interspeech 2008 incorporating SST 08

IRSTLM: an Open Source Toolkit for Handling Large Scale Language Models

Marcello Federico, Nicola Bertoldi, Mauro Cettolo

FBK-irst - Ricerca Scientifica e Tecnologica
Via Sommarive 18, Povo (TN), Italy
<surname>@fbk.eu

Abstract

Research in speech recognition and machine translation is boosting the use of large scale n-gram language models. We present an open source toolkit that permits to efficiently handle language models with billions of n-grams on conventional machines. The IRSTLM toolkit supports distribution of n-gram collection and smoothing over a computer cluster, language model compression through probability quantization, lazy-loading of huge language models from disk. IRSTLM has been so far successfully deployed with the Moses toolkit for statistical machine translation and with the FBK-irst speech recognition system. Efficiency of the tool is reported on a speech transcription task of Italian political speeches using a language model of 1.1 billion four-grams.

Index Terms: Automatic Speech Recognition, Language Modeling, Statistical Machine Translation

1. Introduction

Automatic speech recognition and statistical machine translation are among the tasks in Natural Language Processing that most heavily rely on statistical n-gram language models (LMs). The recent availability of large scale corpora, as those collected from the Web [1, 2], has increased among the research community the demand and interest for building LMs of considerable size. As a consequence, existing software libraries for LMs have been recently updated [3] to cope with the new memory requirements, or novel approaches have been proposed that exploit distributed computations [4].

We present here an open source library that permits to create, store, and access large scale language models with conventional hardware. The library provides training scripts for distributing LM training over a computer cluster through a standard queue manager. The generated LM is stored in the conventional ARPA format, it can be further compiled into a binary format to considerably speed up loading time, n-gram probabilities can be quantized for further space saving, and, finally, the LM can be accessed from disk via memory mapping, resulting in a drastic reduction of memory requirements at a very reasonable time cost.

The toolkit [5][1] so far has been successfully used within the most popular open source toolkit for statistical machine translation, called Moses [6], and with the FBK-irst ASR system for word-lattice re-scoring. In this paper, IRSTLM has been tested by estimating and running a LM of 1.1G 4-grams with a novel large vocabulary phone decoder described in the companion paper [7]. The considered task is the transcription of speeches at the Italian parliament.

[1]IRSTLM is available under a LGPL license from sourceforge.net.

2. Language Model Estimation

LM estimation starts with the collection of n-grams and their frequency counters. Then, smoothing parameters are estimated [8] for each n-gram level; infrequent n-grams are possibly pruned and, finally, a LM file is created containing n-grams with probabilities and back-off weights.

2.1. N-gram Collection

A first bottleneck of the process is the collection and storage of all n-grams into memory. We overcome this issue by splitting and distributing the collection of n-grams statistics over several processes and by making use of efficient data-structure to store n-grams statistics. In order, first the dictionary of the corpus is extracted and split into K word lists, balanced according to word frequency. Then, K independent processes are run, each collecting n-grams beginning with words of a list. The collection of n-grams takes advantage of a dynamic prefix-tree data structure shown in Figure 1. As a difference with previous implementations [9, 3] successors of each 1-gram, 2-gram, ... are stored in memory block allocated on demand. Each block, besides allotting space for a given number of words, uses a specific number of bytes to store word frequencies. In this way, a minimal encoding is used to represent the largest frequency in each block. This strategy permits to cope with the high sparseness of n-grams and with the presence of relatively few highly-frequent n-grams, that require counters encoded with 6 bytes.

2.2. Frequency Smoothing

In the current version, a standard interpolation scheme is considered in combination with two well-established smoothing techniques, namely the Witten-Bell method [10] and a simplified version of the improved Kneser-Ney method [8]. In the former method, smoothing of probabilities from 2-grams is performed separately on each subset of n-grams. For example, frequency smoothing for a 5-gram (v, w, x, y, z) is computed by means of statistics that are local to the subset of n-grams starting with v. Namely, they are the counters $N(v, w, x, y, z)$, $N(v, w, x, y)$, and the number $D(v, w, x, y)$ of different words observed in context (v, w, x, y).

