*INTERSPEECH 2016*
September 8–12, 2016, San Francisco, USA

# Compositional Neural Network Language Models for Agglutinative Languages

*Ebru Arisoy*[1], *Murat Saraclar*[2]

[1]MEF University, Istanbul, Turkey
[2]Bogazici University, Istanbul, Turkey
ebruarisoy.saraclar@mef.edu.tr, murat.saraclar@boun.edu.tr

## Abstract

Continuous space language models (CSLMs) have been proven to be successful in speech recognition. With proper training of the word embeddings, words that are semantically or syntactically related are expected to be mapped to nearby locations in the continuous space. In agglutinative languages, words are made up of concatenation of stems and suffixes and, as a result, compositional modeling is important. However, when trained on word tokens, CSLMs do not explicitly consider this structure. In this paper, we explore compositional modeling of stems and suffixes in a long short-term memory neural network language model. Our proposed models jointly learn distributed representations for stems and endings (concatenation of suffixes) and predict the probability for stem and ending sequences. Experiments on the Turkish Broadcast news transcription task show that further gains on top of a state-of-the-art stem-ending-based $n$-gram language model can be obtained with the proposed models.

**Index Terms**: Language modeling, long short-term memory, sub-word-based language modeling, agglutinative languages

## 1. Introduction

Continuous space language models (CSLMs) have been shown to yield very good performances in speech recognition. The main idea in CSLMs is to embed words in a continuous space and to estimate the probability of word sequences using neural networks. The success of CSLMs, *i.e.*, feedforward neural network language models (NNLMs), over conventional $n$-gram language models is mostly explained by the improved generalization capability of CSLMs, especially for rare and unseen events. In conventional $n$-grams, words are treated as discrete entities. Whereas, CSLMs embed words in a continuous space with the expectation that semantically or syntactically related words will be mapped to nearby locations in the continuous space. As a result, NNLMs are expected to achieve better generalization for unseen or rare n-grams, since a small change in the features will result in a small change in the probability estimation. A comparative study [1] of conventional $n$-gram language models with feedforward NNLMs has advocated that CSLMs have improved generalization capability by showing that feedforward NNLMs improve over conventional $n$-gram language models, especially when the latter model backs off to lower orders.

While feedforward NNLMs [2, 3, 4, 5] predict the next word from a limited history (typically 3-4 previous words), recurrent neural network (RNN) language models [6, 7] have relaxed the limited history restriction by using a self-loop at the hidden layer. Even though the recurrent connections at the hidden layer potentially allow for a very long-range of history, in practice RNNs cannot effectively use information beyond about 5–10 time steps back, due to the well-known "vanishing gradient" problem [8]. The gradient of the error function decays exponentially over time and this reduces the influence of inputs far back in time. Long short-term memory (LSTM) neural networks address this limitation by replacing the nonlinear units in the hidden layer of an RNN with memory blocks that can store values for arbitrary amounts of time [9]. LSTM neural networks have also been applied to language modeling [10] and LSTM NNLMs have been shown to yield superior performance than RNN language models [11, 12] in speech recognition.

In agglutinative languages, words are made up of concatenation of stem and suffixes. As a result many unique word forms evolve from the same stem, resulting in very large vocabularies. Therefore, using words as recognition units results in high out-of-vocabulary (OOV) rates in speech recognition. Increasing the vocabulary size reduces the OOV rate, however, this introduces data sparsity resulting in non-robust estimates in conventional $n$-gram language modeling. The most common solution in speech recognition of agglutinative languages is to use sub-word units in language modeling [13, 14]. Especially, for Turkish, an agglutinative language with rich morphology, stems and endings (concatenation of suffixes) as recognition units provide a good compromise between OOV rate and data sparseness, yielding better accuracies than words [14].

CSLMs can be an ideal choice for agglutinative and morphologically rich languages due to their generalization ability for rare and unseen events. However, when trained on word tokens, CSLMs do not explicitly consider compositionality in word formation in these languages. One way of integrating word compositions into CSLMs is to use stem and suffix feature vectors and to represent the word as the concatenation of these feature vectors. This concatenation approach was investigated in feedforward NNLMs trained for a combined word and morpheme vocabulary and speech recognition accuracy was improved for morphologically rich Egyptian Arabic language [15]. Another way of integrating word compositions into CSLMs is to follow an additive approach, such that, a word can be represented as the sum of vectors representing its morphemes. This approach was explored in LSTM NNLMs trained for a word vocabulary and and speech recognition accuracy was improved for highly inflectional Russian language [16].

