Transactions on Machine Intelligence 1(3) (2018) 172-181

ISSN Online: 2821-1693

Transactions on Machine Intelligence

Journal Homepage: https://tmachineintelligence.ir/index.php/TMI

Part-of-Speech Tagging in Persian Language using Convolutional Neural Network

E. Rahmani[1], S. Sarmadi[2,*]

[1] Computer Engineering and Information Technology Department, Urmia University of Technology, Urmia, Iran
[2] Assistant Professor, Computer Engineering and Information Technology Department, Urmia University of Technology, Urmia, Iran

ARTICLE INFO

Article History:
Received 3 June 2018
Received in revised form 20 July 2018
Accepted 9 September 2018
Available online 23 September 2018

Keywords:
Part-of-Speech Tagging, Word Embedding, Natural Language Processing, Text Corpus

ABSTRACT

Part-of-speech tagging involves identifying the grammatical roles of words within a sentence, such as nouns, verbs, and objects. This process plays a critical role in a variety of natural language processing (NLP) applications, including machine translation, syntactic parsing, spell-checking, and information retrieval. While significant research has been conducted on part-of-speech tagging for many languages, researchers working with Persian encounter unique challenges due to the language's distinctive syntactic and morphological features. Persian is an inflectional language with a complex system of verb conjugations, noun declensions, and word order variations, making it more difficult to apply standard part-of-speech tagging techniques. Traditional methods have utilized a combination of linguistic and statistical models to address these issues, but achieving high accuracy remains a complex task. In this study, we propose the use of a Convolutional Neural Network (CNN) for part-of-speech tagging in Persian. CNNs have demonstrated significant success in various NLP tasks due to their ability to automatically learn feature representations from raw input data, making them particularly effective for language processing tasks that involve complex patterns. The proposed model was evaluated on a large Persian corpus, and the results show that the CNN-based approach achieves a high accuracy rate of 98.55%. This performance indicates the potential of deep learning techniques, specifically CNNs, in overcoming the challenges associated with Persian part-of-speech tagging. The results suggest that CNNs can effectively capture the intricate syntactic and morphological patterns of Persian, providing a reliable method for part-of-speech tagging that can be further extended to other languages with similar complexities.

1. INTRODUCTION

In recent years, natural language has become a crucial means of communication between humans and computers. The vast amount of textual data available and generated daily has made exploring and understanding this data intriguing. One of the successful tools for processing and analyzing complex data such as images, speech, and text

* Corresponding Author: siamaksarmady@uut.ac.ir
Assistant Professor, Computer Engineering and Information Technology Department, Urmia University of Technology, Urmia, Iran

http://dx.doi.org/10.47176/TMI.2018.172

in recent years is neural networks, particularly deep neural networks. These networks have achieved remarkable accuracies in processing the mentioned data, often surpassing the average accuracies of humans.

Labeling phrases and words is a common stage in text processing and analysis. Besides the words themselves, their labels serve as text features in machine learning and other processing methods. The use of neural networks, specifically Multilayer Perceptrons (MLPs), in text processing is not new. However, the utilization of deep neural networks for learning and processing Persian texts has been limited. In this study, a Convolutional Neural Network is used as the primary tool for part-of-speech tagging in textual data.

Various parameters affect the performance of a tagger. The size of the training data, tagging mechanism, the language being processed, implementation method, and of course, the model training settings and parameters all influence the tagging performance. In deep convolutional neural networks, data volume notably impacts this performance. These networks typically require a large amount of data to achieve high accuracy. This article endeavors to attain a suitable level of accuracy by employing appropriate design, parameters, as well as leveraging sufficient data volume.

2. LITERATURE REVIEW

Numerous methods and models have been proposed for part-of-speech tagging in various languages. In this section, we introduce some of the methods employed for part-of-speech tagging in the Persian language. Morteza Akhavat et al. [1] introduced a Persian language tagger using the HMM method. Given the preprocessing operations before tagging, the authors introduced their method as a combination of tagging with morphological rules and HMM. Peyman Pasban et al. utilized a simple MLP neural network for tagging words in Persian texts [2].

