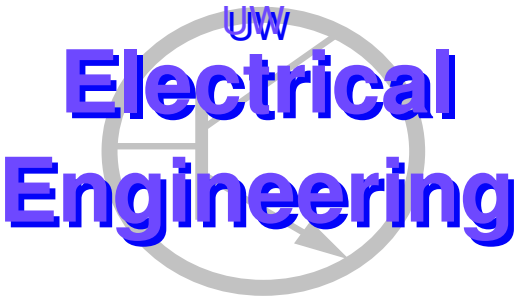
Factored Neural Language Models

Andrei Alexandrescu, Katrin Kirchhoff
andrei@cs.washington.edu, katrin@ee.washington.edu

Dept of CS, University of Washington
Seattle WA, 98195-2350

Dept of EE, University of Washington
Seattle WA, 98195-2500



UWEE Technical Report
Number UWEETR-2006-0014
July 2006

Department of Electrical Engineering
University of Washington
Box 352500
Seattle, Washington 98195-2500
PHN: (206) 543-2150
FAX: (206) 543-3842
URL: http://www.ee.washington.edu

Factored Neural Language Models

Andrei Alexandrescu, Katrin Kirchhoff
andrei@cs.washington.edu, katrin@ee.washington.edu

Dept of CS, University of Washington
Seattle WA, 98195-2350

Dept of EE, University of Washington
Seattle WA, 98195-2500

University of Washington, Dept. of EE, UWEETR-2006-0014

July 2006

Abstract

Language models based on a continuous word representation and neural network probability estimation have recently emerged as an alternative to the established backoff language models. At the same time, factored language models have been developed that use additional word information (such as parts-of-speech, morphological classes, and syntactic features) in conjunction with refined back-off strategies. We present a new type of neural probabilistic language model that learns a mapping from both words and explicit word factors into a continuous space that is then used for word prediction. Additionally, we investigate several ways of deriving continuous word representations for unknown words from those of known words. The resulting model significantly reduces perplexity on sparse-data tasks when compared to standard backoff models, standard neural language models, and factored language models. Preliminary word recognition experiments show slight improvements of factored neural language models compared to all other models.

1 Introduction

Neural language models (NLMs) [1] map words into a continuous representation space and then predict the probability of a word given the continuous representations of the preceding words in the history. They have previously been shown to yield improvements in terms of perplexity and word error rate on medium and large-vocabulary speech recognition tasks [11, 5, 9, 8] when interpolated with backoff models. The main drawbacks of NLMs are computational complexity and the fact that only distributional information (word context) is used to generalize over words, whereas other word properties (e.g. spelling, morphology etc.) are ignored for this purpose. Thus, there is also no principled way of handling out-of-vocabulary (OOV) words. Though this may be sufficient for applications that use a closed vocabulary, the current trend of porting systems to a wider range of languages (especially highly-inflected languages such as Arabic) calls for dynamic dictionary expansion and the capability of assigning probabilities to newly added words without having seen them in the training data.

The improved handling of unseen words or word combinations, in particular in morphologically rich languages, has been the objective of another recently developed model, the *Factored Language Model* (FLM). FLMs [2] use explicit word *factors* such as parts-of-speech (POS), morphological classes, and syntactic

features. FLMs predict words from previous words and their factors. Unseen words or word combinations can thus be modelled more accurately, due to the fact that OOVs may have word features that have been encountered in the training data; thus, the generalization performance of the model is enhanced. A generalized backoff scheme allows multiple probability estimates (based on different word features) to be combined. FLMs have improved perplexity and speech recognition error rate on sparse-data tasks [4, 10].

Here, we introduce a novel type of NLM that improves generalization by using vectors of word factors (stems, affixes, etc.) as input, and we investigate deriving continuous representations for unknown words from those of known words.

