Word-Phrase-Entity Language Models: Getting More Mileage out of N-grams

Michael Levit, Sarangarajan Parthasarathy, Shuangyu Chang, Andreas Stolcke, Benoît Dumoulin

Microsoft Corporation, U.S.A.

{mlevit|sarangp|shchang|anstolck|bedumoul}@microsoft.com

Abstract

We present a modification of the traditional n-gram language modeling approach that departs from the word-level data representation and seeks to re-express the training text in terms of tokens that could be either words, common phrases or instances of one or several classes. Our iterative optimization algorithm considers alternative parses of the corpus in terms of these tokens, re-estimates token n-gram probabilities and also updates within-class distributions. In this paper, we focus on the cold start approach that only assumes availability of the word-level training corpus, as well as a number of generic class definitions. Applied to the calendar scenario in the personal assistant domain, our approach reduces word error rates by more than 13% relative to the word-only n-gram language models. Only a small fraction of these improvements can be ascribed to a larger vocabulary.

Index Terms: class-based LMs, phrase-level LMs

1. Introduction

While n-gram language models continue to prevail in industrial speech recognition, their weaknesses are being increasingly acknowledged and include inability to efficiently generalize from limited training data as well as relatively short memory span. New and more powerful continuous space LMs (such as exponential and NN language models, including RNNs) find their way into mainstream ASR [1, 2]. They are able to generalize from observed histories to unobserved ones by learning word similarities and, in the case of RNNs, also break the context length limitations. In this paper, we postulate that similar goals can be achieved within the n-gram modeling approach by capitalizing on idiosyncratic word patterns in the training data and introducing prior knowledge (human expertise) into the models. A significant body of relevant studies exists.

One example is phrase-based LMs where common and cohesive word sequences are replaced by phrases and used as pseudo-words for n-gram training [3, 4]. Another example is class-based language models with classes being either manually defined or automatically inferred from the data. The manually defined classes usually reflect grammatical categories or various (named) entities while inferred classes (word clusters) typically take advantage of context-similarity to group similar words into classes [5]. Several studies tried combining inferred classes with phrases as well [6, 7].

There are several problems with these approaches: first, phrases are constructed by merging word pairs. This makes phrases of three and more words somewhat tedious to build and evaluate. Second, the merging decisions are based solely on the identity of the words. As a consequence, once the merging decision is made, the affected words will *always* be replaced by phrases, no matter what the context. For instance, *"new york times"* will always be marked as a phrase, even in a sentence: *"mumbai has the population of new york times two"*. The same applies to inferred classes that, in addition, are difficult to modify once formed. On the other hand, manual entity-based classes are often difficult to justify from the probabilistic language modeling perspective. Furthermore, they require an extra tagging step. Taggers, in turn, need to be trained on manually annotated in-domain corpus, which is often not feasible.

In this paper we will focus on combining phrases and classes in a joint modeling approach with a goal of finding a representation of the in-domain data in terms of such phrases and classes that improve language modeling. We are motivated by the desire to preserve simplicity and scalability of the n-gram language models while enhancing their temporal modeling ability via longer phrases (to account for common expressions or idioms) and improving their generalization power via entity-based classes. The Word-Phrase-Entity (WPE) LMs will only instantiate a phrase or an entity in the training corpus if it helps improve likelihood of the data. As an example, consider the bigram *"angelina jolie"*. In the presence of a named entity ACTOR, our algorithm will only cast the bigram as an instance of that NE in the contexts where it makes sense from the likelihood point of view. That is, if the previous history was *"brad pitt and"*, the bigram is unlikely to be replaced by the entity; in fact, in this context, the algorithm might even choose to model *"brad+pitt+and+angelina+jolie"*[1] as a single phrase rather than "ACTOR *and* ACTOR", allowing for longer span trigrams such as *"brad+pitt+and+angelina+jolie wedding photos"* – something we would not be able to get with the word-only representation of the corpus. As a result, the WPE LMs combine the discriminative power of words, generalization power of classes, and context modeling power of common phrases.

We will demonstrate how a few iterations of the EM algorithm starting with just the word-level representation of the training corpus and a number of off-the-shelf NE definitions can produce such representation. Apart from generating WPE language models, we will also show that the generic NE definitions themselves can be adapted to the training corpus further improving the model's fit to the target domain. Using language modeling for identifying NEs has been studied in the literature [8, 9, 10]. Our approach bears some similarity to [9] where weakly supervised generative models are built to locate NEs in texts; however, there are important differences: we go beyond unigrams, we obtain alternative sentence representations from n-best parses and, finally, we use entity probabilities directly in the generative model rather than via regularization.

