Efficient N-gram Language Modeling for Billion Word Web-Corpora

Lars Bungum and **Björn Gambäck**

Norwegian University of Science and Technology
Trondheim, Norway
{larsbun,gamback}@idi.ntnu.no

Abstract

Building higher-order n-gram models over 10s of GB of data poses challenges in terms of speed and memory; parallelization and processing efficiency are necessary prerequisites to build the models in feasible time. The paper describes the methodology developed to carry out this task on web-induced corpora within a project aiming to develop a Hybrid MT system. Using this parallel processing methodology, a 5-gram LM with Kneser-Ney smoothing for a 3Bn word corpus can be built in half a day. About half of that time is spent in the parallelized part of the process. For a serial execution of the script, this time usage would have had to have been multiplied by 250 (corresponding to close to two months of work).

1. Introduction

As part of the language modeling in, for example, a statistical machine translation systems, it is necessary to build n-gram models over very large corpora. The purpose of the language modeling is to help the machine translation system select the correct translation candidate of many, as the graph of possible translations is searched. It is possible to include the language modeling while searching the graph, as well as invoking a separate disambiguation module to select between candidates for a particularly difficult word.

Building higher-order n-gram models over 10s of GB of data poses challenges in terms of speed and memory; parallelization and processing efficiency are necessary prerequisites to build the models in feasible time. The paper describes the methodology developed to carry out this task on web-induced corpora within PRESEMT ("Pattern REcognition-based Statistically Enhanced MT"; `http://www.presemt.eu`), a project developing a hybrid statistical Machine Translation (MT) system. The establishment of a framework which allows for the rapid creation of new large language models of high order is a necessity in such an application.

The n-gram models were built with the standard tool IRSTLM, the IRST Language Modeling Toolkit (Federico and Cettolo, 2007). The IRSTLM framework was adapted to the OpenPBS queue handler in order to distribute the task to a cluster of machines.

The alternative to adapting the parallelization scripts from IRSTLM — or the similar SRILM (Stolcke et al., 2011), or some other already existing, openly-available language modeling toolkit, e.g., RandLM or KenLM described below — would have been to implement a n-gram modeling tool in a parallel programming framework such as MPI or OpenMP. This approach was discarded because it was unlikely to result in any significant gain in performance over the chosen PBS alternative, which produced acceptable results.

Using the adapted scripts we were able to provide the PRESEMT project with higher-order (5- and 7-gram) language models, built and rebuilt according to the needs of the project. Language models of various sorts (lemma-based, word-based, POS-based or combinations thereof) were built, to provide extended information to the search algorithms.

The rest of the paper is laid out as follows. The next section discusses some previous attempts to build very large scale language n-gram models, in particular in the context of statistical machine translation. The corpora used as basis for creating the n-gram models in the present work are described in Section 3. Thereafter the experimental setup and methodology is detailed in Section 4. The core of the paper are the experimental results and statistics on the created models which are given in Section 5. The final section then sums up the discussion and points to the conclusions that can be drawn.

2. Related Work

Brants et al. (2007) investigate how large language models can be built using distributed techniques. They report decreasing perplexity and better n-gram coverage as the number of tokens increase, and also show how the larger n-gram models improve BLEU (Papineni et al., 2002) scores for a given MT task. The approach taken for the distributed compilation of language models is the Map-Reduce framework, where the counting step of the n-gram model creation is parallelized. This process is separated into a *mapping* step where words and keys are gathered, after which they are *reduced* into separate processes making sure the counting of the same keys are done on the same machines. For a 30G size corpus, Brants et al. report a computation time of two days for language models built with Kneser-Ney smoothing.

Several avenues have been taken to the problems of storing and processing huge n-gram language models. To this end, Talbot and Osborne (2007) use a Bloom filter with logarithmically quantized n-gram frequency counts, that is, a *lossy* randomized representation efficiently encoding the n-grams together with their frequency information. This Randomised Language Modelling (now commonly referred to as 'RandLM') can give significant storage space reductions but at the cost of some extra false positives and reduced decoding speed.

In contrast, Pauls and Klein (2011) discuss a number of compact *lossless* implementations based on tabular tries storing only the suffix of the n-gram (the last word) together with an offset encoding the context (the remaining words). Working on the 4B n-gram Google corpus, they encode each n-gram in only 23 bits, in the best case reducing storage requirements to only 1/4 and improving even on the best previous lossy representations. Encoding the context also gives faster processing (since there is no need to look up the context again when moving on to the next word). Combining this with using a direct-mapped cache, Pauls and Klein report obtaining substantial speed-ups (up to 300%).

