Turkish Journal of Electrical Engineering & Computer Sciences
http://journals.tubitak.gov.tr/elektrik/
Research Article
Turk J Elec Eng & Comp Sci
(2023) 31: 581 – 595

doi:10.55730/1300-0632.4003

Deep learning-based Turkish spelling error detection with a multi-class false positive reduction model

Burak AYTAN[1,*], **C. Okan SAKAR**[2]
[1]Department of Computer Engineering, Bahcesehir University, İstanbul, Turkey,
[2]College of Engineering and Technology, American University of the Middle East, Kuwait

Received: 28.09.2022 • **Accepted/Published Online:** 15.03.2023 • **Final Version:** 28.05.2023

Abstract: Spell checking and correction is an important step in the text normalization process. These tasks are more challenging in agglutinative languages such as Turkish since many words can be derived from the root word by combining many suffixes. In this study, we propose a two-step deep learning-based model for misspelled word detection in the Turkish language. A false positive reduction model is integrated into the system to reduce the false positive predictions originating from the use of foreign words and abbreviations that are commonly used in Internet sharing platforms. For this purpose, we create a multi-class dataset by developing a mobile application for labeling. We compare the effect of using different types of tokenizers including character-based, syllable-based, and byte-pair encoding (BPE) approaches together with Long Short-Term Memory (LSTM) and Bi-directional LSTM (Bi-LSTM) networks. The findings show that the proposed Bi-LSTM-based model with the BPE tokenizer is superior to the benchmarking methods. The results also indicate that the false positive reduction step significantly increased the precision of the base detection model in exchange for a comparably less drop in its recall.

Key words: Text normalization, spell checker, tokenizers, long short-term memory, agglutinative languages

1. Introduction

With the increasing use of the internet, there is an increasing need in building reliable natural language processing (NLP) tools. The NLP tasks use the text produced by the users as input for different purposes such as Chatbot development [1], question answering [2], automatic speech recognition [3], speech processing [4], sentiment analysis [5, 6], text classification [7], and machine translation [8]. The companies develop software and applications based on these NLP tasks to provide a better service to their customers and increase their market share.

The massive amount of data produced by the internet community provides an important opportunity for building highly accurate NLP tools. However, the user text retrieved from internet platforms is written with a less standardized language and includes a high rate of misspelled words which worsens the performance of the models developed for the related NLP tasks [9–11]. Therefore, text normalization is an important step to achieve better models and also enables the nonnative readers of the language better understand the content.

Spelling error detection and correction is a critical step for text normalization. In this study, we focus on spelling error detection problems in the Turkish language. For most languages such as English, French, and German, spell checking can be addressed with rule-based or dictionary-based approaches; however, for

*Correspondence: burak.aytan@bahcesehir.edu.tr

the agglutinative languages such as Turkish, Finnish, and Hungarian, in which the meaning of the words is determined by the combination of the root of a word and morpheme, spell checking also requires morphological analysis [12–15]. In these types of languages, the meaning of the word changes with the suffixes combined to the root.

The spelling errors are generally divided into two categories as nonword errors (NWE) and real word errors (RWE) [16, 17]. If the misspelled text is a word that exists in the language, it is called a real word error (RWE), otherwise, it is a nonword error (NWE). For example, the word 'fun' in English might be misspelled as 'gun' or 'vun', in this case, the first is an RWE, while the second is an NWE [16]. These types of errors are more common in English and it is challenging to detect such errors as it requires semantic and context analysis. In our study, we focus on the detection of NWE considering that it is the more common error type in the Turkish language.

The motivation of this study is to propose a highly accurate spelling error detection model for the Turkish language. Due to the use of many suffixes in agglutinative languages, the methods based on the use of predetermined rules do not perform well for the related NLP tasks. These models also produce false positive predictions for the foreign words and abbreviations which are commonly used in Internet sharing platforms. The contributions of this study are as follows:

- Building a deep learning-based model for spelling detection with a high detection rate.
- Developing a mobile application to create a labeled dataset including commonly used foreign and abbreviations in the Turkish web community.
- Integrating a false positive reduction model to reduce the number of false positive detections originating from the use of foreign words, abbreviations, or cases that the base detection model cannot detect.
- Comparing the effect of using different types of tokenizers in spelling error detection problems for the Turkish language.

The rest of this study is organized as follows. In Section 2, the overview of the studies that focus on the detection of spelling errors is given. Section 3 provides a description of the datasets used in this study. The details of the proposed deep learning-based detection models are presented in Section 4. Section 5 gives the experimental results and the related discussion. Section 6 summarizes and concludes the study.

2. Related work

The implementation of applications for the misspelled word detection task relies entirely on external sources such as word dictionaries or tree structures. In general, these methods work well for languages with less complex word formations such as English [18]. However, it is difficult to create comprehensive sources for agglutinative languages like Turkish, which have a high level of morphological complexity. Different methods based on predefined rules, shortest distance algorithms, text similarity metrics, and deep learning has been used in the related studies. In this section, we give an overview of the studies that focus on spelling error detection in the Turkish language.

Solak and Oflazer [13] proposed a rule-based method that consists of three steps. These steps are root determination, morphophonemic checks, and morphological parsing. The main idea is first to find the root of the given text based on a maximal match algorithm. In this algorithm, first, the whole word is searched

in the dictionary which contains only the roots and some irregular stems. If found, it is assumed that the word has no suffixes and therefore does not need to be parsed. Otherwise, one letter is removed from the rightmost of the word and the resulting substring is compared with the whole word again. This process continues until the root is found. After the root of the word is found, the remaining part is considered as a suffix and analyzed morphologically. While performing morphological analysis, the grammar rules of Turkish are determined, and manually defined algorithms are applied. Another approach to morphological analysis is the dynamic programming-based search algorithms which are applied after the root of the word is found [19].

