Commun.Fac.Sci.Univ.Ank.Series A2-A3
Volume 62, Number 1, Pages 71-78 (2020)
DOI: 10.33769/aupse.584380
ISSN 1303-6009 E-ISSN 2618-6462
http://communications.science.ankara.edu.tr/index.php?series=A2A3

# GENERATING TURKISH LYRICS WITH LONG SHORT TERM MEMORY

HAKAN ERTEN, MEHMET SERDAR GUZEL and ERKAN BOSTANCI

ABSTRACT. Long Short Term Memory (LSTM) has gained a serious achievement on sequential data which have been used generally videos, text and time-series. In this paper, we aim for generating lyrics with newly created "Turkish Lyrics" dataset. By this time, there have been studies for creating Turkish Lyrics with character-level. Unlike previous studies, we propose to Turkish Lyrics generator working with word-level instead on character-level. Also, for employing LSTM, we can't send the words as string and words must be vectorized. To vectorize, we tried two ways for encoding the words that are used in dataset and compared them. Firstly, we sample for generating one-hot encoding and then, secondly word-embedding way (Word2Vec). Observational results show us that word- level generation with word-embedding way gives more meaningful and realistic lyrics. Actually, there have not been good results enough to be used for a song because of Turkish Grammar. But, this study encourages authors to work on this field and we do believe that this study will initialize research on this area and lead researchers to contribute to this as well.

## 1. INTRODUCTION

Recurrent Neural Network (RNN) has appeared for solving the problems where sequential data will be used. It can be consecutive time or word or sentence. But, RNN is not an effective method of temporarily keeping long- term data and conventional RNN is hard to train [1]. Problems in RNN use have been solved with LSTM.

Received by the editors: June 29, 2019; Accepted: April 22, 2020.
*Key word and phrases:* LSTM, machine learning, one-hot encoding, word embedding, Turkish lyrics estimation.

Basically, LSTM has input, output, forget gates and a memory cell which can retain series of data. This cell controls the flow of information. LSTM is the state-of-the-art for many complex situations. The original formula of LSTM have been made many years ago [2] [3]. In this paper, we used architecture which is defined details in Colah et al.(2015) [4]. Figure 1 presents the design of LSTM.

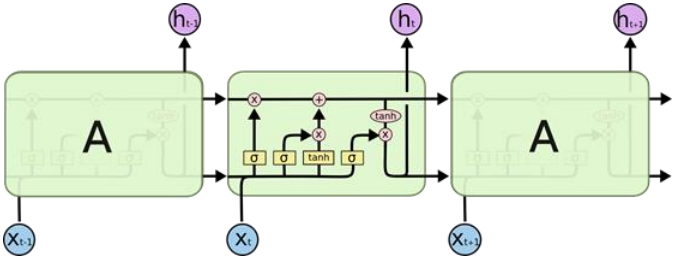


FIGURE 1. LSTM contains for interacting modules.

The model shared by [4], the following equations are identified as follows:

$$f_t = \sigma(W_f \cdot [h_t - 1, x_t] + b_f) \quad (1)$$

$$i_t = \sigma(W_i \cdot [h_t - 1, x_t] + b_i) \quad (2)$$

$$\overline{C}_t = tanh(W_C \cdot [h_t - 1, x_t] + b_C) \quad (3)$$

$$C_t = f_t * C_{t-1} + i_t * \overline{C}_t \quad (4)$$

$$o_t = \sigma(W_o \cdot [h_t - 1, x_t] + b_t) \quad (5)$$

$$h_t = o_t * tanh(C_t) \quad (6)$$

Equivalents of characters are defined below; $t$ : timestep, $x_t$ : input vector at timestep $t$, $h_t$ : hidden state, $W$ : weight matrix of associated gate, $z$ : a vector from concatenation of $x_t$ and $h_{t-1}$, $b$ : bias term, $f_t$ : forget gate, $i_t$ : input gate, $o_t$ : output gate, $C_t$ : memory cell state, $\overline{C}_t$ : candidate state value which will be regulated with $i_t$, * : the element-wise product of vectors at each side, $\sigma$ : sigmoid function, *tanh* : hyperbolic tangent function.