2 Neural Language Models

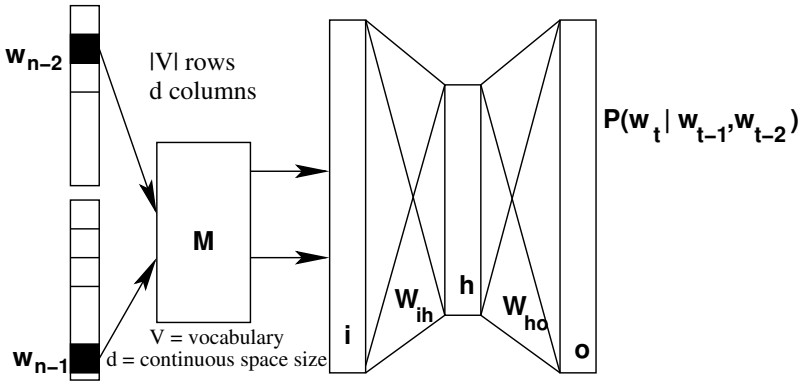


Figure 1: *NLM architecture. Each word in the context maps to a row in the matrix M. The output is next word's probability distribution.*

A standard NLM [1] (Fig. 1) takes as input the previous $n-1$ words, which select rows from a continuous word representation matrix M connecting the discrete features layer to the input $\mathbf{i}$ of the neural network. The next layer's input $\mathbf{i}$ is the concatenation of the rows in M corresponding to the input words. From here, the network is a standard multi-layer perceptron with hidden layer $\mathbf{h} = \tanh(\mathbf{i} * W_{ih} + \mathbf{b}_h)$ and output layer $\mathbf{o} = \mathbf{h} * W_{ho} + \mathbf{b}_o$, where $\mathbf{b}_{h,o}$ are the biases on the respective layers. The vector $\mathbf{o}$ is normalized by the softmax function $f_{softmax}(o_i) = \frac{e^{o_i}}{\sum_{k=1}^{|V|} e^{o_k}}$. Backpropagation (BKP) is used to learn all model parameters, including the M matrix, which is shared across input words. The training criterion maximizes the regularized log-likelihood of the training data.

Efficient training of NLMs is challenging. Every pass through the network requires computing $|V|$ exponentials and summing them over all output units. The computation in the network is dominated by the size of the output layer, $|V|$, which is usually in the 10,000s for a large-vocabulary task.

3 Factored Language Models

Factored LMs [2] represent a word as a vector of factors $w = \langle f_1, f_2, \ldots f_k \rangle$ (where one of the factors is usually an index of the word itself into the vocabulary). The task of a factored n-gram LM is to predict the next word from the factors of the previous $n-1$ words, or a subset thereof. In an FLM, due to the heterogeneity and increased dimensionality of the context, there is no obvious "natural" (e.g. temporal) backoff order. This is because an older factor might be more informative than a more recent factor from a less relevant class. This offers the opportunity to generalize backoff from the classic uni-dimensional scheme to the parallel backoff scheme depicted by the directed graph in Figure 2.

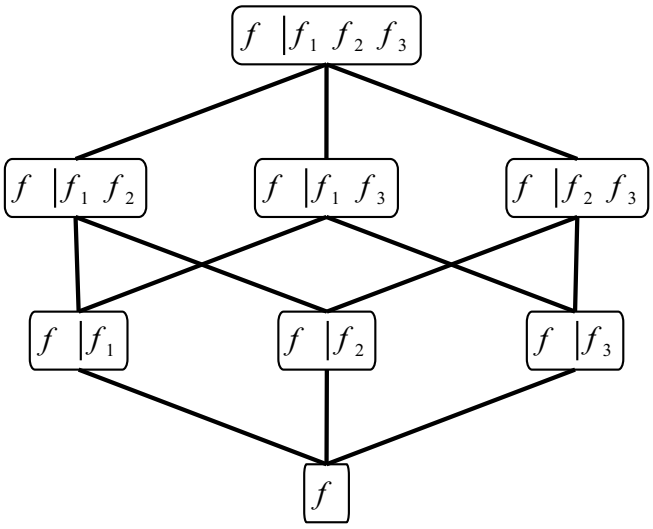


Figure 2: *A backoff graph for F with features F_1, F_2, F_3, in context. Each edge shows a backoff step in which exactly one factor is dropped. In general, however, any number of factors can be dropped in a step*

The challenge is to identify the most helpful factors for each backoff stage to improve the estimate of the probability function. In order to avoid tedious manual model selection, genetic algorithm techniques have been developed for learning the best model structure [4].

4 Generalization in Language Models

An important task in language modeling is to provide reasonable probability estimates for n-grams that were not observed in the training data. This generalization capability is becoming increasingly relevant in current large-scale speech and NLP systems that need to handle unlimited vocabularies and domain mismatches. The smooth predictor function learned by NLMs can provide good generalization if the test set contains n-grams whose individual words have been seen in similar context in the training data. However, most NLMs only have a simplistic mechanism for dealing with words that were not observed at all: OOVs in the test data are mapped to a dedicated class and are assigned the singleton probability when predicted (i.e. at the output layer) and the features of a randomly selected singleton word when occurring in the input.

