Low Cost Portability for Statistical Machine Translation based on N-gram Frequency and TF-IDF

Matthias Eck, Stephan Vogel and Alex Waibel

Interactive Systems Laboratories
Carnegie Mellon University
Pittsburgh, PA, 15213, USA
matteck@cs.cmu.edu, vogel+@cs.cmu.edu, waibel@cs.cmu.edu

Abstract

Statistical machine translation relies heavily on the available training data. In some cases it is necessary to limit the amount of training data that can be created for or actually used by the systems. We introduce weighting schemes which allow us to sort sentences based on the frequency of unseen n-grams. A second approach uses TF-IDF to rank the sentences. After sorting we can select smaller training corpora and we are able to show that systems trained on much less training data achieve a very competitive performance compared to baseline systems using all available training data.

1. Introduction

The goal of this research was to decrease the amount of training data that is necessary to train a competitive statistical translation system regardless of the actual test data or its domain. "Competitive" here means that the system should not produce significantly worse translations compared to a system trained on a significantly larger amount of data.
It is important to note that this is not an adaptation approach as we assume that the test data (and its domain) is not known at the time we select the actual training data.

Statistical machine translation can be described in a formal way as follows:

$$t^* = \arg\max_t P(t \mid s) = \arg\max_t P(s \mid t) \cdot P(t)$$

Here t is the target sentence, and s is the source sentence. P(t) is the target language model and P(slt) is the translation model used in the decoder.
Statistical machine translation searches for the best target sentence from the space defined by the target language model and the translation model.
Statistical translation models are usually either phrase- or word-based and include most notably IBM1 to IBM4 and HMM ([1], [2], [3]). Some recent developments focused on online phrase extraction ([4], [5]).
All models use available bilingual training data in the source and target languages to estimate their parameters and approximate the translation probabilities.
One of the main problems of Statistical Machine Translation (SMT) is the necessity to have large parallel corpora available. This might not be a big issue for major languages, but it certainly is a problem for languages with fewer resources ([6], [7]). To improve the data situation for these languages it is necessary to hire human translators at enormous costs who translate corpora that can later be used to train SMT systems.
Our idea focuses on sorting the available source sentences that should be translated by a human translator according to their approximate importance. The importance is estimated using a frequency based and an information retrieval approach.

2. Motivation

There are three inherently different motivations for the goal of limiting the amount of necessary training data for a competitive translation system. We described those motivations and their applications already in the paper [8].

Application 1: Reducing Human Translation Cost

The main problem of portability of SMT systems to new languages is the involved cost to generate parallel bilingual training data as it is necessary to have sentences translated by human translators.
An assumption could be that a 1 million word corpus needs to be translated to a new language in order to build a decent SMT system.
A human translator could charge in the range of approximately 0.10-0.25 USD per word depending on the involved languages and the difficulty of the text. The translation of a 1 million word corpus would then cost between 100,000 and 250,000 USD.
The concept here is to select the most important sentences from the original 1 million word corpus and have only those translated by the human translators. If it would still be possible to get a similar translation performance with a significantly lower translation effort, a considerable amount of money could be saved.
This could especially be applied to low density languages with limited resources ([6], [7]).

Application 2: Translation on Small Devices

Another possible application is the usage of statistical machine translation on portable small devices like PDAs or cell phones. Those devices tend to have a limited amount of memory available which limits the size of the models the device can actually hold and a larger training corpus will usually result in a larger model. The more recent approaches to online phrase extraction for SMT make it necessary to have the corpus available (and in memory) at the time of translation ([4], [5]).
Given the upper example, a small device might not be able to hold a 1 million word bilingual corpus but e.g. only a corpus

with 200,000 words. The question is now which part of the corpus (especially which sentences) should be selected and put on the device to get the best possible translation system.

Application 3: Standard Translation System

Even on larger devices that do not have rigid limitations of memory, the approach could be helpful. The complexity of online phrase extraction and standard training algorithms depends mainly on the size of the bilingual training data. Limiting the size of the training data with the same translation performance on these devices would speed up the translations.

Another problem is that the still widely used 32 bit machines like the Intel Pentium 4 and AMD Athlon XP series can only address up to 4 gigabytes of memory. There are already bilingual corpora in excess of 4 gigabytes available and therefore it is necessary to select the most important sentences from these corpora to be able to hold them in memory. (The last issue will certainly be resolved by the widespread introduction of 64 bit machines which can theoretically address 17 million terabytes of memory.)

