MORPHEME-BASED FEATURE-RICH LANGUAGE MODELS USING DEEP NEURAL NETWORKS FOR LVCSR OF EGYPTIAN ARABIC

Amr El-Desoky Mousa[1], *Hong-Kwang Jeff Kuo*[2], *Lidia Mangu*[2], *Hagen Soltau*[2]

[1]Human Language Technology and Pattern Recognition – Computer Science Department
RWTH Aachen University, 52056 Aachen, Germany
[2]IBM T. J. Watson Research Center, Yorktown Heights, NY 10598

ABSTRACT

Egyptian Arabic (EA) is a colloquial version of Arabic. It is a low-resource morphologically rich language that causes problems in Large Vocabulary Continuous Speech Recognition (LVCSR). Building LMs on morpheme level is considered a better choice to achieve higher lexical coverage and better LM probabilities. Another approach is to utilize information from additional features such as morphological tags. On the other hand, LMs based on Neural Networks (NNs) with a single hidden layer have shown superiority over the conventional n-gram LMs. Recently, Deep Neural Networks (DNNs) with multiple hidden layers have achieved better performance in various tasks. In this paper, we explore the use of feature-rich DNN-LMs, where the inputs to the network are a mixture of words and morphemes along with their features. Significant Word Error Rate (WER) reductions are achieved compared to the traditional word-based LMs.

Index Terms— language model, morpheme, feature-rich, deep neural network, Egyptian Arabic

1. INTRODUCTION

Egyptian Arabic (EA) is the local colloquial version of Modern Standard Arabic (MSA) spoken in Egypt. It is in fact a low-resource language for which there is no widely available language resources such as written text, pronunciation dictionaries, morphological analyzers, and so forth. Moreover, it is considered one of the morphologically complex languages due to its high degree of inflection and derivation that leads to a very large number of different surface forms derived from the same root. For these reasons, EA is considered as a real challenge for LVCSR systems. Normally, a conventional word-based LVCSR system suffers from high Out-of-vocabulary (OOV) rates and poor LM probability estimates.

An alternative approach to deal with EA is the use of morpheme-based LMs in order to reduce data sparsity, lower the OOV rate and perplexity (PPL), and thereby achieve lower WERs. Morphemes are generated by applying morphological decomposition to words based on linguistic knowledge [1], or based on unsupervised approaches [2]. For MSA, some of the linguistic methods use the Buckwalter Arabic Morphological Analyzer (BAMA) [3]. In fact, almost all the available morphological analyzer tools are specifically designed for MSA. However, one important property about EA is that it shares a large portion of the written vocabulary with MSA. This makes it possible to reuse the MSA morphological analyzers for EA with some acceptable margin of error. In this work, we use the Morphological Analyzer and Disambiguator for Arabic (MADA) [4] as we previously investigated on MSA [5].

Another approach to efficiently exploit sparse training data and reduce the dependence on the discourse domain is to utilize information from additional word features such as morphological tags. Thus, to assign proper features to words and incorporate them in the probability estimation process. This usually yields better smoothing and, hopefully, better generalization to unseen word sequences. The features can be generated based on linguistic methods [6], or via data driven approaches [7]. In this paper, we derived morphological features from MADA.

One of the major disadvantages of the backoff n-gram LM is its poor performance in cases of data sparseness even when efficient smoothing techniques are used like the Modified Kneser-Ney Smoothing [8]. In contrast, Neural Network LMs (NN-LMs) estimate probabilities in a continuous space using single hidden layer (shallow) networks [9, 10]. This NN-LMs have a built-in smoothing capability that helps to achieve better generalization. Recently, Deep Neural Networks (DNNs) with multiple hidden layers have shown the capability to capture higher-level abstract information that are more discriminative to the input features. They have been shown to provide improved performance compared to shallow networks in different tasks [11, 12, 13].

In this work, we explore the use of word-based DNN-LMs. In addition, we use DNNs to estimate morpheme-based LMs having a mixture of words and morphemes as inputs. Moreover, we add word and morpheme features to the DNN inputs. This is a novel approach in which we combine the advantages of using morpheme-based LMs, feature-rich modeling along with the modeling capabilities of DNNs. A related work in [14] explores the use of feature-rich word-based shallow NN-LMs with a focus on PPL improvement only.