In standard back-off n-gram models, OOVs are handled by reserving a small fixed amount of the discount probability mass for the generic OOV word and treating it as a standard vocabulary item. A more powerful backoff strategy is used in factored language models (FLMs) [2], which predict a word jointly from previous words and their factors. The generalized backoff procedure uses the factors to provide probability estimates for unseen n-grams, combining estimates derived from different backoff paths. This can also be interpreted as a generalization of standard class-based models [3]. FLMs have been shown to yield improvements in perplexity and word error rate in speech recognition, particularly on sparse-data tasks [10] and have also outperformed backoff models using a linear decomposition of OOVs into sequences of morphemes. In this study we use factors in the input encoding for NLMs.

5 Factored Neural Language Models

NLMs define word similarity solely in terms of their context: words are assumed to be close in the continuous space if they co-occur with the same (subset of) words. But experience with FLMs suggests that

similarity can also be derived from word shape features (affixes, capitalization, hyphenation etc.) or other annotations (e.g. POS classes). These allow a model to generalize across classes of words bearing the same feature. We thus define a *factored neural language model* (FNLM) (Fig. 3).

Just like for an FLM, the input consists of the previous $n-1$ vectors of factors. The continuous matrix now has $(n-1) \times k$ rows and d columns, where d is the size of the hidden representation, and k is the number of factors. Different factors map to disjoint row subsets of the matrix. It is possible to reflect prior knowledge of the relative contribution of each factor by allowing M to have a non-uniform structure, thus allowing more important factors (as heuristically chosen by the model creator) to map to continuous vectors of higher dimensionality. We decided to defer this option to future work.

The output of the first layer (**M**) is the concatenation of the $(n-1) \times k$ continuous representations of the input. The identity of each factor is encoded by the position of its continuous representation in the vector **i**. Unlike in FLMs, the problem of selecting the proper backoff order is obviated, for there is no more backoff at all: The neural network learns a smooth function that uses all factors in the context and decides their relative importance by means of weight learning. The **h** and **o** layers are identical to the standard NLM's. The function performed by the network is:

$$o = \mathrm{softmax}(W_{ho} \times h + B_o) \quad (1)$$

$$h = \tanh(W_{ih} \times i + B_h) \quad (2)$$

$$i = \left\langle M(f^1_{n-1})\ M(f^2_{n-1})\ \ldots\ M(f^1_{n-2})\ M(f^2_{n-2}) \ldots \right\rangle \quad (3)$$

where W_{ih} and B_h are the weights and bias of the hidden layer, respectively; W_{ho} and B_o are correspondingly the weights and bias of the output layer; and $M(f^i_{n-j})$ is M's row corresponding to the feature i situated at position j before the currently predicted word (the n^{th} word).

Like the standard NLM, the network is trained to maximize the log-likelihood of the data. All weights are trained using the backpropagation (BKP) algorithm with cross-validation on the development set and L2 regularization (the sum of squared weight values penalized by a parameter λ) in the objective function. We noticed that in spite of regularization, the model tends to "saturate" by locking onto the most frequent input patterns, at the expense of the rarely–occurring patterns. To prevent this phenomenon, we skew the distribution of the input data by manipulating the counts as follows:

$$C'(p) = \alpha C_{min} + (1-\alpha)\, C(p) \quad (4)$$

where $C(p)$ is the original count of the pattern (n-gram) p, C_{min} is the count of the least encountered pattern, and $\alpha \in [0, 1]$ is an empirically chosen flattening factor. We chose $\alpha = 0.05$.

We mentioned (section 2) the long training times for neural model. The complexity of the model is dominated by its output layer. Therefore, instead of predicting the probabilities for all words at the output layer directly, we first group words into classes (obtained by Brown clustering) and then compute the conditional probability of each word given its class: $P(w_t) = P(c_t) \times P(w_t|c_t)$. The thin output layer in Fig. 3 depicts this postprocessing step. Theoretically this step trades off some precision to gain speed; in practice, however, we actually noticed that reducing the parameter space (W_{ho} and B_o are smaller when using classes) improves the quality of learning, particularly for a sparse training set. This is a speed-up technique similar to the hierarchical structuring of output units used by [7], except that we use a "flat" hierarchy.

