Document Embedding based Supervised Methods for Turkish Text Classification

Halil İ. Çelenli[1,2], S. Talha Öztürk[1], Gürkan Şahin[1,3], Aydın Gerek[1], Murat C. Ganiz[1]
[1]Marmara Üniversitesi, Mühendislik Fakültesi, Bilgisayar Mühendisliği
[2]Kocaeli Üniversitesi, Mühendislik Fakültesi, Bilgisayar Mühendisliği
[3]Yıldız Teknik Üniversitesi, Elektrik Elektronik Fakültesi, Bilgisayar Mühendisliği
{halil.celenli, stalha, gurkan.sahin, aydin.gerek, murat.ganiz}@marmara.edu.tr

Abstract—**Following the recent increase in the amount of available data, Deep Learning has become the most popular branch of Machine Learning. This trend can also be seen in Natural Language Processing (NLP) especially since textual data can now be scraped from in World Wide Web in vast quantities and used in an unsupervised or semi-supervised manner. For this reason, Deep Learning methods are being used more frequently. In this work we devise several classification methods based on the Paragraph Vector model (a.k.a. Doc2Vec) which represents documents as vectors. These include k-Nearest Neighborhood classifier (k-NN), Support Vector Machines (SVM), Centroid Classifier (CC) that works on paragraph vectors of documents and a custom made method which uses pairwise cosine similarities between documents and class centroids as features in Doc2Vec space. Our experiments use a number of representations and classifiers combined in various ways. On the representation side the Paragraph Vector model is compared with Term Frequency (tf) and Term Frequency-Inverse Document Frequency (tf-idf) using SVM, k-NN, CC and Centroid Features Support Vector Machine (CFSVM) as classifiers.**

Keywords—Text Classification, Doc2Vec, Distributed Vector Representations, Embedding models, Paragraph Vectors

I. INTRODUCTION

In line with the rapid development of NLP technologies the number of published papers on the subject of text classification has also increased. Unstructured textual data now exists in quantities larger than ever seen before, in the form of emails, web pages, presentations, surveys, white papers, etc. spread across the World Wide Web. This abundance of textual data has accelerated Natural Language Processing research. Nowhere else is this as apparent as it is in Deep Learning based text classification methods, which tend to be more data-hungry than other Machine Learning models. The Performances of Deep Learning methods vary depending on the problem to be solved, but in many cases, especially when data is abundant, they outperform other traditional models.

Turkish, the native language of more than 80 million people, ranks highly in the list of commonly used languages worldwide [1]. Despite this, the number of Text Classification studies published on Turkish language corpuses remains low [2,3,4,5]. These works include syntactic and semantic studies.

We conduct our studies on classifying labeled datasets from news articles. Datasets formed from news articles can be considered as low noise datasets as their use of language is regular. On the other hand we also use large corpus of Turkish tweets, which are highly noisy, for the unsupervised training of vector representations.

The organization of this paper is as follows. The background and related work are stated in Section 2. In Section 3 we provide details on preprocessing and classification methods. Section 4 includes information about the experimental setup and libraries and frameworks used. Results and discussion can be found in Section 5. Finally, Section 6 includes concluding remarks.

II. BACKGROUND AND RELATED WORK

Text classification is the problem of labeling natural language text documents with labels from a predefined set of classes or categories. The reader is referred to [6] for a detailed analysis of text classification methods. For a more recent survey see [7]. A distinguishing characteristic of text classification in comparison to other classification applications is the sparseness and high dimensionality of representations. One of the most natural ways to represent text is the bag-of-words approach, in which the dimensionality of the feature space equals the size of the vocabulary (thousands to a few million), and each dimension in a document vector corresponds to a word. The so called "bag" refers to the fact that the order of the words in the text is ignored in this representation. The vector entries can vary based on which variation is used but are usually 0 in the absence of the word, leading to sparse high dimensional representations.

In a recent work of some significance regarding text classification on Turkish text, the effects of different preprocessing methods are examined [8]. They report that stopwords have a negligible effect on Turkish text, and speculate that this may be related to tf-idf term weighting methods. In addition tf-idf vectors indicate semantic similarity (in the form of cosine similarity) between texts better than the bag-of-words method [9].