In a similar fashion, Heafield (2011) introduces a language modeling library called 'KenLM'. He compares regular hash tables to using tries and shows that a linear probing hash table method gives significantly faster processing while the tries produce a lot smaller data structures. Heafield also discusses a lossy compression of the trie pointers which further reduces the necessary storage space, but concludes that the linear probing hash tables are preferable if processing speed is more important than reduced memory usage. On the other hand, RandLM is potentially the currently most memory efficient approach, even though the memory allocation needed by the tries can be further optimized by lossless compression, as shown by Raj and Whittaker (2003).

3. Corpora

In the development of the language modeling scripts, three corpora of English ('enTenTen', with 3.5Bn Words), German ('deTenTen', 3.2Bn Words), and Italian ('itTenTen', 2.2Bn Words) were used, all originating in the previous "Web as Corpus" corpora known as, respectively, UKWaC, DeWaC and ItWaC (Baroni and Kilgarriff, 2006).

The corpora were mined from the web and processed into a "vertical" corpus format, with one word occupying one line, displayed in various forms: the original form, lemma, part-of-speech (POS), or the combinations thereof (Kilgarriff et al., 2010; Kilgarriff et al., 2011).

3.1. Noise in the corpora

After the tagging, the corpora still contained noise, that had to be addressed before building the models. Many of the problems stem from the web-corpora being encoded in a mixture of character sets.

For example, the most notable sources of noise in the German corpus were:

1. Words beginning with special characters (*-Bus*).
2. Higher order special UTF characters (different newlines and so on). This created some headaches before a proper solution could be found. With `less`, the characters get rendered like `U+0084`, etc., but with `cat` and `more` they are invisible (as they perhaps should be). In the LM software they turn up as "token ghosts".
3. Umlauts being rendered differently (from various character set).
4. Words that are split up that clearly are supposed to be one word (such as "*Bewaff- net*").
5. Repeated strings to three occurrences (the corpus introduced many tokens of repeated words).
6. Very long words (usually 50 or 100 characters, possibly created by hammering the keyboard) .

Scripts were written to mitigate these effects, as well as unifying different representations of dates and numbers into the collection tokens `@date` and `@card`.

3.2. Preprocessing

All the corpora were tokenized and part-of-speech tagged with the TreeTagger (Schmid, 1994), and presented in a "vertical format", where each word used one line, and the different forms of the word — original form, lemma, POS, and special lemma + POS (called lempos) — were printed, in a tab-separated form.

Using the data more or less in the same form as it was retrieved from the web was a project research goal in its own right. However, it proved necessary to do some preprocessing to get workable data out of the corpora. Most notably stripping higher-order UTF characters that would crash the IRSTLM or give undesired output, for example, "words" rendered as spaces which would produce spurious n-grams. Hence, before building the language models the corpora were transformed from the vertical format to a horizontal format with one sentence per line encapsulated in `<s> </s>` sentence boundary markers.

Especially the German corpus produced a very large number of unique tokens. The highly compounding nature of the German language resulted in many tokens ending in a "–", as the compounds were used in split mode (enumeration or linebreak) in the web material. Date formats also varied a lot, giving rise to many spuriously unique tokens.

4. Methodology

A 96 node cluster partitioned in equal parts of nodes with 48G and 24G RAM was used to perform the experiments. The cluster uses the Linux operating system, and the OpenPBS job scheduler (`http://www.mcs.anl.gov/research/projects/openpbs`). The IRSTLM software package already had scripts for parallel treatment of data developed for another (closed) version of the PBS system, and this was changed to adhere to the slightly different syntax of OpenPBS.

The processing cycle goes through the following steps:

1. A dictionary is compiled for the whole input corpus.
2. The corpus is sectioned into n sections according to word frequency.
3. N-grams are counted for each of these sections.
4. (Sub-)LM scores are computed for the sections.
5. Files are merged into one LM.

The sectioning of the dictionaries (or the unigram) lists balanced with regard to frequency, would create a list of unigrams for each section, totaling to a similar sum. To reach this sum you would need more unigrams as the frequencies go down. The top unigrams would alone constitute a frequency well above the average frequency per section, but a list with only one entry can not be split up. Steps 3 and 4 are the steps that are carried out in parallel on each node. A bash script submits the jobs to the PBS queue and tells the jobs to delay merging until all jobs have successfully finished.

