.

Turkish differs significantly in its linguistic structure from English. While the differences are wide-spread across many aspects of the language pair, for the purposes of this project we focused on difficulties stemming from Turkish morphology and morpho-syntax. As an agglutinative language, Turkish has a much richer inflectional morphology than English. On the word level, this means that individual morphemes in Turkish often assume semantic and syntactic functions in a well-formed sentence that are, in English, reserved almost exclusively for whole words. This is most apparent in the structure of Turkish verbs, which encode in their morphology among other things the semantic role played by auxiliary verbs that in English would be modifying it. On the phrasal level, the difficulties with morphology also extend to the morphology-syntax interface.[2]

The main problem rich morphology causes for the statistical MT framework is one of data scarcity. Both essential components of the translation system, the translation model and the language model, suffer from the low data to parameter ratio due to scarcity. A thorough morphological analysis is one way of remedying the scarcity problem (Dejean et al., 2003). However, this approach assumes the existence of a morphological transducer, an abstract machine that breaks down the surface form of a word into its stem and morphemes and constructs it back given this stem and list of morphemes. While we had our data available in morphologically analyzed form, we did not have access to the transducer that produced it. Furthermore, developing a fully-functional transducer with appropriate morphology definitions for Turkish and English was impractical due to time and scope limitations. Instead, we picked the approach of approximating a morphological analysis for Turkish and then using this approximate analysis to compensate for the negative effects caused data scarcity in building the translation and language models. Similar efforts have been shown to improve at least word alignment for the English-German language pair (Corston-Oliver and Gamon, 2004). Other methods such as using weighted finite-state transducer composition have also been proposed to aid word alignment in for language pairs of varying morphological complexity (Schafer and Drabek, 2005).

[2] While we are addressing some of the issues involved in Turkish morpho-syntax within the scope of this project (most notably in the phrase extraction algorithm), there are others that we leave mostly unexplored. For example, as a language with free word order Turkish marks subjects and objects in a sentence through morphemes.

In the next section we discuss in detail the components we developed in accordance with our approach and the design decisions we made to render our system sensitive to morphological structure of Turkish.

2 System Components

In addition to the corpora we used, our system consists of three processes that we developed specifically for this task: preprocessing including morphological approximation, word alignment and phrase extraction and scoring. We used these processes in tandem with existing software to stitch together a machine translator. Below we discuss this construction pipeline, including the language model creation, and comment on our data.

2.1 Corpus

Our parallel corpus of English and Turkish text was graciously made available to us by Prof. Kemal Oflazer of Sabanci University, Istanbul, Turkey. It consists of approximately 22,000 aligned sentence pairs drawn across a variety of genres: George Orwell's novel *1984* makes up some of the text along with tourism guides, court hearings and others. A morphologically analyzed version of the corpus was also made available.

The most important thing to notice about this corpus is its relatively small size. Generally, for sufficient training of MT systems corpora at least an order of magnitude larger than ours are used, even when additional scarcity problems introduced by morphology are not of concern. On the other hand, however, the assumption is using large corpora frees one from having to deal separately with morphology. Hence, using smaller corpora is acceptable when additional measures are being taken to prevent data scarcity by morphological processing of some sort. All in all, the size of our corpus was a factor in decreasing the accuracy of the MT system trained on it, although this effect was not debilitating by any means.

2.2 Preprocessing

The preprocessing step prepares the raw corpus text for translation and language model training. The most interesting part of the preprocessing is segmentation, which we discuss in detail below. Additionally, during the preprocessing step we lowercase the text, convert its character encoding from its native encoding to Unicode so that strings of both languages can be represented in a uniform way regardless of the differences in alphabets, and partition it into training, validation and test sets according to the ratio 85:10:5 respectively. These sets are constructed in a uniform way so that each of them exhibits the range of genres found in the unpartitioned corpus.

The segmentation process is essentially responsible for approximating a morphological analysis of Turkish sentences.[3] Using the morphologically analyzed version of the corpus would be desirable here. However, without the appropriate transducer this is completely impractical: if we analyze the text into a set of stems and morphemes and use these units to build our translation model, the output from the translation system would be in the form of these stems and morphemes also, and without a transducer it would be impossible to render these strings into regular Turkish text.