The rest of the paper is organized as follows. In Section 2 we introduce the iterative approach that takes us from words to tokens but also explains how named entity definitions can be optimized. We will then present our experiments in Section 3

[1] We use notation *"a+b"* to denote a multi-word phrase that is treated as a single pseudo-word unit for language modeling purposes.

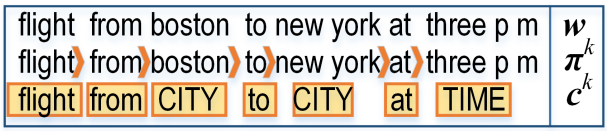

Figure 1: *An example parse* $\boldsymbol{c}^k$ *of sentence* $\boldsymbol{w}$ *induces segmentation* $\boldsymbol{\pi}^k$.

and conclude with an outlook and a summary.

2. From Words to Tokens

Let us start by assuming that our training corpus is already represented as a collection of weighted *token*-level sentences with tokens being either words or phrases, or entities; for example: *"i+would+like+to travel from* CITY *to* CITY*"* with weight 10, where CITY is a predefined NE class. It is straightforward to estimate an n-gram language model from this training corpus by incrementing the count for each n-gram by the weight of the sentence it occurred in. Now imagine that each sentence has its probability mass split among several alternative parses. In the above case, the alternatives could be: *"i+would like+to travel from* CITY *to san+francisco"* with weight 3 or *"i would like to travel+from+boston to* CITY with weight 2. To build a WPE LM, we count token n-grams from all of these alternatives. However, the counters are incremented only by a fraction of the original weight that is equal to the cumulative posterior probability of the parses containing these n-grams. Having built the new LM, we can go back to the original word-level representation of the corpus and parse it with this language model, producing a collection of alternative parses for each of its sentences. After that, the next iteration can begin.

The above approach is essentially a version of the EM algorithm for phrase generation that has been extensively studied in the literature [11, 12]. However, the focus of prior research was put squarely on dealing with multi-word (or multi-character) phrases. In this study, the notion of "token" subsumes multi-word phrases and named entities. The unified terminology is useful for a clear separation between token-level n-gram LMs and intrinsic within-tokens probabilities.

2.1. Re-estimating the Token LM

Consider a sentence of N words $\boldsymbol{w} = w^1 \dots w^N$ and real-valued weight L (that in the beginning could reflect observation counts: $L(\boldsymbol{w}) := \#\boldsymbol{w}$). This sentence can be segmented in K different ways with single- and multi-word tokens competing against each other for words. Let a particular parse $\boldsymbol{c}^k = (c_1^k \dots c_{|\boldsymbol{c}^k|}^k)$, $k = \overline{1,K}$ induce segmentation $\boldsymbol{\pi}^k = (\pi_1^k \dots \pi_{|\boldsymbol{c}^k|}^k)$ that partitions the sentence into a number of word sub-sequences π_i^k, each sub-sequence being accounted by a token instance c_i^k. Figure 1 illustrates this process. Joint likelihood of the sentence and the parse can be obtained via chain rule[2]:

$$P(\boldsymbol{w}, \boldsymbol{c}^k) = \prod_{c_i^k \in \boldsymbol{c}^k} P(c_i^k | h_i^k) P(\pi_i^k | c_i^k). \tag{1}$$

The first term is an n-gram probability of token c_i^k in token n-gram history $h_i^k = c_{i-n+1}^k \dots c_{i-1}^k$. The second one is determined by the nature of the token. For words and phrases, this probability is 1.0, and for a NE class, this is the prior of a particular surface form of this class. Depending on the NE nature, we use one of the following two formalisms to represent it. For (weighted) lists such as personal names or movie titles, we use word tries, and for combinatorial entities such as dates and times, finite state machines (FSMs) are used [13]. Both alternatives allow for efficient decoder implementations.