2. METHODOLOGY

2.1. Word decomposition

Our LM training data is processed using MADA 2.0 tool. MADA is a morphological analyzer and disambiguator tool designed for MSA and built on top of BAMA [4]. It is able to associate a complete set of morphological tags with each word in context. These tags are used to generate robust word diacritization and tokenization. For non-MSA words, MADA produces special *unknown* markers to indicate the inability to analyze the word. To get an idea of how MADA behaves differently with EA than MSA, we performed some measurements on the unknown word rate. For a typical MSA text, the unknown word rate is around 1-3%. However, for some EA text, the unknown word rate is around 10-12%. In addition, MADA produces some additional errors in the known MSA words that are used in EA in a different sense. Given that MADA achieves an accuracy of around 98% for MSA [4], then this means that we can process EA text using MADA with an accuracy of around 80-85%. Based on MADA tokenization, we produce decomposed words in the form of *" prefix+ stem +suffix"*. The '+' sign is used as a marker for full-word recombination. In our previous work in [5, 15], we have found that it is useful to keep in the recognition vocabulary some number of the high frequent full-words without decomposition. This results in hybrid LMs containing words and morphemes in one flat model. For the details of the decomposition process and constraints, see [5].

2.2. Feature derivation

Starting from the MADA morphological tags along with the generated decomposition, we derive two different features, namely *"Lexeme"* and *"Morph"*. Lexeme is an abstraction over the inflected words that groups together all word forms that differ only in one of the morphological categories such as number or gender. Morph is the morphological description of the word; it includes the word Part-of-speech (POS) and indicates whether a conjunction, particle, article or a clitic are agglutinated to the word. The LM training corpus is re-written so that every word is replaced by a vector of features as in the form: {*W-<word>:L-<lexeme>:M-<morph>*}. The same features are similarly defined for morphemes as well as for words. A vector example using Buckwalter transliteration in the case of words is: *wAl$rqyp* $\rightarrow$ {*W-wAl$rqyp:M-conj+art+AJ-FEM-SG:L-$rqy*}. However, in the case of morphemes: *wAl$rqyp* $\rightarrow$ {*W-wAl+:M-conj+art:L-wAl+*} {*W-$rqyp:M-AJ-FEM-SG:L-$rqy*}. Hence, we see that a careful handling of word morphological features could help to produce valid features for morphemes.

2.3. Neural network language models (NN-LMs)

The backoff n-gram LMs perform poorly in cases of data sparseness. Even when large training corpora are used, still extremely small probabilities are assigned to many valid word sequences. The discrete nature of the n-gram LMs makes it difficult to reach high levels of generalization even when efficient smoothing techniques are used [8]. The main issue is the lack of a notion of word similarity. In fact, the use of word features introduces a partial solution to this problem by supporting words with features in cases of sparseness. In contrast, a NN-LM [10] uses a feed-forward NN that maps words into a continuous representation space and predicts the probability of a word given the continuous representations of the preceding words in the history. The projection of words into continuous space is done jointly with the NN training in a single process. This ensures the learning of the most suitable projection matrix that best fits the probability estimation task. Thereby, words that are semantically or grammatically related are hopefully mapped to similar locations in the continuous space. Thus, the similarity is defined as being close in the multi-dimensional feature space. The probability estimates are smooth functions of the continuous word representations, a small change in the input features leads to a small change in the probability estimation. This gives the model a built-in smoothing capability that enables it to achieve better generalization. The NN-LMs have been shown to yield better PPLs and WERs compared to conventional n-gram LMs [16].

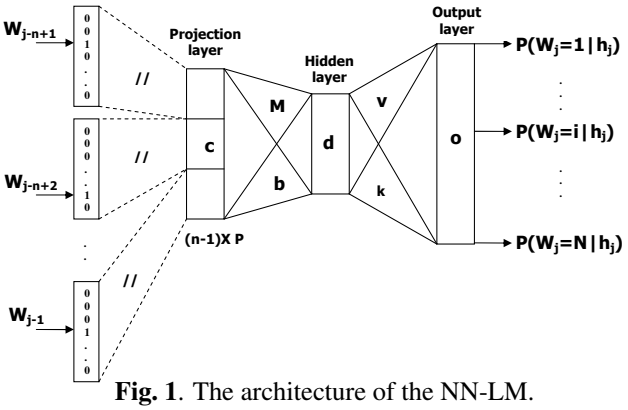

Fig. 1. The architecture of the NN-LM.