Raja and co-workers applied three tagging methods previously used for English, German, and Spanish languages to tag Persian sentences. The first method utilized the TnT model, a type of HMM model, achieving an accuracy of 96%. The second method employed memory-based learning for tagging. In this approach, properties of each word, including its possible tags and a specific number of previous word tags, were used as features for learning. Similarity metrics were also used to measure the distance between new data features and existing class features for classification. This model also achieved an accuracy of around 96%. In the third method, a type of MLE model assigned the most seen tag in the training data to each word, with an accuracy of approximately 94% [3].

To avoid the complexities of Persian language orthography, Persian taggers first convert Persian text into Latin script before tagging. These taggers have utilized the collection of writings by Bijankhan or reduced versions of this collection. To address specific issues within the Bijankhan collection, a more suitable collection called UPEC is used in this research.

Saraji et al. initially preprocess Persian text. In this stage, spaces, half-spaces, and various orthographical forms are converted into a specific and uniform format. Sentences and words are then separated, and part-of-speech tagging of sentence components is performed [4]. The tagging method used is the HunPOS algorithm, which operates based on an HMM (trigram) and allows tagging with various features. This system, in addition to providing suitable speed and accuracy, incorporates an algorithm for morphology analysis and is available as open text. The tagging accuracy in this method has reached 97% [5].

In recent years, there has been a growing interest in convolutional neural networks for text processing and learning due to their high efficiency and speed demonstrated in image processing and speech applications. For instance, Kalchbrenner et al. presented a model for sentence classification that leverages a representation of the semantic content of sentences for this purpose. This type of classification has applications in sentiment analysis, machine translation, text summarization, and many other areas [6].

Kim also introduced a convolutional neural network for sentence classification that uses pre-trained word embeddings for word representation [7]. In this approach, a matrix of word vectors is fed into the neural network as input. Each row of the input matrix represents a vector of a word in the sentence. Convolution filters, where the number of columns matches the dimension of word vectors and the filter length (number of rows) is variable, are applied to this input matrix to obtain new feature vectors. The length of these filters typically ranges from 3 to 7 words, and the number of filters can vary from a few to several tens with different lengths.

Subsequently, max-pooling (selecting the maximum feature value) is performed on the output of each filter, resulting in a fixed-length vector equal to the number of filters. In the final layer, sentence classification is carried out using a feed-forward neural network.

The convolution and pooling operations discard information about the local word order, making the use of convolutional neural networks somewhat inappropriate for sequence labeling tasks. Therefore, it is necessary to incorporate features regarding word positions into the features used for classification. Below, several solutions to this issue are examined.

Zeng et al. [8] and Nguyen et al. [9] have utilized convolutional networks for the task of classifying the relationship between two words in a sentence. In this context, a sentence is taken as input, and the relationship between two specified words, identified by labels e1 and e2, is classified. Both papers employ a combination of word representation and positional representation for word embedding. If a sentence x consists of n words, it can be represented as a sequence of words:

$$X = [x_1, x_2, \ldots, x_n] \quad (1)$$

Each word is represented by a vector wi. To embed the position of each word, the relative distances of that word from entity e1 (xi-e1) and entity e2 (xi-e2) are computed. Using a positional embedding table D (a table randomly initialized to map a distance to a random vector), the position vector for each word is derived based on the relative distances as $p_i = [di_1, di_2]$. Finally, the vector for each word is obtained by concatenating the word representation with the positional representation:

$$x_i = [w_i, d_{i1}, d_{i2}] \quad (2)$$

Sun et al. [10] have also employed a similar method where only one word e in the text is specified. In this article, word representations are obtained by combining the word representation itself with the positional representation of the word. For this purpose, the relative distance of each word in the text from the selected word e is calculated, and the overall vector for that word is obtained as:

$$x_i = [w_i, d_i] \quad (3)$$

Labeling the constituents of speech in a sentence is a sequential classification task, and as previously mentioned, convolutional neural networks are less commonly used for sequence-to-sequence translation tasks. Strubell et al. [11] have introduced a sequence-to-sequence convolutional model that employs multiple convolutional layers, demonstrating higher efficiency for texts and long sequences compared to previous convolutional neural networks. In previous models, typically one convolutional layer is applied to the input sequence. In the architecture of this model, multiple convolutional layers are used on the input sequence. This model has achieved better results and speed in machine translation applications compared to other models. Training and prediction processes in convolutional neural networks are usually faster than recurrent networks like LSTM and GRU due to their simpler computations that also allow for better parallel processing capabilities.