3. Previous Work

This research can generally be regarded as an example of active learning. This means the machine learning algorithm does not just passively train on the available training data but plays an active role in selecting the best training data.

Active learning, as a standard method in machine learning, has been applied to a variety of problems in natural language processing, for example to parsing ([9]) and to automatic speech recognition ([10]).

It is important to note the difference between this approach and approaches to Translation Model Adaptation ([11]) or simple subsampling techniques that are based on the actual test data. Here we assume that the test data is not known at selection time so the intention is to get the best possible translation system for every possible test data.

Our previous work in this area focused on improving the n-gram (type-) coverage by selecting the sentences based on the number of previously unseen n-grams they contain [8]. Section 4.2 will give a short overview over our previous best method.

4. Description of sentence sorting

4.1. Algorithm

The sentences are sorted according to the following very simple algorithm.

```
For all sentences that are not in the sorted list
        Calculate weight of sentences
        Find sentence with highest weight
        Add sentence with highest weight to sorted list
```

The interesting part is the calculation of the weight of each sentence. The weight of a sentence will generally depend on the previously selected sentences.
We present three different schemes to calculate the importance of a sentence. Section 4.2 presents our previous best selection approach and section 4.3 an approach that weights sentences based on the frequency of the unseen n-grams. The method in section 4.4 uses TF-IDF to find sentences that are different from the already seen sentences.

4.2. Previous Best Weighting Scheme

As stated earlier our previous work in this area focused on optimizing the sorting of the sentences based on the n-gram coverage.
The best results were achieved using the following weighting term:

$$\text{previous\_best\_weight}(sentence) = \frac{\sum_{n=1}^{2} \#(\text{unseen } n-\text{grams})}{|sentence|}$$

This means for each sentence, which had not been sorted yet, the number of unseen uni- and bigrams was calculated and divided by the length of the sentence (in words). This gave significantly better results than the baseline systems where the sentences were not weighted.

4.3. Weighting of Sentences Based on N-gram Frequency

The problem with the previous best system is that every unseen unigram gets the same weight. Words that only occur once in the whole training data will be given the same value as higher frequent and probably more important words. The same is certainly true for low- and high-frequency bigrams.
This is why we wanted to make sure that our new weighting schemes focus on high-frequency n-grams and put less weight on lower frequency n-grams. This means the goal here is not necessarily to optimize the coverage of the types but of the tokens.
We use the frequency of the n-grams in the training data to estimate their importance. The first term just sums over the frequencies for every unseen n-gram to get the sentence weight.

$$\text{weight}_j(sentence) = \sum_{n=1}^{j}\left[\sum_{\text{unseen } n-\text{grams}} \text{frequency}(n-\text{gram})\right]$$

The parameter j here determines the n-grams that are considered and was set to values of 1, 2 and 3 in the experiments.
This means an unseen sentence like *"Where is the hotel?"* will have a high weight, especially for data in the tourism domain because we can assume that every n-gram in this sentence is rather frequent.
These simple weighting schemes already show improvements over the baseline systems as shown in the later parts of the paper but they have various shortcomings. They do not take the actual translation cost of the sentence into account. (Translators generally charge per word and not per sentence). This leads to the fact that longer sentences tend to get higher weights than shorter sentences, because they will contain more, and possibly higher frequent, unseen n-grams. The focus on token-coverage is certainly very helpful but longer sentences are more difficult for the training of statistical translation models. (When training the translation model IBM1 for example every possible word alignment between sentences is considered.)

To fix these shortcomings we changed the weighting terms to incorporate the actual length of a sentence by dividing the sum of the frequencies of the unseen n-grams by the length of the sentence:

$$\text{weight}_{\text{j}}(sentence) = \frac{\sum_{n=1}^{j}\left[\sum_{\text{unseen } n-\text{grams}} \text{frequency}(n-\text{gram})\right]}{|sentence|}$$

This changes the weight to – informally speaking – "newly covered tokens in the training data per word to translate".
As noted earlier the algorithms for training translation models in statistical machine translation usually work better (and faster) on shorter sentences. For this reason we also tried to divide by the square of the length of a sentence which prefers even shorter sentences.
Overall the weighting terms can be written as:

$$\text{weight}_{\text{i, j}}(sentence) = \frac{\sum_{n=1}^{j}\left[\sum_{\text{unseen } n-\text{grams}} \text{frequency}(n-\text{gram})\right]}{|sentence|^{i}}$$

We introduce the second parameter *i* here to indicate the exponent of the sentence length (values used in the experiments were 0, 1 and 2).
It is certainly possible to use higher values for *i* and *j* but the results indicated that higher values would not produce better results.