Akın and Akın [20] also proposed a rule-based method that uses a dictionary. The dictionary consists of the roots of words and their various forms. While finding the root of the word, the candidate root is found first and then the searching process continues with the application of the Direct Acyclic Word Graph (DAWG) algorithm [21]. In the morphological parser section, the algorithm continues to add possible suffixes to the root until there are no additional alternatives. The proposed spell checker uses a morphological parser and Damerau-Levenshtein Edit Distance Algorithm [22]. Another rule-based method consisting of seven normalization layers with proper noun detection and vowel restoration operations was proposed by Eryigit and Torunoglu [23]. This method generates a single candidate word for the misspelled word. Each layer is designed to address specific types of errors that appear in social media texts. In this method, the abbreviation list from TDK (Turkish Language Institution) and some foreign words were also taken into consideration.

In one of the more recent studies, Colakoglu et al. [24] designed a machine translation approach for Turkish text normalization. In this study, they used both statistical machine translation and neural machine translation methods. A 6-g character-level language model, KenLM [25], was used in the statistical machine translation part, and an encoder-decoder architecture, OpenNMT [26], was used in the neural machine translation part. In another study, Buyuk [27] proposed a sequence-to-sequence model for Turkish normalization. The seq2seq model was an LSTM-based model and a character-based tokenizer was used by separating words into letters. The contribution of this study compared to the other seq2seq models was mainly in the tokenization part. For this task, the first three consonants of nonspelling words were fed as input to the seq2seq model.

Safaya et al. [28] proposed a Hunspell-based [29] method for the Turkish language. Hunspell is a kind of spell checker originally designed for the Hungarian language. It has been applied to many languages with rich morphology, complex word combinations, and character encoding [29]. In another recent study [30], a morphological analyzer was used to detect incorrect words, and words that could not be parsed into a morpheme sequence by this analyzer were defined as nonstandard. Afterwards, all the words defined as nonstandard were passed by the Turkish entity recognizer and the recognized words were eliminated.

3. Dataset description

In this section, we provide a description of the training and test datasets used in the experiments. We also give detailed information about the functions used to create misspelled words.

3.1. Training data set

For the training and testing of the detection models, several datasets from different sources were used in our study. The dataset including the correctly spelled words was created using a dictionary obtained from TDK (Turkish Language Institution). An extended version of this dataset including person and tense suffixes was also created. Then, proper nouns including person and location names were added to this data summing up to

1 million words.

The dataset obtained from TDK consists of nontypo words. Therefore, we applied some operations to create misspelled words from the correct words. The functions used for this purpose are shown in Table 1. As a result, a dataset consisting of 1 million correctly and 7 million incorrectly spelled words were obtained. The main detection model was built using this dataset.

Table 1. Functions developed to create the misspelled words for model training.

Function name	**Detail**	**Example**
Remove vowel	Some or all of the vowels in the words are deleted	“merhaba”→“merhba”
Add vowel	Some vowels in the word are duplicated	“geliyorum”→“geeliyorum”
Asciify	Turkish characters are converted to English characters	“ağlıyorum”→“agliyorum”
Changing two characters	Randomly changing the places of two consecutive characters	“kelime”→“keilme”
Adding last character	Duplicating the last character of the word	“erzurum”→“erzurumm”
Known mistakes	The most common typos in Turkish	”seviyorum”→“seviyom”

The samples that are classified as misspelled words are considered as positive predictions in our study. The false positive reduction model was designed to further reduce the false positives predictions of the base models which are the correctly spelled words that are labeled as misspelled. To train this model, first the base detection model was applied on data of size 38 GB obtained from Turkish Wikipedia, Turkish OSCAR, and some news sites. The words classified as misspelled (positive predictions) at this step were sorted according to the frequency of occurrence from largest to smallest. Then, a mobile application has been developed to review the most frequent words with the aim of approving and correcting the predictions of the detection model shown in Figure 1. The words in this list have been categorized under four categories during this labeling process: (1) correctly classified as positive (misspelled words that are also predicted as misspelled), (2) incorrectly classified as positive (correctly spelled words that are predicted as misspelled), (3) abbreviation, and (4) foreign word. The number of words reviewed in this stage was 10,000, 35% of which were misspelled, 30% foreign, 20% abbreviations, and 15% correctly spelled.

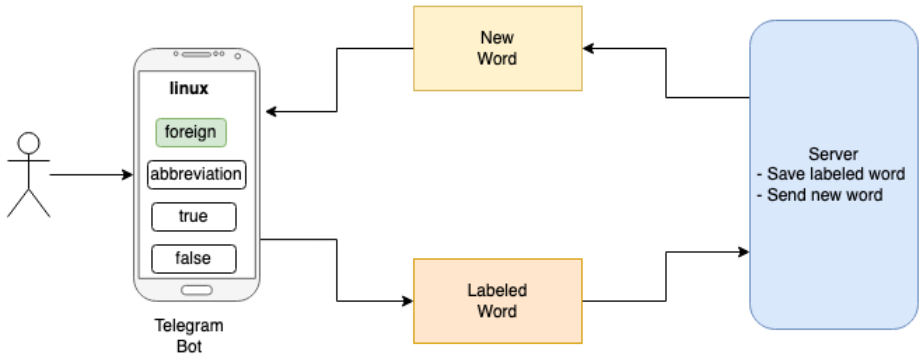


Figure 1. Mobile application developed for word-based labeling with the aim of improving the detection ability of the system.

3.2. Test data set

The use of a classical cross-validation procedure in which the test set is extracted from the original set may result in biased results in our study as we generated the misspelled words with some predefined functions in an artificial way. For this reason, we manually tagged the words retrieved from a microblogging site, Twitter, and generated test data consisting of user text. We obtained 2422 words from the retrieved content of which 1974 was spelled correctly and 448 were misspelled. The detection system with and without the false positive reduction model was built using the training set and the obtained model was applied to this test set consisting of real user text.

4. Proposed methods

In this study, we propose two spelling error detection models. The first model is the base detection model classifying each word as correctly spelled or misspelled. The second model includes a further false positive reduction model applied to the words classified as misspelled by the base detection model. The false positive reduction model was designed as a multi-class classification problem assigning each word to one of the true positive, false positive, foreign, and abbreviation classes. In this section, first, we give an overview of the tokenizers used for both of the detection models. Then, we give the details of the base detection model, false positive reduction model, and the proposed combined detection model.