Recently deep learning has become very popular and new neural network methods have been developed accordingly. One of these methods is called the paragraph vector model (a.k.a. Doc2Vec). Some of these works are listed below

- Bag-of-words vocabulary representation is compared to the Doc2Vec representation and Convolutional Neural Network (CNN) classification models on a social media platform that exchanges ideas among investors The CNN model is said to give the best result [10].
- Classification of damp-heat syndrome with data gathered from Chinese medical hospitals has been worked. Here various word representations (tf-idf, Latent Semantic Analysis (LSA), Word2Vec, Word2Vec+tf-idf and Doc2Vec) methods are compared. In addition, SVM, Decision Tree, Random Forest and k-NN are used as classifiers, The best result is obtained by combining Word2Vec + tf-idf word representation with k-NN [11].
- The authors of [12] attempt to do sentiment analysis by using Paragraph Vector model on Turkish and English twitter data. Only Distributed Bag of Words (DBOW) and Distributed Memory (DM) architectures comparisons were made. It is shown in this work that the DBOW architecture gives better results than the DM architecture.
- In a study on Turkish texts, the use of Word2Vec vectors was compared with the bag-of-words model. Word2Vec vectors have been shown to give better results than the tf-idf [13].

III. Overview of Applied Methods

A. *Preprocessing Methods*

We do not use any stemmer tool in our experiments. For preprocessing we only convert the words to lowercase and remove special characters.

B. *Classification Methods*

We use the following classifiers in our study: Support Vector Machines (SVM), k-Nearest Neighborhood (k-NN), Centroid Classifier (CC) and Centroid Feature Support Vector Machine (CFSVM). The last two methods are new classifiers discussed below. In our experiments with use the default parameters of the Scikit-learn [14] library for all classifiers except where stated otherwise.

The SVM is a well-known supervised classifier [15]. Although it is a mathematically sophisticated model, SVM achieves high classification rates in many areas. SVM is in essence a large-margin linear binary classifier. SVM works by finding a hyper-plane that separates the two sets of data points belonging to the two different classes in such a way that the distance between the separating hyper-plane and the nearest data points (a.k.a the margin) is the largest possible. These nearest points are the so called support vectors. There are several kernels that can be used to augment the SVM model. In our experiments we observe that the linear kernel performs much better than the others. Therefore, the reader should take note that all references to the SVM model in this paper refer to the linear kernel version. We use a linear kernel in favor of the default radial-basis function (RBF) kernel.

The k-NN classifier is an instance based lazy classifier that does not require training [16]. Classification is done by observing the labeled of the nearest k neighbors of the data point under consideration, and picking the most frequent. Closeness can be measured with any metric, including Euclidean distance, or similarity function, e.g. cosine similarity. k is an empirically picked hyperparameter and depends on dataset and application. The k-NN algorithm is very easy to implement, but unfortunately is computationally expensive, especially for large datasets. The k hyperparameter of the k-NN classifier was empirically chosen as 3 in place of the default value of 5.

The CC is a lazy learning instance-based classifier. To classify an unknown document, Centroid algorithm identifies the nearest class centroid in a given feature space. For our experiments we have our own implementation of the Centroid Classifier.

The CFSVM classifier is a meta-version of the SVM classifier. First, the class centroids are computed as in Centroid Classifier. After that cosine similarity is computed between each instance and each class centroid. These values form a new feature space for the instances. In this case the dimension of this the new feature space will equal the number of classes. So this can be seen as a drastic dimensionality reduction scheme in which the dimension of document vectors is reduced to the number of classes. This process is depicted in Figure 1. Finally, documents in this new feature space is fed into an SVM classifier along with their class labels represented in a column vector named *y*. A similar approach is employed in [17] with several different classifiers. This meta-classifier is also self-implemented.

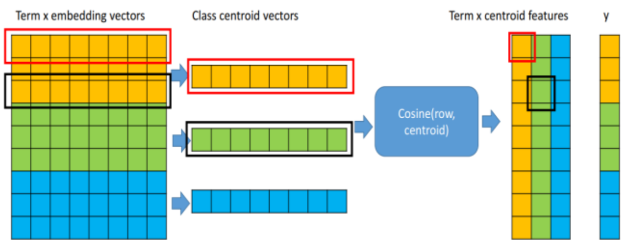

Fig.1. Illustration of the feature extraction phase of CFSVM classifier. At the end of the process dimensionality is reduced to the number of classes.

