Syntactic Features for Arabic Speech Recognition

Hong-Kwang Jeff Kuo, Lidia Mangu, Ahmad Emami, Imed Zitouni, Young-Suk Lee

IBM T.J. Watson Research Center
Yorktown Heights, NY 10598
{hkuo,mangu,emami,izitouni,ysuklee}@us.ibm.com

***Abstract*—We report word error rate improvements with syntactic features using a neural probabilistic language model through N-best re-scoring. The syntactic features we use include exposed head words and their non-terminal labels both before and after the predicted word. Neural network LMs generalize better to unseen events by modeling words and other context features in continuous space. They are suitable for incorporating many different types of features, including syntactic features, where there is no pre-defined back-off order. We choose an N-best re-scoring framework to be able to take full advantage of the complete parse tree of the entire sentence. Using syntactic features, along with morphological features, improves the word error rate (WER) by up to 5.5% relative, from 9.4% to 8.6%, on the latest GALE evaluation test set.**

I. Introduction

Word n-gram models have been extremely successful as language models (LMs) for speech recognition. The model provides a conditional probability of a predicted word given the context of $n - 1$ previous words:

$$P(w_i | w_{i-n+1} \dots w_{i-1}) \tag{1}$$

In practice, one uses a context no longer than three words (corresponding to a 4-gram LM) because the number of parameters can potentially grow exponentially with the context length. Given a fixed training set, as n is increased, the number of unique n-grams that can be reliably estimated is reduced.

However, modeling long-span dependency can help the language model better predict words. Consider the following partial sentence: "The police searched ten homes in the zone *for* ..." Suppose we want to predict the word "*for*." A conventional 4-gram language model would use the previous three words "in the zone" which intuitively is not very predictive of the word "*for*"; a more predictive context word would be "searched."

Such long-span context features could be extracted from a syntactic parse tree. One way to model longer span dependencies is to use head word information, similar to the Structured Language Model (SLM) [1], [2]. The SLM exploits syntactic structure for language modeling. A left-to-right binary parser was used to obtain multiple possible partial parses of the partial sentence (up to the word before the predicted word). The two previous exposed head words, along with their non-terminal labels are used as context features to compute the probability of the current word.

The features we use in this paper are inspired by those used in the SLM. Our work makes simplifications compared with [1], resulting in the following differences. First we do not use a strictly left-to-right binary parser. Instead we use a full sentence parser in an N-best re-scoring framework. This allows us to take advantage of any features that can be extracted from the complete parse tree of the entire sentence. Another advantage to this approach is that the quality of the parser can potentially be better than one that is constrained to be left-to-right binary, working on partial sentences. Furthermore, an off-the-shelf parser can be used without further customization.

We use a neural network to model the probabilities. A neural network language model (NNLM) has the potential to model long contexts without exponentially increasing the number of parameters [3], [4], [5].

Compared to the prior work done with syntactic language models that used read speech of grammatical sentences such as Wall Street Journal [1], [4], our current experiments are performed on broadcast news and conversations. The baseline performance against which we measure improvements is an extremely competitive system with all helpful acoustic and language modeling techniques already incorporated, including using a large un-pruned n-gram language model and NNLM with word context features [6]. Moreover, our framework is a simple one of N-best re-scoring that can be easily implemented without any modification of the decoder.

More recently, syntactic features were used in a discriminative language model training framework [7]. The framework allows flexible incorporation of features such as context-free rule productions, constituent/head features, and head-to-head dependencies. However, as common with discriminative training methods that considers acoustic scores in optimizing for WER, it requires decoding the training speech data and generating N-best hypotheses, which may be time consuming, specific to the set of acoustic models used to generate the N-best hypotheses, and may not be able to take advantage of text data without corresponding acoustic data.

In this paper, we achieved up to 5.5% relative improvement in WER using syntactic and morphological features incorporated in an NNLM. We also report improvements on both broadcast news (more clearly spoken) and broadcast conversations (more spontaneous) speech data. To our knowledge, our work is the first in recent history to report this level of improvement through the use of syntactic features in a state-of-the-art large vocabulary speech recognizer. This is also the first reported result for a non-English (specifically Arabic) system.