4.4. Weighting of sentences based on TF-IDF

The second approach for the weighting of sentences is based on a different idea and uses an information retrieval method (TF-IDF) to attach a weight to sentences.

TF-IDF similarity measure

TF-IDF is a similarity measure widely used in information retrieval. The main idea of TF-IDF is to represent each document by a vector in the size of the overall vocabulary. Each document D (this will be a sentence or a set of sentences in our case) is then represented as a vector $(w_1, w_2, ..., w_m)$ if m is the size of the vocabulary. The entry w_k is calculated as:

$$w_k = tf_k * \log(idf_k)$$

- tf_k is the term frequency (TF) of the k-th word in the vocabulary in the document D i.e. the number of occurrences.
- idf_k is the inverse document (IDF) frequency of the k-th term, given as

$$idf_k = \frac{\#\text{documents}}{\#\text{documents containing k - th term}}$$

The similarity between two documents is then defined as the cosine of the angle between the two vectors.

Sentence weighting with TF-IDF

The idea now is to use TF-IDF to find the most *different* sentence compared to the already selected sentences and give this one the highest importance - this means we just select the sentence with the lowest TF-IDF score (compared to the already selected sentences) next.
The first sentence here has to be randomly selected because there is nothing to compare the available sentences against in the first step. The randomly selected sentence could be:

1.	*Where is the hotel?*

In the next step the TF-IDF score for every still available sentence compared to this sentence is calculated.
Sentences that do not have a single common word with this sentence will get the lowest possible TF-IDF score of 0 and one of those will again be selected, for example:

1.	*Where is the hotel?*
2.	*I had soup for dinner.*

At some point there will be no more sentences left that only contain unseen words so every sentence will get a positive TF-IDF score. The lowest TF-IDF score will then be for sentences that have the fewest number of already seen words and the highest document frequency for these words. A selected sentence in this example could be:

1.	*Where **is** the hotel?*
2.	*I had soup for dinner.*
3.	*This **is** fine.*

This sentence only shares the word *"is"* with the already sorted sentences. The word *"is"* most likely has a very high document frequency, thus a low IDF score which leads to an overall low score for this particular sentence.
A sentence like *"We ate dinner at a restaurant."* will get a higher score because the shared word *"dinner"* is certainly less frequent than *"is"* and will get a higher IDF score.
The TF score in this example would be the same so it can be ignored. In the next iteration the TF score for *"is"* in the sorted sentences will be higher, which in turn lowers the chances to select another sentence with *"is"*.
This means overall that this weighting scheme will make sure that at the beginning new and unseen words are covered and it will give more weight to higher frequent words later, which is the same behavior as the weighting schemes presented in section 4.3.

A more information-retrieval centered motivation for the TF-IDF method could be: We always select the sentence with the topic that is *"furthest away"* from the topic(s) of the sentences we already sorted. This will make sure that we cover all possible topics that are in our training data and might come up in the test data.

Generalizing TF-IDF for N-grams

TF-IDF can easily be generalized to n-grams by using every n-gram as an entry in the document vectors (instead of only using words). We tried this for n-grams up to bigrams and plan on doing experiments with higher n-grams.
The following section 5 will give an overview over the experiments that were done using the three presented approaches to sort sentences according to their estimated importance.

5. Experiments English-Spanish

5.1. Test and Training Data

The full training data for the translation experiments consisted of 123,416 English sentences with 903,525 English words (tokens). This data is part of the BTEC corpus ([12]) with relatively simple sentences from the travel domain. The whole training data was also available in Spanish (852,362 words).
The testing data which was used to measure the machine translation performance consisted of 500 lines of data from the medical domain.
All translations in this task were done translating English to Spanish.

5.2. Machine Translation System

The applied statistical machine translation system uses an online phrase extraction algorithm based on IBM1 lexicon probabilities ([3], [13]). The language model is a trigram language model with Kneser-Ney-discounting built with the SRI-Toolkit ([14]) using only the Spanish part of the training data.
We applied the standard metrics introduced for machine translation, NIST ([15]) and BLEU ([16]).