Due to resource constraints, IRSTLM uses similar scripts to section up the building of the LMs also when running on a serial architecture, as memory requirements of building a large model could easily exhaust any machine. Instead of carrying out the steps above after another, the tasks could be submitted to a queue handler, performing them in parallel (with the exception of the first step, dictionary collection, and the last step, merging).

Since the same scripts were used, the parallel processing reliably gave the same output as serial processing, and the speed-up factor (though influenced by the load on the cluster) was easy to assess.

The changes in adapting the Sun Grid PBS script to the OpenPBS format mostly related to the control of work flow. It is possible to specify that specific jobs submitted to the queue will be halted until the successful execution of others. To avoid submitting too many jobs at once, the shell script waited for the dictionary compilation before the n-gram counting and sub-LM (as described above) jobs were submitted. The jobs were sent to the queue at the same time, where each individual sub-LM job depended on the corresponding job for n-gram counting, as the n-gram counts for each section needed to be compiled before their respective sub-LM (LM for that section) could be derived. This was controlled by letting each sub-LM job submitted to the queue handler depend on the successful execution of the corresponding n-gram counting job.

The IRSTLM framework can output LMs in an internal format, the ARPA LM format, as well as a compiled version for quicker access with IRSTLM tools (the local platform is Linux/amd64, but the compile step can be done on any architecture).

5. Results

The corpora mentioned in Section 3. were sectioned into `TRAIN` and `TEST` corpora with a Perl script randomly drawing 10% of the lines into the latter for testing purposes.

The sizes of the corpora are shown in Table 1.

For each language, two corpora were extracted, a lemma corpus, containing only the lemmata in succession and one corpus consisting of full forms. The minor differences in size are explained by idiosyncrasies in the extraction methods from the original format of the corpora.

For all corpora, 5- and 7-gram models were built with Kneser-Ney smoothing. For the *fullform* corpora, 5-gram models with and without pruning of singleton n-grams (i.e., n-grams occurring only once in the corpus) were built as well. Due to preprocessing discarding some tokens from the *fullform* corpora, that were not discarded in their lemmatized

English	TRAIN		TEST	
	Lines	Words	Lines	Words
Lemma	108.4	3150.5	12.0	350.3
Fullform	107.4	3122.6	11.9	347.0

(a) Size of the English Corpora

German	TRAIN		TEST	
	Lines	Words	Lines	Words
Lemma	141.8	2837.0	15.8	315.3
Fullform	141.1	2809.0	15.7	312.2

(b) Size of the German Corpora

Italian	TRAIN		TEST	
	Lines	Words	Lines	Words
Lemma	78.5	2913.4	8.7	323.9
Fullform	77.9	2851.2	8.7	317.0

(c) Size of the Italian Corpora

Table 1: Size of training and test corpora.
Figures reported in million lines and words.

versions, the number of words reported in Table 1 are lower for the *fullform* corpora.

5.1. Corpus Statistics

Evaluating language models is most interesting relative to a specific task such as Machine Translation (MT) or spell checking. In this work, no specific task was employed, and the held-out test corpora were instead used to compute perplexity and out-of-vocabulary (OOV) statistics. For the pruned models, perplexity statistics on the `TEST` corpora were collected. For the unpruned corpora this was not possible due to memory constraints.

The IRSTLM (Federico et al., 2010) language modeling software's standard functionality offers computation of the above-mentioned statistics for a corpus. This was utilized for all three languages and all corpus types.

The results on the `TEST` corpora for nine language models are shown in Table 2. The difference in word numbers in the `TEST` between the dictionary sizes shown in Table 1 are explained by how the sentence boundary markers are counted. In the former table, they are counted once per sentence, whereas all markers are counted in the latter, making the difference equate to the line number.

For the Lemma-based language models, the 7-gram models have the lowest perplexity, whereas the opposite is the case for the Fullform-based models where the 5-gram models have lower perplexity (although just barely) for English and Italian, with and without the effect of the OOV words taken into account. An exception to this picture is the German corpus, where the 7-gram model has markedly lower perplexity with the 7-gram model also for the Fullform version of the corpus.

