Spelling and Grammatical Error Detection for Informal Turkish Texts with Morphologically Sensible Models

MSc Research Project
MSCAI1_JAN23, Master of Science in Artificial Intelligence

Onur Bayram
Student ID: x22186662

School of Computing
National College of Ireland

Supervisor: Muslim Jameel Syed

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Onur Bayram………………………………………………………………………………………		
Student ID:	x22186662………………………………………………………………………………………..……		
Programme:	Master of Science in Artificial Intelligence	**Year:**	January 2023……….
Module:	MSc Research Practicum ………………………………………………………………….………		
Supervisor:	Muslim Jameel Syed …………………………………………………………….………		
Submission Due Date:	31/01/2024…………………………………………………………………………….………		
Project Title:	Spelling and Grammatical Error Detection for Informal Turkish Texts with Morphologically Sensible Models		
Word Count:	8171………………………………	**Page Count:**	24……………………………….……..

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.
ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature: [illegible]………………………………………………………………………………………

Date: 31/01/2024……………………………………………………………………………………………

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	**x**
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Spelling and Grammatical Error Detection for Informal Turkish Texts with Morphologically Sensible Models

Onur Bayram
x22186662

Abstract

Turkish is a morphologically rich language with unique characteristics such as agglutination and vowel harmony. This makes it challenging to create efficient spelling and grammatical error detection models for informal Turkish texts. Existing perspectives in the field are not enough to consider the unique characteristics of Turkish language, especially for informal texts, leading to poor precision. In this research, the project proposes to develop and discuss a morphologically sensible sequential deep learning models to aim spelling and grammatical error detection for informal Turkish texts. The proposed models are recurrent neural networks (RNN), Gated Recurrent Unit (GRU), Long Short-Term Memory (LSTM), Bidirectional GRU (Bi-GRU), and Bidirectional LSTM (Bi-LSTM); the models are all types of neural network architectures, especially designed for handling sequential data. They benefit for informal Turkish-specific tasks. The models are trained and tested equal to two million rows dataset, consisting of both formal and informal Turkish texts from Turkish news, Wikipedia, and Twitter. Each of the proposed models has an accuracy of 97%. Detailed results of the 5 proposed models are presented in this paper based on classification report, confusion matrix, accuracy-loss plots, and discussion. The proposed models are highly effective to fill the void in Turkish natural language processing and improving the accuracy of spelling and grammatical error detection for informal Turkish texts. The research also checks and displays misspelled words for the put into practice informal texts with 5 text experiments, one case study for each of the proposed models, in the implementation section.

Keywords: Natural Language Processing, Neural Network Architectures, Spelling and Grammatical Error Detection, Text Analysis, Turkish Language

1 Introduction

Natural language processing (NLP) activities utilize text supplied by users as input for various applications. The increasing utilization of the internet requires the creation of reliable natural language processing systems. While the quantity of valuable textual data increases daily due to microblog and social media platforms, most of the data gathered from these sources lacks a formal structure, posing challenges for its utilization in natural language processing systems.
The primary challenge in NLP lies in effectively collecting the nuanced contextual information present in input data, which may consist of a single word, a sentence, or even a paragraph. While the terminology of natural language remains fixed, there are many potential interpretations for these phrases. For this reason, deeper understanding of a word's definition is inadequate; one must also catch the surrounding circumstances in which it is used. Various

neural network designs exist for representing textual context. Neural network-based architectures have been successfully utilized in various domains. They are delivering extraordinary outcomes (Batmaz, 2022). Texts written by users on internet platforms frequently deviate from conventional language norms and exhibit several instances of misspelled words. This can pose a challenge for natural language processing models to perform effectively. Text normalization is a technique used to rectify these issues and enhance the comprehensibility of the text for models. Additionally, it aids individuals with imperfect language proficiency in comprehending the material more effectively (Aytan and Sakar, 2023). Identifying and detecting spelling and grammatical errors is an essential part of text normalization.

