SRILM at Sixteen: Update and Outlook

Andreas Stolcke[1] Jing Zheng[2] Wen Wang[2] Victor Abrash[2]

[1] *Microsoft Speech Labs,*
Mountain View, California, U.S.A.
anstolck@microsoft.com

[2] *Speech Technology and Research Laboratory, SRI International*
Menlo Park, California, U.S.A.
{zj,wwang,victor}@speech.sri.com

***Abstract*—We review developments in the SRI Language Modeling Toolkit (SRILM) since 2002, when a previous paper on SRILM was published. These developments include measures to make training from large data sets more efficient, to implement additional language modeling techniques (such as for adaptation and smoothing), and for client/server operation. In addition, the functionality for lattice processing has been greatly expanded. We also highlight several external contributions and notable applications of the toolkit, and assess SRILM's impact on the research community.**

I. Introduction

The SRI Language Modeling Toolkit (SRILM for short) is an open source software toolkit for statistical language modeling and related tasks. It was first conceived and implemented—with minimal functionality—in 1995, followed by a first public (beta) release in 1999. Since then SRILM has found wide use in the speech and natural language research community.

A 2002 paper [1] presented an overview of the toolkit's design and functionality, of which we provide only a brief summary here. This paper takes stock of SRILM developments since then, as well as extensions and applications of SRILM. We also summarize activities in the SRILM user community, give an overall assessment of developments so far, and point out possible future directions.

II. SRILM Overview

SRILM is a toolkit for building and evaluating statistical language models (LMs). Most of the LM types it supports are based on N-gram statistics, including the standard back-off models, with an array of standard smoothing algorithms (Good-Turing, Witten-Bell, Kneser-Ney, etc.). Models based on word classes (automatically induced or externally defined) are also supported. SRILM implements various methods for interpolating and adapting LMs, as well as pruning for trading off size against performance. Once an LM is trained, it can be evaluated or used in a variety of standard tasks, such as perplexity computation, N-best and lattice rescoring, and text tagging and segmentation.

SRILM consists of three software layers. The core functionality is implemented in C++, comprising classes for containers (arrays, associative maps, tries, N-gram counts), application-related data structures (N-best lists, lattices, confusion networks, text statistics), smoothing methods, and the LMs themselves. The latter are arranged in a class hierarchy, reflecting the fact that many LM types are variants of more basic types, for which most of the implementation can be shared. The second layer, and the one most relevant to most users, is a set of executable tools that carry out the standard tasks of LM building and application, as well as manipulation of lattices, N-best lists, and confusion networks. A third layer comprises tools implemented as scripts in the Bourne shell, Gawk, and Perl languages for miscellaneous tasks that are carried out primarily through simple manipulation of text files or on top of the second-layer tools. The tools in the second and third layers are designed to combine via Unix pipes to carry out more complex tasks.

III. New developments

Since 2002, SRILM has seen a fairly regular stream of incremental releases (about 2 to 3 per year), each of which typically contributed some new functionality, as well as bug fixes and portability improvements. New functionality can be grouped into the areas of efficient and handling of large data sets, language modeling algorithms, and lattice and N-best processing.

A. Efficiency and large LM training

The past ten years have seen an explosion in the availability of public training data for language modeling purposes. This was partly due to government funding research programs with the aim to improve the performance for speech recognition and machine translation systems. Second, ever-growing audio archives of broadcast programs are now available for web access and transcription costs have been lowered through outsourcing to commercial services. Finally, textual data is now available in essentially unlimited amounts through the exponential growth of the world-wide web.

SRILM data structures for N-gram models were not originally engineered with memory footprint in mind, focusing instead on programming convenience and speed. Since memory requirements grow roughly linearly with the number of N-grams in training, this can lead to problems when processing the ever-increasing amounts of training data. Other LM toolkits have since been published that are engineered specifically for

memory compactness and avoiding the limitations of random-access memory by utilizing disk-based data structures [2], [3].

While memory sizes in typical research machines certainly have not grown on the same scale as the increase in LM training data, they have grown somewhat (in our case, from around 500MB when SRILM was first developed to tens of gigabytes today). This, in combination with several incremental measures described below, has proven sufficient so far to keep up with the data demands at least in a research environments.