C. *The Paragraph Vector Model*

Paragraph Vector (PV) model [18] is a Deep Learning model that uses paragraph vectors to extract document relationships. It automatically performs the feature selection process itself using an artificial neural network. Artificial neural networks were used to create a model between words and documents based on both syntactic and semantic relations.

Initially a dictionary of unique words is constructed with a single pass over the corpus thus providing a unique index for each word. Each document (or so called paragraph) is also similarly indexed. Two embedding matrices, one for words, and one of paragraphs, are randomly initialized. The rows of these matrices are the vector representations for words and paragraphs respectively.

The PV model has two possible neural architectures described below. In both architectures the embeddings are trained with backpropagation and gradient descent algorithms as is usual in neural networks.

Distributed Memory (DM) is based on the neural language model in [19] which predicts the next word in a sequence of words. The sequence of words is of fixed length and sampled from a window moving over the text. It is represented as a combination of the vectors representing its constituent words. The combination operation can be a simple averaging or concatenation. This process is visualized in in Figure 2.

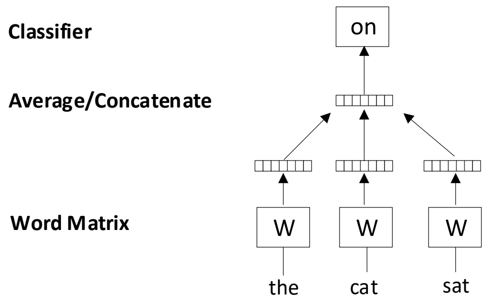

Fig. 2. The language model predicts that the word “on” will follow the sequence of words “the”, “cat” and “sat” (adapted from [18])

The DM architecture adds paragraph vectors on top of this language model by including the paragraph vectors in the combination step as seen in Figure 3. This allows each paragraph vector to represent the context in which the words occur, and thus contains semantic information on the paragraph itself.

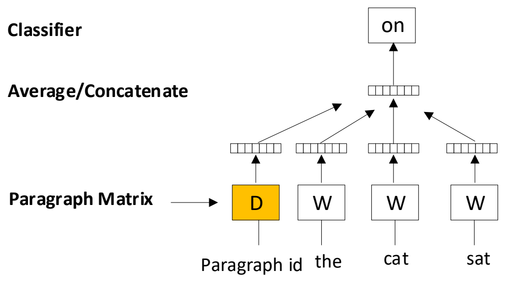

Fig. 3. Using DM , the sequence of words “the”, “cat” and “sat” are input to the model and the word “on” is expected the predicted output (adapted from [18])

The Distributed Bag of Words (DBOW) architecture is inspired by the Word2Vec Skip-gram architecture [20]. Here the paragraph vector by itself is used to predict the representative vectors of its constituent words. The DBOW architecture requires less memory than the DM architecture. Details are shown in Figure 4.

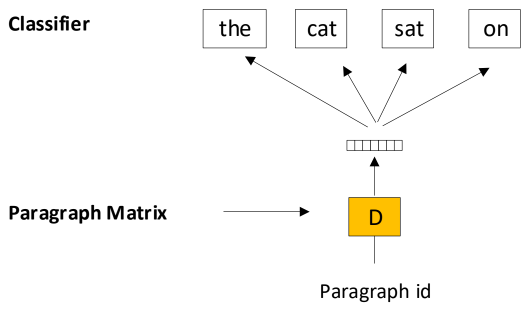

Fig. 4. DBOW architecture in which the paragraph vector is used to predict the vectors of its constituent words ” (adapted from [18]).

IV. EXPERIMENTAL SETUP

1150Haber and tweet datasets have been retrieved from the website of the Kemik Natural Language Processing group of Yıldız Technical University[1]. Among the used datasets are also Bilcol [21] and Hurriyet6c1k [8]. The characteristics, provenance, and usage details of these datasets are described below.

The tweet dataset contains 19 million untagged tweets, however for our experiments we found that it suffices to use the first 8 million tweets from this dataset.

The Bilcol dataset consists of 2 million sentences.

The 1150Haber dataset includes Turkish newspaper news. It consists of 1150 Turkish news texts in 5 classes (economy, magazine, health, political, sports) and 230 documents of each class.

The dataset Hurriyet6c1k was collected for an earlier work [22]. It includes news from 2010 to 2011 which appeared on the Turkish newspaper Hürriyet. It is contains six categories and 1000 documents for each category. Categories in this dataset are Dünya (world), ekonomi (economy), güncel (current), spor (sports), siyaset (politics), yaşam (life).