The paper is organized as follows. Section II describes the syntactic features extracted from the parse tree. Section III gives a brief review of neural network language models and explains why they are particularly appropriate for incorporating syntactic features. Section IV describes how scores from different models are combined in N-best re-scoring. Section V explains the additional morphological processing step prior to parsing that is used for Arabic and other morphologically rich languages such as Czech, Turkish, German, etc. Section VI describes the experimental setup, followed by the results in Section VII and conclusion in Section VIII.

II. Syntactic Features

Our goal is to use syntactic parse structure to improve speech recognition. Parse structure can potentially enable us to incorporate long-span context words and syntactic features from the parse tree. Specifically, we will be extracting head words and non-terminal labels from the parse tree as context features. Many references on syntactic parsing and head words are available, e.g. [8]. The parser we use in this paper is a maximum entropy based parser [9].

Consider a contrived sentence and syntactic parse tree in Figure 1. Suppose we are trying to predict the word "*for*." A traditional 4-gram context would consist of the words "in the zone" and as already noted, these words are not very predictive of the word "*for*," compared with a context word like "searched." Using this example, we will show the types of syntactic features we use and how they are extracted.

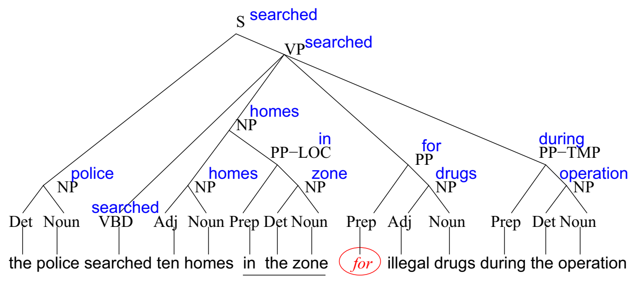

S searched
VP searched
NP homes
PP–LOC in
NP police
NP homes
NP zone
PP for
NP drugs
PP–TMP during
NP operation
VBD searched
Det Noun VBD Adj Noun Prep Det Noun Prep Adj Noun Prep Det Noun
the police searched ten homes in the zone for illegal drugs during the operation

Fig. 1. Example of sentence and parse tree

In our work, we assume that we have access to the whole parse tree, not just a partial parse of words before the predicted word. As commonly done, head words are first deterministically assigned to all non-terminal nodes using a set of linguistic rules. For pre-terminal nodes, the head word is defined to be the word in the leaf under the node. We perform the following operations to extract the first previous exposed head word and its associated non-terminal label:

1) Start at the leaf corresponding to the predicted word (w_i) and the leaf corresponding to the previous context word (w_{i-1}).
2) From each leaf, go up the tree until the two paths meet at the lowest common ancestor (LCA).
3) Cut the link between the LCA and the child that is along the path from the context word. The head word and non-terminal label of this child are chosen as context features hw_{-1} and nt_{-1}.

Figure 2 shows an example of how the first previous exposed head word is derived. Starting at the predicted word "*for*" and the previous word "*zone*," two paths are traced up the tree. They meet at the lowest common ancestor node, labeled "VP" with head word "searched." The link between the VP node and its child labeled "NP," which is along the path from the context word "*zone*," is then cut. The child node labeled "NP" with head word "homes" now forms the root node of the sub-tree. The head word hw_{-1}="homes" and non-terminal label nt_{-1}="NP" are extracted as context features to be used to predict the word "*for*."

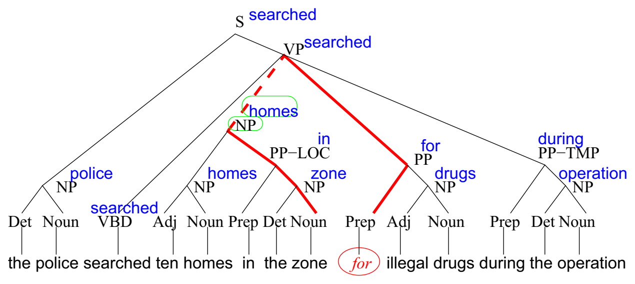

S searched
VP searched
NP homes
PP–LOC in
NP police
NP homes
NP zone
PP for
NP drugs
PP–TMP during
NP operation
VBD searched
Det Noun VBD Adj Noun Prep Det Noun Prep Adj Noun Prep Det Noun
the police searched ten homes in the zone for illegal drugs during the operation

Fig. 2. Extracting first previous head word and non-terminal label

Notice that we have crafted our definition of exposed head words so that the head word feature will never be the same as the predicted word (to avoid the awkward situation where a word is used to predict itself). This is true because head words always percolate up and not down, and after cutting the link to the LCA, the exposed head word is in a sub-tree that is disconnected from the predicted word.