4.1. Tokenizers

Tokenization is the process of breaking a sentence, paragraph, or text into words or smaller parts [31]. It is a crucial step in the construction of a highly accurate spelling error detection model, particularly for agglutinative languages. Each piece formed at the end of the transaction is called a token [31]. In our study, we implemented three types of tokenizers, the details of which can be seen in Table 2. Employing a character-based tokenizer involves the segmentation of words into individual characters, whereby each character constitutes a token. Alternatively, our second approach involved the use of a syllable-based tokenizer, wherein words are partitioned into syllables and each syllable is subsequently utilized as a token, as exemplified by the information presented in Table 2.

Table 2. An overview of the tokenizers used for the tokenization process in the detection models.

Tokenizer	Detail	Example
Character based tokenizer	each character of the word is used as a token	“istanbul”→ [’i’,’s’,’t’,’a’,’n’,’b’,’u’,’l’]
Syllable based tokenizer	words are divided into syllables and each syllable is used as a token	“konuşuyorum”→ [’ko’,’nu’,’şu’,’yo’,’rum’]
Byte-Pair encoding (BPE) tokenizer	this tokenizer creates tokens by looking at the frequency of the character groups in the corpus	“sabahlayabilmek”→ [’sabah’,’layabilme’,’k’]

In conjunction with the character and syllable-based tokenizers, we incorporated a byte-pair encoder (BPE) tokenizer [32] into our methodology, a technique commonly utilized in language models such as BERT (Bidirectional Encoder Representations from Transformers). This approach involves the generation of tokens by evaluating the frequency of character groupings present within the corpus, as outlined in Table 2. Through the implementation of the BPE tokenizer, frequently utilized words are categorized as tokens, and any suffixes

may be subsequently processed as distinct tokens. The role of the corpus is fundamental to the success of this method, as the BPE tokenizer determines the tokens based on the frequency of character groupings observed within the corpus. Accordingly, we employed the BPE tokenizer utilizing two distinct corpora. The initial iteration was created employing solely correctly spelled words, whereas the second iteration was formulated using a more expansive corpus that incorporated both correctly and misspelled words. The results obtained from both iterations are detailed within the results section. Figure 2 shows the distribution of token lengths obtained with both the BPE and syllable-based tokenizers.

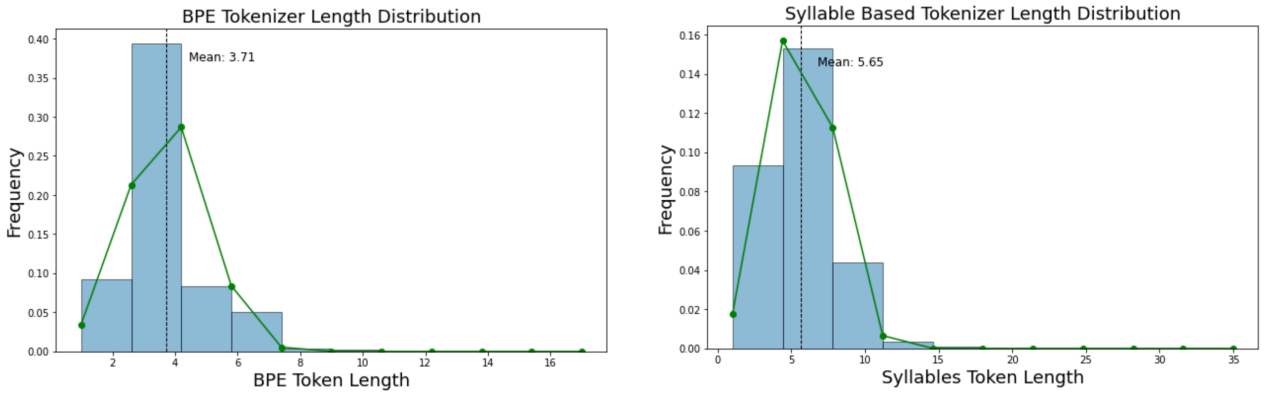


Figure 2. Tokenizer-based token length distribution.

4.2. Base detection model

The base detection model shown in Figure 3 was designed as a binary classification problem consisting of two hidden Bi-LSTM layers. This model takes a word as an input and gives the output whether the word is written correctly. The recurrent connections of the LSTM architecture enable the model to evaluate the previous character or word parts together with the next ones in the detection of spelling errors. The most important difference between LSTM and transformer architectures is the way they handle long-range dependencies. The transformer architecture utilizes self-attention mechanisms to explicitly model long-range dependencies, making it better suited for handling long sequences compared to LSTM [33, 34]. However, as the proposed spelling error detection model is word-based, its input does not necessitate the handling of extensive sequences. Specifically, as demonstrated in Figure 2, the mean token length associated with BPE and syllable-based tokenizers is 3.71 and 5.65, respectively, with 99% of token lengths being less than 9 for BPE and less than 15 for syllable-based tokenizer. Hence, we choose an LSTM-based model rather than a transformer-based one in the proposed detection model.

As seen in Figure 3, the architecture of the proposed model consists of two hidden layers with 512 LSTM nodes in each one. The model takes the whole word as input and then separates it into tokens by the use of the tokenizers given in Table 2. This process is followed by an embedding layer including 300 nodes. The obtained representations are fed to two hidden layers each one consisting of 512 Bi-LSTM nodes. The fully connected layer that maps the hidden layer representations to the binary outputs consists of 1024 nodes. This model was tested with different tokenizers, the corpus used for the tokenization, and with the replacement of the Bi-LSTM layers [35] with LSTM [36].