The techniques that help accommodate large data and models can be summarized as follows:

1) **Count-of-count (metacount) statistics:** When count cutoffs on N-grams are used, as they usually are in training large LMs, it is not necessary to load all N-gram counts into memory. Rather, the count-of-count statistics required for discounting algorithms can be computed off-line (in constant memory), and for N-grams that fall below the frequency threshold only these count-of-counts (how many distinct N-grams of a certain frequency in a given context occurred) need to be passed to the LM estimation algorithm. This method is encapsulated in the SRILM `make-big-lm` script. (Note that the gathering of N-gram counts per se is not limited by memory when using recursive count merging as in `merge-batch-counts`).
2) **Vocabulary subsetting:** At test time, the data can usually be split into manageable chunks, each of which utilizes only a small subset of the vocabulary. All LM types now support a loading method triggered by `ngram -limit-vocab`, whereby only those LM parameters are loaded that intersect with a given vocabulary subset.
3) **Binary file formats:** N-gram counts and backoff LMs now support binary formats that are inherently faster to read and, more importantly, support very fast loading with `-limit-vocab`, efficiently skipping over file portions that are outside a given vocabulary. While the format is binary, it is still portable between different byte-orders and machine word sizes.
4) **Indirect integer storage:** Large counts require 64 bits for encoding, yet this would entail wasted space for most N-grams, which are still infrequent. We therefore implemented an integer data type that stores small count values inline using 2- or 4- byte words, while referencing a table for large values that do not fit in 15 or 31 bits, respectively.
5) **Optimized heap allocation:** We lowered the overall overhead in heap memory allocation (for bookkeeping) by providing a special-purpose allocator that keeps lists of small memory chunks, as are typically found in N-gram trie data structures. (This, too, exploits the fact that N-gram frequency distributions have a long tail of infrequent N-gram types, leading to a large number of trie nodes with small number of children.) For a 6-gram LM with 690M N-grams, this reduced memory usage by 30% (from 33GB to 23GB).
6) **Destructive LM merging:** Since large LMs are typically constructed as a static interpolation of component LMs by N-gram merging, we made N-gram LM merging destructive, so that the total memory demand is only the sum of the result LM and the smaller of the two inputs. This way, very large LMs can be built by successively merging the component models.

B. Language modeling algorithms

A major algorithmic addition in SRILM is support for **factored language models** (FLMs), proposed by Bilmes and Kirchhoff [4], who implemented FLMs within the SRILM framework. FLMs represent words as vectors of discrete features, which allows the prediction of words to be factored in a number of ways. This treatment is particularly appropriate for modeling languages with complex morphology, such as Arabic [5].

The array of N-gram **smoothing algorithms** has been rounded out by adding the original (unmodified) Kneser-Ney (KN) method [6] and simple additive smoothing. The latter is useful for instructive purposes, and can give better results when the goal is not to minimize model perplexity (e.g., when using N-gram models for classification tasks).

It was recently pointed out [7] that KN-smoothed backoff models perform poorly when pruned heavily (e.g., for use in speech decoding). This problem results in part because KN-smoothed models by their nature are not a good model for the joint probabilities of lower-order N-grams. As a partial remedy, we added an option to specify a second LM (presumably smoothed not using KN) to the pruning algorithm for use in computing lower-order N-gram marginals.

SRILM now also supports **deleted interpolation** (also known as Jelinek-Mercer smoothing) [8] as an alternative classical smoothing approach. Although it is no longer state of the art, this method was included because it confers a practical advantage when very large training corpora are used. In our implementation, known as a **count-based LM**, the model is stored only as a set of interpolation weights, along with a pointer to the N-gram count statistics. Together with the ability to load only those counts that are relevant to a given test set vocabulary, this enables the use of essentially unlimited N-gram statistics.

Count-based LMs were specifically motivated by the recent publication of **large N-gram corpora** by Google and Yahoo. To leverage these data sources, a method for reading N-gram counts in the Google format was added (including efficient reading for sub-vocabularies), as well as several other functions that enable effective use of the data (e.g., in machine translation systems [9]). A vocabulary mapping mechanism was incorporated to allow the given count statistics to be used even though the target task might require different text normalization, such as when Web data is used for speech recognition LMs. Also, because the published Web data statistics were subjected to count cutoffs, we devised a way to extrapolate the missing lower count-of-count statistics that are required for KN and Good-Turing discounting [9].