Our work, on the other hand, provides the first Turkish Lyrics generation with word-level and using a newly created dataset. On websites, there are some works about this topic. However, there are not any word-level works with Turkish Lyrics. Compared with character-level, because of keeping words in integrity on word-level

generation, there is no unclear word in results. This is the biggest advantage of word-level usage. Besides generation, the vectorization methods have been compared (one-hot encoding and word2vec). Our experimental evaluations show that using word-level generation with word2vec embedding deduce more meaningful words than one-hot encoding for a Turkish song phrase.

## 2. Related Works

Machine learning gather lots of attention from the researchers and is applied into different research fields [5,6]. Text generation is one of the most popular topic for using RNN. In one of the recent works, Sutskever et al. [7] presented an strength of RNN. In this character-level work, language model is originated. In 2013, the same work (originating language model) has been implemented with LSTM and originated better language model. Graves et al. [8]

Generating lyrics has been analyzed in current works. There are several works in various languages and types of music. Potash et al. [9] proposes not just generating rap lyrics, also to create a structure. In this work, authors corpus with line-by-line. They predict the next line in already existing lines. Malmi et al. [10] presents a tool which generates online rap lyrics (deepbeat.org), offers an information-retrieval approach, demonstrates the rhyme density and uses line-by-line prediction. There are also other websites which create a song with your choices.

## 3. Methodology

### 3.1 Datasets

The general Looking for a inviolate Turkish lyrics for this study, we acquired a good col- lection of lyrics website 1. This corpus has also been used in a blog [11], but for an only singer's songs (Sezen Aksu). Composing a corpus, We decided to join the songs of singers singing the same style. The all set contained 932 song lyrics in txt format, as well as basic meta-data including only the lyrics without any other data. In order to use only words, all the punctuation marks and spaces have been erased from the corpus. Hence, we didn't want to punctuation marks spaces to be perceived as a word and desired to employ only Turkish alphabet characters. Since the two words in the Turkish words can have different meanings, we checked the spelling of all words one by one (Totally 85662 words). In order to prepare for training, there are several other processes. Firstly, if all the words in the corpus had been utilized,

a layer of tens of thousands of dimensions would have been used. To keep away from this large numbers, we decided to filter uncommon words (frequency = 10). If a word is less than 10 times in the corpus, this one is ignored. After the ignored words have been removed, all the remaining words have been written in a dictionary file. Secondly, since words can't be applied as string, All the words in dictionary have been needed to put into bits and vectorized. We split the dataset 98 for training and 2 for testing.

### 3.2 Wordembeddings (Word2vec)

Word embedding is one of the most favorite model of vectorization for text files. This model can find structure of a word, relation and similarities with other words. Word embedding is a vector representation of words. Word2Vec is a style which learns word embedding using shallow neural network. Word2vec is one of the most popular methods in recent years. Word2vec can be obtained with two methods: Skip Gram and Common Bag of Words (CBOW).

In this paper, firstly Keras Embedding Layer which is initialized with random weights have been used. In training, all the weights will be updated and will learn an embedding for all words. In our model, we have defined an embedding layer with our size of words and a vector space of 1024 dimensions in which words will be embedded. Some relational sequences have been obtained.

### 3.3 One-hot encoding

As said above, words cannot be used as string. One-hot encoding which is binary vector array is another way of model that can represent each integer value. It uses a simple binary vector. Firstly, the number equivalents of the letters are kept. Then, all words are expressed according to these equivalents. In this model, there are not any relation between words. Only the integer values of words can be used. Because of that, Generating lyrics with using this model, any relational results haven't been taken.