The implementation of the improved Kneser-Ney method differs from the original one because it does not make use of so-called modified frequencies for the lower order n-grams. How-

Accepted after peer review of full paper

list index	dictionary size	number of 5-grams: observed	distinct	non-singletons
0	4	217M	44.9M	16.2M
1	11	164M	65.4M	20.7M
2	8	208M	85.1M	27.0M
3	44	191M	83.0M	26.0M
4	64	143M	56.6M	17.8M
5	137	142M	62.3M	19.1M
6	190	142M	64.0M	19.5M
7	548	142M	66.0M	20.1M
8	783	142M	63.3M	19.2M
9	1.3K	141M	67.4M	20.2M
10	2.5K	141M	69.7M	20.5M
11	6.1K	141M	71.8M	20.8M
12	25.4K	141M	74.5M	20.9M
13	4.51M	141M	77.4M	20.6M
total	4.55M	2.2G	951M	289M

Table 1: Distributed LM training with the English Gigaword corpus. The table reports 5-gram statistics with $K = 14$ splits.

ever, specific discounting constants are applied to n-grams occurring once, twice, thrice and more than thrice in the training data. Such statistics are collected independently in each n-gram list, for each n-gram order, and finally summed up.

Then, K LM files are estimated by just reading through the n-gram files, which are indeed not loaded in memory, and by applying the smoothing statistics of the selected method. During this phase, pruning of infrequent n-grams is also permitted. Finally, all LM files are joined, global 1-gram probabilities are computed and added, and a single large LM file, in the standard ARPA format [3], is generated.

2.3. Compilation

The final textual LM can be compiled into a binary format to be efficiently loaded and accessed at run-time. Our implementation follows the one adopted by the CMU-Cambridge LM Toolkit [11] and well analyzed in [12]. Briefly, n-grams are stored in a data structure which privileges memory saving rather than access time. In particular, single components of each n-gram are searched, via binary search, into blocks of successors stored contiguously (Figure 2). Further improvements in memory savings are obtained by quantizing both back-off weights and probabilities.

2.4. Compression

Quantization provides an effective way of reducing the number of bits needed to store floating point variables. [13] showed that best results with a statistical machine translation decoder were achieved with the so-called *binning method*. This method partitions data points into uniformly populated intervals or bins. Bins are filled in a greedy manner, starting from the lowest value. The center of each bin corresponds to the mean value of all its points. Quantization with 8-bits is applied separately at each n-gram level and distinctly to probabilities or back-off weights.

An insight about why the binning method performs well is that it discriminates well probability values. Indeed what we wish from a quantization method is that it should permit to directly compare (i.e. to test for $<$ or $>$) as many as possible of the different probabilities in our population. Notice that by

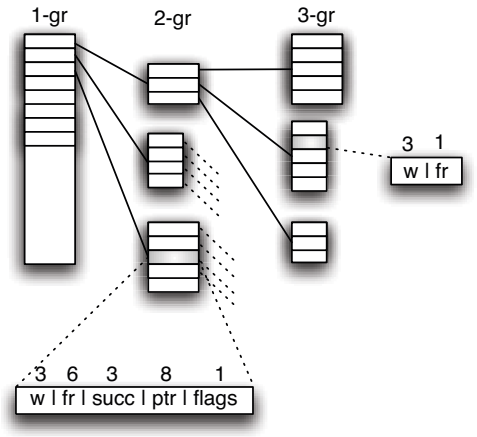

Figure 1: Dynamic data structure for storing n-grams. Blocks of successors are allocated on demand and might vary in the number of entries (depth) and bytes used to store counters (width). Size in bytes is shown to encode words (w), frequencies (fr), and number of (succ), pointer to (ptr) and table type of (flags) successors.

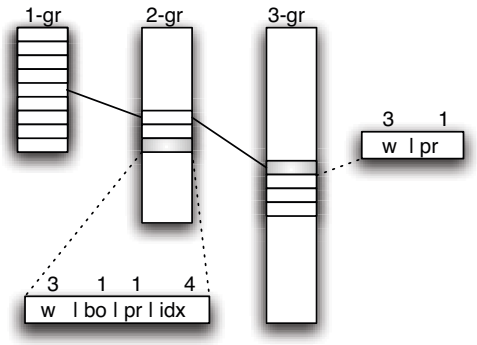

Figure 2: Static data structure for 3-gram LMs. For the bigram and trigram levels we show the number of bytes used to encode single words (w), quantized back-off weights (bo) and probabilities (pr), and start index of successors (idx).

quantizing we cannot compare probabilities falling in the same cluster/bin, but only members from different clusters.