3. DATASET

Given the large number of parameters in deep neural networks, utilizing large training datasets is crucial for achieving optimal performance. One available text dataset in Persian is the "Bijan-Khan" dataset, where all words are fully labeled. These texts are sourced from online resources and encompass a wide range of writing styles stored with different encoding schemes. These variations and minor flaws can impact the accuracy of natural language processing tools. Furthermore, Saraji et al. [4] have enhanced and normalized this text dataset. The new set of texts, named UPEC, adheres to all rules regarding spacing and half-spacing in the Persian language. Additionally, this dataset utilizes a 31-label set. In this research, this text dataset has been utilized for training and testing a CNN neural network.

4. PERSIAN WORD EMBEDDING

To process sentences and text using neural networks, the input sequence of words must first be represented with a numerical vector. One of the best and most efficient methods for converting words into vectors is word embedding. Among the commonly used methods for word embedding are CBOW [15], Skip-gram, and [16] GLoVe. In this study, Persian words are embedded using these three methods, and they are subsequently utilized in the following stages. For each of these models, a vocabulary file and a file containing word vectors have been prepared. In the vocabulary file, each word is assigned an identifier, and in the vector file, the vector of each word is specified with the word's identifier.

5. CONVOLUTIONAL NEURAL NETWORK FOR PART-OF-SPEECH TAGGING

As mentioned earlier, for creating sequence-to-sequence models, multi-layer convolutional neural networks can be employed [11]. These neural networks create hierarchical representations of the input sequence. In these models, words close to each other are connected in shallower layers, while words farther apart are related in deeper layers. Hierarchical structures operate faster compared to recurrent structures. This speed is due to parallel computations in convolutional networks, whereas in RNNs, computations need to be done iteratively for multiple cycles to calculate outputs.