In this paper, we investigate compositional LSTM NNLMs for Turkish and propose a new LSTM architecture for integrating word compositionality into sub-word-based language modeling. We explore both the concatenative and additive approach in the proposed LSTM NNLM architecture. The rest of the paper is organized as follows: Section 2 explains the compositional LSTM NNLM. Experiments and results are described in Section 3, and Section 4 presents conclusions.



http://dx.doi.org/10.21437/Interspeech.2016-1239

## 2. Compositional Neural Network Language Model

In this section we will briefly explain the LSTM neural network language model and then give the details of the proposed compositional LSTM NNLM architectures.

### 2.1. LSTM Neural Network Language Model

Long Short-Term Memory neural networks are recurrent neural networks proposed to solve the vanishing gradient problem of RNNs. Figure 1 shows a basic RNN/LSTM language model architecture. Here, the neural network language model is composed of input, projection, hidden and output layers. Each word in the vocabulary is encoded by 1-to-$V$ coding where $V$ is the size of the input vocabulary. In 1-to-$V$ coding, each word in the vocabulary is represented by a $V$ dimensional sparse vector where only the index of that word is 1, shown with a dark spot at the input layer in Figure 1, and the rest of the entries are 0. Then the input vector is mapped into a linear projection layer. The projection layer is followed by a recurrent hidden layer and the hidden layer is connected to the output layer. Each target at the output layer corresponds to a word in the vocabulary so that the output layer produces a probability distribution over the predicted word. For instance, the dark spot at the output layer in Figure 1 represents the probability of the $i$'th word in the vocabulary given the previous words in the input sequence as the history. Since the computational complexity of the model is dominated by the multiplications at the output layer, a shortlist containing the most frequent $N$ words in the input vocabulary is used as the output vocabulary. An out-of-output-vocabulary token is used to predict the probability of the words not seen in the output vocabulary.

Formally, given an input vector sequence $X = \{x_1, \cdots, x_T\}$ and an output vector sequence $Y = \{y_1, \cdots, y_T\}$, RNN activations are calculated as follows:

$$h_t = \tanh(W_{xh}x_t + W_{hh}h_{t-1} + b_h) \quad (1)$$

$$y_t = W_{hy}h_t + b_y \quad (2)$$

where $h_t$ denotes the hidden layer vector, $W_{xh}$ denotes the input-to-hidden-layer weight matrix, $W_{hh}$ denotes the hidden-to-hidden-layer weight matrix and $W_{hy}$ denotes the output-to-hidden-layer weight matrix. The values $b_h$ and $b_y$ denote the hidden and output layer biases, respectively. Note that in Equation 1, $x_t$ represents the projection layer vector of $w_t$, the word at time $t$ in the input word sequence.

In RNN language modeling, the conditional word probabilities $P(w|h)$ are calculated using the softmax operation as follows:

$$p(w_t = i|w_{t-1}, h_{t-2}) = \frac{\exp(y_t^i)}{\sum_{j=1}^{N}\exp(y_t^j)} \quad (3)$$

where $y_t^i$ represents the $i$th element of the output vector $y_t$. Note that using the previous word as well as the previous hidden layer state while predicting the probability of the next word in RNNs translates to predicting the $n$-gram probability using a long-range of previous words in the history.

Even though RNNs potentially utilize arbitrarily long histories, in practice the effective context length of an RNN is quite limited. LSTMs remedy this limitation by replacing the nonlinear units in the hidden layer of an RNN with memory blocks containing memory cells for storing values; and multiplicative gates for reading (*output*), writing (*input*), and resetting (*forget*) these values. A memory cell can be used to store information for long periods, and gates collect activations from both inside and outside a memory block to update a memory cell's value.