After the first previous exposed head word has been extracted, the second previous exposed head word is obtained using a similar procedure, with the constraint that the node corresponding to the second head word is different from the first. Specifically,

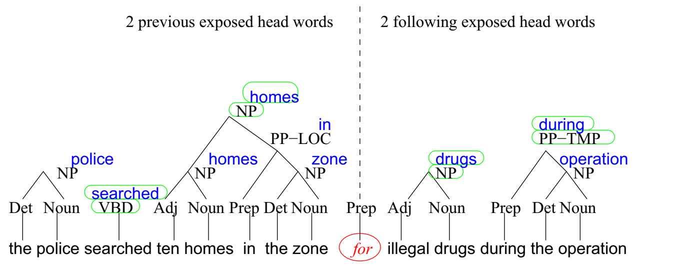
1) Set $k = 2$
2) If $i - k < 1$, return hw_{-2}="<s>" and nt_{-2}="S"
3) Otherwise, start at the leaf corresponding to the predicted word (w_i) and the leaf corresponding to the context word (w_{i-k}).
4) From each leaf, go up the tree until the two paths meet at the lowest common ancestor (LCA).
5) Cut the link between the LCA and the child that is along the path from the context word.
6) If this child node has previously been chosen, set $k = k + 1$ and go to step (2).
7) Otherwise, return the head word and non-terminal label of this child node as context features hw_{-2} and nt_{-2}.

In our example, as we consider the context words w_{i-k} (*the, in, homes, ten*) in succession, we find that after cutting the link to the LCA, the root node of the sub-tree is the same as that for the first exposed head word, with head word "homes" and non-terminal label "NP." Only when we reach the context word "searched" do we end up with the second previous exposed head word. Like the Structured Language Model [1], pre-terminal labels (part-of-speech tags) are treated the same as non-terminal labels. Thus the second previous exposed head word and non-terminal label for our example are hw_{-2}="searched" and nt_{-2}="VBD."

In addition to the previous exposed head words, we also consider exposed head words following the predicted word. This is a new type of feature not considered in [1]. Using such "non-causal" features requires more careful probability normalization, and it would not be possible to use them in decoding or lattice re-scoring. In our case, because we are treating the output of each model as a score from a different information source, and we are combining the scores at sentence-level in an N-best re-scoring framework, we are able to use such models without any problems.

Figure 3 shows the result of extracting two previous and two following exposed head words and their corresponding non-terminal labels. As mentioned earlier, the two previous exposed head words and their associated non-terminal labels in this example are hw_{-2}="searched," nt_{-2}="VBD," hw_{-1}="homes," and nt_{-1}="NP." The two following exposed head words and their associated non-terminal labels are hw_{+1}="drugs," nt_{+1}="NP," hw_{+2}="during," and nt_{+2}="PP-TMP." These features are used to help predict the word "*for*."

At each predicted position, the SLM [1] produces many partial parses and hence different exposed head words. The most likely partial parse of the SLM may not result in the same previous exposed head words as what we obtain. For example, in the SLM, the most probable partial parse may result in a node "VP(searched)" joining "VBD(searched)" and "NP(homes)" as a sub-tree. Thus the two previous head words would be "searched" and "police" (instead of "homes" and "searched" in our case). The notion of exposed head words in this paper is inspired by but not necessarily identical to that described in [1]. Our version of exposed head words is chosen to be easily computable from a full parse tree, with a guarantee that the exposed head words are different from the word being predicted.