The difference in perplexity between the Lemma and Fullform-based models are markedly greater for the DE

English	Nw	PP	PPwp	Noov	OOV
5-Lemma	338,4	66,466.56	1,483.36	517,501	0.15%
7-Lemma	338,4	65,720.84	**1,466.72**	517,501	0.15%
5-Fullform	335,1	59,605.54	**1,432.67**	564,070	0.17%
7-Fullform	335,1	59,812.53	1,437.65	564,070	0.17%

(a) EN TEST Corpus

German	Nw	PP	PPwp	Noov	OOV
5-Lemma	299,6	29,740.41	2,088.30	13,46,978	0.45%
7-Lemma	299,6	29,264.45	**2,054.88**	13,46,978	0.45%
5-Fullform	296,5	62,139.56	4,606.35	1,422,892	0.48%
7-Fullform	296,5	60,932.71	**4,516.89**	1,422,892	0.48%

(b) DE TEST Corpus

Italian	Nw	PP	PPwp	Noov	OOV
5-Lemma	315,2	81,825.78	1,682.07	433,929	0.14%
7-Lemma	315,2	79,060.37	**1,625.22**	433,929	0.14%
5-Fullform	308,3	96,281.84	**2,208.01**	482,946	0.16%
7-Fullform	308,3	99,458.00	2,280.85	482,946	0.16%

(c) IT TEST Corpus

Table 2: Statistics of the TEST corpora.

N_w is the total number of words in the evaluation corpus,
PP is the perplexity, and
PP_{wp} reports the contribution of out-of-vocabulary (OOV) words to the perplexity.
The out-of-vocabulary word term OOV is defined as $N_{oov}/N_w * 100$,
with N_{oov} being the number of OOV words.

corpus, suggesting a higher degree of noise in this corpus as discussed in Section 3.1.

5.2. Dictionary Growth Curves

In addition to the statistics above, a *dictionary growth curve* was obtained, that is, a curve showing the amount of n-grams above the orders 0–9, with the OOV frequency in each category when testing on the TEST corpus.

The Dictionary Growth Curves (DCGs) are shown in Tables 3a–3f. The first three columns of each of the tables show the percentage of words in the training corpus whose frequencies are over 0 (all of them, 100%), those having a frequency over 1 (40%), etc.

The reader will notice that the number of dictionary entries (unigrams) should be the same as the number of unigrams in an unpruned language model. However, due to the implementation Kneser-Ney smoothing, the singleton n-grams for order $1, 2$ were pruned away in the process.

Again, the markedly higher number of dictionary entries for the DE corpora stand out.

5.3. N-grams Counts

We also extracted the numbers of each n-gram level from the corpora, which easily done by looking at the header of the output language model files.

The number of n-grams in the models for all three languages are shown in Table 4. For the English and Italian corpora, the 4-grams were the most frequent n-gram order in the language models, whereas the highest amount of non-singleton n-grams was found in the bigram category for the German corpus.

In addition, unpruned models were built for the Fullform corpora. Comparing those to the singleton-pruned language models shows that the amount of n-grams quickly gets enormous. When zipped down, the unpruned models required about 31G of storage (in the intermediate iArpa format), and were thus not very workable for standard machinery. Using the quantization functionality of the IRSTL toolkit might be a solution to this.

5.4. Computation Times

Because of the differences in load on the cluster over time, it is difficult to report accurate computation times. However, as an indication, the whole process of building any one of these language models would take 8–12 hours depending on the cluster load. If the load was low, it was possible to build three such models simultaneously within the same time frame.

Freq	Entries	Percent	Freq	OOV
>0	7,507,448	100.00%	<1	0.15%
>1	3,072,234	40.92%	<2	0.22%
>2	2,074,727	27.64%	<3	0.26%
>3	1,628,826	21.70%	<4	0.29%
>4	1,362,639	18.15%	<5	0.32%
>5	1,185,035	15.78%	<6	0.35%
>6	1,054,988	14.05%	<7	0.37%
>7	956,259	12.74%	<8	0.39%
>8	877,378	11.69%	<9	0.41%
>9	813,647	10.84%	<10	0.43%

(a) DCG for English Lemma Corpus

Freq	Entries	Percent	Freq	OOV
>0	8,203,706	100.00%	<1	0.17%
>1	3,335,431	40.66%	<2	0.24%
>2	2,280,889	27.80%	<3	0.28%
>3	1,801,269	21.96%	<4	0.32%
>4	1,516,574	18.49%	<5	0.35%
>5	1,325,480	16.16%	<6	0.38%
>6	1,185,591	14.45%	<7	0.40%
>7	1,079,341	13.16%	<8	0.43%
>8	994,459	12.12%	<9	0.45%
>9	925,466	11.28%	<10	0.47%