To obtain new maximum-likelihood (ML) estimates for the n-gram probabilities $P(c|h)$ for token c and its token-level history h from a single sentence $\boldsymbol{w}$, we accumulate parse-specific observation counts:

$$\begin{aligned} P^{\text{ML}}(c|h,\boldsymbol{w}) &= \sum_k \frac{\#ch}{\#h}\bigg|_{\boldsymbol{c}^k} P(\boldsymbol{c}^k|\boldsymbol{w}) \\ &= \sum_k \frac{\#ch}{\#h}\bigg|_{\boldsymbol{c}^k} \frac{P(\boldsymbol{w},\boldsymbol{c}^k)}{\sum_{\boldsymbol{c}} P(\boldsymbol{w},\boldsymbol{c})}, \end{aligned} \tag{2}$$

with notation $\cdot|_{\boldsymbol{c}^k}$ used to signify counting on a single parse $\boldsymbol{c}^k$. Taking averages over an entire corpus $\{\boldsymbol{w}_j\}$ closes the EM loop:

$$\begin{aligned} P^{\text{ML}}(c|h) &= \sum_j P(\boldsymbol{w}_j) P^{\text{ML}}(c|h,\boldsymbol{w}_j) \\ &= \sum_{\boldsymbol{w}} \frac{L(\boldsymbol{w})}{\sum_{\bar{\boldsymbol{w}}} L(\bar{\boldsymbol{w}})} P^{\text{ML}}(c|h,\boldsymbol{w}) \\ &= \sum_{\boldsymbol{w}} \sum_k L'(\boldsymbol{w},k) \frac{\#ch}{\#h}\bigg|_{\boldsymbol{c}^k}, \end{aligned} \tag{3}$$

where the weights of each parse are set to:

$$L'(\boldsymbol{w},k) := \frac{L(\boldsymbol{w})}{\sum_{\bar{\boldsymbol{w}}} L(\bar{\boldsymbol{w}})} * \frac{P(\boldsymbol{w},\boldsymbol{c}^k)}{\sum_{\boldsymbol{c}} P(\boldsymbol{w},\boldsymbol{c})} \tag{4}$$

The maximization step (3) can be naturally extended to incorporate smoothing techniques such as discounting. In fact, this process is equivalent to building an n-gram LM from a new representation of the training corpus that now consists of a weighted collection of token-level parses of the original sentences. The entire cycle of Eqs. (1–4) can be repeated until convergence is achieved.

2.2. Optimizing Named Entities

The re-estimation formulae from the previous section will update token-level n-gram LM, but we can also tune up named entities in the same maximum-likelihood fashion. For NEs modeled as word tries, counting is simple. If π is a particular surface form of NE-token c, then its probability is computed as:

$$\begin{aligned} P^{\text{ML}}(\pi|c) &= \sum_j P(\boldsymbol{w}_j) P^{\text{ML}}(\pi|c,\boldsymbol{w}_j) \\ &= \sum_{\boldsymbol{w}} \sum_k L'(\boldsymbol{w},k) \frac{\#(c,\pi)}{\#c}\bigg|_{\boldsymbol{c}^k} \end{aligned} \tag{5}$$

For combinatorial entities such as times and dates, the re-estimation is similar, except it is carried out for each emanating arc (and final costs) of each state in a normalized deterministic FSM. Because training corpora are limited in size, the ML estimator from Eq. (5) suffers from over-training. On the other hand, the generic NE definitions might be robust, but lack domain specificity. Therefore, a trade-off needs to be established

[2] For the sake of discussion, we can assume a unique segmentation corresponding to each parse.

between the two. In our experiments, on each iteration t we achieve this by regularizing posterior $P^{\text{ML}}_{(t)}$ with prior $P^{\text{prior}}_{(t)}$ via linear interpolation:

$$P_{(t)}(\pi|c) := (1-\lambda_{(t)})P^{\text{ML}}_{(t)}(\pi|c) + \lambda_{(t)}P^{\text{prior}}_{(t)}(\pi|c) \quad (6)$$

with iteration-specific inertia $\lambda_{(t)}$ defined as:

$$\lambda_{(t)} := \begin{cases} 1.0, & \text{if } t < \kappa \text{ or } Z_{(t)}(c) < \theta_1 \\ \lambda^{0.5^{(t-\kappa)}}, & \text{otherwise.} \end{cases} \quad (7)$$

For the first κ iterations, the prior is determined by the initial LM definition; after that: $P^{\text{prior}}_{(t)}(\pi|c) := P_{(t-1)}(\pi|c)$. Parameters θ_1, κ and λ are set empirically and $Z_{(t)}(c)$ is the normalization factor

$$Z(c) = \sum_{\boldsymbol{w}} \sum_{k} L'(\boldsymbol{w}, k)\#c|_{\boldsymbol{c}^k} \quad (8)$$

from iteration t. $Z_{(t)}(c)$ can be seen as the observation count of c computed at this iteration.