2 previous exposed head words
2 following exposed head words
homes
NP
in
PP−LOC
police
NP
homes
NP
zone
NP
searched
VBD
drugs
NP
during
PP−TMP
operation
NP
Det Noun VBD Adj Noun Prep Det Noun Prep Adj Noun Prep Det Noun
the police searched ten homes in the zone for illegal drugs during the operation

Fig. 3. Previous and following exposed head words and non-terminal labels

III. Neural Network LM

It was mentioned earlier that neural network models are a good choice when modeling rich and long dependencies. This is definitely the case in our setup where we use head words and non-terminal tags (rich dependencies), the use of which results in long (> 4) contexts.

The basic idea behind neural network language modeling is to project words into a continuous space and let a neural network learn the prediction in that continuous space, where the model estimation task is presumably easier than the original discrete space [3], [4], [5]. The continuous space projections, or *feature vectors*, of the preceding words (or context features) make up the input to the neural network, which then will produce a probability distribution over a given vocabulary. The feature vectors are randomly initialized and are subsequently learned, along with the parameters of the neural network, so as to maximize the likelihood of the training data. The model achieves generalization by assigning to an unseen word sequence a probability close to that of a "similar" word string seen in the training data. The similarity is defined as being close in the multi-dimensional feature space. Since the probability function is a smooth function of the feature vectors, a small change in the features leads to only a small change in the probability.

To compute the conditional probability $P(y|x_1, x_2, \cdots, x_m)$, where $x_i \in V_i$ (*input vocabulary*) and $y \in V_o$ (*output vocabulary*), the model operates as follows: First for every $x_i, i = 1 \cdots m$, the corresponding feature vector (continuous space projection) is found. This is simply a table lookup operation that associates a real vector of fixed dimension d with each x_i. Secondly, these m vectors are concatenated to form a vector of size $m \cdot d$. Finally this vector is processed by the neural network which produces a probability distribution $P(.|x_1, x_2, \cdots, x_m)$ over vocabulary V_o at its output.

Note that the input and output vocabularies V_i and V_o are independent of each other and can be completely different. Training is achieved by searching for parameters Φ of the neural network and the values of feature vectors that maximize the penalized log-likelihood of the training corpus:

$$L=\frac{1}{T}\sum_t logP(y^t|x_1^t,\ldots,x_m^t;\Phi)-R(\Phi) \quad (2)$$

where superscript t denotes the t^{th} event in the training data, T is the training data size and $R(\Phi)$ is a regularization term, which in our case is a factor of the L2 norm squared of the hidden and output layer weights.

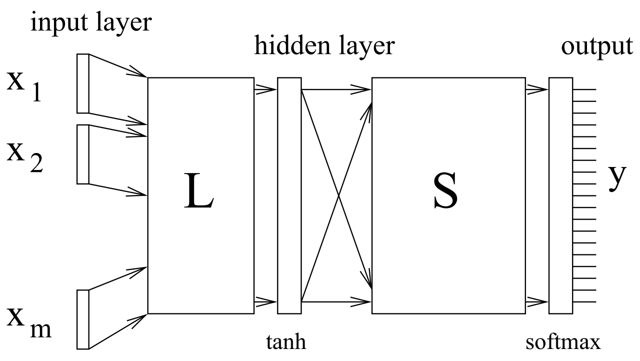


Fig. 4. The neural network architecture

The model architecture is given in Figure 4 [3], [4], [5]. The neural network is fully connected and contains one hidden layer. The operations of the input and hidden layers are given by:

$$\vec{f}=(f_1,...,f_{d\cdot m})=(\vec{f}(x_1),\vec{f}(x_2),\cdots,\vec{f}(x_m))$$
$$g_k=\tanh\left(\textstyle\sum_j f_j L_{kj}+B_k^1\right) \qquad k=1,2,...,h$$

where $\vec{f}(x)$ is the d-dimensional feature vector for token x. The weights and biases of the hidden layer are denoted by L_{kj} and B_k^1 respectively, and h is the number of hidden units.