The hidden layer activations of an LSTM neural network is computed as follows:

$$i_t = \sigma(W_{xi}x_t + W_{hi}h_{t-1} + W_{ci}c_{t-1} + b_i) \quad (4)$$

$$f_t = \sigma(W_{xf}x_t + W_{hf}h_{t-1} + W_{cf}c_{t-1} + b_f) \quad (5)$$

$$c_t = f_t c_{t-1} + i_t \tanh(W_{xc}x_t + W_{hc}h_{t-1} + b_c) \quad (6)$$

$$o_t = \sigma(W_{xo}x_t + W_{ho}h_{t-1} + W_{co}c_t + b_o) \quad (7)$$

$$h_t = o_t \tanh(c_t) \quad (8)$$

where $i_t$, $f_t$, $o_t$ and $c_t$ represent the input gate, forget gate, output gate and cell activation vectors at time $t$, respectively. The matrices $W_{**}$ denote the weight matrices between various layers, gates, and cells; *e.g.*, $W_{xi}$ represents the weight matrix between the input layer and the input gates. The gate and cell bias terms are denoted as $b_i$, $b_f$, $b_o$ and $b_c$; and $\sigma(\cdot)$ is the logistic sigmoid function. For language modeling, after computing $h_t$, conditional word probabilities at the output layer are calculated using Eqs. (2) and (3). Since LSTMs solve the vanishing gradient problem in RNNs, the $n$-gram probabilities in LSTM neural network language models contain a much longer history context than RNN language models.

### 2.2. Compositional LSTM Language Model

The main idea in the compositional neural network language model is to transfer word compositions into the neural network architecture. NNLMs are expected to learn word similarities, in other words, they should learn to map similar words to nearby locations in the continuous space. For instance the continuous space representation for the word "France" is expected to be close to the continuous space word representation of other countries, *i.e.*. "Germany", and the relation between the singular and plural forms of the same word, *i.e.*, "cat-cats", is expected to be encoded in the continuous space vectors. It has been shown that with proper training of the word embeddings, linguistic relations can be learned by the word embeddings and simpler architectures have been shown to be more successful than RNN LMs in learning syntactic and semantic relations [17, 18]. However, these analyses were performed for English and the type of relations that might be learned by LSTM NNLMs and the effect of these models on speech recognition performance have not been fully investigated for morphologically rich languages.

The proposed architecture for the compositional LSTM Neural Network Language Model is shown in Figure 2. Here the input vocabulary consists of stems and endings. Each stem and ending in the vocabulary are encoded by 1-to-$S$ and 1-to-$E$ codings where $S$ and $E$ are the number of stems and endings in the input vocabulary. This input layer is followed by a $2d$-dimensional linear projection layer, where $d$-dimensional continuous space representations of stem and ending corresponding to an input word are concatenated. The discrete to continuous space mapping is a look-up table whose $i$'th row corresponds to the continuous space feature representation of the $i$'th token in the vocabulary, where the tokens are stems and endings. A special symbol, $\phi$, is used when a word has no ending and the projection layer for that token is taken as a zero vector. Note that Figure 2 shows the projection layer as the stack of two linear layers. We use this representation in order demonstrate both the concatenative and additive approaches in the same figure. In the concatenative approach, the weight matrix mapping the first projection layer to the second one is an identity matrix and can

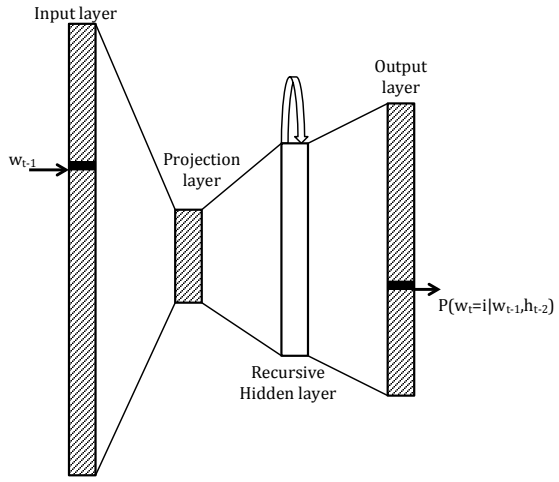


Figure 1: RNN/LSTM language model architecture

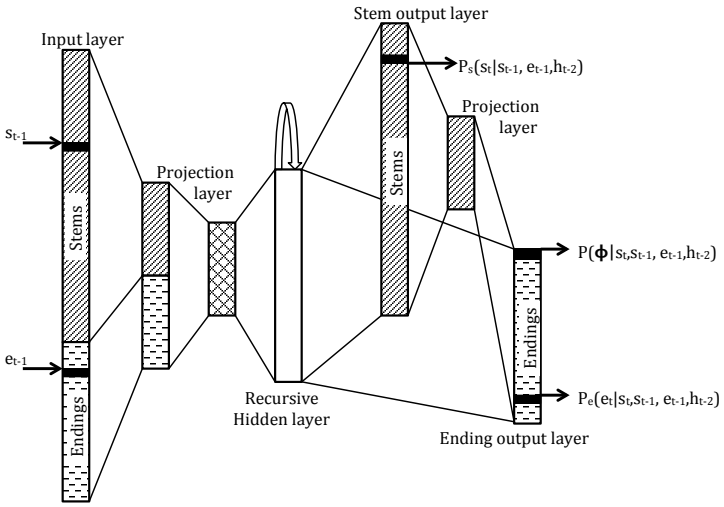


Figure 2: Compositional RNN/LSTM architecture.