2.3. Initialization

A sensible initialization is crucial for successful optimization. In our case, we need to:

- suggest a set of candidate multi-word phrases
- build the initial token-level LM.

We start by extracting all word sub-sequences π of length up to L from the training corpus. For each π, we query all NE-tokens c for prior $P^{\text{prior}}(\pi|c)$ and, if it is positive, increment the count of c by $\#\boldsymbol{w} * P^{\text{prior}}(\pi|c)$. Corpus-wide counts of π are accumulated as well. If they exceed the required minimum threshold, π is designated as a phrase token. In addition, all regular words and classes, no matter what their occurrence count, become tokens as well. After that, a unigram LM is trained from these statistics as the initial token-level LM. All subsequent iterations build LMs of higher order.

We are aware of the distorted probability space that ensues from substring counting, where the same word span can give rise to several tokens without sharing probability mass among them. At this point, we do not have a proper justification for the cold start like that, except that it turns out to work well in practice. However, certain precautions need to be exercised. Specifically, special attention needs to be paid to NE surface forms that are common words or expressions, such as movie titles *"it"*, *"her"* or *"up"*. To avoid NEs with these surface forms being instantiated in every sentence with these words, we first identify them using a general purpose off-the-shelf language model, and then ban them from participating in parsing for the first few iterations.

2.4. Limiting Phrase Lexicon

Most of the studies involving derived multi-word units, keep model complexity at bay by applying criteria such as MDL. Similarly, we re-examine our inventory of phrase tokens at every iteration, and purge all phrase-tokens c that have not been observed often enough: $Z_{(t)}(c) < \theta_2$.

3. Experiments

3.1. Implementation

We employed the SRILM toolkit [14] to train and compute probabilities with n-gram language models (Witten-Bell smoothing was selected to support fractional observation counts), but also to perform decoding on a trellis that we use to encode the search space for parsing. One of the two criteria we utilize to measure optimization progress is the aggregated sentence probability $P(\boldsymbol{w}) = \sum_{\boldsymbol{c}} P(\boldsymbol{w}, \boldsymbol{c})$ measured on the unseen test set. The summation is carried over the lattice of parses. For the experiments below, we conduct a fixed number of iterations ($T = 10$), but perplexity change could be used as an alternative stopping criterion as well. The second metric is word error rate (WER) observed when recognizing utterances from the test set using the trained token-level LM.

The WPE LM treats NEs as classes. Furthermore, for each selected phrase, a new pseudo-word is created with a concatenated pronunciation. While co-articulation effects may offer additional improvement opportunities for multi-word phrases [4], we leave them for future explorations. The minimum phrase count is set to $\theta_2 = 10.0$, and the pool of candidate phrases consists of all word sub-sequences of length ≤ 6.

3.2. Data

We expect multi-level LMs to be particularly helpful for bootstrapping narrowly defined domains with a limited inventory of named entities. High personalization potential or dynamic NEs whose contents are subject to rapid changes offer additional motivation for WPE LMs. Consequently, only a moderate amount of training data can be expected, without the ability to tag it manually.[3] As an example of such a setup, we have selected the calendar scenario from the personal assistant domain, where people use natural language to set up, query or edit appointments (but also perform other related tasks) in their smart phone calendar. Below are four examples of in-domain utterances:

- *am i free on saturday*
- *move appointment with rebecca to next week*
- *will john be at marketing meeting*
- *three pm every monday and wednesday yoga class*

Our training set (180K words in 20K unique sentences) was obtained via crowd-sourcing and our test set (20K words) contains transcribed utterances from a Windows Phone calendar application. The selection of NEs reflects expectations of the domain at hand (see Table 1) and their initial weights are either uniform or reflect authors' experience from prior applications.

Table 1: *Named entities used in the experiment.*

NE	type	size	examples
city	trie	2000	new york city, boston
state	trie	55	hawaii, california, p a
first name	trie	1000	john, mary
week day	trie	7	tuesday
date	FSM	16st/132arcs	march first two thousand
time	FSM	11st/72arcs	seven twenty p m

3.3. Results

Our baseline is a word-level n-gram language model built from the training sentences. Table 2 compares perplexity, out-of-vocabulary (OOV) rate and WER obtained with the WPE LM

[3] Automatic taggers for specific domains would still require sizable amounts of in-domain training material.

trained on the same corpus using formulae from Section 2.1. The WPE LMs were trained with 10 iterations of EM. On each iteration, five best parses were used to estimate a new language model. The table shows that WPE models improve perplexity (despite their larger vocabulary) and decrease WER by almost 11%.