The approximation works as a way of exploiting certain patterns in Turkish morphology:

- Morphemes almost always occur as suffixes, rather than prefixes or infixes. This means that the morphologically analyzed form of a Turkish word is a stem, followed by zero or more suffixes.
- Morphemes rarely have more than one vowel in them. Syllables in Turkish also have one and only one vowel in them. Hence there exists a rough correspondence between morphemes and the last few syllables of a Turkish word.
- While there are many types of morphemes, there are only a few morpheme tokens for each type. Therefore, it's not unreasonable to assume that each morpheme token can be considered in a class of its own without much loss of precision.

With these considerations in mind, we enlist the help of an external package, Zemberek, an open-source Java library for performing simple Turkish NLP tasks like spell checking, syllabification and stemming (Zemberek WEBSITE). Segmentation of each Turkish word works as follows: first Zemberek is consulted for a list of true stem candidates for this word. From this list the longest candidate is picked. This is intended to minimize the noise added from derivational morphology. Zemberek's stemmer returns, in addition to the root guessed for the word in question, all derivational forms of that root. Since derivational suffixes always precede inflectional suffixes in Turkish, picking the longest candidate amounts to a rough guess at the true stem for that word.

Once a candidate is chosen as the true stem for the word, the longest common substring of this true stem and the surface form of the word itself is taken to be the stem approximation. The remaining portion of the word, if any, is then syllabified and each syllable is taken to be a suffix approximation. The following table shows some examples of this process:

Surface form	**Correct Analysis**	**Our approximation**	**English Translation**
kelebeğin	kelebek #in	kelebe #ğin	the butterfly's
yörelerdeki	yöre #ler #de #ki	yöre #ler #de #ki	in the areas
alkollü	alkol #lü	alkol #lü	alcoholic

This segmentation approach has three main advantages. First, it provides a reasonable approximation to a morphological analysis, without requiring the machinery of a morphological transducer. Second, it reduces the data scarcity in a meaningful way – both the translation model and the language model benefit from this significantly as we

[3] It is also responsible for approximating a morphological analysis for English sentences, but we cheat a bit here: we use the morphologically analyzed version of the English side of the corpus directly. Since we only test our system on similarly analyzed test sentences, this has no practical drawbacks.

discuss later on. Finally, with the help of the language model, this approach provides an easy solution to composing back surface forms from morphemes: just append suffixes to the preceding stem. Since we keep our suffixes as surface forms, no additional compositional rules are necessary.

The segmentation process produces a pair of files for each language: one with only stems, the other with stems and suffixes. Next section explains how these files are utilized.

2.3 Two-level Giza++ Alignments

Once we produce segmented versions of the training set, we use the Giza++ toolkit (Och, 2000) to learn word alignments. This is a two-step process. First, we align bidirectionally the stemmed parallel corpora.[4] These alignments are then intersected to get high precision, one-to-one alignment points. These aligned stem pairs essentially make up a "stem dictionary," which we then use for the second round of Giza++ training. Provided with an initial dictionary of word correspondences, Giza++ obeys the "dictionary constraint" during the its first iteration over the training data:

> In parallel sentences "$e_1 \ldots e_n$" and "$f_1 \ldots f_m$", e_i and f_i are counted as a co-occurrence pair if one of two conditions is met: 1) e_i and f_i occur as an entry in the dictionary, or 2) e_i does not occur in the dictionary with any f_j ($1 \leq j \leq m$) and f_i does not occur in the dictionary with any e_j ($1 \leq j \leq n$).

This effectively gives the training cycle an initial bias towards aligning stems with stems and suffixes with suffixes. However, this is only a slight bias, which is just as well: it turns out that often we want to align Turkish suffixes with whole English words (as is the case with most English prepositions).

At the end of this step two alignment files are obtained from Giza++: Turkish source sentences aligned with English target sentences and vice versa. These alignment files are then passed on to the next step, which extracts word and phrase alignments from them.

2.4 Word Alignment, Phrase Extraction, and Phrase Scoring

The next step in the process is responsible for processing the two unidirectional alignment files produced in the previous step into a phrase translation table that captures the translation model. This, in turn, is done in two parts. First, we create a bidirectional lattice of alignment points for each unidirectional alignment pair. Then we extract and count phrases based on these alignment points.