be ignored. In the additive approach, the weight matrix is used to sum the concatenated projection vectors and the second stack layer is a $d$-dimensional vector. Then the projection layer is fed to the recursive hidden layer and the hidden layer is followed by two output layers: (i) stem output layer for predicting the probability of the stem given the history, (ii) ending output layer for predicting the probability of the ending given the history. After predicting the probability distribution over the predicted stem, a linear projection is applied to the stem output layer and a $d$-dimensional projection layer is obtained for the predicted stem. This projection layer is also fed to the ending output layer to predict the probability of the ending. The ending output layer also predicts the probability of the existence of an ending since the predicted word might consist of a stem only. The empty ending symbol, $\phi$, is not included in the softmax operation for the endings. The probability of the predicted ending at the output layer is computed as follows:

$$P(e_t \mid s_t, s_{t-1}, e_{t-1}, h_{t-2}) = \begin{cases} P(\phi \mid s_t, s_{t-1}, e_{t-1}, h_{t-2}) \text{ for no ending,} \\ (1 - P(\phi \mid s_t, s_{t-1}, e_{t-1}, h_{t-2})) \\ \quad \times P_e(e_t \mid s_t, s_{t-1}, e_{t-1}, h_{t-2}) \text{ otherwise.} \end{cases}$$

Finally the probability of stem and the ending given the history is computed as follows:

$$P(s_t, e_t \mid s_{t-1}, e_{t-1}, h_{t-2}) = P_s(s_t \mid s_{t-1}, e_{t-1}, h_{t-2}) \times P(e_t \mid s_t, s_{t-1}, e_{t-1}, h_{t-2}).$$

The proposed architecture allows us to distinguish between the vector spaces for stems and endings so that the neural network might learn the groupings of similar stems and similar endings separately.

# 3. Experimental Results

## 3.1. Baseline ASR System

In our research, we used word and stem+ending based LVCSR systems developed for automatic transcription of Turkish Broadcast News (BN). The acoustic model was built on a Turkish BN corpus containing 184 hours of BN recordings with

Table 1: WERs for the word and stem+ending baseline systems.

| Models | Heldout (%) | Test (%) |
|---|---|---|
| Baseline (word) | 12.4 | 13.4 |
| Baseline (stem+ending) | 11.7 | 12.8 |

Kaldi [19] using deep neural networks. Separate held-out (3.1 hours) and test (3.3 hours) data were used to evaluate the system performance. The Turkish web corpus [20] (182.3 M words) collected from major news portals and the reference transcriptions of the acoustic model training data (1.3 M words) were used in building generic and in-domain language models respectively for the BN transcription system. The word vocabulary contains the most frequent 200 K words in the training data, resulting in a 2% OOV rate on the test data. A Turkish morphological parser [20] was used to segment the words in text data into stems and endings. A word was segmented into 1.5 units on average. The most frequent 200K types in the segmented text data were chosen as the vocabulary for the stem-ending-based system, resulting in 0.2% OOV on the test data. Language models with interpolated Kneser-Ney smoothing were built using SRILM toolkit [21]. 3-gram language models were built for the word-based system and 4-gram language models were built for the stem-ending-based system. Generic and in-domain language models were linearly interpolated. The word error rate (WER) results on heldout and test sets are given in Table 1. Stem-ending-based system outperforms the word-based system by 0.7% on the held-out data and 0.6% on the test data.

## 3.2. LSTM Language Models

LSTM NNLMs were trained on the reference transcriptions of the acoustic model training data (1.3 M words). We have trained a regular LSTM language model using the architecture given in Figure 1 and two compositional language models, with additive and concatenate approaches, using the architecture given in Figure 2. Among the 200 K types in the stem and ending vocabulary, we used the most frequent 100 K types as the input vo-

cabulary. The input vocabulary contains 85.6 K stems and 14.4 K endings. In order to reduce the computational complexity of the models, we used a shortlist containing the most frequent 20 K types in the vocabulary. The output vocabulary contains approximately 16 K stems and 4 K endings. Stems and endings in the input vocabulary were represented with 180-dimensional vectors and LSTM NNLMs were trained with 300-dimensional hidden layers.