## 4. Model and Experiment

### 4.1 Model

As expressed, previous works show the power of LSTM to model language. In our paper, using word embedding, after embedding layer Keras Bidirectional LSTM layer which has 128 dimensions and dense layer with the size of words have been

used. Dense layer connected to a single softmax output layer. To prevent overfitting, dropout function have been used. We employed sparse categorical cross entropy as loss function and Adam function for optimizer. When using one-hot encoding, embedding layer have not been used predictably and as loss function employed categorical cross entropy. As you see, there are two differences between in two models.

```
>>>Sentence[0]:
['onlar' , 'biraz' , 'terkedilmiş' , 'biraz' , 'küskün']
>>>next_Word[0]:
['çocuktular']
>>>Sentence[1]:
["'biraz' , 'terkedilmiş' , 'biraz' , 'küskün' , 'çocuktular']
>>>next_Word[1]:
['sanki']
```

FIGURE 2. Sequence and next word examples.

## 4.2 Training the network architecture

For implementation, Python implementation of an LSTM from Enrique a2 have been used. First of all, the words of all the dataset have been splitted. 85662 words in text and 13539 unique words have been found. Then, to keep away from this large dimensionality, we calculated the frequency of the words and filtered uncommon words (frequency = 5). In next step, Sequences have been created and filtered. If a sentence contains an ignored Word, this sentence will not be used for training. After that, Sequences have been shuffled and splitted training set. (Training sequence number: 2963 and test seq. number: 61) After these steps, Model have been implemented and started training to learn weights. In training, our model tried to predict the next word at the end of the sentence. Our sentence consists of 10 words and the next word is the target word. The sentence and the target word compose the sequence. Our model works are given as follow:

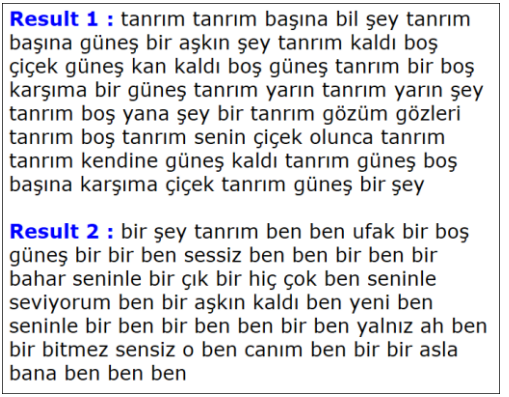


FIGURE 3. Word2vec results.

## 5. EXPERIMENTAL EVALUATION

### 5.1 Implementation details

For Word embedding and one-hot encoding architectures, we have trained the models for 100 epochs with a learning rate 0.01 and batch size 32. Mostly, after nearly 40 iterations, Good results have been obtained. The result was a 60-word lyrics. The Model is trained on a 12 GB NVIDIA TitanX GPU based laptop computer.

### 5.2 Results and discussion

In this section, we firstly present the word embedding result. Sample Lyrics is given in Figure 3. Secondly, one-hot encoding results are given in Figure 4.

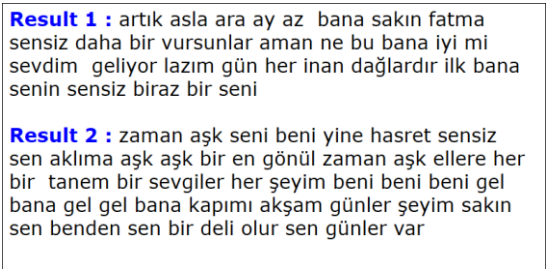


FIGURE 4. One-hot encoding results.

As a result, 85662 words have been trained but only 1052 of them used for estimation. We could not use thousands of words because of appendixes. Because Turkish is an agglutinative language. Because of that, some sentences which is created are not steady. Also, the meanings of two words of the same writing can be separated. This event can prevent the sentences from connecting the words properly. Dilemmas are frequently used in Turkish. Because of that, number of dilemmas are more than normal words. In most of the sentences generated due to the use of dilemmas too much, dilemmas took place almost nearly all the words.