So let P be the total population size and P_j the population of cluster/bin j ($j = 1, .., J$). The quantity we would like to maximize is:

$$\sum_j P_j(P - P_j) \text{ subject to } \sum_j P_j = P$$

where the number $P_j(P - P_j)$ says that each element of the cluster/bin j can be discriminated from the rest of the population. It easily follows that the optimal solution is $P_j = P/J$ for all j, which is exactly the criterion applied by the binning algorithm.

The quantization algorithm can be applied to any LM represented with the ARPA format. Quantized LMs can also be converted into a binary format that can be efficiently uploaded at decoding time.

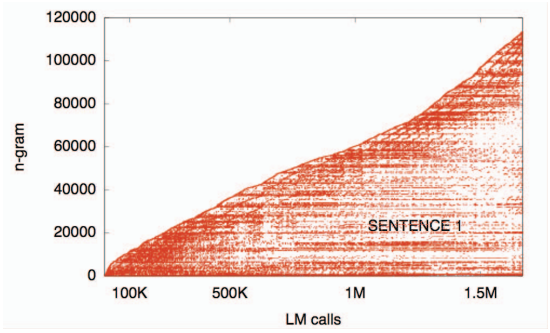

Figure 3: LM calls during machine translation of a text: each point corresponds to a specific 3-gram call.

3. Efficient Access

One motivation of this work is the assumption that efficiency, both in time and space, can be gained by exploiting peculiarities of the way the LM is used by the hosting program, i.e. an ASR or SMT decoder. An analysis of the interaction between an SMT decoder and the LM was carried out, that revealed some important properties. Figure 3 plots all calls to a 3-gram LM by a decoder computing the translation of a text. The plot shows typical locality phenomena, that is the LM n-grams are accessed in nonuniform, highly localized patterns. Locality is mainly temporal, namely the first call of an n-gram is easily followed by other calls of the same n-gram. We expect to observe a very similar behavior also by an ASR system when decoding a single sentence. This property suggests that gains in access speed can be achieved by exploiting a cache memory in which to store already called n-grams. Moreover, the relatively small amount of involved n-grams makes viable the access of the LM from disk on demand. Both techniques are briefly described.

3.1. Caching of probabilities

Caching based on hash tables is used to store all final n-gram probabilities requested by the decoder, LM states used to recombine theories, as well as all partial n-gram statistics computed by accessing the LM structure. In this way, binary searches at every level of the LM tables are reduced at a minimum. All cache memories can be reset before decoding each single input set.

3.2. Memory Mapping

The data structure shown in Figure 2 permits indeed to efficiently exploit the so-called *memory mapped* file access.[2] Memory mapping basically permits to include a file in the address space of a process, whose access is managed as virtual memory (see Figure 4).

During decoding of a single sentence, only those n-grams, or better memory pages, of the LM that are actually accessed are loaded into memory. Once decoding of the input sentence is completed, all loaded pages are released, so that resident memory is available for the n-grams of the following sentence. A remarkable feature is that memory-mapping also permits to share the same address space among multiple processes, so that the same LM can be accessed by several decoding processes running on the same machine.

[2]POSIX-compliant operating systems and Windows support some form of memory-mapped file access.

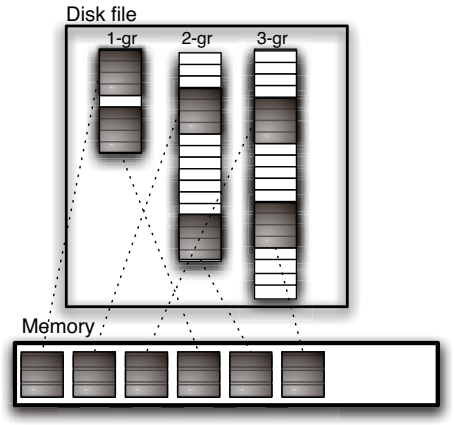

Figure 4: Memory mapping of the LM on disk. Only the memory pages (grey blocks) of the LM that are accessed while decoding the input sentence are loaded in memory.