### 3.3. Results

The ASR performance of the LSTM neural network language models were evaluated in 100-best rescoring. LSTM neural network language models were log linearly interpolated with the baseline 4-gram stem+ending language model and the interpolation weights were optimized using the simplex algorithm to minimize the WER on the development set. We use the simplex algorithm implementation in the SRILM toolkit [21]. The results given in Table 2 show the effectiveness of LSTM NNLMs for different architectures. Even though all LSTM NNLMs outperform the baseline stem-ending-based $n$-gram language model, the concatenative and the additive architectures do not provide any further gain over their regular counterpart.

Table 2: WERs for the LSTM NNLMs

| Models | Heldout (%) | Test (%) |
|---|---|---|
| Baseline (stem+ending) | 11.7 | 12.8 |
| LSTM NNLM (regular) | 10.9 | 12.1 |
| LSTM NNLM (concatenative) | 11.0 | 12.1 |
| LSTM NNLM (additive) | 11.0 | 12.1 |

## 4. Analysis and Discussion

In order to analyze the differences between the regular and compositional LSTM NNLMs, we visualized the projection layers using t-Distributed Stochastic Neighbor Embedding (t-SNE) [22]. For clarity, the most frequent 5 K types were included in the analysis. A comparison of the plots for the regular LSTM NNLM presented in Figure 3 and the compositional (concatenative) LSTM NNLM given in Figure 4 clearly indicate that the latter approach is able to provide a separation of stems and endings. Furthermore, the compositional approach yields meaningful clusters for the endings. For example, the cluster in the far right in Figure 4 contains the following endings corresponding to possessive suffixes: +in +ın +ların +lerin +'nin +nin +'ın +nin +'in +nun +lerinin +larının +inin +un +'nın +sinin +sının. This example suggests that the continuous space representation of these endings are close to each other and that the NNLM is able to learn that although the endings such as +in and +ın are different in surface form, they correspond to the same lexical form and serve the same morphosyntactic function.

## 5. Conclusions

In this paper we have investigated various continuous space language models for an agglutinative language, Turkish. In particular, we have proposed compositional neural network language models based on the morphological segmentation of words into stems and endings. The proposed LSTM NNLM models jointly learn distributed representations for stems and endings and predict the probability for stem and ending sequences. The experimental results on the Turkish Broadcast news transcription task show that the LSTM NNLMs yield further gains on top of a state-of-the-art stem-ending-based $n$-gram language model. However, the compositional LSTM NNLMs do not provide any further gains over the regular LSTM NNLM. We would like to note that for these experiments the LSTM NNLMs were trained on a relatively small text corpus and we are currently working on extending this work to a larger text corpus.

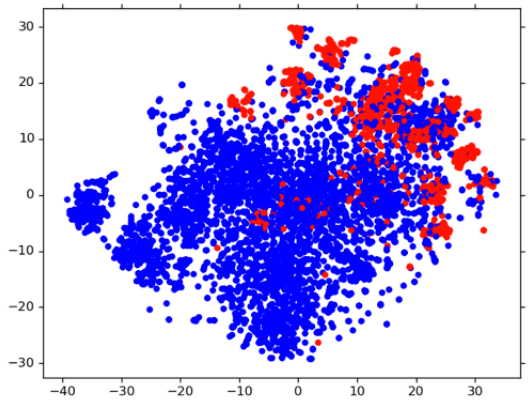


Figure 3: t-SNE plot of embeddings for the regular LSTM NNLM (blue: stem, red: ending)

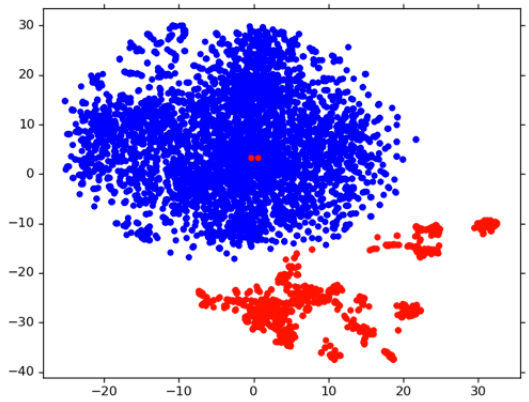


Figure 4: t-SNE plot of embeddings for the compositional (concatenative) LSTM NNLM (blue: stem, red: ending)

## 6. Acknowledgements

Ebru Arisoy was supported in part by the TUBITAK-BIDEB 2232 Program (Project No: 115C037).