Two additional methods from the literature for N-gram model adaptation and combination have been implemented. **Unigram marginal adaptation** [10] combines an existing N-gram LM with unigram statistics from an adaptation set by scaling the N-gram conditional probabilities to conform to the unigram distribution. **Log-linear interpolation** of LMs [11] is similar to standard linear interpolation except that probabilities are combined multiplicatively rather than additively (i.e., the combination is additive in the log probability space). Both these methods require renormalization of the modified N-gram probabilities, which makes them relatively more expensive to compute than standard linear interpolation. This drawback is mitigated by caching the normalization terms once computed, given that the same word histories tend to reoccur, especially in N-best and lattice rescoring.

A simple mechanism for **N-gram-based approximation** of any of the implemented non-standard LMs has been added. The user supplies a standard N-gram LM to `ngram -rescore-ngram` and all N-gram conditional probabilities are recomputed according to the LM specified by the remaining options. For example, a log-linearly interpolated model could be precomputed for all N-grams of interest and stored as a single standard LM. The quality of the approximation is determined by the set of N-grams included in the "rescored" model.

SRILM now includes a simple **client-server implementation** that allows connecting LM computation and applications over a TCP/IP network connection. The `ngram` tool when started in server mode listens for connections on a specified port, reads requests for N-gram probabilities, and writes back results according to the LM options specified. On the client side, the `ngram` tool is invoked with an option specifying the host and port number on which the server is found. The client has an option for caching LM results for fast reuse to reduce latency.

The SRILM network protocol corresponds to a minimal subset of the SRILM API and also includes requests for determining the history length used in an N-gram estimate, and the corresponding backoff weight (enough information to allow efficient lattice rescoring). The client-server approach has obvious uses in network-based LM applications. For example, evaluation of large LMs may be delegated to a machine with large memory, thereby reducing the footprint of LM applications and avoiding the latency incurred by reading the LM at startup.

Before an LM can be estimated one has to perform **vocabulary selection**. SRILM now includes the tool `select-vocab` that ranks vocabulary words from a union of training corpora, based on their corpus frequencies as well as the relevance of each corpus relative to a held-out test set [12]. Conveniently, training and held-out statistics may be given by text files, N-gram count files, or LMs.

C. Lattices and N-best processing

Language models are often applied to lattices and N-best lists of hypotheses, and SRILM from early on had facilities for handling these data structures. A major enhancement since 2002 has been support for lattices in **HTK Standard Lattice Format**. This allows separate encoding of different kinds of scores (acoustic, pronunciation, language model) from the recognizer, and efficient replacing and reweighting of scores when new models are applied to recognition output. Confusion networks are also accepted as an input lattice format, allowing further rescoring after confusion network construction.

The lattice expansion algorithm has been generalized to work for LMs with arbitrary finite history length, either in exact form or taking advantage of the LM's backoff structure [13]. It is also now possible to rescore lattices with higher-order LMs without explicitly expanding them, saving time and memory.

Based on HTK lattices, several generally useful algorithms have been implemented that for the most part are independent of language models. N-best lists can be generated from HTK lattices, using either an A-star (stack) or Viterbi-style decoding algorithm.

Based on the weighted input scores on lattice hypotheses, posterior probabilities can be computed and used for pruning lattices. Posterior probabilities may also be used in computing the weighted counts of all N-grams found in a lattice, which in turn is the basis of many applications, such as speaker and language recognition based on phone lattices. To enable word spotting applications one can generate an index of all word lattice N-grams with their time offsets and posterior probabilities.

A major addition is the generation of confusion networks (CNs) directly from lattices (previously confusion networks could only be generated from N-best lists). While inspired by the original CN algorithm of [14], we decided to implement a new algorithm that is similar to one proposed by [15] and does not rely on word time marks or pronunciations to partition word hypotheses into confusion sets. The algorithm starts with the highest-probability path through the lattice, and then successively aligns those partial lattice paths to the CN that lead to and from remaining nodes, until all word hypotheses are accounted for. The resulting algorithm has lower complexity than the one in [14], requiring time that is only linear in the product of the number of lattice nodes and the length of the final CN.