Figure 1 shows the architecture of a standard NN-LM. Assuming that the vocabulary size is N, each vocabulary word is represented by a binary N dimensional indication vector having a value of *one* at the index of that word and *zero* elsewhere. The input to the NN is the concatenated indication vectors of the $n-1$ history words. A linear projection layer is used to map each word to its continuous representation. This encoding simplifies the calculation of the projection layer since we only need to copy the i^{th} row of the $N \times P$ dimensional projection matrix. The projection matrix is tied for all history words. The continuous feature vectors of the history words are concatenated together to form the input of the hidden layer. This hidden layer has H hidden units with hyperbolic tangent activation function. This is followed by an output layer with N target units that use the softmax function to produce the posterior probabilities $P(w_j = i|h_j)$. These posteriors make up the LM probabilities of each word in the vocabulary given a specific history h_j.

Let the linear activities of the projection layer be c_l with $l = 1, ..., (n-1)P$, $\mathbf{M} = [m_{jl}]$ is the weight matrix between the projection and the hidden layer, $\mathbf{V} = [v_{ij}]$ is the weight matrix between the hidden and the output layer, b_j and k_i are the biases of the hidden and the output layers respectively, then the operations performed by the NN are:

$$d_j = tanh\left(\sum_{l=1}^{(n-1)P} m_{jl}c_l + b_j\right) \quad \forall j = 1, ..., H \quad (1)$$

$$o_i = \sum_{j=1}^{H} v_{ij}d_j + k_i \quad \forall i = 1, ..., N \quad (2)$$

$$p_i = \frac{e^{o_i}}{\sum_{r=1}^{N} e^{o_r}} = P(w_j = i|h_j) \quad \forall i = 1, ..., N \quad (3)$$

These operations are dominated by the $H \times N$ multiplications at the output layer. Therefore, we use a shortlist of output targets containing only the most frequent vocabulary words. The network is trained using the standard Back-propagation algorithm with the cross-entropy loss function.

2.4. Deep neural network language models (DNN-LMs)

A Deep Neural Network LM (DNN-LM) [17] is similar to the one in Figure 1 but employs several hidden layers of non-linearities. This deep architecture has been found to improve the performance over the single hidden layer NN across different tasks [18]. The reason is that the upper layers of the DNN represent more abstract concepts that explain the input observation, whereas lower layers extract low-level features. In [13], the DNN-LMs are investigated for the English Wall Street Journal (WSJ) speech recognition task and was found to improve both PPLs and WERs over the shallow NN-LMs.

2.5. Feature-rich deep neural network language models

To enhance the probability estimation of the DNN-LMs, we add lexeme and morph features to the inputs of the DNN (see Section 2.2). The vocabulary of lexemes and morphs is concatenated to the main hybrid word/morpheme vocabulary. Thereby, a unified binary indication vector can be used to encode word, lexeme or morph inputs to the DNN. For a given predicted word, the history words are expanded by adding features. Then, all the words and features in the history are encoded as binary indication vectors that are concatenated together and used as inputs to the DNN. Assuming that w is the predicted word, h is the history words, $\bar{h}$ is the features of the history, then the feature-rich DNN-LM is estimating the probability distribution $P(w|h, \bar{h})$. For example, in this paper, the estimated probability distribution is $P(w_t|w_{t-1}, l_{t-1}, m_{t-1}, w_{t-2}, l_{t-2}, m_{t-2})$ (3-gram like model), where l is the lexeme and m is the morph.