5.3. Baseline and Previous Best Systems

The baseline system that uses all available training data achieved a NIST score of 4.19 [4.03; 4.35][1] and a BLEU score of 0.141 [0.129; 0.154][1].
For the baseline systems that do not use all available training data we selected sentences based on the original order of the training corpus and trained the smaller systems from this data. The second "baseline" systems were trained using the previous best approach presented in section 4.2.
Translation systems trained on these (smaller) data sets give the scores shown in diagrams 1 and 2. The diagrams clearly illustrate that after a rather steep increase of the scores until the translation of approximately 400,000 words the scores of the baseline increase only slightly until they reach the final score for the system using all available training data.
The previous best selection especially benefits at the beginning for a lower number of translated words and hits a NIST score of 4.0 at 170,000 translated words, which is very close to the confidence interval and only about 5% worse than the best overall score. A NIST score of 4.1 is already achieved at 220,000 translated words and 2% worse than the final baseline of 4.19. At 10,000 translated words the previous best system achieves a NIST score of 2.56, compared to a baseline of 2.04.

[1] 95% confidence intervals

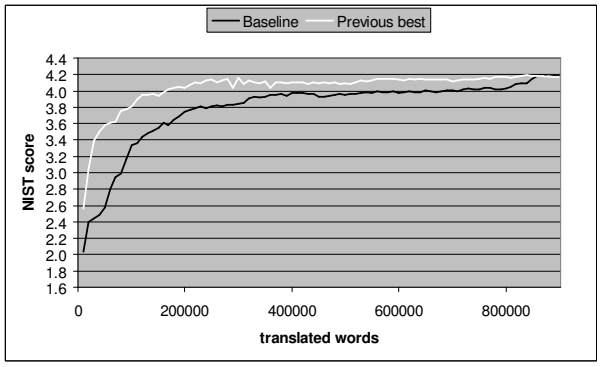

Diagram 1: NIST scores for Baseline and Previous best

The picture is similar for the BLEU scores. The previous best selection reached a BLEU score of 0.13 at 400,000 translated words. The reason for the necessity to translate more words to reach a BLEU score in the confidence interval of the final system could be that the BLEU score puts higher importance on fluency. Larger systems might benefit from more robust estimations of the larger language models.

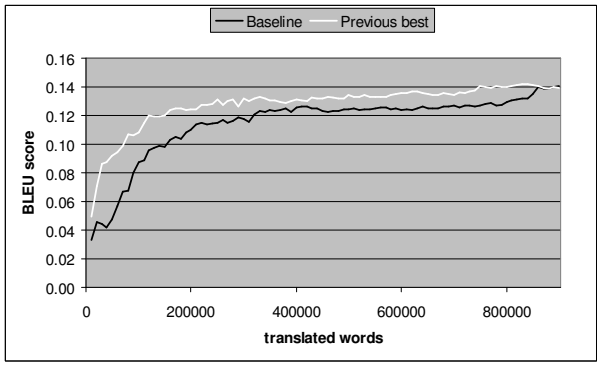

Diagram 2: BLEU scores for Baseline and Previous best

5.4. Translation Results

Because of the limited space we will only show diagrams for the NIST scores for each experiment. This can be justified as the graphs for the BLEU scores showed basically the same behavior.
We did also not include the graph for the previous best system in the diagrams because the new approaches did not always clearly improve over the previous best system and this would have led to even more close-packed diagrams.

Results for term weight$_{0,j}$

Diagram 3 illustrates the NIST scores for systems where the sentences were sorted according to weight$_{0,j}$.
If the optimization only uses the frequency sum of previously unseen unigrams to rank sentences, the systems score significantly higher than the baseline for very small amounts of training data. But the steep increase stops very soon and the systems fall slightly below the baseline, recover towards the end, and finish on the same scores.
These problems are clearly fixed by incorporating the bi- and trigrams into the optimization process. The scores no longer

fall beyond the scores of the baseline systems but stay consistently higher.
The systems optimized on uni- and bigrams ($weight_{0,2}$) are not significantly different from the systems for uni-/bi- and trigrams ($weight_{0,3}$) but show a very similar performance with slight advantages for the uni- and bigram-systems.
Unfortunately both systems do not outperform the previous best method as they reach a NIST score of 4.0 at 230,000 and 240,000 translated words and a score of 4.1 at 300,000 and 320,000 translated words. However all three systems achieve better NIST scores at very small amounts of training data with the same NIST score of 2.72 for 10,000 translated words.