4. Experiments

Speech recognition experiments were carried out in the context of a large vocabulary Italian speech transcription task. In the following, a brief overview of the experimental set-up is provided; details can be found in the companion paper [7].

4.1. System Architecture

The system architecture consists of two modules: a HMM-based speech recognizer, which processes the speech and generates a confusion network at the level of phones, and a SMT system which, processing the confusion network, outputs the final word transcription.

4.2. Data

The `news` corpus consists of 606M words from Italian newspapers and transcriptions of Italian Broadcast news. The phrase translation tables of the SMT system were built from the `news` corpus, where the phone transcriptions of utterances were paired with the corresponding word sequences. Different LMs were estimated for the SMT system from the `news` corpus and from the `itwac` 1.6G word text corpus [1].

Tests were performed on the SITACAD corpus, a speech database collected from the Italian Parliament. Six parliament sessions (about 5 hours and half, 43K words) were manually transcribed and equally split in one test and one development set. It is worth to noticing that the domain of training data is quite different from that of development and test texts. As a reference for results presented in the following, the best WERs we obtained with a fully HMM based ASR system on those development and test sets were 10.14% and 12.12%, respectively.

4.3. Set-up

On the whole `news` corpus, 3-gram LMs were estimated with the Witten-Bell and the simplified improved Kneser-Ney smoothing methods presented in Section 2.2. For comparison purposes, a 3-gram LM smoothed through the original Kneser-Ney method was also estimated. Moreover, two 4-gram LMs were estimated on the `news` and the `itwac` corpora with the Witten-Bell smoothing technique for a total of more than 1.1G n-grams (363+740 millions). Pruning of rare n-grams was not

applied by purpose, in order to stress the gains in memory efficiency achieved by our implementation.

4.4. Results

Table 2 collects the most interesting figures recorded in phone-decoding experiments with different LMs on the development set. Experiments were carried out on dual Intel/Xeon 64bit 3.2GHz CPUs equipped with 8Gb of RAM. The "kit" column refers to the LM toolkit employed by Moses during decoding. The "process size" column refers to the virtual memory assigned to processes. RTRs measure the real-time ratio of the SMT decoding with respect to the duration of the input speech; the ASR decoding is not considered since common to all our experiments (by the way, its RTR is about 1.5).

The table includes three blocks of three entries each. Blocks refer to different smoothing methods: Witten Bell (`wb`), original (`kn`) and simplified (`skn`) improved Kneser-Ney, respectively. The first two entries of each block differ for the library (SRILM vs. IRSTLM) used in runtime for loading/accessing the same LM. The third entry regards the compressed version (`q`, Section 2.4) of the LM under investigation in the block.

First of all, it can be noted that the IRSTLM toolkit allows significant savings of memory space both for file storing and run-time usage, through the binarization (Section 2.3) of the LM. Moreover, the compression of the LM results in a 40% reduction of disk space, a 33% reduction of process size and a small speed-up in decoding, at no cost at all in performance.

Secondly, on this task no significant WER difference is observed by changing the smoothing method of the LM.

The last row of the table shows that thanks to memory mapping (Section 3.2), the decoder can run with a very large LM without affecting decoding time. Accessing the LM in this way avoids indeed the overhead caused by memory swapping operations that would occur if the full LM was loaded in memory.

LM(s)	kit	file size (Gb)	proc. size (Gb)	RTR	WER %
wb-3gr-news	sri	7.6	6.7	.65	14.91
wb-3gr-news	irst	2.0	2.4	.76	14.96
wb-3gr-q-news	irst	1.2	1.6	.75	14.94
kn-3gr-news	sri	7.6	6.7	.68	14.76
kn-3gr-news	irst	2.0	2.4	.79	14.73
kn-3gr-q-news	irst	1.2	1.6	.79	14.77
skn-3gr-news	sri	7.6	6.7	.65	14.89
skn-3gr-news	irst	2.0	2.4	.76	14.95
skn-3gr-q-news	irst	1.2	1.6	.76	15.00
wb-4gr-q-(nws+itwc)	irst	3.6+7.3	11.3	.82	12.78

Table 2: Run-time figures on dev set for different LMs.