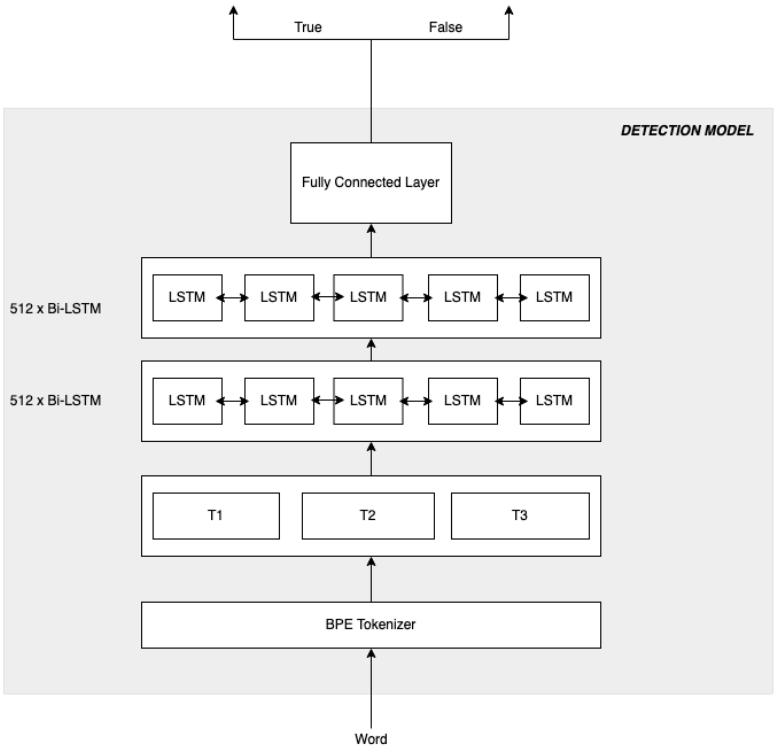


Figure 3. Architecture of the Bi-LSTM based spelling error detection model.

4.3. False positive reduction model

The base detection model was trained using Turkish words and the distorted versions of these words were obtained using the functions given in Table 1. Since these words were retrieved from a dictionary for the training of the base detection model, the corpus generated in this way does not contain foreign words and abbreviations that are frequently used in Turkish. Therefore, foreign words and abbreviations are labeled as misspelled words by the models which results in an increase in the error rate of the models. In this study, a false positive reduction model was trained to reduce the number of false positive predictions originating from such detections. The architecture of the false positive reduction model along with the base detection model is shown in Figure 4.

The false positive reduction model is trained only with the character-based tokenizer since the abbreviations and foreign words are not in the word pool we generated based on the dictionary. These words are also not suitable for Turkish syllable structure. As seen in Figure 4, the character-based tokenizer is followed by two hidden layers. Each hidden layer has 512 Bi-LSTM nodes. The last layer is a fully connected layer mapping the obtained representations to one of the four classes.

4.4. Proposed combined model

The proposed word checker model shown in Figure 4 consists of the combination of two different models. First, the words are separated into tokens with the related tokenizer and the base detection model that addresses a binary classification problem is applied to determine the correctly spelled and misspelled words. The experiments