Finally, the `nbest-optimize` tool for score weight optimization has been generalized to also allow BLEU score optimization, as required for machine translation.

D. Portability

Given the diversity of the user community, we have tried to keep the code as portable as possible. Originally developed on Unix/Linux platforms, it now builds and runs on MacOS and Windows systems as well. Windows builds can use either the Cygwin porting layer or the native Visual C++ environment. Cygwin supports some functionality that is otherwise not available (compressed file I/O through pipes, the client/server implementation based on Unix sockets), but has the limitation of 32bit word size. A test suite that covers all the major

functions ensures that the expected results are obtained on all platforms (up to small discrepancies due to differences in machine arithmetic).

SRILM has also been ported to ARM-processor-based platforms, and is embedded both in SRI's DynaSpeak® small-footprint speech recognition engine [16] and in the SRInterp cross-platform statistical machine translation engine, which supports speech-to-speech translation applications on Android smartphones [17].

IV. Extensions and Applications

SRILM from the start was designed to be an extensible platform that others could use to build new algorithms while effectively leveraging the facilities already implemented. It is therefore very satisfying to see that several researchers have based their implementations on SRILM and released extensions to it as separate packages. Among these are "semantic LMs" based on latent semantic analysis [18], random forest LMs [19], LM smoothing using the hierarchical Pitman-Yor process and power law discounting [20], and maximum entropy LMs [21].

Equally satisfying, SRILM has found widespread use as a ready-made toolbox that can be used to implement state-of-the-art LMs in a wide range of applications. Various speech recognition systems, such as the open-source system developed at RWTH Aachen [22], make use of SRILM for their language models. Open-source machine translation systems such as Moses [23] and Joshua [24] also use SRILM, as does the commercial SYSTRAN engine [25]. Other natural language applications include text-to-speech alignment [26], tagging and word segmentation for Semitic languages [27], Chinese word segmentation [28], and handwriting recognition [29]. SRILM has also found applications in bio-informatics [30].

Finally, we are happy to see that SRILM has found wide use in academic teaching of natural language processing and speech recognition subjects. Courses at Carnegie Mellon, the Center for Spoken Language Understanding, Mississippi State, Stanford, and the University of Washington (to name only those we are aware of) have incorporated SRILM into their course materials and hands-on assignments.

V. User Community

As of this writing, SRILM has been downloaded about 40,000 times from http://www.speech.sri.com/projects/srilm/, corresponding to 15,299 unique email addresses. Downloaders by default join a mailing list to receive future release announcements, from which they can opt out, however, or remove themselves later (the mailing list software also automatically removes invalid email addresses). This `srilm-announce` list at present contains 5841 email addresses. A second mailing list requires explicit signing up, and is meant for users who take a more active interest in the toolkit and sometimes answer questions about its use. This `srilm-user` mailing list currently has 366 members, and results in a searchable archive of questions and answers about SRILM usage and possible future extensions. To gauge the extent to which our toolkit has contributed to research we can query scholar.google.com, which reports that the 2002 SRILM paper [1] was cited in 1699 research publications.

While the great majority of SRILM users work on not-for-profit or government-sponsored research (the legitimate fields of use when the software is obtained for free), the toolkit has also been licensed for commercial use.[1]

VI. Summary and Outlook

SRILM has proven itself as a popular toolkit for the purposes it was designed to serve: as a ready-made set of tools for state-of-the-art language modeling in a wide range of application domains (speech recognition, text-based natural language processing, machine translation, and bio-informatics), as well as a platform for LM research that continues to be extended with new functionality.

The toolkit's basic software design has aged remarkably well given the many extensions in functionality over the past sixteen years. This is not to say that SRILM could not use some reengineering, such as an overhaul of datastructures to accommodate very large datasets, including disk-based container abstractions.

Several advances in LM research in recent years seem to show consistent gains and wider adoption, such as class-based exponential models [31] and neural network-based LMs [32]. Clearly, adding some of these new approaches to the toolkit would serve its users well, and we hope that contributions from the community will continue to be forthcoming.

Acknowledgments

We thank all those who helped improve SRILM with their feedback. Special thanks go to those who contributed code, documentation, or portability improvements: Jeff Bilmes, Elad Dinur, Kevin Duh, Mike Frandsen, Dustin Hillard, Anand Venkataraman, and Deniz Yuret (with apologies for inadvertent omissions).