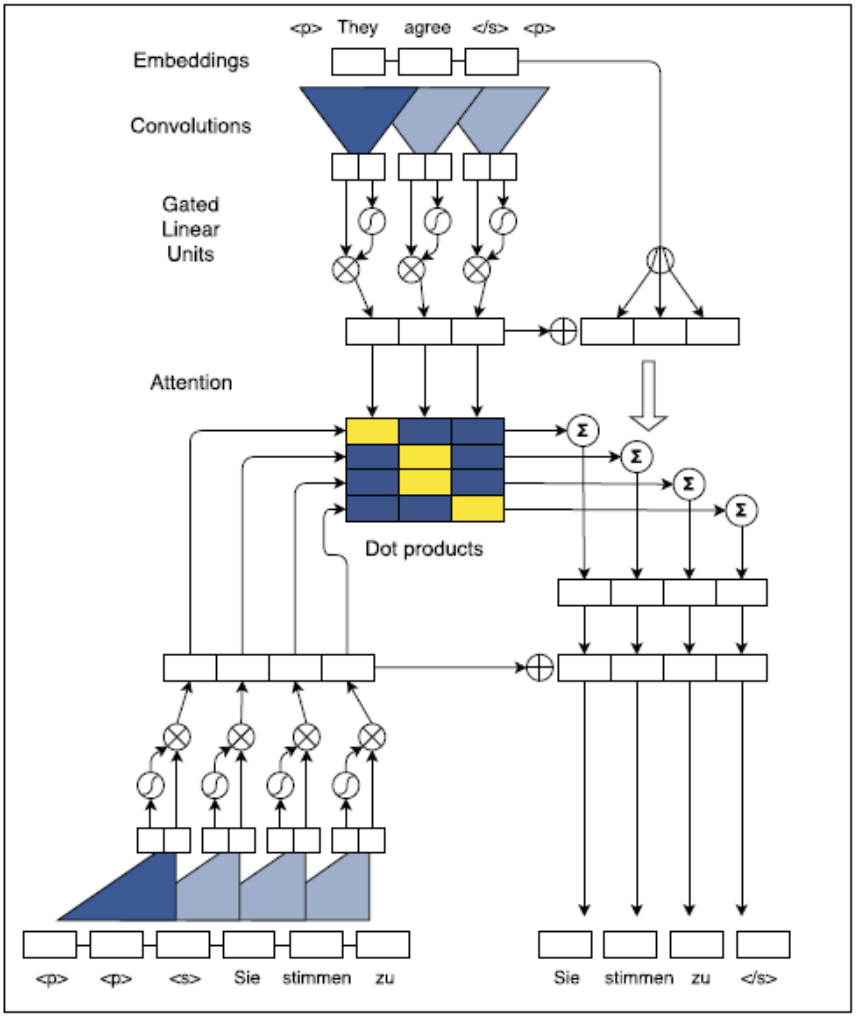

Fig. 1. Sequence-to-Sequence Convolution Architecture [12].

Gehring et al. [12] introduced a Sequence-to-Sequence Convolution architecture (Figure 1) for the task of machine translation from English to German, achieving notable results and high speed. This network employs an attention mechanism in each decoding layer.

The architecture utilized in this study for labeling components of the Persian language closely resembles the model employed by Gehring and his team. The implementation carried out for this paper is openly available and was done using the Chainer library and the Python language [13]. This library is commonly used for implementing deep neural networks [14]. In the following sections, the network architecture in use will be elaborated upon. Figure 1 can be divided into four sections. In the first section or the encoder (top part of the figure), embeddings of English language word sequences (training data and data to be translated) are generated and fed into a convolutional layer. In the second section (bottom left part), the embedded vectors of German language word sequences in the training data are passed through a convolutional layer. In the third section (center of the figure), the attention mechanism operates, receiving outputs from the first and second sections. The fourth section (bottom right) pertains to predicting the output German word sequence for test data, for which translation is forecasted. Further details on the components and the convolution model will be elucidated in subsequent sections.

5.1. Creating Embedding Vectors

As depicted in Figure 1, the input English word sequence in this model is transformed into embedded vectors before entering the encoder. Similarly, the German word sequence in the training data is also transformed into embedded vectors before entering the decoder.

To achieve this, the input sequence $X=[x_1,\ldots,x_m]$ is converted into embedding vectors $W=[w_1,\ldots,w_m]$, where $w_j \in R^f$ is extracted from a word embedding matrix D ($D \in R^{(v\times f)}$), where v is the vocabulary size and f is the size of word vectors). Additionally, the model maintains the order of input words by incorporating a position vector P ($P=[p_1,\ldots,p_m]$ where $p \in R^f$) that assigns a position based on the order of each word in the sequence. Subsequently, the two vectors W and P are combined to form the representation of the input word sequence $e=[w_1+p_1,\ldots,w_m+p_m]$.

A similar process is carried out for the sequence given to the decoder network (bottom left part of Figure 1), obtaining representations specific to the German language. The inclusion of positional information in the representation vectors of words has proven to be highly effective in the performance of this network.

5.2. Convolution Block Structure

Figure 1 illustrates the structures of the encoder (top part) and decoder (bottom left) networks, which share a similar architecture. In both networks, a sequence of words is passed through a one-dimensional convolutional block followed by a non-linear function. Non-linear functions allow the network to either extract features from the entire input string or focus on specific elements as needed.

The output of the lth block for the encoder network is denoted as $z^l = [z_1^l, \ldots, z_m^l]$, and for the decoder network as $h^l = [h_1^l, \ldots, h_n^l]$.