At the output layer of the network we have:

$$z_k=\textstyle\sum_j g_j S_{kj}+B_k^2 \qquad k=1,2,...,|V_o|$$
$$p_k=\frac{e^{z_k}}{\sum_j e^{z_j}} \qquad k=1,2,...,|V_o| \tag{3}$$

with the weights and biases of the output layer denoted by S_{kj} and B_k^2 respectively. The softmax layer (Equation 3) ensures that the outputs are valid probabilities and provides a suitable framework for learning a probability distribution.

The k^{th} output of the neural network, corresponding to the k^{th} item y_k of the output vocabulary, is the desired conditional probability: $p_k = P(y^t = y_k|x_1^t,...,x_m^t)$.

The neural network weights and biases, as well as the input feature vectors, are learned simultaneously using stochastic gradient descent training via back-propagation algorithm, with the objective function being the one given in Equation 2. Details of the implementation, speed-up techniques, as well as the probability normalization and optimal NN configuration, are described in [6].

One great advantage of this model is that context length (number of inputs) can be increased, resulting in at most linear increase in model size, in contrast to exponential growth for regular n-gram models. Another advantage is that one does not have to define a back-off order of the context features. The model converges to the same solution no matter how the context features are ordered, as long as the ordering is consistent. This makes the neural network a very suitable model for capturing longer and richer probabilistic dependencies.

IV. N-best Re-scoring

We choose to use N-best re-scoring instead of lattice re-scoring due to the following reasons. First, if the difference in oracle error rate between lattice and N-best is small, we will not lose much potential performance. Another reason is simplicity of implementation: we can easily use an off-the-shelf parser to perform full sentence parsing of each N-best hypothesis. Moreover, as mentioned earlier, N-best re-scoring allows us to use models that include "non-causal" context features, as well as to combine heterogeneous models that predict different types of tokens, e.g. words or morphs.

In order to be able to use this framework, we first need to find a suitable value for N, to ensure that there is enough margin for improvement, as measured by the N-best oracle word error rate. The number of hypotheses to be extracted for each task depends on the lengths of the utterances to be decoded and the quality of the baseline acoustic and language models. As soon as we fix the set of N-best hypotheses, we take the various language models and compute the corresponding scores for each hypothesis. Given that the various modeling techniques require different representations of the sentences, based on words or morphs, we need to combine the scores at a sentence level. The weight optimization for the different sources is done using the simplex algorithm, and we use the one implemented in the SRI LM toolkit [10].

V. ATB Segmentation

Arabic is a morphologically rich language. Before parsing Arabic sentences, it is necessary to decompose or segment words into smaller units. Thus an additional step of segmentation is required at the beginning, before parsing and syntactic feature extraction. This would also be applicable to other morphologically rich languages such as Czech and Turkish.

In Arabic morphology, most morphemes are comprised of a basic word form (the root or stem), to which affixes can be attached to form whole words. Arabic white-space delimited words may be then composed of zero or more prefixes, followed by a stem and zero or more suffixes. The segmentation process consists of separating the normal white-space delimited words into (hypothesized) prefixes, stems, and suffixes, which become the tokens for further processing.

In this paper we use the Arabic Treebank (ATB) segmentation, which is a *light* segmentation adopted to the task of manually writing parse trees in the ATB corpus [11]. This type of segmentation considers splitting the word into affixes if and only if it projects an independent phrasal constituent in the parse tree. As an example, in the word ومكتبته (wmktbth — and his library), the independent phrasal constituents are: (i) conjunction و (*w* — and); (ii) noun and the head of a Noun Phrase (NP) مكتبة (*mktbp* — library); and (iii) a pronoun (PRON) ه (h — his). The ATB segmentation would be و+ مكتبت +ه (w+ mktbt +h).

A full segmentation of this word (i.e., morphological segmentation) will separate the suffix ة (*p* — feminine marker) from the word مكتبة (*mktbp* — library): (w+ mktb +p +h). Since the ة (and generally all the suffixes which are gender marks) are not independent constituents, they are not considered for ATB segmentation. Thus, the ATB segmentation scheme considers splitting only a subset of prefixes and suffixes from the stem. Note also that ATB segmentation does not separate "ال" (*Al* — the) from words.

Several algorithms for automatic segmentation of Arabic text have been presented in the recent years. In this paper we use a generalization of the algorithm presented in [12] within a *weighted finite state transducer* (WFST) framework as described in [13] .