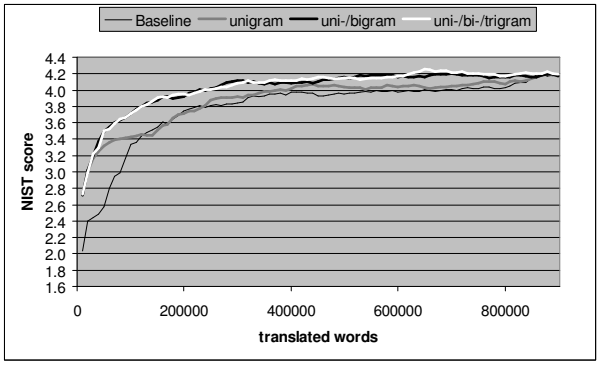

Diagram 3: NIST scores for sentences sorted according to $weight_{0,j}$

Results for term $weight_{1,j}$

The difference between the term $weight_{0,j}$ and $weight_{1,j}$ is the incorporation of the length of a sentence. The frequency sum of the unseen n-grams is divided by the number of words in the respective sentence to get the weight for the sentence.
Diagram 4 illustrates the associated NIST scores.

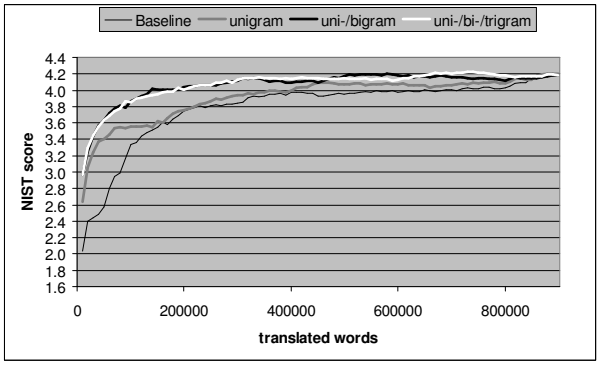

Diagram 4: NIST scores for sentences sorted according to $weight_{1,j}$

A comparison with Diagram 3 shows that the NIST scores for the sorting of the sentences according to $weight_{1,j}$ are even better than for the term $weight_{0,j}$.
We see a very similar behavior for the unigrams and an improvement for the optimizations based on uni- and bigrams and uni-/bi- and trigrams compared to $weight_{0,j}$.
We also do not see any significant differences between the scores for those two optimizations. The performance is very similar with only slight advantages for the optimization based on uni- and bi-grams ($weight_{1,2}$). For this system a NIST score of 4.0 was already reached at 140,000 translated words (190,000 for $weight_{1,3}$) while 4.1 was reached at 300,000 translated words (280,000 for $weight_{1,3}$). It is again possible to outperform the baseline and previous best systems at 10,000 translated words with a NIST scores of 2.64 ($weight_{1,1}$) and 2.97 ($weight_{1,2}$ and $weight_{1,3}$).

Results for term $weight_{2,j}$

As explained in section 4.3 we tried to prefer shorter sentences in term $weight_{2,j}$ by dividing the frequency sum of the unseen n-grams by the square of the number of words in the respective sentence. Diagram 5 illustrates those scores.
The scores overall are similar to the earlier diagrams. The term $weight_{2,2}$ reaches a NIST score of 4.0 at 180,000 (220,000 for $weight_{2,3}$) translated words, and a NIST score of 4.1 at 220,000 translated words (270,000 for $weight_{2,3}$).
The systems again outperform the other systems for 10,000 translated words with NIST scores of 3.02 for $weight_{2,3}$ and 2.98 for $weight_{2,2}$ ($weight_{2,1}$ gets a NIST score of only 2.56).