In the encoder, the input $x \in \mathbb{R}^{k\times d}$ is obtained by concatenating k inputs into d dimensions. The output $Y \in R^{2d}$, twice the dimensions of the input elements, is produced. Subsequent layers operate on the k outputs of the previous layer. A Gated Linear Unit (GLU) is used as a non-linear function, applying a simple mechanism to the output of the convolution $Y = [A, B] \in \mathbb{R}^{2d}$:

$$v([A,B]) = A \otimes \sigma(B) \tag{4}$$

The inputs A and B are non-linear functions in V. The symbol $\otimes$ denotes element-wise multiplication, and the output $v([A,B]) \epsilon \mathbb{R}^d$ is half the size of Y. The gates $\sigma(B)$ control which inputs of A are relevant to the current context. To activate deep convolutional networks, residual connections connect the input of each convolution to the output block.

5.3. Multi-Step Attention Mechanism

A distinct attention mechanism is employed for each layer of the decoder. To compute attention, the current decoder state h_i^l is combined with an embedded vector g_i as follows:

$$d_i^l = W_d^l h_i^l + b_d^l + g_i \tag{5}$$

For the lth decoder layer, the attention mechanism a_{ij}^l at step i and input element j is calculated through a dot product between d_i^l and each output z_j^u from the last encoder block u:

$$a_{ij}^l = \frac{\exp\,(d_i^1 .\, z_j^u)}{\sum_{t=1}^{m} \exp(d_i^1 .\, z_t^u)} \tag{6}$$

The conditional input c_i^l for the current decoder layer is a weighted sum of encoder outputs as well as embedded input elements e_j (shown in the rightmost part of Figure 1):

$$c_i^l = \sum_{j=1}^{m} a_{ij}^l (z_j^u + e_j) \tag{7}$$

This method differs from recursive approaches that calculate attention and weighted sums solely based on the encoder output z_j^u. The encoder output z_j^u effectively represents input texts, while e_j captures specific information about input elements, making them both highly suitable for prediction tasks.

Finally, a probability distribution over T elements that may be the next y_{i+1} is calculated by passing h_i^L through a linear layer with weights W_0 and bias b_0:

$$p(y_{i+1}|y_1, \dots, y_i, x) = softmax(W_0 h_i^l + b_0) \epsilon \mathbb{R}^T \tag{8}$$

As previously mentioned, this model is utilized for machine translation. In this study, the model is applied for labeling speech components, with a key distinction being the fixed input and output lengths in labeling tasks compared to the variable lengths encountered in machine translation scenarios.

In [12], standard normal initialization is used for embedding English and German words as well as positional embeddings. In this research, apart from standard initialization, four embedding methods GloVe vectors, CBOW vectors, Skip-gram vectors, and random vectors—are employed for input embeddings and compared against each other in terms of performance.

6. EXPERIMENTS AND RESULTS

In this section, the results of experiments related to command labeling of Persian sentences using a CNN neural network are presented. To evaluate the CNN neural network, embedded vectors were utilized with various methods. Rand vectors were randomly initialized and optimized during training. CBOW and Skip-gram vectors, both known as word2vec, were obtained using two different methods [15]. GloVe vectors were prepared following the method described by Pennington et al. [16]. Normal vectors, similar to the approach presented in the article by [12], were initialized with random numbers following a standard normal distribution.

Due to substantial differences in labeling accuracy among these models, the results are depicted in two separate graphs (Figure 2 and Figure 3).