VI. Experimental Setup

Arabic broadcast transcription is a core component of DARPA's Global Autonomous Language Exploitation (GALE) program. In this paper, we assess the usefulness of syntax-based language models on the best system that IBM fielded in the January 2009 evaluation. The acoustic model is a discriminatively-trained Subspace Gaussian Mixture

Model [14] trained on 1400h of transcribed audio and is used in a cross-adaptation scheme [15]. We have 16 sources of language model training data totaling 1.3 billion words: transcripts of the audio data, the Arabic Gigaword corpus, newsgroup and weblog data, etc. Our baseline language model is a traditional 4-gram LM interpolated with a 6-gram NNLM (context of 5 previous words). The 4-gram LM has a vocabulary of 774K words and is obtained by interpolating modified Kneser-Ney smoothed [16] 4-gram models built on each of the 16 sources. The un-pruned interpolated 4-gram model consists of roughly 500M n-grams. The NNLM is trained on a subset of the LM training data deemed most relevant to the task, specifically only the audio transcripts (12M words) of broadcast news and conversations [6]. For the NNLM, we used 30 features for the input vectors, 100 hidden units, and up to 60 epochs of training. The NNLM imparts probabilities on only the most frequent 20k words, giving zero probabilities to the rest of the vocabulary. Interpolating with the 4-gram LM ensures that the final LM probabilities of words not in the 20k list are non-zero (though not normalized).

The NNLMs using syntactic features are also trained on the same subset of data, with configurations similar to the 6-gram NNLM. However, because parsing requires ATB segmentation of words into smaller units (which we can call morphs or segmented words), both the predicted words and context words are actually such segmented words. As noted previously, ATB segmentation is very light, so on average, the ratio of the number of segmented words to that of the original words is about 1.2. The output vocabulary is limited to the most frequent 20k segmented words, and we normalize using a background 4-gram LM trained on the ATB segmented audio transcripts (see the normalization equation in [6]).

Note that different models may require different representations of the input sentence (e.g. words in the case of the baseline LM and morphs in the case of the syntax-based LM). Given that we are performing N-best re-scoring, and not lattice re-scoring, we need not combine scores at the word level but only at the sentence level. Thus we are able to take advantage of the different tokenizations of a sentence, and combine successfully a word-predicting NNLM with a morph-predicting syntax-based NNLM. For the NNLMs that use syntactic features, we also add the five previous segmented words as context features because NNLMs are able to easily handle larger contexts unlike traditional n-gram LMs.

These are the specific steps of processing during training:

1) ATB segmentation
2) Syntactic parse
3) Head word assignment
4) Feature extraction
5) Training neural network language model

To parse the Arabic corpus, we used a maximum entropy parser as described in [17], [9]. The parser was trained on the Arabic Treebank: Part 1 v 2.0 (LDC2003T06), Part2 v 2.0 (LDC2004T02), Part 3 v 1.0 (LDC200T11) and various BN data released under the DARPA GALE program. This parser was trained for other purposes, not specifically for this paper, but in the future, we hope to investigate improving the parser specifically for improving speech recognition.

In order to be able to incorporate the syntactic features described in II, we need to extract a set of N-best hypotheses. Table I shows that for our system with a 1-best WER of 9.3%, the 100-best oracle WER is 6.1%, compared to 5.2% for the lattice. Given the large improvement over the baseline, and the small difference compared to the lattice oracle WER, we use 100-best hypotheses for all experiments described here.

TABLE I
N-BEST VS. LATTICE ORACLE WER

	WER
1-best	9.3%
100-best oracle	6.1%
Lattice oracle	5.2%

During testing, N-best sentences are processed as follows:

1) ATB segmentation
2) Syntactic parse
3) Head word assignment
4) Feature extraction
5) Assign sentence score using syntax-based NNLM
6) Simplex weight optimization of various scores, including acoustic model, baseline language model, and syntax-based NNLM
7) Extract new 1-best hypothesis

We report results on a variety of test sets, dev07(2.6h), dev08(3h) and eval08U(3h) (un-sequestered portion of eval08).