We use datasets Bilcol [21] and Turkish Tweets to train PV model. Trained models were used to classify 1150Haber and Hurriyet6c1k datasets.

Table 1 provides statistics on each dataset.

[1] www.kemik.yildiz.edu.tr/?id=28

TABLE I. THE CHARACTERSTICS OF 1150HABER, HURRIYET_6C_1K AND BILCOL [21] DATASETS

Datasets	# of terms	Avg. # of terms per paragraph
Hurriyet6c1k	18280	124.61
1150haber	11040	127.41
Bilcol [21]	30816387	14.10

In recognition of its mature and highly popular Machine Learning ecosystem, the Python language was chosen for developing our experimental codebase. We used the popular Scikit-learn library for its Machine Learning models, and Gensim [22] for its Doc2Vec (Paragraph Vector) model.

V. RESULTS AND DISCUSSION

The baseline accuracy rates with SVM and k-NN classifiers fed the tf and the tf-idf representations of the 1150Haber and Hurriyet6c1k dataset are displayed in Table 2. These accuracies are averaged over 10-fold Cross Validation.

TABLE II. CLASSIFICATION ACCURACY OF 1150HABER AND HURRIYET6C1K DATASETS

	tf		**tf-idf**	
Training Data	k-NN	SVM	k-NN	SVM
1150Haber	42	92	88	**95**
Hurriyet6c1k	39	77	72	**81**

The SVM model performs best on both datasets with tf-idf weighting.

TABLE III. CLASSIFICATION ACCURACIES OF 1150HABER AND HURRIYET6C1K DATASETS WITH PARAGRAPH VECTOR USING VARIOUS PARAMETERS

	1150Haber		**Hurriyet6c1k**	
Training Data and Parameters	k-NN	SVM	k-NN	SVM
2 million tweets, 5 epochs, DBOW	**91**	**93**	**73**	**76**
2 million tweets, DM	71	89	49	68
5 million tweets, DM	66	88	44	68
8 million tweets, DM	69	87	45	68
2 million from Bilcol [21], 5 epochs, DBOW	75	85	52	63
2 million Bilcol [21], 300 epochs, DBOW	47	81	33	62
2 million Bilcol [21], 500 epochs, DBOW	45	82	33	62

Models were trained using 2, 5 and 8 million tweets and Bilcol [21]. The Gensim Doc2Vec model hyperparameter values common to the experiments listed in Table 3 are as follows: Window_size = 15, Min_count = 1, Sampling_threshold=1e-5, Negative_size=5, Vector_size=300.

As seen on Table 3, the best result is obtained by a model trained using 2 million tweets, 5 epochs and DBOW.

We have also run experiments by varying training/test set ratios. We use two different datasets, two different term weighting methods and four different classification algorithms in order to determine the effects of different term weighting methods on Turkish text. As explained earlier, we use tf-idf and PV models to represent documents. In our experiments we used the following classifiers: SVM, k-NN, CC and CFSVM. For the k-NN model the number of neighbors is selected as 3. Paragraph vectors were trained over the Bilcol corpus [21]

Gensim Doc2Vec model hyperparameter values common to the experiments listed in Table 4, 5, 6 and 7 are as follows: Window_size = 15, Iter= 300, Sampling_threshold = 1e-5, Negative_size=5, Vector_size=300.

We run experiments with several different classifiers including k-NN, CC, CFSVM, SVM and three different representation schemes: tf, tf-idf and Paragraph Vector. We found that the paragraph vector representation produces better results compared to tf-idf representations for SVM classifiers when the amount of training data is limited. The results of SVM classifiers with two different representations are shown in the tables 4 and 5. We note that, as seen in tables 6 and 7, The k-NN classifier paired with PV representations trained on Bilcol [21] did not perform as well as the k-NN classifier paired with tf-idf representation trained on the same corpus. This is in contrast to experiments listed in Tables 2 and 3, in which the representations were trained on the tweet dataset.