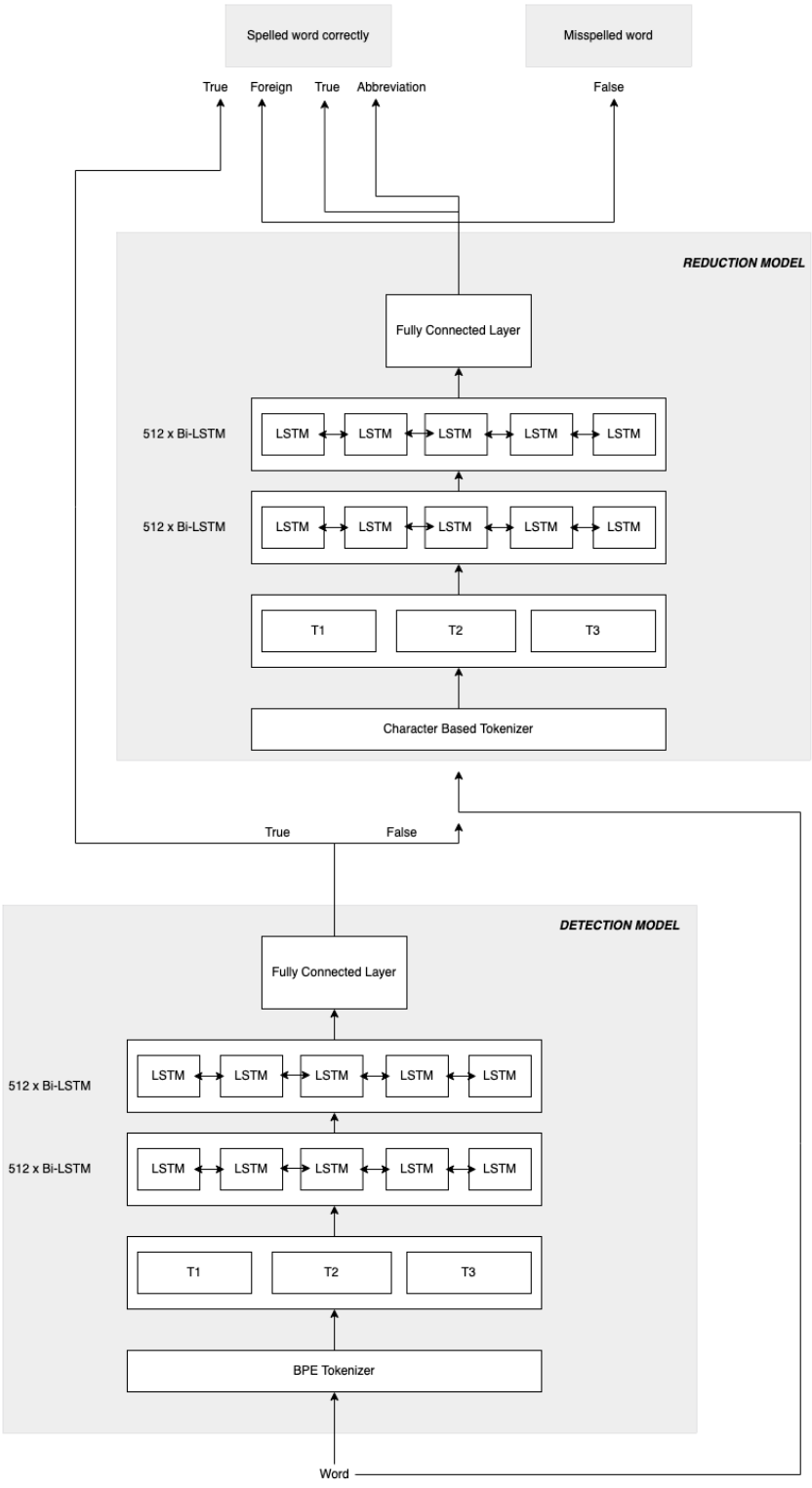


Figure 4. Architecture of the proposed spelling error detection model

detailed in Section 5 showed that the base detection model achieved the best results with 512 Bi-LSTM nodes and two hidden layers which were also used in the combined model. The words that are classified as correctly spelled are eliminated from the dataset. The remaining words that are labeled as misspelled by the base detection model are fed to the false positive reduction model which was detailed in Section 4.3. The purpose of this model is to recheck the words that were marked as misspelled by the first model and hence reduce the false positive rate. Finally, the model produces one of the four predictions, three of which indicate that the word does not contain a spelling error.

5. Results and discussion

In this section, first, we present a detailed evaluation of the proposed model with various tokenizers and neural network architectures. Then, we give the comparative results with the previous related studies that address spelling error detection problems in the Turkish language.

5.1. Evaluation of the proposed model

The proposed model is evaluated with different combinations of tokenizers and neural network architectures. We used character-based, syllable-based, and BPE tokenizers with different corpus sizes in the tokenization layer. This layer was followed by two different neural network architectures, LSTM and Bi-LSTM, each one with one, two, and three hidden layers. Each hidden layer consists of 512 nodes.

Table 3 shows the results in terms of accuracy and F1 score metrics. The initial experiments using LSTM architecture for training showed that word piece-based tokenization yields a higher success rate in the detection of both correct and misspelled words. As seen in Table 3, F1 score of 0.65 obtained using LSTM with character-based tokenization increased to 0.71 when the tokenization is performed using the syllable-based method. Therefore, the experiments with the Bi-LSTM architecture were performed using only syllable-based and BPE tokenization methods.

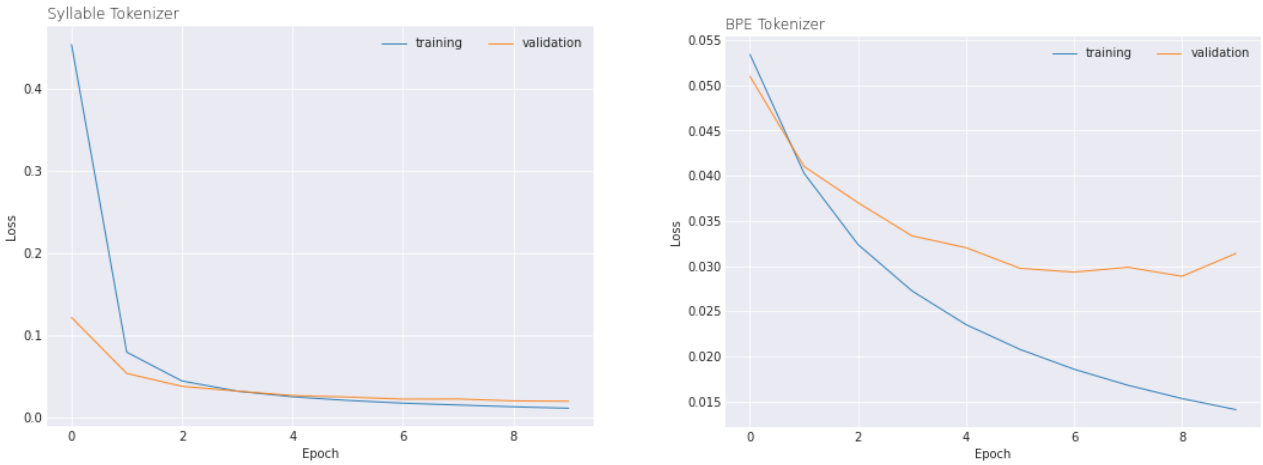
Table 3. Results obtained on test set with different combinations of tokenizers and neural network architectures.

Model name	Tokenizer	Tokenizer corpus	Hidden layer	Correct words accuracy	Misspelled words accuracy	Overall accuracy	F1 score
512 LSTM	Character	All words	2	90%	70%	86%	0.65
512 LSTM	Syllable	All words	2	92%	75%	88%	0.71
512 LSTM	BPE	Correct words	1	95%	84%	93%	0.8
512 LSTM	BPE	Correct words	2	95%	88%	94%	0.8
512 LSTM	BPE	Correct words	3	96%	88%	94%	0.84
512 Bi-LSTM	BPE	All words	2	96%	90%	94%	0.86
512 Bi-LSTM	Syllable	Correct words	2	92%	90%	91%	0.79
512 Bi-LSTM	BPE	Correct words	1	96%	90%	95%	0.86
512 Bi-LSTM	BPE	Correct words	2	96%**	92%**	95%**	0.87**
512 Bi-LSTM	BPE	Correct words	3	96%	91%	95%	0.87

As seen in Table 3, the success rate of the proposed model with the syllable-based tokenization significantly increased when LSTM architecture was replaced with Bi-LSTM architecture for the training of the detection model. Therefore, we further trained Bi-LSTM architecture with BPE tokenization. The results showed that tokenization with BPE yielded higher accuracy and F1 score than syllable-based tokenization. We also changed the corpus used for BPE-based tokenization and observed that using only the correct words as the tokenizer

corpus gave slightly higher accuracy and F1 score compared to the tokenizer corpus built using all words. As seen in Table 3, the highest accuracy of 95% and F1 score of 0.87 were achieved using BPE and only correct words for tokenization and Bi-LSTM architecture with 2 hidden layers for training. In Figure 5, the training and validation loss graphs of the Bi-LSTM models on the training and validation sets obtained syllable-based tokenizer and BPE tokenizer are shown. As seen, BPE tokenizer gave lower loss values on both training and validation sets.