VII. RESULTS

Next we explicitly list the different features used in the various NNLMs in our experiments. Table II shows the features used in each NNLM and a designated name for each model. The first model named **Word** is the 6-gram NNLM used in the baseline, where the model predicts the current word based on a context of five previous words ($w_{-5} \dots w_{-1}$). The second model **Syn1** predicts the current segmented word, based on a context of two previous exposed head words and associated non-terminal labels, as well as the five previous segmented words ($sw_{-5} \dots sw_{-1}$). The third model **Syn2** is the same as Syn1 except that it uses two following exposed head words, rather than two previous head words.

TABLE II
CONTEXT FEATURES USED IN DIFFERENT NNLMS

NNLM	#feat	Context features
Word	5	$w_{-5}w_{-4}w_{-3}w_{-2}w_{-1}$
Syn1	9	$hw_{-2}nt_{-2}hw_{-1}nt_{-1}sw_{-5} \dots sw_{-1}$
Syn2	9	$hw_{+1}nt_{+1}hw_{+2}nt_{+2}sw_{-5} \dots sw_{-1}$

Table III shows the improvement in WER using syntactic features. Note that dev07 was used to tune the weights of the different model scores via the simplex algorithm. The table shows that using two previous exposed head words and non-terminal labels (Syn1) resulted in a relative WER improvement of about 2-3% (10.6% to 10.3% on dev08 and

9.1% to 8.9% on eval08U). A similar improvement is obtained when using two following head words and non-terminal labels (Syn2). When these two models are combined, we get the best performance: a relative WER improvement of 2.8% (10.6% to 10.3%) on dev08 and 5.5% (9.1% to 8.6%) on eval08U. Compared to a traditional 4-gram LM, word and syntax-based NNLMs together provided a relative WER improvement of 5.5% (10.9% to 10.3%) on dev08 and 8.5% (9.4% to 8.6%) on eval08U. These improvements are particularly impressive because they are on top of our very best system.

TABLE III
WER RESULTS

LM	dev07	dev08	eval08U
4-gram	9.5%	10.9%	9.4%
4-gram+Word	9.3%	10.6%	9.1%
4-gram+Word+Syn1	9.0%	10.3%	8.9%
4-gram+Word+Syn2	9.0%	10.4%	8.8%
4-gram+Word+Syn1+Syn2	8.9%	10.3%	8.6%

Table IV shows the WER of the broadcast news (BN) and broadcast conversations (BC) portions of the eval08U test set. It is seen that syntactic features are helping both BN and BC. Specifically, through the use of syntactic features, for BN, the WER improves by 4.6% (6.5% to 6.2%) and for BC, the WER improves by 5.0% (12.1% to 11.5%).

TABLE IV
BN AND BC RESULTS

LM	eval08U	BN	BC
4-gram	9.4%	6.9%	12.5%
4-gram+Word	9.1%	6.5%	12.1%
4-gram+Word+Syn1	8.9%	6.5%	11.7%
4-gram+Word+Syn2	8.8%	6.3%	11.7%
4-gram+Word+Syn1+Syn2	8.6%	6.2%	11.5%

In examining some of the errors corrected by using syntactic features, we observe that many of the words appear to be acoustically confusable. For example, *wSlt* (she arrived) was corrected to *wSlp* (link). Better pronunciation or acoustic models may correct such errors, but given that such models are imperfect, the syntactic features are helping to choose the correct words. In some cases, the erroneous and corrected words have very different meanings, so the improvements in word error rate may also lead to improvements in machine translation from Arabic to English.

VIII. CONCLUSION

In this paper, we achieve significant word error rate improvements on a state-of-the-art system by using syntactic features. The results are obtained using an off-the-shelf parser and simple syntactic features incorporated in a neural network language model (NNLM). Since we are using N-best rescoring, we are able to combine scores at the sentence level from multiple heterogeneous models using the simplex algorithm. This strategy allows for access to the full parse tree of the entire sentence for any syntax-based modeling. In addition, to accommodate any other type of modeling, each model has the flexibility to represent or tokenize the sentence differently and to make predictions of different units (e.g. words, morphs, tags, sub-trees, nodes, etc.) as long as it can assign a score to the sentence.