There are two possible approaches to use the probabilities of the feature-rich DNN-LM. The first is to perform N-best rescoring for sentences expanded with features. In this case, to combine the feature-rich DNN-LM with the standard n-gram LM, we need to do N-best score combination. This is because a direct interpolation of both models is not possible. The second approach is to perform lattice rescoring. In this paper, we investigate only the second approach leaving the first one as a future work. In order to perform lattice rescoring, we need to estimate the probability $P(w|h)$ from the distribution $P(w|h, \bar{h})$. This is done as follows:

$$P(w|h) = \sum_{\bar{h}} P(w, \bar{h}|h) = \sum_{\bar{h}} P(w|h, \bar{h})P(\bar{h}|h) \quad (4)$$

The distribution $P(w|h, \bar{h})$ is obtained from the DNN-LM ($\sum_w P(w|h, \bar{h}) = 1$). The probability $P(\bar{h}|h)$ is the probability of some features given history words. The summation of Equation 4 is performed over all possible features $\bar{h}$ that occur for the words of h in the training data. For the unrelated features, the probabilities $P(\bar{h}|h)$ are considered zeros. To estimate $P(\bar{h}|h)$, we could make the assumption that for a certain $\bar{h}$, $P(\bar{h}|h)$ is close to 1.0, whereas for other $\bar{h}$, it is close to 0.0. Under this assumption, a maximum approximation is used to estimate $P(w|h)$ in Equation 5. Alternatively, we can assume that $P(\bar{h}|h)$ is uniformly distributed such that $P(\bar{h}|h) = 1/N(\bar{h})$, where $N(\bar{h})$ is the total number of possible features $\bar{h}$ of h. This leads to Equation 6.

$$P(w|h) = \max_{\bar{h}} P(w|h, \bar{h}) \quad (5)$$

$$P(w|h) = \frac{1}{N(\bar{h})} \sum_{\bar{h}} P(w|h, \bar{h}) \quad (6)$$

The empirical results have shown that the second approximation performs better in practice. In order to apply Equation 6, we extract all the n-grams of the required length from the lattices and expand them by adding all possible features that occur for each history in the training data. The probabilities $P(w|h, \bar{h})$ are extracted from the DNN-LM. Then, the averaging of Equation 6 is performed. The features are then removed to obtain word conditional probabilities that are used to rescore the lattices. We can also interpolate these probabilities with those obtained from the DNN-LM without features.

3. EXPERIMENTAL SETUP

Our LVCSR system is trained on the Egyptian CallHome Arabic (ECA) corpus consisting of around 16h of transcribed telephone conversational speech. The acoustic models (AMs) are quint-phone across-word models trained with Maximum Likelihood (ML) and Discriminative Training (DT) based on boosted Maximum Mutual Information (bMMI). Our LM training corpora have around 7 Million running words including: acoustic transcriptions (140k words), web text (5M words), extra sources (1.5M words). We build word-based and morpheme-based systems. The first uses a 350k vocabulary of full-words, whereas the second uses a 250k vocabulary *[5k full-words + 245k morphemes]*. The number of full-words is selected so as to minimize the WER over the devel-

opment set. A grapheme-based lexicon is used with pronunciations similar to word orthographies. The speech recognizer works in 2 passes. The first pass uses feature space Maximum Likelihood Linear Regression (fMLLR) adaptation. The second pass uses MLLR adaptation. In each pass, a word- or morpheme-based 3-gram LM smoothed using the modified Kneser-Ney smoothing is used to construct the search space and to produce lattices, these lattices are rescored using different NN-LMs[1]. All the LM training corpora are used to estimate the 3-gram LMs using interpolation of the three available text sources, whereas only the first two text sources are used to train the NN-LMs. This is to speed-up the NN training process since the remaining text source is found almost not influential to the final 3-gram LM PPL. We build a separate 3-gram NN-LM for each text source. Then, we interpolate the two models together with the conventional 3-gram LM. We follow the best reported settings of NN-LMs in [13]. Let d be the feature dimension at the projection layer, h is the number of units for each hidden layer, l is the number of hidden layers, and v is the size of the output layer. Then, we use $[d = 120; h = 500; l = 1 \text{ to } 4; v = 10k \text{ to } 20k]$. Recognition performance is evaluated on ECA evaluation set [ECA-eval: 1.7h]. Parameter tuning is performed on [ECA-dev: 3.6h].