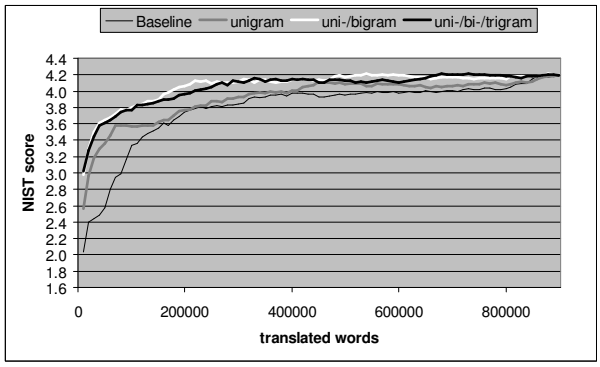

Diagram 5: NIST scores for sentences sorted according to $weight_{2,j}$

Results for TF-IDF based sorting

Diagram 6 shows the scores for the optimization based on TF-IDF for unigrams and uni-/bigrams.
In this case the original TF-IDF (based only on unigrams) slightly outperforms the TF-IDF based on uni- and bigrams but both approaches do not show better results than the earlier weighting terms.

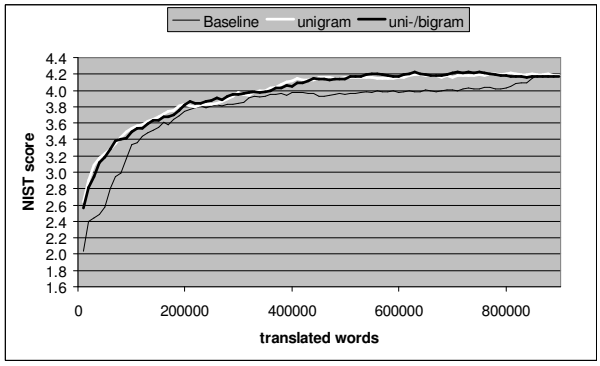

Diagram 6: NIST scores for sentences sorted according to TF-IDF

5.5. Overview

Table 1 compares the results achieved by the different methods with a special focus on small amounts of data. We give NIST scores for 10,000; 20,000; 50,000 and 100,000 translated words. The last 2 columns show the number of translated words (in thousands) necessary to achieve NIST scores of 4.0 and 4.1. (Best values for each column are printed bold.)

	Score for 10k translated words	Score for 20k translated words	Score for 50k translated words	Score for 100k translated words	Translated words for 4.0 (NIST)	Translated words for 4.1 (NIST)
Baseline	2.04	2.40	2.58	3.34	650k	850k
Previous best	2.56	3.05	3.56	3.81	170k	**220k**
$weight_{0,1}$ (unigram)	2.72	3.00	3.31	3.42	380k	760k
$weight_{0,2}$ (uni-/bigram)	2.72	3.02	3.49	3.72	230k	300k
$weight_{0,3}$ (uni-/bi-/trigram)	2.72	3.00	3.50	3.71	240k	320k
$weight_{1,1}$ (unigram)	2.64	2.05	3.40	3.55	410k	450k
$weight_{1,2}$ (uni-/bigram)	2.97	3.25	3.63	**3.86**	**140k**	300k
$weight_{1,3}$ (uni-/bi-/trigram)	2.97	3.29	3.63	3.85	190k	280k
$weight_{2,1}$ (unigram)	2.56	2.98	3.36	3.57	400k	450k
$weight_{2,2}$ (uni-/bigram)	2.98	**3.30**	**3.65**	3.80	180k	**220k**
$weight_{2,3}$ (uni-/bi-/trigram)	**3.02**	3.27	3.62	3.77	220k	270k
TF-IDF (unigram)	2.63	2.90	3.23	3.53	360k	390k
TF-IDF (uni-/bigram)	2.57	2.82	3.19	3.50	370k	430k

Table 1: Performance Overview

One might argue that improvements at very small data sizes are not relevant, as the translations will still be very deficient. This might be the case, but there are applications where even a low-quality translation can be helpful ([17]). And as we showed in [8] - some translations are surprisingly good, even for very small amounts of training data.

6. Future Work

The presented weighting schemes could certainly incorporate other features of the original training data.

The pure frequency based approach "tries" to cover every n gram once and then does not consider it anymore. It might be helpful to have a goal of covering every n-gram a number of times to get better estimates of translation probabilities.

The TF-IDF based sorting did not yet show improvements over the earlier approaches. We hope that it will be beneficial to further investigate this idea and maybe combine it with the other methods.

Both presented methods give a high weight to function words at the beginning. This is not necessarily desirable so it could be helpful to lower the impact of function words and increase the weight of (high-frequent) content words. Especially the NIST score could benefit from correctly translated content words, as it incorporates the information gain in the score calculation.