For future work, we intend to assess the impact of the parser quality on the speech recognition performance. Also, we will investigate other syntactic features and other languages.

ACKNOWLEDGMENT

We would like to acknowledge the support of DARPA under Grant HR0011-06-2-0001 for funding part of this work. We are grateful to Dimitra Vergyri for advice on simplex optimization in the SRI LM toolkit. We also thank Jeff Sorensen and Stan Chen for their software and Fadi Biadsy for helpful comments.

REFERENCES

[1] C. Chelba and F. Jelinek, “Exploiting syntactic structure for language modeling,” in *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and Seventeenth International Conference on Computational Linguistics*, C. Boitet and P. Whitelock, Eds. San Francisco, California: Morgan Kaufmann Publishers, 1998, pp. 225–231.
[2] C. Chelba, “Exploiting syntactic structure for natural language modeling,” Ph.D. dissertation, The Johns Hopkins University, Baltimore, MD, 2000.
[3] Y. Bengio, R. Ducharme, P. Vincent, and C. Jauvin, “A neural probabilistic language model,” *Journal of Machine Learning Reseach*, vol. 3, 2003.
[4] A. Emami and F. Jelinek, “A neural syntactic language model,” *Machine Learning*, vol. 60, pp. 195–227, Sep. 2005.
[5] H. Schwenk, “Continuous space language models,” *Comput. Speech Lang.*, vol. 21, no. 3, 2007.
[6] A. Emami and L. Mangu, “Empirical study of neural network language models for Arabic speech recognition,” in *Proc. ASRU 2007*, Kyoto, Japan, Dec. 2007, pp. 147–152.
[7] M. Collins, B. Roark, and M. Saraclar, “Discriminative syntactic language modeling for speech recognition,” in *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics*, Ann Arbor, MI, 2005, pp. 507–514.
[8] D. M. Bikel, *Statistical Parsing Exposed: Viewing the Model as Data*. Saarbrücken, Germany: VDM Verlag, 2009.
[9] A. Ratnaparkhi, “Learning to parse natural language with maximum entropy models,” *Machine Learning*, vol. 34, no. 1-3, pp. 151–175, Feb. 1999.
[10] A. Stolcke, “SRILM - an extensible language modeling toolkit,” in *Proc. ICSLP*, Colorado, 2002.
[11] M. Maamouri, A. Bies, T. Buckwalter, and W. Mekki, “The Penn Arabic Treebank: Building a large-scale annotated Arabic corpus,” in *Proceedings of NEMLAR Conference on Arabic Language Resources and Tools*, Cairo, Egypt, 2004.
[12] Y.-S. Lee, K. Papineni, S. Roukos, O. Emam, and H. Hassan, “Language model based Arabic word segmentation,” in *ACL*, 2003, pp. 399–406.
[13] M. Mohri, F. C. N. Pereira, and M. Riley, “A rational design for a weighted finite-state transducer library,” in *Proceedings of the Second International Workshop on Implementing Automata*, D. Wood and S. Yu, Eds. Berlin-NY: Springer-Verlag, September 1998, pp. 144–158, vol. 1436 of Lecture Notes in Computer Science.
[14] D. Povey, “Subspace Gaussian Mixture Models for Speech Recognition,” Microsoft Research, Tech. Rep. MSR-TR-2009-64, 2009.
[15] H. Soltau, G. Saon, B. Kingsbury, H.-K. J. Kuo, L. Mangu, D. Povey, and A. Emami, “Advances in Arabic speech transcription at IBM under the DARPA GALE program,” *IEEE Trans. Audio, Speech, and Language Process.*, vol. 17, no. 5, pp. 884–894, Jul. 2009.
[16] S. F. Chen and J. Goodman, “An empirical study of smoothing techinques for language modeling,” Center for Research in Computing Technology, Harvard University, Cambridge, Massachusettes, Tech. Rep. TR-10-98, August 1998.
[17] A. Ratnaparkhi, “A linear observed time statistical parser based on maximum entropy models,” in *Proc. Empirical Methods on Natural Language Processing 1997*, Providence, RI, 1997.