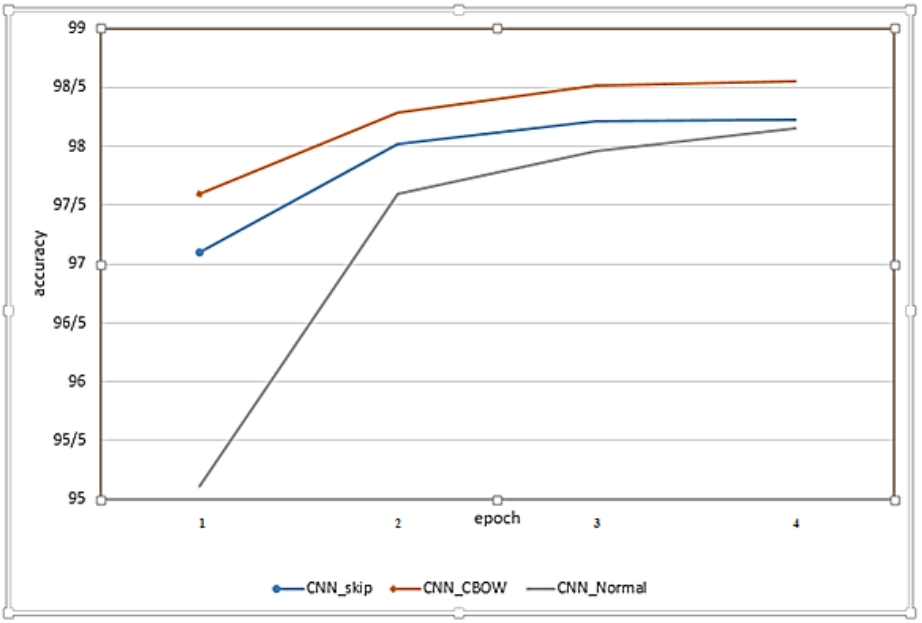

Fig. 2. Comparison of labeling accuracy using CBOW, Skip-gram, and Normal vectors.

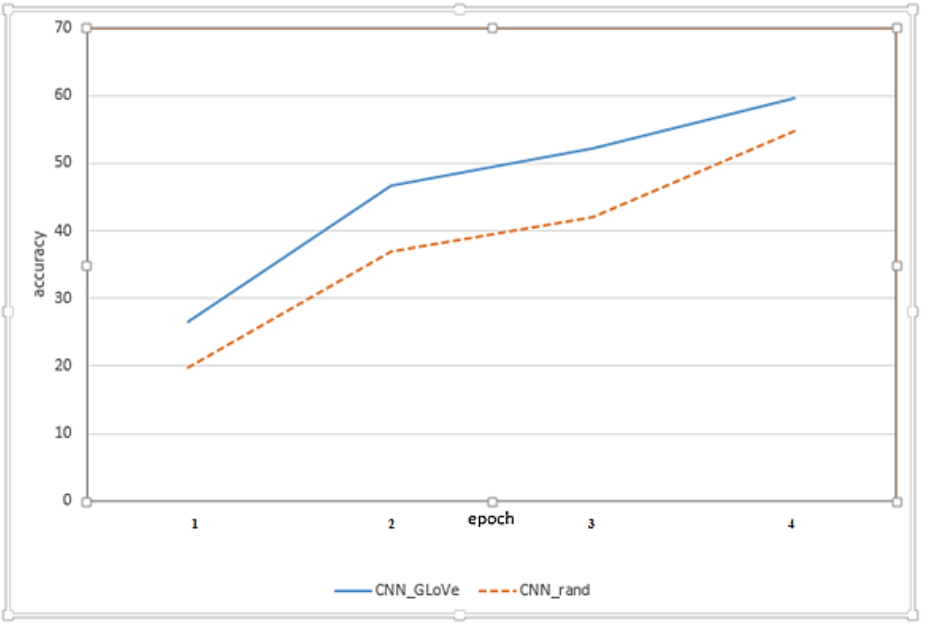

Fig. 3. Comparison of labeling accuracy using Rand and GloVe vectors.

As evident in Figures 2 and 3, CBOW vectors outperform other vectors in terms of accuracy. These vectors achieved an accuracy of 98.55% after 4 training epochs. GloVe vectors, due to the low number of training epochs in this study, only reached an accuracy of 60% compared to other methods. This accuracy slightly surpassed that of Rand random vectors, which reached a maximum accuracy of 55%.

Given the superior performance of CBOW vectors, an investigation into the impact of training data quantity and training time on these vectors is conducted. When using these vectors, an average of 405 minutes is required to complete each training epoch on the hardware utilized. Figure 4 illustrates the relationship between training time

and accuracy. Based on the graph, it can be inferred that dedicating more time will only marginally enhance the model's accuracy.