# 7. References


[1] I. Oparin, M. Sundermeyer, H. Ney, and J.-L. Gauvain, "Performance analysis of neural networks in combination with n-gram language models," in *Proceedings of ICASSP*, 2012.

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

[3] H. Schwenk and J.-L. Gauvain, "Training neural network language models on very large corpora," in *Proceedings of HLT-EMNLP*, 2005, pp. 201–208.

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

[5] H.-K. J. Kuo, E. Arisoy, A. Emami, and P. Vozila, "Large scale hierarchical neural network language models," in *Proceedings of Interspeech*, Portland, Oregon, USA, 2012.

[6] T. Mikolov, M. Karafiat, L. Burget, J. Cernocky, and S. Khudanpur, "Recurrent neural network based language model," in *Proceedings of Interspeech*, 2010, pp. 1045–1048.

[7] T. Mikolov, S. Kombrink, L. Burget, J. Cernocky, and S. Khudanpur, "Extensions of recurrent neural network language model," in *Proceedings of ICASSP*, 2011, pp. 5528–5531.

[8] Y. Bengio, P. Simard, and P. Frasconi, "Learning long-term dependencies with gradient descent is difficult," *IEEE Transactions on Neural Networks*, vol. 2, no. 5, pp. 157–166, 1994.

[9] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 8, no. 9, pp. 1735–1780, 1997.

[10] M. Sundermeyer, R. Schlüter, and H. Ney, "LSTM neural networks for language modeling," in *Proceedings of Interspeech*, 2012.

[11] D. Soutner and L. Müller, "Application of LSTM neural networks in language modelling," *Lecture Notes in Computer Science*, pp. 105–112, 2013.

[12] E. Arisoy, A. Sethy, B. Ramabhadran, and S. Chen, "Bidirectional recurrent neural network language models for automatic speech recognition," in *Proceedings of ICASSP*, 2015.

[13] M. Creutz, T. Hirsimäki, M. Kurimo, A. Puurula, J. Pylkkönen, V. Siivola, M. Varjokallio, E. Arısoy, M. Saraçlar, and A. Stolcke, "Morph-based speech recognition and modeling of out-of-vocabulary words across languages," *ACM Transactions on Speech and Language Processing*, vol. 5, no. 1, pp. 1–29, 2007.

[14] E. Arısoy, D. Can, S. Parlak, H. Sak, and M. Saraçlar, "Turkish Broadcast News Transcription and Retrieval," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 17, no. 5, pp. 874–883, 2009.

[15] A. E.-D. Mousa, H.-K. J. Kuo, L. Mangu, and H. Soltau, "Morpheme-based feature-rich language models using deep neural networks for lvcsr of egyptian arabic," in *Proceedings of ICASSP*, 2013.

[16] D. Renshaw and K. B. Hall, "Long short-term memory language models with additive morphological features for automatic speech recognition," in *Proceedings of ICASSP*, 2015.

[17] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *CoRR*, vol. abs/1301.3781, 2013. [Online]. Available: http://arxiv.org/abs/1301.3781

[18] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Advances in Neural Information Processing Systems 26*, 2013, pp. 3111–3119.

[19] D. Povey *et al.*, "The kaldi speech recognition toolkit," in *IEEE 2011 Workshop on Automatic Speech Recognition and Understanding*. IEEE Signal Processing Society, Dec. 2011.

[20] H. Sak, T. Güngör, and M. Saraçlar, "Turkish language resources: Morphological parser, morphological disambiguator and web corpus," in *Proceedings of GoTAL, LNAI 5221*, 2008, pp. 417–427.

[21] A. Stolcke, "SRILM–An extensible language modeling toolkit," in *Proceedings of ICSLP*, vol. 2, Denver, 2002, pp. 901–904.

[22] L. van der Maaten and G. Hinton, "Visualizing high-dimensional data using t-sne," *Journal of Machine Learning Research*, vol. 9, pp. 2579–2605, 2008.