In Figure 6, the advantage of using the BPE tokenizer with only correct words for the tokenization process over its alternatives is demonstrated over two exemplary Turkish words. In the first example, the misspelled word "izliyiceler", which should be spelled as "izleyiciler (audiences)", has been labeled as a correctly spelled word by the character-based and syllable-based models. The reason for this false negative prediction is that the misspelled form "izliyiceler" is suitable for the harmony of the language in terms of the use of letters and syllables. On the other hand, as seen in Figure 6, the misspelling in this word has been successfully detected by the BPE-based model since the BPE tokenizer performs a kind of dictionary-based control. In the second example, the misspelled word "herkez", which should be spelled as "herkes (everyone)", has been labeled as a correctly spelled word by the character-based and syllable-based models. As seen, this common mistake in the Turkish language was successfully detected by the model that is based on the tokenizer.



(a) Syllable-based tokenizer (b) BPE-based tokenizer

Figure 5. Loss graphs of the Bi-LSTM models on the training and validation sets

5.2. Comparison to the related studies

In this section, we compare the Turkish language spelling error detection ability of the proposed models with the related works. We also applied two state-of-the-art transformer-based language models, BERT and RoBERTa (Robustly Optimized BERT Pretraining Approach) on the same dataset for comparison purposes. For this purpose, we modified the BERTurk[37] and RoBERTaTurk[7] architectures, which are pretrained Turkish language models based on the BERT and RoBERTa, respectively, by adding task-specific layers on top of them for detecting spelling errors in Turkish text. These models have been specifically trained on Turkish language data and are therefore able to perform NLP tasks on Turkish text with a high degree of accuracy.

Table 4 presents the comparative results in terms of precision, recall, F1 score, and overall accuracy metrics. We applied both versions of the proposed model, the base detection model and the combined model.

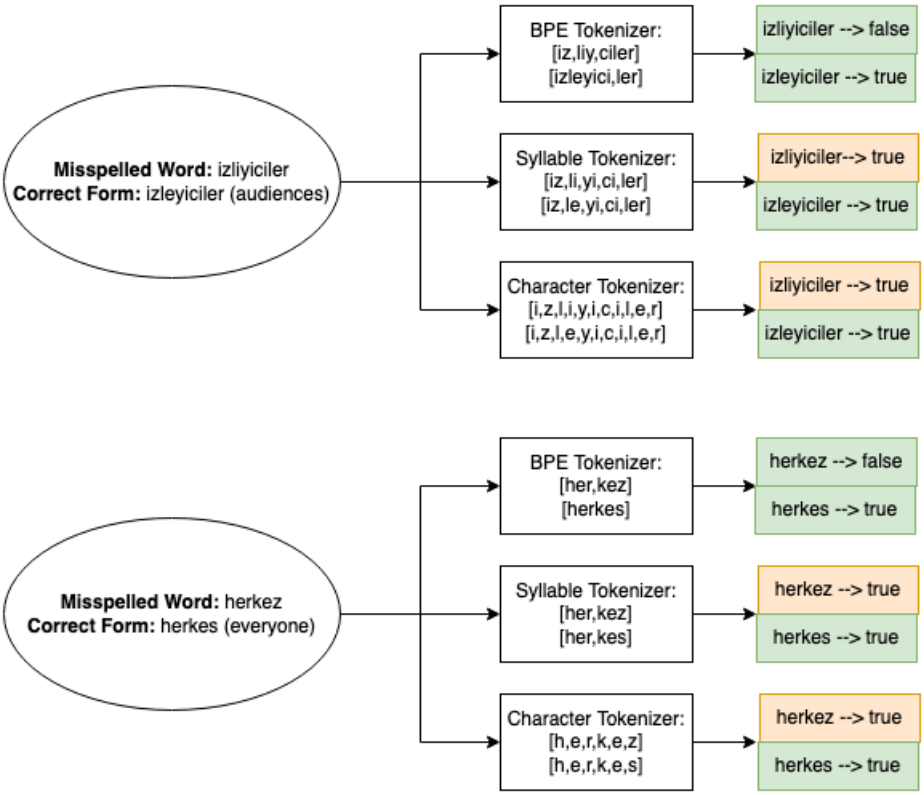


Figure 6. Predicted labels for two exemplaires of misspelled words by the neural network models that use different tokenizers

As detailed in Section 4, the combined model includes an additional step, the false positive reduction model applied only on the samples labeled as misspelled words by the base spelling error detection model. We should also note that the models were applied on the same test set described in Section 3 consisting of 2422 words of which 1974 were spelled correctly (negative) and 448 were misspelled (positive).

As seen in Table 4, the highest F1 score and overall accuracy with 0.91% and 96.6%, respectively, were achieved by the proposed combined model. It was also observed that while the highest precision was also obtained with this model, the model used by the Turkish spelling corrector available in Microsoft Office software gave higher recall with comparably very low precision. Similarly, Zemberek, which is a commonly used NLP library in the Turkish language, achieved a higher recall than both of the proposed models. However, it is seen that the precision of this model was lower than all models. On the other hand, the results showed that the proposed combined model yielded a very balanced precision and recall while achieving the highest precision. The fine-tuned transformer-based models, BERTurk and RoBERTaTurk, gave 0.83 and 0.84 F1 scores, respectively.

The comparison of the base detection model and combined model seen in Table 4 indicated that an increase of 0.9 points in precision is achieved against a decrease of 0.3 points in recall with the application of the false positive reduction model. Accordingly, the F1 score and overall accuracy obtained with the base detection

Table 4. Comparison of the existing and proposed models on the test set.