5. Conclusions

We have presented the IRSTLM toolkit. It includes a method for efficiently estimating and handling large scale n-gram LMs for the sake of statistical machine translation and automatic speech recognition. LM estimation is performed by splitting the task with respect to the initial word of each n-gram, and by merging the resulting sub-LMs. Language models are stored in a conventional ARPA format, can be compressed through probability quantization, and compiled into a compact data structure for quick upload and access. During decoding, lazy-loading of huge LM files is supported through memory mapping, that permits to load into memory only portions of the LM that are needed to decode a single sentence.

6. References

[1] M. Baroni and M. Ueyama, "Building general and special-purpose corpora by web crawling," in *Proc. of the NIJL International Symposium, Language Corpora: Their Compilation and Application*, 2006, pp. 31–40. [Online]. Available: http://clic.cimec.unitn.it/marco/publications/bu_wac_kokken_formatted.pdf

[2] M. Lapata and F. Keller, "Web-based models for natural language processing," *ACM Transactions on Speech and Language Processing*, vol. 1, no. 2, pp. 1–31, 2006.

[3] A. Stolcke, "Srilm - an extensible language modeling toolkit," in *Proc. of ICSLP*, Denver, Colorado, 2002. [Online]. Available: http://www.speech.sri.com/cgi-bin/run-distill?papers/icslp2002-srilm.ps.gz

[4] T. Brants, A. C. Popat, P. Xu, F. J. Och, and J. Dean, "Large language models in machine translation," in *Proc. of the Joint Conference EMNLP/CoNLL*, Prague, Czech Republic, 2007, pp. 858–867. [Online]. Available: http://acl.ldc.upenn.edu/D/D07/D07-1090.pdf

[5] M. Federico and M. Cettolo, "Efficient Handling of N-gram Language Models for Statistical Machine Translation," in *Proc. of ACL-SMT workshop*, Prague, Czech Republic, 2007.

[6] P. Koehn, H. Hoang, A. Birch, C. Callison-Burch, M. Federico, N. Bertoldi, B. Cowan, W. Shen, C. Moran, R. Zens, C. Dyer, O. Bojar, A. Constantin, and E. Herbst, "Moses: Open source toolkit for statistical machine translation," in *Proc. of Demo and Poster Sessions of the ACL Meeting*, Prague, Czech Republic, 2007, pp. 177–180. [Online]. Available: http://aclweb.org/anthology-new/P/P07/P07-2045.pdf

[7] N. Bertoldi, M. Federico, D. Falavigna, and M. Gerosa, "Fast speech decoding through phone confusion networks," in *Proc. of Interspeech*, Brisbane, Australia, 2008.

[8] S. F. Chen and J. Goodman, "An empirical study of smoothing techniques for language modeling," *Computer Speech and Language*, vol. 4, no. 13, pp. 359–393, 1999.

[9] F. Wessel, S. Ortsmann, and H. Ney, "Implementation of word based statistical language models," in *Proc. of SQEL Workshop on Multilingual Information Retrieval Dialogs*, 1997, pp. 55–59.

[10] I. H. Witten and T. C. Bell, "The zero-frequency problem: Estimating the probabilities of novel events in adaptive text compression," *IEEE Trans. Inform. Theory*, vol. IT-37, no. 4, pp. 1085–1094, 1991.

[11] P. Clarkson and R. Rosenfeld, "Statistical language modeling using the CMU-Cambridge toolkit," in *Proc. of Eurospeech*, Rhodes, Greece, 1997, pp. 2707–2710.

[12] E. Whittaker and B. Raj, "Quantization-based language model compression," in *Proc. of Eurospeech*, Aalborg, Denmark, 2001, pp. 33–36.

[13] M. Federico and N. Bertoldi, "How many bits are needed to store probabilities for phrase-based translation?" in *Proc. of the Workshop on SMT*. New York City: ACL, June 2006, pp. 94–101. [Online]. Available: http://www.aclweb.org/anthology/W/W06/W06-3113