6 Handling Unknown Factors in FNLMs

In an FNLM setting, a subset of a word's factors may be known or can be reliably inferred from its shape although the word itself never occurred in the training data. For example, the root, suffix, and POS of the neologism "posterized" are common and easy to infer meaningfully, although the word is not in the

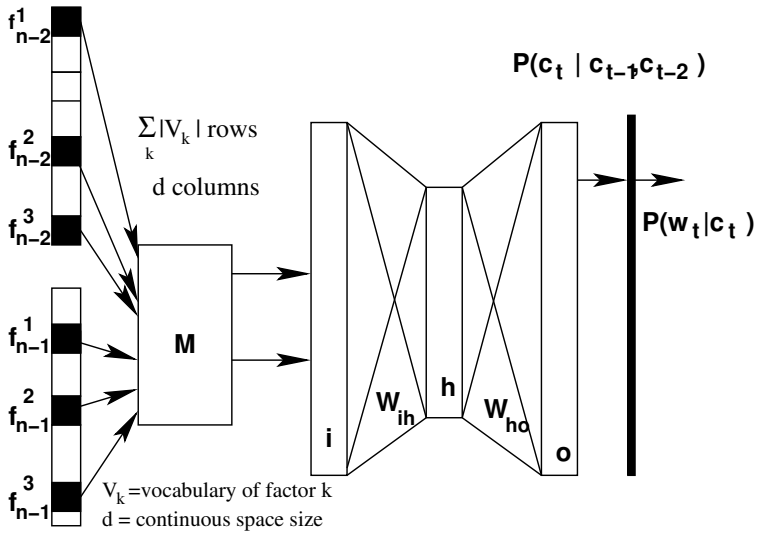


Figure 3: *FNLM architecture. Input vectors consisting of word and feature indices are mapped to rows in M. The final multiplicative layer outputs the word probability distribution.*

vocabulary or has too low a count in the training set to contribute to the model. The FNLM can use the continuous representation for these known factors directly in the input. If unknown factors are still present, new continuous representations are derived for them from those of known factors of the same type. This is done by averaging over the continuous vectors of a selected subset of the words in the training data, which places the new item in the center of the region occupied by the subset. For example, proper nouns constitute a large fraction of OOVs, and using the mean of the rows in M associated with words with a proper noun tag yields the "average proper noun" representation for the unknown word. We have experimented with the following strategies for subset selection: NULL (the null subset, i.e. the feature vector components for unknown factors are 0), ALL (average of all known factors of the same type); TAIL (averaging over the least frequently encountered factors of that type up to a threshold of 10%); and LEAST, i.e. the representation of the single least frequent factors of the same type. The prediction of OOVs themselves is unaffected since we use a factored encoding only for the input, not for the output (though this is a possibility for future work).

7 Data and Baseline Setup

7.1 Perplexity

We evaluate our approach by measuring perplexity on two different language modeling tasks. The first is the LDC CallHome Egyptian Colloquial Arabic (ECA) Corpus, consisting of transcriptions of phone conversations. ECA is a morphologically rich language that is almost exclusively used in informal spoken communication. Data must be obtained by transcribing conversations and is therefore very sparse. The present corpus has 170K words for training ($|V| = 16026$), 32K for development (dev), 17K for evaluation (eval97). The data was preprocessed by collapsing hesitations, fragments, and foreign words into one class each. The corpus was further annotated with morphological information (stems, morphological tags) obtained from the LDC ECA lexicon. The OOV rates are 8.5% (development set) and 7.7% (eval97 set), respectively.

The second corpus consists of Turkish newspaper text that has been morphologically annotated and disambiguated [6], thus providing information about the word root, POS tag, number and case. The vocabulary size is 67510 (relatively large because Turkish is highly agglutinative). 400K words are used for training,

Model	ECA ($\cdot 10^2$)		Turkish ($\cdot 10^2$)	
	dev	**eval**	**dev**	**eval**
baseline 3gram	4.108	4.128	6.385	6.438
hand-optimized FLM	4.440	4.327	4.269	4.479
GA-optimized FLM	4.325	4.179	6.414	6.637
NLM 3-gram	4.857	4.581	4.712	4.801
FNLM-NULL	5.672	5.381	9.480	9.529
FNLM-ALL	5.691	5.396	9.518	9.555
FNLM-TAIL 10%	5.721	5.420	9.495	9.540
FNLM-LEAST	5.819	5.479	10.492	10.373