4. EXPERIMENTS

Table 1 shows the WERs and PPLs for word- and morpheme-based systems using the baseline 3-gram LMs and different NN-LMs interpolated with the 3-gram LMs. We can see the significant improvement in WER (2.9% absolute) using the morpheme-based LMs compared to the word-based LMs. In addition, significant improvements are achieved in both WERs and PPLs as a result of using NN-LMs. The largest gain is acquired by the first hidden layer. Going to more deep NN-LMs leads to little further improvements in WERs and PPLs. The best performance of the word-based system occurs with a 3-layer NN-LM. Whereas, the best performance for the morpheme-based system occurs with a 2-layer NN-LM. However, the PPL is almost not improved beyond the 2-layers. Using feature-rich NN-LMs, we achieve little more improvements. The final best results are presented in Table 2, where three morpheme-based LMs are interpolated together; namely the conventional 3-gram LMs, the NN-LMs, and the feature-rich NN-LMs. The final best WER is 55.8% achieved using 2-layer NN-LMs. This is improved by 3.9% (absolute) compared to the WER of the baseline word-based system in Table 1. Compared to the baseline morpheme-based system, we achieved 1.0% (absolute) improvement. These WER improvements are considered statistically significant (p-value ≤ 0.1) using the test proposed in [19].

5. CONCLUSIONS

We proposed a novel approach that combines the benefits of: morpheme-based LMs, feature-rich modeling, along with the recently explored DNN-LMs to perform LVCSR for Egyptian Arabic. Most of the obtained WER improvement is achieved by using morpheme-based LMs. The second most influential approach is the use of DNN-LMs. Therein, the largest improvement is obtained by the first hidden layer, whereas little additional improvement is acquired by the deeper DNN-LMs. Moreover, the use of morphological features in feature-rich DNN-LMs introduces little further improvements. The morpheme-based modeling increases the lexical coverage and reduces the data sparseness. The feature-rich modeling promotes the generalization capability of the LMs. Whereas, the use of DNN-LMs allows for efficient smoothing and higher discrimination capabilities. The best performance is achieved by interpolating the following models: the conventional n-gram LM, the DNN-LM, and the feature-rich DNN-LM. As a future work, we explore the effect of the following on the DNN-LMs: pretraining strategies, increasing the context length, N-best rescoring, optimization of layer sizes.

[1]NN-LM is a generic name for neural network LMs with l hidden layers

Table 1. *WERs [%] & PPLs over ECA-eval corpus for* **WB***: 350k word-based system OOV = 1.2%,* **MB***: 250k morpheme-based system (5k full-words + 245k morphemes) OOV = 0.75%; using a conventional 3-gram LM, NN-LM, and* **fNN-LM***: feature-rich NN-LM.*

	WB		MB	
LM	**PPL**	**WER**	**PPL**	**WER**
3-gram	330	59.7	308	56.8
3-gram + NN-LM				
1 layer	316	59.2	286	56.1
2 layers	315	59.4	285	**56.0**
3 layers	318	**59.1**	287	56.2
4 layers	319	59.2	288	56.0
3-gram + fNN-LM				
1 layer	309	59.2	280	56.0
2 layers	306	59.1	281	**56.0**
3 layers	307	**59.0**	278	56.0
4 layers	308	59.1	282	56.1

Table 2. *WERs [%] & PPLs over ECA-eval corpus for a 250k morpheme-based system (5k full-words + 245k morphemes) using an interpolated LM: 3-gram + NN-LM + fNN-LM;* **fNN-LM***: feature-rich NN-LM.*

# layers for NN/fNN-LM	**PPL**	**WER**
1 layer	286	56.0
2 layers	285	**55.8**
3 layers	285	56.0
4 layers	287	55.9

6. ACKNOWLEDGMENTS

This work was supported by the DARPA BOLT Program under Contract No. HR0011-12-C-0015. We would like to thank Ebru Arisoy for her valuable support and discussions.

7. REFERENCES

[1] G. Choueiter, D. Povey, S. Chen, and G. Zweig, "Morpheme-based language modeling for Arabic LVCSR," in *Proc. IEEE Int. Conf. on Acoustics, Speech, and Signal Processing*, vol. 1, Toulouse, France, May 2006, pp. 1053 – 1056.

[2] M. Creutz, "Induction of the morphology of natural language: Unsupervised morpheme segmentation with application to automatic speech recognition," Ph.D. dissertation, Helsinki Univ. of Technology, Finland, 2006.