(b) DCG for English Fullform Corpus

Freq	Entries	Percent	Freq	OOV
>0	19,300,334	100.00%	<1	0.45%
>1	7,404,928	38.37%	<2	0.64%
>2	4,841,598	25.09%	<3	0.76%
>3	3,706,439	19.20%	<4	0.86%
>4	3,042,511	15.76%	<5	0.94%
>5	2,606,399	13.50%	<6	1.01%
>6	2,293,273	11.88%	<7	1.07%
>7	2,056,786	10.66%	<8	1.13%
>8	1,870,568	9.69%	<9	1.18%
>9	1,719,753	8.91%	<10	1.22%

(c) DCG for German Lemma Corpus

Freq	Entries	Percent	Freq	OOV
>0	20,775,474	100.00%	<1	0.48%
>1	8,163,441	39.29%	<2	0.68%
>2	5,463,441	26.30%	<3	0.81%
>3	4,253,230	20.47%	<4	0.92%
>4	3,543,995	17.06%	<5	1.01%
>5	3,071,589	14.78%	<6	1.09%
>6	2,731,179	13.15%	<7	1.15%
>7	2,470,988	11.89%	<8	1.21%
>8	2,263,809	10.90%	<9	1.27%
>9	2,096,181	10.09%	<10	1.32%

(d) DCG for German Fullform Corpus

Freq	Entries	Percent	Freq	OOV
>0	6,475,359	100.00%	<1	0.14%
>1	2,778,546	42.91%	<2	0.20%
>2	1,903,603	29.40%	<3	0.24%
>3	1,511,911	23.35%	<4	0.27%
>4	1,275,415	19.70%	<5	0.30%
>5	1,116,931	17.25%	<6	0.32%
>6	1,000,423	15.45%	<7	0.34%
>7	911,485	14.08%	<8	0.36%
>8	840,714	12.98%	<9	0.38%
>9	782,787	12.09%	<10	0.40%

(e) DCG for Italian Lemma Corpus

Freq	Entries	Percent	Freq	OOV
>0	7,365,655	100.00%	<1	0.16%
>1	3,169,535	43.03%	<2	0.22%
>2	2,222,344	30.17%	<3	0.27%
>3	1,786,668	24.26%	<4	0.31%
>4	1,525,238	20.71%	<5	0.34%
>5	1,348,829	18.31%	<6	0.37%
>6	1,219,203	16.55%	<7	0.39%
>7	1,119,484	15.20%	<8	0.41%
>8	1,040,111	14.12%	<9	0.44%
>9	974,612	13.23%	<10	0.46%

(f) DCG for Italian Fullform Corpus

Table 3: DCG curves for the six corpus types

On average, the parallelized jobs would take about 1.5 hours, with the exception of the jobs counting n-grams based on the most frequent unigrams, that could take up to 5 hours to finish. The merging of the sub-language models would have to wait for all the sub-models to finish, and hence it was necessary to wait for these initial jobs to exit. Theoretically it would be possible to find an ideal number of jobs to minimize the total computation time, where the smaller jobs would be made big enough to correspond to the jobs based counting n-grams for the most frequent unigrams. This would lead to a smaller total computation time, being easier on the cluster, but would not change the total time the script

English	Fullform		
Order	5 pruned	5 unpruned	7 pruned
1-gr	3.3	3.3	3.3
2-gr	135.3	135.3	135.2
3-gr	165.6	668.8	165.6
4-gr	**222**	1,451.9	**222.0**
5-gr	179.4	**2,026.2**	179.4
6-gr			115,4
7-gr			72,1
In total	705.7	4,285.5	893.3

(a) EN Corpus n-gram counts

German	Fullform		
Order	5 pruned	5 unpruned	7 pruned
1-gr	8.1	8.1	8.1
2-gr	237.0	237.0	**237.9**
3-gr	168.5	842.9	168.5
4-gr	**180.5**	1,493.3	180.5
5-gr	128.1	**1,844.2**	128.1
6-gr			79.6
7-gr			52.6
In total	722.4	4,425.5	854.7

(b) DE Corpus n-gram counts

Italian	Fullform		
Order	5 pruned	5 unpruned	7 pruned
1-gr	3.1	3.1	3.1
2-gr	131.2	131.2	131.2
3-gr	169.7	670.3	169.7
4-gr	**225.1**	1,454.8	**225.1**
5-gr	174.8	**1,991.3**	174.8
6-gr			114.2
7-gr			79.2
In total	704.1	4,250.9	897.6

(c) IT Corpus n-gram counts

Table 4: N-gram counts for pruned and unpruned 5,7-gram models from the Fullform EN/DE/IT corpora. Figures reported in million n-grams.

needs to return with a language model.