TABLE IV. CLASSIFICATION ACCURACIES OF 1150HABER DATASET WITH CC, CFSVM, SVM

Training Set %	**CC + PV**	**CFSVM + PV**	**SVM + tf-idf**
1	**65,93**	59,82	22,67
3	84,17	**84,75**	72,48
5	86,85	**88,20**	81,70
10	89,77	**90,75**	89,62
30	90,57	91,73	**93,42**
50	90,90	92,16	**93,91**
70	90,78	92,12	**94,46**
80	91,26	92,57	**94,65**
90	90,96	91,83	**94,43**
99	91,67	91,67	**94,17**

TABLE V. CLASIFICATION ACCURACIES OF HURRIYET6C1K DATASET WITH CC, CFSVM, SVM

Training Set %	CC + PV	CFSVM + PV	SVM + tf-idf
1	**60,64**	60,27	57,19
3	**66,81**	66,78	66,45
5	68,25	68,37	**69,57**
10	69,34	69,61	**73,53**
30	70,26	70,72	**78,11**
50	70,61	71,45	**79,89**
70	70,49	71,23	**80,96**
80	70,52	70,79	**81,35**
90	70,98	71,95	**81,55**
99	69,50	70,50	**83,50**

TABLE VI. CLASIFICATION ACCURACIES OF 1150HABER DATASET WITH K-NN

Training Set Size %	k-NN + PV	k-NN + tf-idf
1	29,23	**42,89**
3	45,09	**62,11**
5	52,34	**70,30**
10	58,88	**78,97**
30	64,11	**86,41**
50	66,78	**87,88**
70	71,33	**88,90**
80	72,17	**89,87**
90	71,83	**89,48**
99	70,83	**88,33**

TABLE VII. CLASIFICATION OF ACCURACIES HURRIYET6C1K DATASET WITH K-NN

Training Set Size %	k-NN + PV	k-NN + tf-idf
1	30,08	**47,22**
3	38,72	**55,94**
5	41,06	**59,60**
10	46,43	**62,99**
30	52,89	**67,95**
50	55,87	**70,03**
70	56,79	**71,96**
80	57,83	**71,66**
90	57,90	**71,82**
99	58,83	**75,50**

Here, using the tf in the 1150Haber dataset, both k-NN and SVM values were improved by 10-fold Cross Validation. In the Hurriyet6c1k dataset, only the classification of the k-NN classifier was developed using tf. In addition higher performance has been achieved in k-NN classifiers using tf-idf in datasets in contrast to experiments described in Table 3.

As seen in Tables 4 and 5, Centroid Classifier paired with paragraph vector outperforms the other two models when the size of the training data on which the classifier is trained is overly scarce (1-3%). On the other hand, as seen in Table 4, the CFSVM model outperforms the others when the training data ratio is 3-10%, demonstrating that it works well when the training data is limited but not overly scarce. SVM + tfidf performs best when more data is available (10%+).

VI. CONCLUSION

In this study, we developed a classifier based on class centroid similarity features and a centroid based classifier, both employing the Paragraph Vector model (a.k.a. Doc2Vec) which represents documents as vectors. We conducted several experiments using a number of representations and classifiers combined in various ways. Our results show that, in situations where data is scarce, the best accuracy is obtained by classifiers paired with document embedding vectors which are trained using Distributed Bag of Words (DBOW) architecture for 5 epochs. As an interesting observation, training the Doc2Vec document embeddings model using more epochs decreases classification accuracy. We suspect that this is a form of overfitting of the document embeddings model that can only be seen when they are used for supervised classification. As far as we know, this has not been mentioned in earlier literature on the subject of neural embeddings. We conclude that the paragraph vector representation, also known as Doc2Vec, produces better results compared to tf-idf representations for SVM classifiers when the amount of training data is limited.

For future work, we are planning to modify document embedding models that are trained using unlabeled data to make use of labeled data for text classification purposes.

ACKNOWLEDGEMENTS

This work is supported in part by The Scientific and Technological Research Council of Turkey (TÜBİTAK) grant number 116E047. Points of view in this document are those of the authors and do not necessarily represent the official position or policies of the TÜBİTAK. The authors would like to thank Murat Çetiner, Hakan Bilici, Furkan Şahin, Onurcan Karabulut, and Bekir Çankaya for their help in the project.

REFERENCES

[1] R. Blumberg, S. Atre, "The Problem With Unstructured Data", Information Management Magazine, February 2003

[2] A. Deniz and H. E. Kiziloz, "Effects of various preprocessing techniques to Turkish text categorization using n-gram features," in *2017 International Conference on Computer Science and Engineering (UBMK)*, 2017, pp. 655–660.