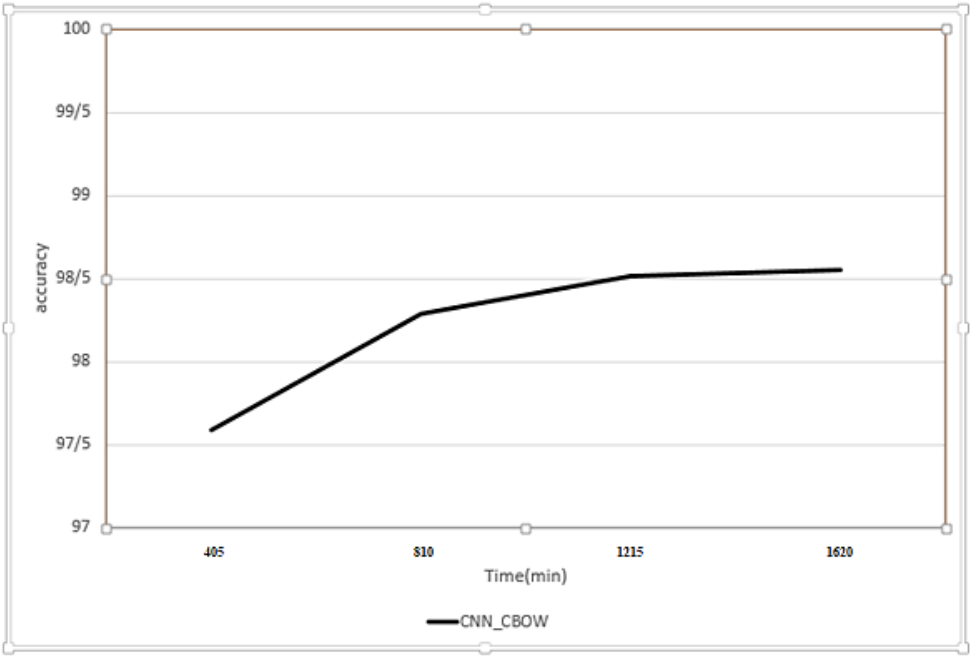

Fig. 4. Impact of Increased Training Time on Model Accuracy (using CBOW vectors)

In Figure 5, the influence of training data quantity (number of sentences) on model accuracy is presented. The highest accuracy (98.55%) was achieved when the entire training data (66,718 sentences) was utilized. The graph indicates that the effect of data quantity on accuracy improvement may potentially exceed the impact of time spent on training. By further increasing the amount of data, it is likely possible to reach an accuracy above 99%.

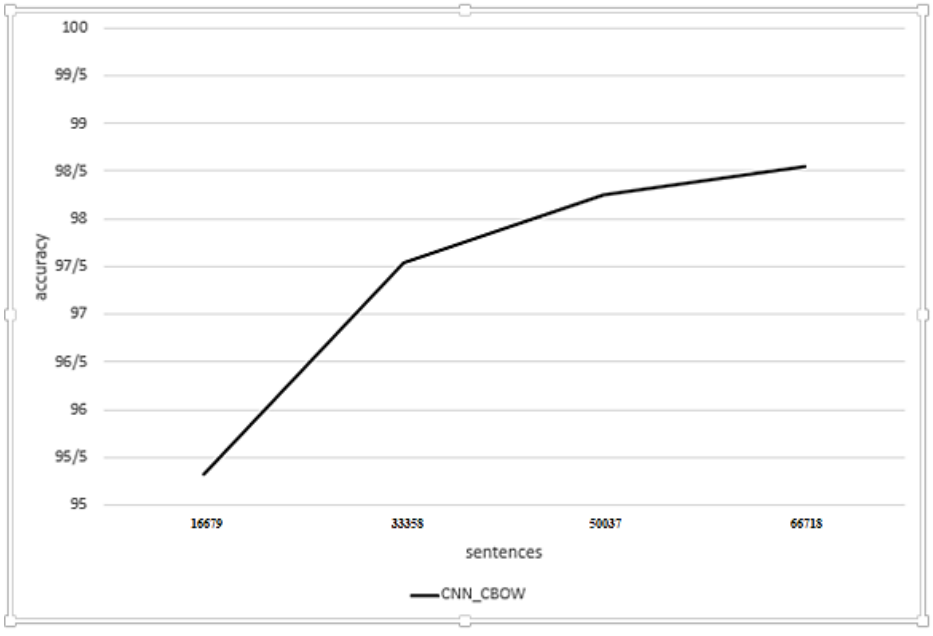

Fig. 5. Impact of Increased Training Data on Model Accuracy (using CBOW vectors)