Table 1: *Average probability (scaled by 10^2) of known words with unknown words in order-2 context*

#	Model	ECA dev		ECA eval		Turkish dev		Turkish eval	
		no unk	w/unk	no unk	w/unk	no unk	w/unk	no unk	w/unk
1	Baseline 3-gram	191	176	183	172	827	569	855	586
2	Class-based LM	221	278	219	269	1642	1894	1684	1930
3	1) & 2)	183	169	178	167	790	540	814	555
4	Word-based NLM	208	341	204	195	1510	1043	1569	1067
5	1) & 4)	178	165	173	162	758	542	782	557
6	Word-based NLM	202	194	204	192	1991	1369	2064	1386
7	1) & 6)	175	162	173	160	754	563	772	580
8	hand-optimized FLM	187	171	178	166	827	595	854	614
9	1) & 8)	182	167	174	163	805	563	832	581
10	genetic FLM	190	188	181	188	761	1181	776	1179
11	1) & 10)	183	166	175	164	706	488	720	498
12	factored NLM	189	173	190	175	1216	808	1249	832
13	1) & 12)	169	155	168	155	724	487	744	500
14	1) & 10) & 12)	**165**	**155**	**165**	**154**	**652**	**452**	**664**	**461**

Table 2: *Perplexities for baseline backoff LMs, FLMs, NLMs, and LM interpolation*

100K for development (11.8% OOVs), and 87K for testing (11.6% OOVs). The corpus was preprocessed by removing segmentation marks (titles and paragraph boundaries).

7.2 Speech Recognition

We also evaluated our system in a speech recognition task for the aforementioned ECA corpus. The recognition system used was the SRI DECIPHER^{TM} system, which makes use of multiple recognition passes. The front-end consists of 52 mel-frequency cepstral coefficients (13 base coefficients + 1st + 2nd +3rd differences), reduced with HLDA to 39 dimensions. Mean and variance as well as vocal tract length normalization are performed for speaker clusters (the waves for each conversation side were clustered into an average of 3 speaker clusters). Continuous-density, genonic hidden Markov models with 128 Gaussians per genone are used. The system contains approximately 220 genones. The decoder uses a multipass approach: In the first pass (Stage 1), N-best hypotheses are generated using phoneloop-adapted non-crossword models and a bigram LM. Maximum word posterior hypotheses are obtained using N-best ROVER, which are then used to train speaker-adaptive training (SAT) and maximum-likelihood linear regression (MLLR) transforms for each speaker. The adapted models are used in the second pass to produce bigram lattices. The lattices are rescored with a trigram LM and are used as recognition networks for the following passes. Two more passes are performed, one using adapted non-crossword maximum-mutual-information (MMI) trained models, and one using adapted crossword maximum-likelihood trained models. Both sets of N-best lists in the system (N

= 2000) is rescored with the language models described above. Given that the corpus consists of recordings of highly informal phone conversations, the accuracy of the baseline is very low and hard to improve upon.

8 Experiments and Results

8.1 Perplexity

We first investigated how the different OOV handling methods affect the average probability assigned to words with OOVs in their context. Table 1 shows that the average probabilities increase compared to the strategy described in Section 4 as well as other baseline models (standard backoff trigrams and FLM, further described below), with the strongest increase observed for the scheme using the least frequent factor as an OOV factor model. This strategy is used for the models in the following perplexity experiments.