[3] D. Kılınç, A. Özçift, F. Bozyigit, P. Yıldırım, F. Yücalar, and E. Borandag, "TTC-3600: A new benchmark dataset for Turkish text categorization," *J. Inf. Sci.*, vol. 43, no. 2, pp. 174–185, Apr. 2017.

[4] Z. H. Kilimci, S. Akyokus, and S. I. Omurca, “The effectiveness of homogenous ensemble classifiers for Turkish and English texts,” in *2016 International Symposium on INnovations in Intelligent SysTems and Applications (INISTA)*, 2016, pp. 1–7.

[5] Z. H. Kilimci and M. C. Ganiz, “Evaluation of classification models for language processing,” in *2015 International Symposium on Innovations in Intelligent SysTems and Applications (INISTA)*, 2015, pp. 1–8.

[6] C. C. Aggarwal and C. Zhai, “A Survey of Text Classification Algorithms,” in *Mining Text Data*, Boston, MA: Springer US, 2012, pp. 163–222.

[7] M. Allahyari *et al.*, “A Brief Survey of Text Mining: Classification, Clustering and Extraction Techniques.” p. 13, 2017.

[8] D. Torunoglu, E. Cakirman, M. C. Ganiz, S. Akyokus, and M. Z. Gurbuz, “Analysis of preprocessing methods on classification of Turkish texts,” in *2011 International Symposium on Innovations in Intelligent Systems and Applications*, 2011, pp. 112–117.

[9] J. A. Bullinaria and J. P. Levy, “Extracting semantic representations from word co-occurrence statistics: A computational study,” *Behav. Res. Methods*, vol. 39, no. 3, pp. 510–526, Aug. 2007.

[10] S. Sohangir and D. Wang, “Finding Expert Authors in Financial Forum Using Deep Learning Methods,” in *2018 Second IEEE International Conference on Robotic Computing (IRC)*, 2018, pp. 399–402.

[11] Wei Zhu, Wei Zhang, Guo-Zheng Li, Chong He, and Lei Zhang, “A study of damp-heat syndrome classification using Word2vec and TF-IDF,” in *2016 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*, 2016, pp. 1415–1420.

[12] M. Bilgin and İ. F. Şentürk, “Sentiment analysis on Twitter data with semi-supervised Doc2Vec,” in *2017 International Conference on Computer Science and Engineering (UBMK)*, 2017, pp. 661–666.

[13] G. Şahin, “Turkish document classification based on Word2Vec and SVM classifier,” in *2017 25th Signal Processing and Communications Applications Conference (SIU)*, 2017, pp. 1–4.

[14] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot an Edouard Duchesnay Pedregosa, al Matthieu Brucher, M. Perrot matthieuperrot, and cea Edouard Duchesnay, “Scikit-learn: Machine Learning in Python,” *J. Mach. Learn. Res.*, vol. 12, pp. 2825–2830, 2011.

[15] J. Platt: Fast Training Of Support Vector Machines Using Sequential Minimal Optimization. In B. Schoelkopf And C. Burges And A. Smola, Editors, Advances In Kernel Methods - Support Vector Learning, 1998

[16] D.W. Aha, D. Kibler, And M.K. Albert, “Instancebased Learning Algorithms.”, Machine Learning, 6 , Pp. 37-66, 1991

[17] M. Taşpınar, M. C. Ganiz, and T. Acarman, “A Feature Based Simple Machine Learning Approach with Word Embeddings to Named Entity Recognition on Tweets,” Springer, Cham, 2017, pp. 254–259.

[18] Q. Le and T. Mikolov, “Distributed Representations of Sentences and Documents,” p. 9, 2014.

[19] Y. Bengio *et al.*, “A Neural Probabilistic Language Model,” *J. Mach. Learn. Res.*, vol. 3, pp. 1137–1155, 2003

[20] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean, “Distributed Representations of Words and Phrases and their Compositionality,” 2013.

[21] C. Özköse and M. F. Amasyalı, “Tümce Öğelerinden Hayat Bilgisi Çıkarımı,” *Türkiye Bilişim Vakfı Bilgi. Bilim. ve Mühendisliği Derg.*, vol. 5, no. 2, Jun. 2012.

[22] R. Rehurek and P. Sojka, “Software Framework for Topic Modelling with Large Corpora,” *Proc. Lr. 2010 Work. NEW CHALLENGES NLP Fram.*, pp. 45--50, 201