Method	Precision	Recall	F1 Score	Accuracy
Zemberek [20]	0.74	0.94	0.83	92.8%
Microsoft Office	0.78	**0.95**	0.86	94.3%
Eryigit et al [23]	0.76	0.83	0.80	92.1%
Colakoglu et al [24]	0.89	0.86	0.88	95.5%
BERTurk [37]	0.84	0.74	0.83	92.6%
RoBERTaTurk [7]	0.85	0.75	0.84	93.4%
Hunspell [28]	0.84	0.71	0.82	92.2%
Proposed Base Detection Model	0.82	0.94	0.88	95.2%
Proposed Combined Model	**0.91**	0.90	**0.91**	**96.6%**

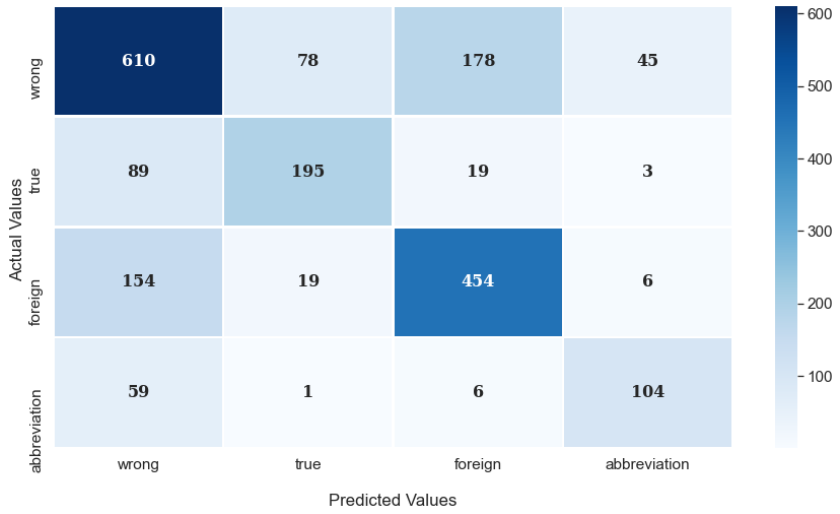


Figure 7. Predictions of the false positive reduction model on the test set.

model increased by 0.3% and 1.4%, respectively. McNemar's test pointed out that the difference between the F1 score of the combined model and all other models is statistically significant. To perform a further evaluation of the false positive reduction model, we tested its discriminative ability with the train-test procedure on the samples labeled using the mobile application detailed in Section 3. Eighty percent of the dataset was used for training and validation purposes, while the remaining samples were used for testing. The results given in Figure 7 show that overall accuracy of 67% was achieved on this four-class problem. The results reveal that the accuracy of the model in separating the foreign words from the other classes is 72% which is higher than the accuracies obtained for the other classes. On the other hand, the lowest class-based accuracy was obtained for the abbreviation class with 61%.

6. Conclusions

In this study, we proposed a spelling error detection model for the Turkish language. The model consists of two main steps. In the first step, a model trained using Bi-LSTM with BPE tokenizer for the binary classification task that separates the correctly spelled words and misspelled words is applied to the given word. In the second

step, the samples labeled as positive, i.e. misspelled words, are fed to the false positive reduction model in which the false positive predictions are aimed to be reduced.

For the training of the false positive reduction model, a dataset was built and labeled using a mobile application developed. As a contribution of this study, during the labeling process, foreign words and abbreviations were also determined, and thus, different from the existing studies the proposed combined model is able to determine not only Turkish words but also foreign words and abbreviations including social media terms commonly used in the Turkish spoken language. The findings showed that the false positive reduction model significantly increased the precision of the base detection model in exchange for a comparably less drop in its recall. Besides, the combined model yielded a higher F1 score and accuracy than the existing studies addressing spelling error detection problems. The results also showed that using the BPE tokenizer for tokenization improves the detection ability of the LSTM-based model.

The user texts on online platforms such as social media posts or product reviews on e-commerce websites include a high rate of misspelled words. The model-based text correction studies require a clean text database for model training. In future research, the proposed detection model with high error detection accuracy can be used to prepare a clean dataset to form a basis for a deep learning-based spelling error correction model. Furthermore, it is worth noting that while we address word-level spelling detection problems in our paper, transformer-based architectures are anticipated to outperform LSTM models in accurately modeling a sentence-based spelling detection task due to their attention mechanism and superior ability to represent contextual information. In the future, we aim to extend our research by developing a sentence-based spelling detection and correction approach for the Turkish language and provide an extensive comparison with the model proposed in this paper and transformer-based models.

References

[1] Rapp A, Curti L, Boldi A. The human side of human-chatbot interaction: A systematic literature review of ten years of research on text-based chatbots. International Journal of Human-Computer Studies. 2021;151:102630.3. https://doi.org/10.1016/j.ijhcs.2021.102630

[2] Singh D, Reddy S, Hamilton W, Dyer C, Yogatama D. End-to-end training of multi-document reader and retriever for open-domain question answering. Advances in Neural Information Processing Systems. 2021; 34:25968-81.5

[3] Ali A, Nakov P, Bell P, Renals S. WERd: Using social text spelling variants for evaluating dialectal speech recognition. In: 2017 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU). IEEE; 2017; 7141-7148. https://doi.org/10.1109/ASRU.2017.8268928

[4] Bellegarda JR, Monz C. State of the art in statistical methods for language and speech processing. Computer Speech & Language. 2016; 35:163-84.10. https://doi.org/10.1016/j.csl.2015.07.001

[5] Birjali M, Kasri M, Beni-Hssane A. A comprehensive survey on sentiment analysis: Approaches, challenges and trends. Knowledge-Based Systems. 2021; 226:107134.12. https://doi.org/10.1016/j.knosys.2021.107134

[6] Altuner AB, Kilimci ZH. A novel deep reinforcement learning based stock price prediction using knowledge graph and community aware sentiments. Turkish Journal of Electrical Engineering and Computer Sciences. 2022;30 (4):1506-24.14. https://doi.org/10.55730/1300-0632.3862

[7] Aytan B, Sakar CO. Comparison of Transformer-Based Models Trained in Turkish and Different Languages on Turkish Natural Language Processing Problems. In: 2022 30th Signal Processing and Communications Applications Conference (SIU). IEEE; 2022. p. 1-4.17 (in Turkish with an abstract in English). https://doi.org/10.1109/SIU55565.2022.9864818

[8] Rivera-Trigueros I. Machine translation systems and quality assessment: a systematic review. Language Resources and Evaluation. 2021:1-27.19. https://doi.org/10.1007/s10579-021-09537-5