We compare the perplexity of word-based and factor-based NLMs with standard backoff trigrams, class-based trigrams, FLMs, and interpolated models. Evaluation was done with (the "w/unk" column in Table 2) and without (the "no unk" column) scoring of OOVs, in order to assess the usefulness of our approach to applications using closed vs. open vocabularies. The baseline Model 1 is a standard backoff 3-gram using modified Kneser-Ney smoothing (model orders beyond 3 did not improve perplexity). Model 2 is a class-based trigram model with Brown clustering (256 classes), which, when interpolated with the baseline 3-gram, reduces the perplexity (see row 3). Model 3 is a 3-gram word-based NLM (with output unit clustering). For NLMs, higher model orders gave improvements, demonstrating their better scalability: for ECA, a 6-gram (w/o unk) and a 5-gram (w/unk) were used; for Turkish, a 7-gram (w/o unk) and a 5-gram (w/unk) were used. Though worse in isolation, the word-based NLMs reduce perplexity considerably when interpolated with Model 1. The FLM baseline is a hand-optimized 3-gram FLM (Model 5); we also tested an FLM optimized with a genetic algorithm as described in [4] (Model 6). Rows 7-10 of Table 2 display the results. Finally, we trained FNLMs with various combinations of factors and model orders. The combination was optimized by hand on the dev set and is therefore most comparable to the hand-optimized FLM in row 8. The best factored NLM (Model 7) has order 6 for both ECA and Turkish. It is interesting to note that the best Turkish FNLM uses only word factors such as morphological tag, stem, case, etc. but not the actual words themselves in the input. The FNLM outperforms all other models in isolation except the FLM; its interpolation with the baseline (Model 1) yields the best result compared to all previous interpolated models, for both tasks and both the unk and no/unk condition. Interpolation of Model 1, FLM and FNLM yields a further improvement. The parameter values of the (F)NLMs range between 32 and 64 for d, 45-64 for the number of hidden units, and 362-1024 for C (number of word classes at the output layer).

8.2 Speech Recognition

We investigated how replacing the original language model (a standard smoothed trigram) with our language model affected global performance of the speech recognition system. Table 3 shows error rates for five systems. The baseline is the existing speech recognition system using a 3-gram language model. The rest of the models are the same as those discussed: the factored n-gram model optimized by using genetic algorithms, two word-based neural models, and the factored neural language model. The absolute improvement obtained by the factored neural language model on the dev set is an encouraging 1.5%, but only 0.3% on the eval set. None of the competing models could improve over the baseline on the eval set, confirming the difficulty of the task. We believe that these results are encouraging but need to be more substantial to be conclusive.

Model	dev	eval
Baseline	53.4%	57.5%
Factored	53.2%	57.4%
Neural 3-gram (words only)	53.2%	57.4%
Neural 7-gram (words only)	53.1%	57.5%
Factored neural	**52.9%**	**57.2%**

Table 3: Speech recognition error rate for the ECA corpus

9 Conclusion

We have introduced FNLMs, which combine neural probability estimation with factored word representations and different ways of inferring continuous word features for unknown factors. On sparse-data Arabic and Turkish language modeling task FNLMs were shown to outperform all comparable models (standard backoff 3-gram, word-based NLMs) except FLMs in isolation, and all models when interpolated with the baseline. These conclusions apply to both open and closed vocabularies. On a particularly difficult speech recognition task, the factored neural model performed marginally better than the baseline and all comparable models.

Acknowledgments

This work was funded by NSF under grant no. IIS-0326276 and DARPA under Contract No. HR0011-06-C-0023. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of these agencies.

References

[1] Y. Bengio, R. Ducharme, and P. Vincent. A neural probabilistic language model. In *NIPS*, 2000.

[2] J.A. Bilmes and K. Kirchhoff. Factored language models and generalized parallel backoff. In *HLT-NAACL*, 2003.

[3] P. F. Brown, V. J. Della Pietra, P. V. deSouza, J. C. Lai, and R. L. Mercer. Class-based n-gram models of natural language. *Computational Linguistics*, 18(4), 1992.

[4] K. Duh and K. Kirchhoff. Automatic learning of language model structure. In *COLING 2004*, 2004.

[5] A. Emami and F. Jelinek. Exact training of a neural syntactic language model. In *ICASSP 2004*, 2004.

[6] D. Hakkani-Tür, K. Oflazer, and G. Tür. Statistical morphological disambiguation for agglutinative languages. *Journal of Computers and Humanities*, 36(4), 2002.

[7] F. Morin and Y. Bengio. Hierarchical probabilistic neural network language model. In *AISTATS*, 2005.

[8] H. Schwenk. Training neural network language models on very large corpora. In *HLT/EMNLP*, 2005.

[9] H. Schwenk and J.L. Gauvain. Neural network language models for conversational speech recognition. In *ICSLP 2004*, 2004.

[10] D. Vergyri, K. Kirchhoff, K. Duh, and A. Stolcke. Morphology-based language modeling for arabic speech recognition. In *ICSLP*, 2004.

[11] P. Xu, A. Emami, and F. Jelinek. Training connectionist models for the structured language model. In *EMNLP 2003*, 2003.