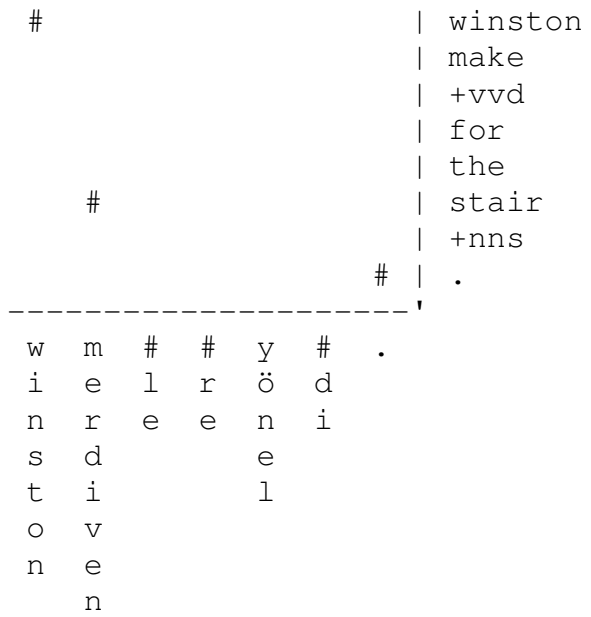
We acquire alignment points according to the "Grow-Diag-Final" algorithm discussed in (Koehn et al., 2003). The idea is to begin with the intersection of the two Giza++ alignments, only add new alignment points that exist in the union of the two Giza++ alignments and connect at least one previously unaligned word. This is done by

[4] Since Giza++ is an implementation of the IBM word alignment models, it produces 1-to-N alignments. Therefore, bidirectional training is generally used, and these bidirectional alignments are further processed to obtain alignment points as we discuss below.

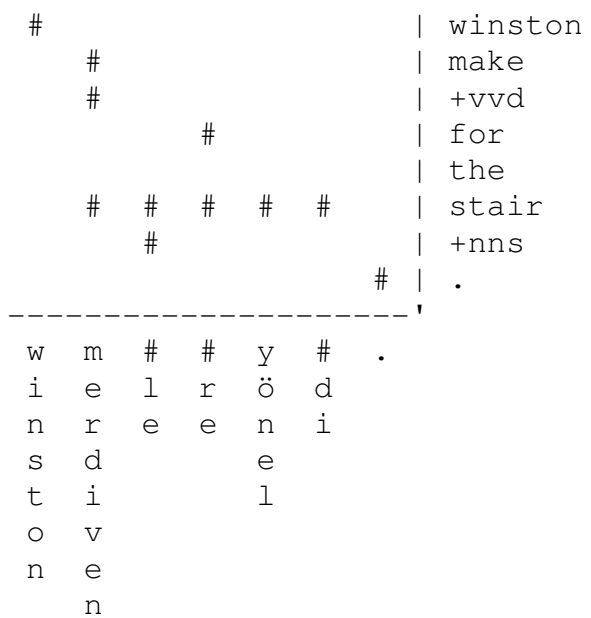
first “growing” to only those points that are directly adjacent to existing alignment points. When no new alignment point can be added this way, non-adjacent alignment points are added, with otherwise the same requirements.

To deal more correctly with suffix alignments, we perform an additional, final step, which we call *filling morphological holes*. During this process each suffix in the Turkish sentence is aligned with those English words (or suffixes) that the preceding Turkish token (whether it is a stem or a suffix) is aligned with.

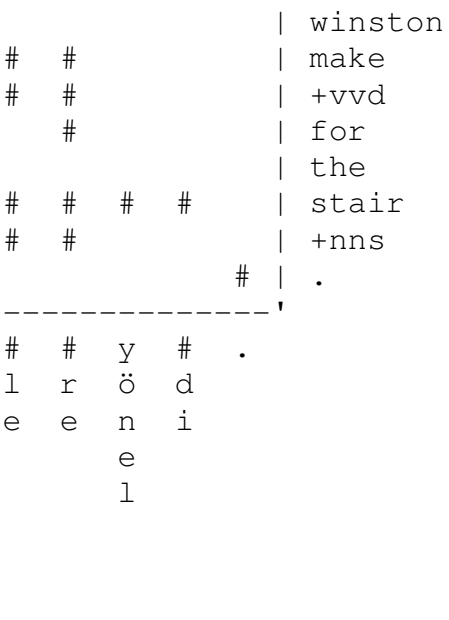
We provide below an example of the word alignment process:



1. After two unidirectional alignments are intersected.



2. After applying “Grow-Diag-Final.”



3. After filling morphological holes.

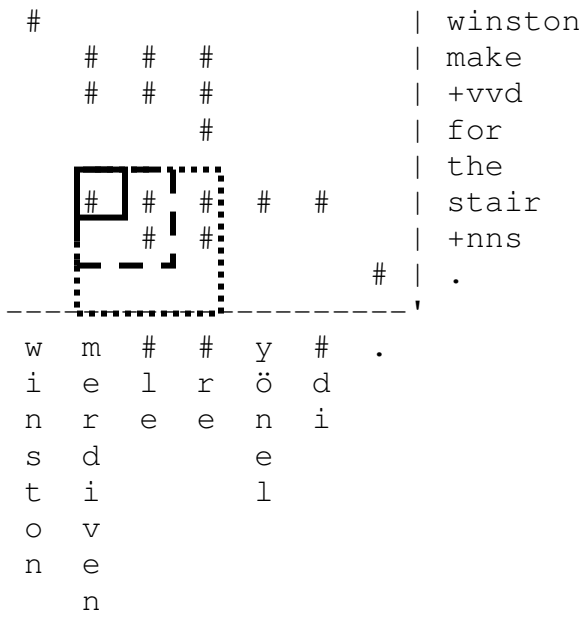
Once these alignment points are determined, the next step is to extract and count phrases that are consistent with them. However, what is consistent with an alignment takes on a different meaning when the aligned units are stems and suffixes than when they are whole words. In our case the formal definition of a bilingual phrase (Zens et al., 2002) becomes thus inapplicable. Instead we use a custom phrase extraction algorithm that is based on the following assumptions:

- Each Turkish suffix can be aligned to an English stem or a suffix. Each English suffix, on the other hand, should be aligned to a Turkish suffix.
- Suffix alignments are independent of each other in that phrase alignments involving more than one Turkish suffix without a stem are meaningless.
- A Turkish phrase can be suffix, in which case it should be aligned to one English token only, or a stem and suffixes following it, or a set of such phrases. These should be aligned to similarly defined English phrases (note, however, that we don’t consider English suffixes by themselves to be phrases.

The algorithm works by considering Turkish strings consistent with the above definition of a phrase, extracting aligned English phrases, and then counting these extracted pairs. Suffix phrases for Turkish are extracted from high-precision intersected alignments that also ensure 1-to-1 correspondence. Extracting stem phrases is trickier. We start by considering each Turkish stem aligned with an English stem. This alignment point is also an initial phrase alignment. Then we expand from this alignment point by considering gradually increasing spans over the Turkish and English sentences. If an alignment is found then a new phrase that is consistent with that alignment and all the previous alignments leading up the initial stem alignment is constructed. This is repeated

until we hit the end of either the English or the Turkish sentence, or we add no new alignment points.

The example below demonstrates this process:



Begin with aligned stems “merdiven” and “stair”:

Extracted at iteration 0 (solid lines)	merdiven : stair
Extracted at iteration 1 (dashed lines)	merdiven #le : stair +nns
Extracted at iteration 2 (dotted lines)	merdiven #le #re : stair +nns

This example also demonstrates a key advantage of our phrase extraction algorithm. The correct morphological analysis of “merdivenlere” is, in our notation, “merdiven #ler #e.” Now if we assume that “#le” approximates “#ler” and “#re” approximates “#e” then we have the following phrases:

[1] merdiven : stair
[2] merdiven #ler : stair #s[5]
[3] merdiven #ler #e : stair #s

Given Turkish morphology not only 1 and 2 but all three of the above phrases are consistent in that if “the stairs” is the object of the English sentence than it should likely be translated as “merdivenlere” and not “merdivenler.” In any case generating these translation options are an important step in the direction of improving data scarcity problems.

[5] We use the surface form notation here to better illustrate the point.