Work described here was done while the first author was with SRI International. The development of SRILM has been supported in part by the Defense Advanced Research Projects Agency (DARPA) under Contracts MDA972-02-C-0038 and HR0011-06-C-0023. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of DARPA.

References

[1] A. Stolcke, "SRILM—an extensible language modeling toolkit", in J. H. L. Hansen and B. Pellom, editors, *Proc. ICSLP*, vol. 2, pp. 901–904, Denver, Sep. 2002.

[2] P. Nguyen, J. Gao, and M. Mahajan, "MSRLM: a scalable language modeling toolkit", Technical Report MSR-TR-2007-144, Microsoft Research, Nov. 2007, https://research.microsoft.com/pubs/70505/tr-2007-144.pdf.

[3] M. Federico, N. Bertoldi, and M. Cettolo, "IRSTLM: An open source toolkit for handling large scale language models", *in Proc. Interspeech*, pp. 1618–1621, Brisbane, Australia, Sep. 2008.

[1] Inquiries about commercial use should be directed to http://www.speechatsri.com/utils/contact.php .

[4] J. A. Bilmes and K. Kirchhoff, "Factored language models and generalized parallel backoff", in M. Hearst and M. Ostendorf, editors, *Proc. HLT-NAACL*, pp. 4–6, Edmonton, Alberta, Canada, Mar. 2003. Association for Computational Linguistics.

[5] K. Kirchhoff, D. Vergyri, J. Bilmes, K. Duh, and A. Stolcke, "Morphology-based language modeling for conversational Arabic speech recognition", *Comp. Speech Lang.*, vol. 20, pp. 589–608, Oct. 2006.

[6] R. Kneser and H. Ney, "Improved backing-off for M-gram language modeling", *in Proc. ICASSP*, vol. 1, pp. 181–184, Detroit, May 1995.

[7] C. Chelba, T. Brants, W. Neveitt, and P. Xu, "Study on interaction between entropy pruning and kneser-ney smoothing", *in Proc. Interspeech*, pp. 2422–2425, Makuhari, Japan, Sep. 2010.

[8] F. Jelinek and R. L. Mercer, "Interpolated estimation of Markov source parameters from sparse data", *in Proc. Workshop on Pattern Recognition in Practice*, Amsterdam, May 1980. North-Holland.

[9] W. Wang, A. Stolcke, and J. Zheng, "Reranking machine translation hypotheses with structured and web-based language models", *in Proceedings IEEE Workshop on Speech Recognition and Understanding*, pp. 159–164, Kyoto, Dec. 2007.

[10] R. Kneser, J. Peters, and D. Klakow, "Language model adaptation using dynamic marginals", in G. Kokkinakis, N. Fakotakis, and E. Dermatas, editors, *Proc. EUROSPEECH*, vol. 4, pp. 1971–1974, Rhodes, Greece, Sep. 1997.

[11] D. Klakow, "Log-linear interpolation of language models", in R. H. Mannell and J. Robert-Ribes, editors, *Proc. ICSLP*, vol. 5, pp. 1695–1698, Sydney, Dec. 1998. Australian Speech Science and Technology Association.

[12] A. Venkataraman and W. Wang, "Techniques for effective vocabulary selection", *in Proc. EUROSPEECH*, pp. 245–248, Geneva, Sep. 2003.

[13] F. Weng, A. Stolcke, and A. Sankar, "Efficient lattice representation and generation", in R. H. Mannell and J. Robert-Ribes, editors, *Proc. ICSLP*, vol. 6, pp. 2531–2534, Sydney, Dec. 1998. Australian Speech Science and Technology Association.

[14] L. Mangu, E. Brill, and A. Stolcke, "Finding consensus in speech recognition: Word error minimization and other applications of confusion networks", *Comp. Speech Lang.*, vol. 14, pp. 373–400, Oct. 2000.

[15] D. Hakkani-Tür, F. Béchet, G. Riccardi, and G. Tür, "Beyond ASR 1-best: Using word confusion networks in spoken language understanding", *Comp. Speech Lang.*, vol. 20, pp. 495–514, Oct. 2006.