## 6. Conclusion

In this paper, we collected a new Turkish Lyrics dataset. The dataset is opening up interesting directions to explore. In order to set the benchmarks in this dataset, we experimented a Turkish word-level lyrics generator. We also introduce vectorization methods (Word2vec and one-hot encoding). The experimental results indicates that the generation of lyrics with Turkish songs may be quite difficult because of the Turkish Grammar and sentence structure. Maybe, line-by-line generation can give better results for a song. Upon the publication of the paper, all the dataset will be made available to public.

## References

[1] Crash Course in Recurrent Neural Networks for Deep Learning, https://machinelearningmastery.com/ crash-course-recurrent-neural-networks-deep-learning/, accessed: 2019-06-15, 2010. 2

[2] Hochreiter, S., Schmidhuber, J., Long Short Term Memory, Technical Report FKI-207-95, URL http://citeseerx.ist.psu.edu/viewdoc/ similar? doi=10.1.1.51.3117&type =ab.2.

[3] Hochreiter, S., Schmidhuber, J., Long Short Term Memory, Neural Compu- tation, 9(8):17351780,November 1997. ISSN 0899-7667. doi: 10.1162/neco URL http://www.bioinf.jku.at/publications/older/2604.pdf. 2

[4] Understanding LSTM Networks, https://colah.github.io/posts/ 2015-08-Understanding-LSTMs/, accessed: 2019-06-17, 2015. 2

[5] Karim, A. M., Güzel, M. S., Tolun, M. R., Kaya, H., Çelebi, F. V., A new generalized deep learning framework combining sparse autoencoder and taguchi method for novel data classification and processing mathematical problems in engineering, Article ID 3145947, (2018), 13 pages.

[6] Karim, A. M., Güzel, M. S., Tolun, M. R., Kaya, H., Çelebi, F. V. A new framework using deep auto-encoder and energy spectral density for medical waveform data classification and processing, *Biocybernetics and Biomedical Engineering*, 39 (2019), 148–159.

[7] Sutskever, I., Martens, J., Hinton, G., Generating Text with Recurrent Neural Networks, *in: Proceedings of the 28th International Conference on Machine Learning, ICML*, (11 March 2011), 1017–1024,
[8] Graves, A., Mohamed, A.R., Hinton, G., Speech recognition with deep recur- rent neural networks, *in: IEEE international conference on acoustics, speech and signal processing, IEEE*, (March 2013), 6645–6649.
[9] Potash, P., Romanov, A., Rumshisky, A., Ghostwriter: Using an lstm for automatic rap lyric generation, *in: Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, (March 2015), 1919–1924.
[10] Malmi, E., Takala, P., Toivonen, H., Raiko, T., Gionis, A., DopeLearning: A Computational Approach to Rap Lyrics Generation, *in: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM*, (March 2016), 195–204.
[11] "Sezen aksu sarkisi yazan yapay zeka diyorum", 15/05/2019. https://medium.com/@tuncerergin/ sezen-aksu-sarkisi-yazan-yapay-zeka-diyorum-cd327001b7c4, Access date: 2019-05-31, 2019.

*Current Address:* Hakan ERTEN: Ankara University, Department of Computer Engineering, Ankara TURKEY
*E-mail:* hakanerten09@gmail.com
*ORCID:* http://orcid.org/0000-0002-4208-1537

Current Address: Mehmet Serdar GUZEL (Corresponding author): Ankara University, Department of Computer Engineering, Ankara TURKEY
E-mail: mguzel@ankara.edu.tr
ORCID: http://orcid.org/0000-0002-3408-0083

Current Address: Erkan BOSTANCI: Ankara University, Department of Computer Engineering, Ankara TURKEY
E-mail: ebostanci@ankara.edu.tr
ORCID: http://orcid.org/0000-0001-8547-7569