7. CONCLUSION

This article addressed the labeling of speech components in the Persian language using a sequence-to-sequence convolutional neural network. Various embedded vectors were employed for word representation in this method. The introduced architecture managed to achieve an accuracy of 98.55% using CBOW vectors, a satisfactory accuracy level surpassing the examined methods.

Transparency Statement

The data supporting this study are available upon reasonable request to the corresponding author, subject to ethical and confidentiality considerations.

Acknowledgments

We would like to express our gratitude to all individuals who contributed to this project.

Declaration of Interest

The authors declare that they have no competing interests.

Funding

This research received no specific grant from any funding agency, commercial, or not-for-profit sectors.

REFERENCES

[1] Okhovvat, M., & Bidgoli, B. M. (2011). A hidden Markov model for Persian part-of-speech tagging. Procedia Computer Science, 3, 977–981. https://doi.org/10.1016/j.procs.2010.12.160

[2] Passban, P., Liu, Q., & Way, A. (2016). Boosting neural POS tagger for Farsi using morphological information. ACM Transactions on Asian and Low-Resource Language Information Processing (TALLIP), 16(1), 4. https://doi.org/10.1145/2934676

[3] Raja, F., et al. (2007). Evaluation of part of speech tagging on Persian text.

[4] Seraji, M., Megyesi, B., & Nivre, J. (2012). A basic language resource kit for Persian. In Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC 2012) (pp. 23–25). European Language Resources Association.

[5] Seraji, M. (2011). A statistical part-of-speech tagger for Persian. In Proceedings of NODALIDA 2011 (pp. 11–13). Riga, Latvia.

[6] Kalchbrenner, N., Grefenstette, E., & Blunsom, P. (2014). A convolutional neural network for modelling sentences. arXiv preprint arXiv:1404.2188. https://doi.org/10.3115/v1/P14-1062

[7] Kim, Y. (2014). Convolutional neural networks for sentence classification. arXiv preprint arXiv:1408.5882. https://doi.org/10.3115/v1/D14-1181

[8] Zeng, D., et al. (2014). Relation classification via convolutional deep neural network. In Proceedings of COLING.

[9] Nguyen, T. H., & Grishman, R. (2015). Relation extraction: Perspective from convolutional neural networks. In Proceedings of the HLT-NAACL Workshop on Visual Analytics (pp. 1–10). https://doi.org/10.3115/v1/W15-1506

[10] Sun, Y., et al. (2015). Modeling mention, context and entity with neural networks for entity disambiguation. In Proceedings of IJCAI.

[11] Strubell, E., et al. (2017). Fast and accurate sequence labeling with iterated dilated convolutions. arXiv preprint arXiv:1702.02098. https://doi.org/10.18653/v1/D17-1283

[12] Gehring, J., et al. (2017). Convolutional sequence to sequence learning. arXiv preprint arXiv:1705.03122.

[13] Soskek. (2017). Convolutional sequence to sequence learning (Gehring et al., 2017) by Chainer. GitHub. https://github.com/soskek/convolutional_seq2seq

[14] Chainer. (n.d.). A powerful, flexible, and intuitive framework for neural networks. https://chainer.org/

[15] Mikolov, T., et al. (2013). Distributed representations of words and phrases and their compositionality. In NIPS'13: Proceedings of the 26th International Conference on Neural Information Processing Systems (Vol. 2, pp. 3111–3119).

[16] Pennington, J., Socher, R., & Manning, C. D. (2014). GloVe: Global vectors for word representation. In Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP) (pp. 1532–1543). https://doi.org/10.3115/v1/D14-1162