[9] Aggarwal CC. Machine learning for text: An introduction. In: Machine learning for text. Springer 2018; 1-16.https://doi.org/10.1007/978-3-319-73531-3_1

[10] Varma R, Verma Y, Vijayvargiya P, Churi PP. A systematic survey on deep learning and machine learning approaches of fake news detection in the pre-and post-COVID-19 pandemic. International Journal of Intelligent Computing and Cybernetics. 2021; 23. https://doi.org/10.1108/IJICC-04-2021-0069

[11] Yildiz B, Emekci F. Name spell-check framework for social networks. Turkish Journal of Electrical Engineering and Computer Sciences. 2016;24 (4):2194-204.25. https://doi.org/10.3906/elk-1402-92

[12] Anbukkarasi S, Varadhaganapathy S. Neural network-based error handler in natural language processing. Neural Computing and Applications. 2022;34 (23):20629-38. https://doi.org/10.1007/s00521-022-07489-7

[13] Solak A, Oflazer K. Design and implementation of a spelling checker for Turkish. Literary and linguistic computing. 1993 ;8 (3):113-30. https://doi.org/10.1093/llc/8.3.113

[14] Yessenbayev Z, Kozhirbayev Z, Makazhanov A. KazNLP: A pipeline for automated processing of texts written in Kazakh language. InInternational Conference on Speech and Computer 2020; 657-666. Springer, Cham. https://doi.org/10.1007/978-3-030-60276-5_63

[15] Ozer H, Korkmaz EE. Transmorph: a transformer based morphological disambiguator for Turkish. Turkish Journal of Electrical Engineering and Computer Sciences. 2022;30 (5):1897-913. https://doi.org/10.55730/1300-0632.3912

[16] Choudhury M, Thomas M, Mukherjee A, Basu A, Ganguly N. How difficult is it to develop a perfect spell-checker? A cross-linguistic analysis through complex network approach. arXiv preprint physics/0703198. 2007 Mar 21. https://doi.org/10.48550/arXiv.physics/0703198

[17] Singh S, Singh S. HINDIA: a deep-learning-based model for spell-checking of Hindi language. Neural Computing and Applications. 2021 ;33 (8):3825-40. https://doi.org/10.1007/s00521-020-05207-9

[18] Hassan H, Menezes A. Social text normalization using contextual graph random walks. In Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers) 2013 ;(pp. 1577-1586).

[19] Oflazer K. Spelling correction in agglutinative languages. arXiv preprint cmp-lg/9410004. 1994. https://doi.org/10.48550/arXiv.cmp-lg/9410004

[20] Akın AA, Akın MD. Zemberek, an open source NLP framework for Turkic languages. Structure. 2007;10 (2007):1-5.

[21] Balík M. Implementation of directed acyclic word graph. Kybernetika. 2002;38 (1):91-103.

[22] Damerau FJ. A technique for computer detection and correction of spelling errors. Communications of the ACM. 1964 ;7 (3):171-6. https://doi.org/10.1145/363958.363994

[23] Eryigit G, Torunoglu-Selamet Di. Social media text normalization for Turkish. Natural Language Engineering. 2017;23 (6):835-75. https://doi.org/10.1017/S1351324917000134

[24] Çolakoğlu T, Sulubacak U, Tantuğ AC. Normalizing non-canonical Turkish texts using machine translation approaches. In The 57th Annual Meeting of the Association for Computational Linguistics 2019 Jul 28. The Association for Computational Linguistics.

[25] Heafield K. KenLM: Faster and smaller language model queries. In Proceedings of the sixth workshop on statistical machine translation 2011 (pp. 187-197).

[26] Klein G, Kim Y, Deng Y, Senellart J, Rush AM. Opennmt: Open-source toolkit for neural machine translation. arXiv preprint arXiv:1701.02810. 2017 . https://doi.org/10.48550/arXiv.1701.02810

[27] Büyük O. Context-dependent sequence-to-sequence turkish spelling correction. ACM Transactions on Asian and Low-Resource Language Information Processing (TALLIP). 2020;19 (4):1-6. https://doi.org/10.1145/3383200

[28] Safaya A, Kurtuluş E, Göktoğan A, Yuret D. Mukayese: Turkish NLP Strikes Back. arXiv preprint arXiv:2203.01215. 2022 Mar 2. https://doi.org/10.48550/arXiv.2203.01215

[29] Viktor T, Gyorgy G, Halácsy P, Kornai A, Laszlo N et al. Hunmorph: open source word analysis.InWorkshop on Software 2005; 16:77-85

[30] Demir S, Topcu B. Graph-based Turkish text normalization and its impact on noisy text processing. Engineering Science and Technology, an International Journal. 2022; 35:101192. https://doi.org/10.1016/j.jestch.2022.101192

[31] Webster JJ, Kit C. Tokenization as the initial phase in NLP. InProceedings of the 14th conference on Computational linguistics-Volume 4 1992: pp. 1106-1110. https://doi.org/10.3115/992424.992434

[32] Sennrich R, Haddow B, Birch A. Neural machine translation of rare words with subword units. arXiv preprint arXiv:1508.07909. 2015 . https://doi.org/10.48550/arXiv.1508.07909

[33] Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L et al. Attention is all you need. Advances in neural information processing systems. 2017;30. https://doi.org/10.48550/arXiv.1706.03762

[34] Raffel C, Shazeer N, Roberts A, Lee K, Narang S et al. Exploring the limits of transfer learning with a unified text-to-text transformer. Journal of Machine Learning Research. 2020;21 (140):1-67.

[35] Zhang S, Zheng D, Hu X, Yang M. Bidirectional long short-term memory networks for relation classification. In Proceedings of the 29th Pacific Asia conference on language, information and computation 2015 Oct (pp. 73-78).

[36] Hochreiter S, Schmidhuber J. Long short-term memory. Neural computation. 1997;9 (8):1735-80. https://doi.org/10.1162/neco.1997.9.8.1735

[37] Schweter S. Berturk-bert models for Turkish. 2020;3770924. https://doi.org/10.5281/zenodo.30