As the final step of creating the translation model, each extracted phrase pair is scored with its relative frequencies with respect to either the English or the Turkish phrase. Hence two phrase translation probability distributions are calculated:

$$p(e \mid t) = \text{count}(e, t) / \sum_f \text{count}(e, t) \quad \text{and} \quad p(e \mid t) = \text{count}(e, t) / \sum_f \text{count}(e, t)$$

The translation model we produce at the end of this step, coupled with the language model we produce in the next step, form the input to the decoder.

2.4 The Language Model

We trained our language model on the segmented Turkish training corpus using SRI Language Modeling toolkit (Stolke, 2002) with Kneser-Ney smoothing. The assumption was that our approximation to a morphological analysis would be accurate enough to reduce perplexity of the model to a reasonable level. Our results indicate that this was indeed the case.

In contrast to word alignment modeling, language modeling for morphologically rich languages in general, and for Turkish in particular, has received somewhat significant attention in the NLP research community (e.g. Siivola et al., 2003).[6] Since our simple approach yielded good enough results, and due to time and scope restraints, we leave evaluation of an intricate language modeling scheme within the context of our translation system as further work.

3 Evaluation: Qualitative and Quantitative Results

To evaluate our system we developed a baseline translator using a standard assembly line training for the translation model (Koehn, 2004a), SRI Language Modeling Toolkit on the original training corpus for the language model, and Pharoah as the decoder (Koehn, 2004b). This baseline was compared with two translation systems: one that uses a single translation model (phrase translation probabilities; referred below as "Turkalator 1"), and another that uses a double translation model (phrase translation probabilities and inverse phrase translation probabilities; referred below as "Turkalator 2"). We used the BLEU metric to obtain a quantitative results regarding the relative performances of the systems we compared. These are given in the table below:

	Baseline	**Turkalator 1**	**Turkalator 2**
BLEU Score	9.12	16.80	17.00
1-gram precision score	32.9	44.6	45.8

[6] It seems that the main reason for this is an impetus that comes from speech recognition research, which is more rooted in comparison to statistical MT research.

2-gram precision score	12.4	22.3	23.0
3-gram precision score	5.9	12.5	13.2
4-gram precision score	2.9	7.1	7.5

Three conclusions can be drawn from these results. First, neither Turkalator nor the baseline system performs that well. During our work on this project it became increasingly obvious to us that this is indeed what we should have expected. The truth is putting together a decent translation system for a language that is structurally very different from major European languages requires quite a substantial amount of work beyond what existing software can offer.

Second, Turkalator performed significantly better (84 % to be exact) than the baseline translator, indicating that at least one of the components we developed for it was an important step in the right direction. This was also to be expected since – as it should be clear from the previous sections – we employed techniques specifically tuned for the special task at hand and with careful attention paid to relevant linguistic facts.

Third, the performance of the system that uses bidirectional phrase translation probabilities was better, but only slightly than that of the system that uses a single phrase translation table. This indicates that while using two translation models simultaneously results increases system accuracy, this increase is by no means comparable to increases in performance resulting from attending to the specific features of the English-to-Turkish MT.

Quantitative evaluation of translation quality is an issue still very much open to debate. Particularly, researchers generally use BLUE metric as a rough guide, preferring human judgment as the ultimate metric. Thus it is wise to consider the actual translations Turkalator produces along with the baseline translations to get a better idea about translation quality.

Before looking at individual translations, let us turn to general patterns. The first pattern to observe is that the baseline translator almost always produces translations where several English words remain un-translated. This is a clear indication of the scarcity problem that we have focused on throughout. On the other hand, we were glad to see that this problem was largely solved for Turkalator; in fact, it almost always produced entirely Turkish translations. There was, however, a cost for this: it seems that in reducing scarcity we have introduced some noise to our phrase alignments. Even though the output of our system is almost entirely Turkish, most of it is also rather meaningless.[7]

Finally, let us consider specific translations. Here is an example:

English input: “it was three years ago .”

[7] It is worth noting that while whole sentences are very hard to make sense of phrases are usually much easier to understand. An interesting extension to the present work would be exploring these methods in a smaller scope, e.g. for noun phrases only. One could go in the other direction as well by augmenting the translation system with some basic knowledge of Turkish syntax and morpho-syntax to ensure more meaningful sentence productions.

Baseline translation: "bu was üç yıl ago ." [this was three years ago.]

Turkalator translation:"bununla üç yıl önceydi ." [with this it was three years ago]

Turkish reference: "üç yıl önceydi ."