6. Conclusions

Experimenting with building and rebuilding n-gram models built from large corpora requires efficient computation. In this paper we have shown how they can be efficiently built using the IRSTLM framework, adapted to the OpenPBS job scheduler. Although the machinery used can be considered high-end, such equipment is available for many universities and research organizations today.

With web corpora. noise can be a problem, and we have identified steps that can be taken to reduce the number of unique tokens that are not members of the language, but rather have been produced as the result of idiosyncrasies in the corpus processing.

Comparing models of different orders and built on lemmas, nouns, verbs, etc., in a final application (in this case Machine Translation) is also of value. When dealing with large corpora, it is possible to extract valuable linguistic information about the languages, as the perplexity of corpus samples of a language wanders asymptotically towards the perplexity of the language itself.

It is interesting to note the low degree of out-of-vocabulary words, also when using corpora that retain capitalization and inflected forms (as in the Fullform corpora above), an indication of the benefit of large data.

Acknowledgments

The research leading to these results has received funding from the European Community's Seventh Framework Programme (FP7/2007–2013) under grant agreement nr 248307 (PRESEMT). Thanks to André Lynum, Erwin Marsi and the anonymous reviewers for useful comments on earlier drafts of the article.

7. References

Marco Baroni and Adam Kilgarriff. 2006. Large linguistically-processed web corpora for multiple languages. In *Proceedings of the Eleventh Conference of the European Chapter of the Association for Computational Linguistics: Posters & Demonstrations*, pages 87–90, Trento, Italy. ACL.

Thorsten Brants, Ashok C. Popat, Peng Xu, Franz J. Och, and Jeffrey Dean. 2007. Large language models in machine translation. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 858–867, Prague, Czech Republic, June. ACL.

Marcello Federico and Mauro Cettolo. 2007. Efficient handling of n-gram language models for statistical machine translation. In *Proceedings of the Second Workshop on Statistical Machine Translation*, pages 88–95, Prague, Czech Republic, June. ACL.

Marcello Federico, Nicola Bertoldi, and Mauro Cettolo, 2010. *IRST Language Modeling Toolkit, Version 5.50.02: User Manual*. FBK-irst, Trento, Italy, November.

Kenneth Heafield. 2011. KenLM: Faster and smaller language model queries. In *Proceedings of the Sixth Workshop on Statistical Machine Translation*, pages 187–197, Edinburgh, Scotland, July. ACL.

Adam Kilgarriff, Siva Reddy, Jan Pomikálek, and Avinesh PVS. 2010. A corpus factory for many languages. In Nicoletta Calzolari, Khalid Choukri, Bente Maegaard, Joseph Mariani, Jan Odijk, Stelios Piperidis, Mike Rosner, and Daniel Tapias, editors, *Proceedings of the Seventh International Conference on Language Resources and Evaluation (LREC'10)*, pages 904–910, Valletta, Malta, may. European Language Resources Association (ELRA).

Adam Kilgarriff, Avinesh PVS, and Jan Pomikálek. 2011. Comparable corpora BootCaT. In Iztok Kosem and Karmen Kosem, editors, *Proceedings of eLex 2011, Electronic Lexicography in the 21st Century: New Applications for New Users*, pages 122–128, Bled Slovenia, November. Trojina, Institute for Applied Slovene Studies.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Morristown, NJ, USA. ACL.

Adam Pauls and Dan Klein. 2011. Faster and smaller n-gram language models. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 258–267, Portland, Oregon, USA, June. ACL.

Bhiksha Raj and Ed Whittaker. 2003. Lossless compression of language model structure and word identifiers. In *Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing*, page 388–391. IEEE.

Helmut Schmid. 1994. Probabilistic part-of-speech tagging using decision trees. In *Proceedings of International Conference on New Methods in Language Processing*, volume 12, pages 44–49, Manchester, UK.

Andreas Stolcke, Jing Zheng, Wen Wang, and Victor Abrash. 2011. SRILM at Sixteen: Update and outlook. `http://www-speech.sri.com/pubs/papers/Stol1112:SRILM/document.pdf`.

David Talbot and Miles Osborne. 2007. Randomised language modelling for statistical machine translation. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 512–519, Prague, Czech Republic, June. ACL.