It might be reasonable for some applications to also consider the target language part of the training data when sorting the sentences. This is certainly not possible if the goal is to limit the effort for human translators and the target sentences are not even available at selection time. It could however be included in the selection of training data for small devices because here the translations will already be available.

7. Conclusion

We presented two new weighting schemes to sort training sentences for statistical machine translation according to their importance for the translation performance.

The first method mainly tries to improve the token coverage while taking the sentence length into account. We are able to outperform our baseline and our previously best system and see especially nice improvement for very small data sizes. The focus on token coverage is achieved by using the frequency of the previously unseen n-grams as the basis for the sentence weight.

We also presented a second idea that bases the sorting of the sentences on the similarity measure TF-IDF, but we did not see improvements over the first method.

8. References

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

[2] Stephan Vogel, Hermann Ney, and Christoph Tillmann, 1996. *HMM-based Word Alignment in Statistical Translation.* Proceedings of Coling 1996, Copenhagen, Denmark.

[3] Stephan Vogel, Ying Zhang, Alicia Tribble, Fei Huang, Ashish Venugopal, Bing Zhao, and Alex Waibel. 2003. *The CMU Statistical Translation System.* Proceedings of MT Summit IX, 2003. New Orleans, LA, USA.

[4] Chris Callison-Burch, Colin Bannard and Josh Schroeder. 2005. *Scaling Phrase-Based Statistical Machine Translation to Larger Corpora and Longer Phrases.* Proceedings of ACL 2005, Ann Arbor, MI, USA.

[5] Ying Zhang and Stephan Vogel. 2005. *An Efficient Phrase-to-Phrase Alignment Model for Arbitrarily Long Phrases and Large Corpora.* Proceedings of EAMT 2005, Budapest, Hungary.

[6] Tony McEnery, Paul Baker, Lou Burnard. 2000. *Corpus Resources and Minority Language Engineering.* Proceedings of LREC 2000, Athens, Greece.

[7] Alon Lavie, Katharina Probst, Erik Peterson, Stephan Vogel, Lori Levin, Ariadna Font-Llitjós, and Jaime Carbonell. 2004. *A Trainable Transfer-based Machine Translation Approach for Languages with Limited Resources.* Proceedings of EAMT 2004, Malta.

[8] Matthias Eck, Stephan Vogel, and Alex Waibel. 2005. *Low Cost Portability for Statistical Machine Translation based on N-gram Coverage*. Proceedings of MTSummit X 2005. Phuket, Thailand.

[9] Rebecca Hwa. 2004. *Sample selection for statistical parsing.* Computational Linguistics vol. 30, no. 3.

[10] Teresa. M. Kamm and Gerard G. L. Meyer. 2002. *Selective Sampling of Training Data for Speech Recognition.* Proceedings of HLT 2002, San Diego, CA, USA.

[11] Almut Silja Hildebrand, Matthias Eck, Stephan Vogel and Alex Waibel. 2005. *Adaptation of the Translation Model for Statistical Machine Translation based on Information Retrieval.* Proceedings of EAMT 2005, Budapest, Hungary.

[12] Toshiyuki Takezawa, Eiichiro Sumita, Fumiaki Sugaya, Hirofumi Yamamoto, and Seiichi Yamamoto. 2002. *Toward a Broad-coverage Bilingual Corpus for Speech Translation of Travel Conversation in the Real World.* Proceedings of LREC 2002, Las Palmas, Spain.

[13] Stephan Vogel, Sanjika Hewavitharana, Muntsin Kolss, and Alex Waibel. 2004. *The ISL Statistical Translation System for Spoken Language Translation.* Proceedings of the International Workshop on Spoken Language Translation, Kyoto, Japan.

[14] SRI Speech Technology and Research Laboratory. 1995-2005. SRI Language Modeling Toolkit. http://www.speech.sri.com/projects/srilm/

[15] George Doddington, 2001. *Automatic Evaluation of Machine Translation Quality using n-Gram Co-occurrence Statistics.* NIST Washington, DC, USA.

[16] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. *BLEU: a Method for Automatic Evaluation of Machine Translation.* Proceedings of ACL 2002, Philadelphia, PA, USA.

[17] Ulrich Germann. 2001. *Building a Statistical Machine Translation System from Scratch: How Much Bang Can We Expect for the Buck?* Proceedings of the Data-Driven MT Workshop of ACL 2001. Toulouse, France.