Table 2: *WPE LM versus word-level n-gram LM.*

LM	test ppx	word OOV (%)	WER (%)
3-grams; words	57.9	4.5	19.99
5-grams; words	58.0	4.5	19.97
3-grams; WPE	48.9	3.8	**17.81**
5-grams; WPE	49.1	3.8	17.72

Since one of the advantages of the WPE LM is that it models longer spans, the table also reports on 5-gram versions of the experiments. They show that only a small fraction of the improvements due to WPE LM can be regained by increasing the n-gram order of the word-level LM. This is in part due to the relatively small size of the training corpus. Another reason for a very weak improvement due to 5-grams is the potential slight mismatch between the training and test sets.

Next, we want to see how much of the WER improvement can be attributed to the trivial fact that WPE LMs have larger vocabulary (via NE classes). On our test set, the 3-gram baseline LM encountered 140 more OOV words than the 3-gram WPE LM. This would account for about 0.7% WER, far less than the overall improvement we have observed. Thus, the advantage of the WPE LM consists in more than just having higher-coverage vocabulary.

As for the generated phrases, the 3-gram run above selected about 650 of them. Most of the phrases are intuitively plausible (e.g. *"yoga+class"* or *"when+is+my+next"*, but there are also a few, such as *"two+pm"*, that one would expect to be identified as entity tokens **if** our goal had been to do NE tagging. However, from the language modeling perspective, it is cheaper to consider some occurrences of those common sequences as independent tokens. This demonstrates the advantage of our approach compared to a typical class-based LM.

As a next step, we add NE optimization from Section 2.2 starting with iteration $\kappa = 3$. Parameter θ_1 was set to 2.0. Note that these hyper-parameters were not tuned but rather reflected the authors' prior experience. With initial inertia $\lambda = 0.5$, the WER falls to 17.30%, a total of more than 13.5% relative improvement from the baseline. Note that all NEs appear to contribute to the observed gains. For instance, if we remove FSM-based entities DATE and TIME from the setup, WER will rise to 18.87%, about half way between the full model and WPE LM with just phrases but no entities in it (WER=19.69%). The plot in Figure 2 shows the effect of the initial inertia λ on the WER of the final WPE LM. The lowest error rates of 17.25% are achieved with $\lambda = 0.75$, though the optimal value is expected to change depending on the size of the training corpus.

Finally, we would like to see how the size of the training corpus affects recognition accuracy. Figure 3 shows how WERs of word-only baseline and WPE LM are changing as we restrict the training material to ever smaller subsets. We see that WER improvements remain relatively stable with a more significant increase for a training subset of just 300 examples where lexicon size started playing a greater role. Thus, while unable to experiment with larger training sets directly, the observed stability of improvements can be interpreted as a hint that WPE

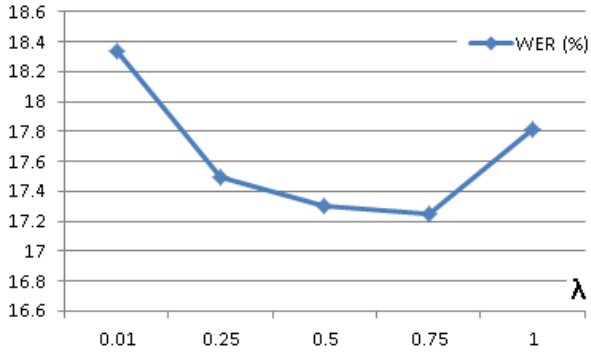

Figure 2: *Effect of the initial inertia λ on the WER.*

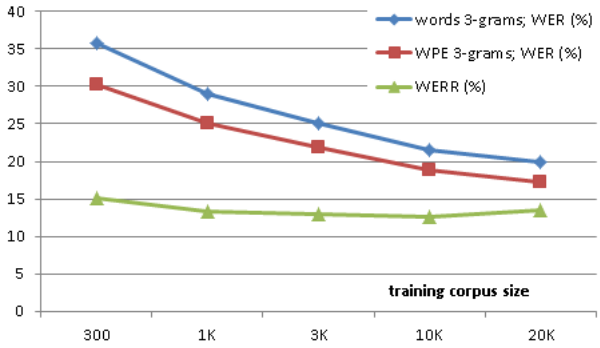

Figure 3: *WPE improvements remain relatively stable as a function of training corpus size.*

will continue outperforming word-level LM for at least somewhat larger training corpora.