[16] H. Franco, J. Zheng, J. Butzberger, F. Cesari, M. Frandsen, J. Arnold, V. R. R. Gadde, A. Stolcke, and V. Abrash, "DynaSpeak: SRI's scalable speech recognizer for embedded and mobile systems", *in Proc. 2nd International Conference on Human Language Technology Research*, pp. 25–30, San Diego, 2002.

[17] J. Zheng, A. Mandal, X. Lei, M. Frandsen, N. F. Ayan, D. Vergyri, W. Wang, M. Akbacak, and K. Precoda, "Implementing SRI's Pashto speech-to-speech translation system on a smartphone", *in Proc. IEEE Spoken Language Technology Workshop*, pp. 121–126, Berkeley, CA, Dec. 2010.

[18] M. Pucher and Y. Huang, "Latent semantic analysis based language models for meetings", in S. Renals and S. Bengio, editors, *Machine Learning for Multimodal Interaction: Second International Workshop, MLMI 2005*, vol. 3869 of *Lecture Notes in Computer Science*. Springer, 2005.

[19] Y. Su, F. Jelinek, and S. Khudanpur, "Large-scale random forest language models for speech recognition", *in Proc. Interspeech*, pp. 598–601, Antwerp, Aug. 2007.

[20] S. Huang and S. Renals, "Hierarchical Pitman-Yor language models for ASR in meetings", *in Proceedings IEEE Workshop on Speech Recognition and Understanding*, pp. 124–129, Kyoto, Dec. 2007.

[21] T. Alumäe and M. Kurimo, "Efficient estimation of maximum entropy language models with N-gram features: An SRILM extension", *in Proc. Interspeech*, pp. 1820–1823, Makuhari, Japan, Sep. 2010.

[22] D. Rybach, C. Gollan, G. Heigold, B. Hoffmeister, J. Lööf, R. Schlüter, and H. Ney, "The RWTH Aachen University open source speech recognition system", *in Proc. Interspeech*, pp. 2111–2114, Brighton, U.K., Sep. 2009.

[23] 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. 45th Annual Meeting of the ACL, Interactive Poster and Demonstration Sessions*, pp. 177–180, Prague, 2007.

[24] Z. Li, C. Callison-Burch, C. Dyer, J. Ganitkevitch, S. Khudanpur, L. Schwartz, W. N. G. Thornton, J. Weese, and O. F. Zaidan, "Joshua: An open source toolkit for parsing-based machine translation", *in Proc. 4th Workshop on Statistical Machine Translation*, pp. 135–139, Athens, Mar. 2009.

[25] "SRI International and SYSTRAN announce licensing agreement for SRI's language modeling toolkit", http://www.systransoft.com/systran/news-and-events/press-release/sri-international-and-systran-announce-licensing-agreement, Sep. 2009.

[26] A. Katsamanis, M. P. Black, P. G. Georgiou, L. Goldstein, and S. Narayanan, "Sailalign: Robust long speech-text alignment", *in Proc. Workshop on New Tools and Methods for Very Large Scale Research in Phonetic Sciences*, Philadelphia, Jan. 2011.

[27] R. Bar-Haim, K. Simaan, and Y. Winter, "Choosing an optimal architecture for segmentation and POS-tagging of Modern Hebrew", *in Proc. ACL Workshop on Computational Approaches to Semitic Languages*, 2005.

[28] "bigram-apps", http://ostatic.com/bigram-apps, 2007.

[29] S. Quiniou, E. Anquetil, and S. Carbonnel, "Statistical language models for on-line handwritten sentence recognition", *in Proc. 8th International Conference on Document Analysis and Recognition*, pp. 516–520, Seoul, Korea, 2005.

[30] P. A. Reche, J. P. Glutting, and E. L. Reinherz, "Rankpep: Prediction of mhc-restricted ligands", http://immunax.dfci.harvard.edu/Tools/rankpep_help.html, 2002.

[31] S. F. Chen, "Shrinking exponential language models", *in Proc. Annual Conference of the North American Chapter of the ACL*, pp. 468–476, Boulder, CO, 2009.

[32] Y. Bengio, R. Ducharme, P. Vincent, and C. Jauvin, "A neural probabilistic language model", *Journal of Machine Learning Research*, vol. 3, pp. 1137–1155, 2003.