[3] L. Lamel, A. Messaoudi, and J. Gauvain, "Investigating morphological decomposition for transcription of Arabic broadcast news and broadcast conversation data," in *Interspeech*, vol. 1, Brisbane, Australia, Sep. 2008, pp. 1429 – 1432.

[4] N. Habash and O. Rambow, "Arabic diacritization through full morphological tagging," in *Proc. Human Language Technology Conf. of the North American Chapter of the ACL*, vol. Companion, Rochester, NY, USA, Apr. 2007, pp. 53 – 56.

[5] A. El-Desoky, C. Gollan, D. Rybach, R. Schlüter, and H. Ney, "Investigating the use of morphological decomposition and diacritization for improving Arabic LVCSR," in *Interspeech*, Brighton, UK, Sep. 2009, pp. 2679 – 2682.

[6] G. Maltese, P. Bravetti, H. Crépy, B. Grainger, M. Herzog, and F. Palou, "Combining word- and class-based language models: A comparative study in several languages using automatic and manual word-clustering techniques," in *Proc. European Conf. on Speech Communication and Technology*, Aalborg, Denmark, Sep. 2001, pp. 21 – 24.

[7] T. Matsuzaki, Y. Miyao, and J. Tsujii, "An efficient clustering algorithm for class-based language models," in *Proc. Human Language Technology Conf. of the North American Chapter of the ACL*, vol. 4, Edmonton, Canada, May 2003, pp. 119 – 126.

[8] S. F. Chen and J. Goodman, "An empirical study of smoothing techniques for language modeling," in *Proc. Annual Meeting of the Association for Computational Linguistics*, ser. ACL '96. Stroudsburg, PA, USA: Association for Computational Linguistics, 1996, pp. 310 – 318.

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

[10] H. Schwenk, "Continuous space language models," *Computer Speech and Language*, vol. 21, no. 3, pp. 492 – 518, Jul. 2007.

[11] A. Mohamed, G. Dahl, and G. E. Hinton, "Deep belief networks for phone recognition," in *NIPS Workshop on Deep Learning for Speech Recognition and Related Applications*, Whistler, BC, Canada, 2009.

[12] F. Seide, G. Li, X. Chen, and D. Yu, "Feature engineering in context-dependent deep neural networks for conversational speech transcription," in *Proc. IEEE Automatic Speech Recognition and Understanding Workshop*, Honolulu, Hawaii, USA, Dec. 2011, pp. 24 – 29.

[13] E. Arisoy, T. Sainath, B. Kingsbury, and B. Ramabhadran, "Deep neural network language models," in *NAACL-HLT 2012 Workshop: Will We Ever Really Replace the N-gram Model? On the Future of Language Modeling for HLT*, Montreal, Canada, Jun. 2012, pp. 20 – 28.

[14] A. Alexandrescu and K. Kirchhoff, "Factored neural language models," in *Proc. Human Language Technology Conf. of the North American Chapter of the ACL*, ser. NAACL-Short '06. Stroudsburg, PA, USA: Association for Computational Linguistics, 2006, pp. 1 – 4.

[15] M. Shaik, A. El-Desoky, R. Schlüter, and H. Ney, "Using Morpheme and Syllable Based Sub-words for Polish LVCSR," in *Proc. IEEE Int. Conf. on Acoustics, Speech, and Signal Processing*, Prague, Czech Republic, May 2011, pp. 4680 – 4683.

[16] H. Schwenk and J. Gauvain, "Training neural network language models on very large corpora," in *Proc. of the Conf. on Human Language Technology and Empirical Methods in Natural Language Processing*, ser. HLT '05. Stroudsburg, PA, USA: Association for Computational Linguistics, 2005, pp. 201 – 208.

[17] Y. Bengio, "Learning deep architectures for AI," *Foundations and Trends in Machine Learning*, vol. 2, no. 1, pp. 1 – 127, Jan. 2009.

[18] T. Sainath, B. Kingsbury, and B. Ramabhadran, "Improvements in using deep belief networks for large vocabulary continuous speech recognition," IBM, Speech and Language Algorithms Group, Tech. Rep., 2012.

[19] N. Parihar and J. Picone, "DSR Front End LVCSR Evaluation - AU/384/02," Aurora Working Group, European Telecommunications Standards Institute, France, Tech. Rep., Dec. 2002.