First and foremost, this example illustrates how the baseline suffers from data scarcity: it fails to find translations for "was" and "ago."[8] Our translation on the other hand doesn't do badly, producing the correct translation, but with an extra word. It is not very difficult to see how this word got introduced to the translation: the subject in the Turkish reference translation is hidden; if not this token would be corresponding to "it." As such, however, the translator aims to find a good translation of "it" and picks "bununla."

Here is another example:

English input: "regulation on annual paid leave"

Baseline translation: "yönetmeliği yıllık ücretli leave"

Turkalator translation: "yıllık ücretli izin hakkında tüzük"

Turkish reference: "yıllık ücretli izin yönetmeliği"

In this case our translation looks rather strange in comparison to the reference translation at a first blush. But anyone who speaks Turkish can tell you that they mean more or less the same thing: the difference is, in fact, similar to that of "regulation on annual paid leave" and "annual paid leave regulation." This indicates that on some level our translation is even better than the reference in that it stays true to the phrase structure of the original input.

4 Further Work

We end by offering some directions for extensions to the preliminary investigations instantiated in the present work.

The most obvious extension involves the use of a fully-functional morphological transducer to pre- and post-process the text so that there would be no need for a mere approximation. A somewhat more ambitious endeavor might be to deal correctly with morpho-syntax of Turkish. This could prove especially useful in extracting accurate phrase alignments.

These are some broad directions one could take. Rest assured there are many aspects of English – Turkish statistical MT challenge that are worth taking up and will, hopefully, be taken up as larger resources become readily available.

[8] A word about translating the verb "to be" into Turkish: the Turkish counterpart of this verb is almost always either hidden or is a morpheme like "idi" or "#di." Therefore it is understandable that in the lack of morphological analysis of any sort, the baseline model fails to establish an alignment between "was" and "#ydi" as is the case here.

5 Acknowledgement

We would like to thank Prof. Kemal Oflazer of Sabanci University, who by presenting us very graciously with the parallel corpus made this project possible.

6 Work Distribution

For the final project, Gorkem did everything that had to do with Turkish morphology, and Jonathan did everything else. More specifically Gorkem designed and implemented the segmentation, word alignment and phrase extraction tools. Jonathan put together the entire baseline system. He also stitched together parts of Turkalator so that it would run smoothly with the Pharaoh decoder.

7 References

Corston-Oliver, Simon and Michael Gamon. 2004. Normalizing German and English Inflectional Morphology to Improve Statistical Word Alignment. In *Proceedings of the Conference of the Association for Machine Translation in the Americas*. 48-57.

Dejean, Herve, Eric Gaussier, Cyril Goutte and Kenji Yamada. 2003. Reducing Paramater Space for Word Alignment. In *Proceedings from the HLT-NAACL 2003 workshop on Building Parallel Texts.* 23-26.

Koehn, Philip, Frank J. Och and Daniel Marcu. 2003. Statistical Phrase Based Translation. In *Proceedings of the Joint Conference on HLT-NAACL*. 48-54.

Koehn, Philip. 2004a. Pharaoh: Training Manual.

Koehn, Philip. 2004b. Pharaoh: a Beam Search Decoder for Phrase-Based Statistical Machine Translation Models.

Och, Franz J. 2000. Giza++: Training of Statistical Translation Models. Available at http://www-i6.informatik.rwthaachen.de/~och/software/GIZA++.html.

Schafer, Charles and Elliott F. Drabek. 2005. Models for Inuktitut-English Word Alignment. In *Proceedings of the ACL Workshop on Building and Using Parallel Texts.* 79-82.

Siivola, Vesa, Teemu Hirsimaki, Mathias Creutz and Mikko Kurimo. 2003. Unlimited Vocabulary Speech Recognition Based on Morphs Discovered in an Unsupervised Manner. In *Proceeding of Eurospeech'03.* 2293–2296.

Stolcke, Andreas. 2002. SRILM – an Extensible Language Modeling Toolkit. In *Proceedings of the International Conference on Spoken Language Processing.*

Zemberek. Available at https://zemberek.dev.java.net/.

Zens, Richard, Frank J. Och, and Hermann Ney. 2002. Phrase-based Statistical Machine Translation. In *Proceedings of the German Conference on Artificial Intelligence.*