4. Future Work and Conclusion

We presented a method for building token-level language models that combine words, common word phrases and classes such as named entities. Our iterative algorithm maximizes likelihood of the training corpus by re-expressing it in terms of the tokens while avoiding hard tagging decisions for phrases and classes in favor of flexible context-specific decisions for each instance. Starting with word-level text representation and a collection of generic class-definitions (weighted lists and/or FSMs), we improve perplexity of unseen data and reduce WER. Our next goal is to extend this approach to personalize user-specific NE definitions (such as names from user's address book) that could be loaded at run-time. In addition, we intend to explore alternatives to the cold start initialization where the initial unigram language model is estimated from substrings, n-grams of different lengths share the probability space and some common phrases need to be artificially blocked for the first few iterations. We also plan to apply the WPE-paradigm to continuous space language models such as RNN. Finally, the effect of noise in the training material needs to be investigated to tackle cases without textual training data. The reported experiments demonstrated how WPE n-gram LM can improve recognition accuracy for narrowly defined domains with limited training data. Specifically for the Calendar domain, our algorithm reduced WER by close to 11%, and more than 13.5% when named entities were optimized jointly with the language model.

5. References

[1] Chen, S. F.: "Shrinking Exponential Language Models"; in proc. of HLT, NA ACL, pp.468–476, 2009.

[2] Mikolov, T., Karafiát, M., Burget, L., Cernocký, J. and Khudanpur, S.: "Recurrent Neural Network Based Language Model"; in proc. of Interspeech, pp.1045–1048, 2010.

[3] Kuo, H. K. J. and Reichl, W.: "Phrase-based Language Models for Speech Recognition"; in proc. of Eurospeech, 1999.

[4] Saon, G., Padmanabhan, M.: "Data-driven Approach to Designing Compound Words for Continuous Speech Recognition"; in IEEE trans on Speech and Audio Processing, 9(4), 2001, pp.327–332.

[5] Brown, P. F., deSouza, P. V., Mercer, R. L., Della Pietra, V. J. and Lai, J. C.: "Class-based n-gram Models of Natural Language"; in Comput. Linguistics, pp.467–479, 18(4), 1992.

[6] Ries, K., Buo, F. D. and Waibel, A.: "Class Phrase Models for Language Modeling"; in proc. of ICSLP, Philadelphia, PA, 1996.

[7] Zitouni, I., Mari, J. F., Smail, K.i and Haton, J. P.: "Variable-length Sequence Language Models for Large Vocabulary Continuous Dictation Machine"; in proc. of Eurospeech, Budapest, Hungary, 1999.

[8] Pinto, D., Branstein, M., Coleman, R., Croft, W. B., King, M., Li, W. and Wei, X.: "QuASM: a System for Question Answering Using Semi-structured Data"; in proc. of 2nd ACM/IEEE-CS joint conference on Digital libraries, pp.46–55, 2002.

[9] Tan, B. and Peng, F.: "Unsupervised Query Segmentation Using Generative Language Models and Wikipedia"; in proc. of 17th int. conf. on WWW, pp.347–356, 2008.

[10] Wang, C. K., Hsu, B., Chang, M. W. and Kiciman, E.: "Simple and Knowledge-intensive Generative Model for Named Entity Recognition"; Microsoft Research, 2013.

[11] Olivier, D. C.: "Stochastic Grammars and Language Acquisition Mechanisms"; PhD Thesis, Harvard University, 1968.

[12] Deligne, S. and Bimbot, F.: "Language Modeling by Variable Length Sequences: Theoretical Formulation and Evaluation of Multigrams"; in proc. of ICASSP, pp.169–172, 1995.

[13] Mohri, M., Pereira, F. and Riley, M.: "The Design Principles of a Weighted Finite-state Transducer Library"; Theoretical Computer Science, 231(1), pp.17–32, 2000.

[14] Stolcke, A.: "SRILM — an Extensible Language Modeling Toolkit"; in proc. of Interspeech, 2002.