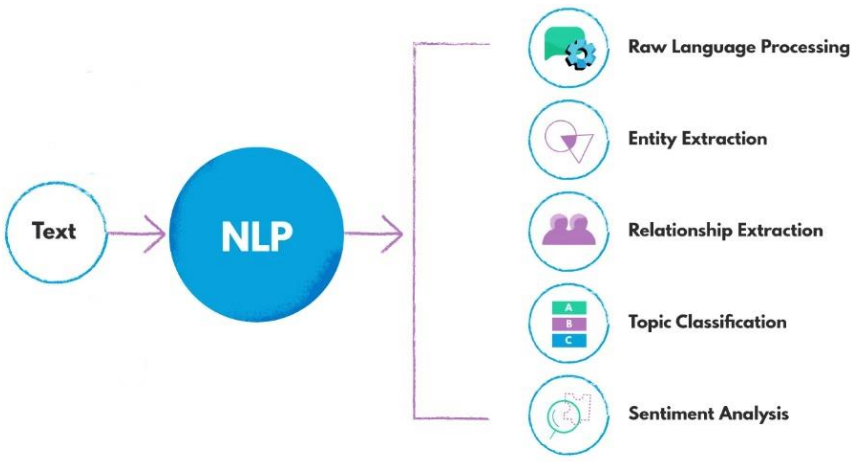


Figure 1. Diagram of Natural Language Processing

The research conducted by Uz and Eryigit explores the difficulties associated with spell checking in Turkish language. In Turkish language, where words are created by combining a root with morphemes, due to the contextual dependence of word meaning, morphological analysis is necessary for spell checking. For spell checking systems, the intricate relationship between morphology and semantics poses unique challenges (Uz and Eryigit, 2023).
Eryigit and Torunoglu-Selamet's article introduces a social media text normalization solution for Turkish language, Turkish language characterized by agglutination and distinctive normalization difficulties. The system partitions candidate generation into distinct cascaded modules, which target seven distinct error categories. Two manually standardized datasets for Turkish in the Web 2.0 domain were presented (Eryigit and Torunoglu-Selamet, 2017). Adali and Eryigit's article also address two significant issues, the restoration of vowels and diacritics, in the normalization of social media text. Their work proposes a hybrid approach to select one of the morphologically accurate outputs from the first stage by combining a language validator and a discriminative sequence classifier. They propose a model independent of language and does not require manual training set annotation. The efficacy of their methodology was evaluated using both synthetic data and real social media data (Adali and Eryigit, 2014).
Even though many grammar and spell-checking tools work well, they still have trouble with agglutinative languages like Turkish, especially when it comes to informal writing. The aim of this research project is to investigate this question; How well can sequential deep learning models perform in detecting informal Turkish texts as a classification task, and how can these

models be optimized to handle the idiosyncratic features of Turkish language? In this project, a recurrent neural networks (RNN) model, a Gated Recurrent Unit (GRU) model, a Long Short-Term Memory (LSTM) model, a Bidirectional GRU (Bi-GRU), and a Bidirectional LSTM (Bi-LSTM) model is developed for text classification in informal Turkish texts. Beyond these deep learning models, the study also analyzed misspelled words in informal Turkish texts using a combination of dictionary-based method.

The rest of this research paper is organized as follows. In Section 2, Related Work, the overview of the studies that focus on the detection of spelling and grammatical errors is demonstrated. Section 3 describes the Research Methodology. Section 4 presents Design Specification while Section 5 presents Implementation and Evaluation. Lastly, Section 6 discusses Conclusion and Future Work.

2 Related Work

An overview of research on the possible contributions of Turkish grammar and spell detection and text normalization towards attaining the aims is provided in this section. Multiple studies have been carried out on Turkish language. Over the past few years, there has been a significant advancement and numerous groundbreaking findings in study in this field.

Sonmez and Ozgur discuss the challenges posed by the informal style of social media material in their research, highlighting its intricate nature when it comes to automated analysis using natural language techniques. Their proposed approach for unsupervised text normalization stands out because to its utilization of not only lexical, but also contextual and grammatical information obtained from a word association network created from an extensive unlabeled social media text corpus. The authors highlight the effectiveness of their context-aware technique by achieving the highest F-score performance, surpassing traditional methods that depend on normalization dictionaries (Sonmez and Ozgur, 2014). Marsan et al. contribute to the subject of linguistic resource development by introducing a novel method for converting dependencies to constituencies, tailored specifically for Turkish language. They make the method work better by adding a bootstrap aggregating meta-algorithm and using a morphological analyzer and a feature-based machine learning model. The work provides evidence for a useful and effective conversion procedure, which promotes the development of new constituency treebanks and serves as excellent training material for natural language processing resources (Marsan et al.,2022).

Taylan et al. offer valuable insights in their study on grammar pedagogy, specifically focusing on the challenges of instructing non-native speakers in English writing proficiency. The authors provide a writing tool that utilizes a deep learning model to provide immediate feedback on grammatical issues in student writing, specifically emphasizing the precision of form and rule acquisition (Taylan et al., 2019). Yildirim and Yildiz analyze the essential requirement for text normalization in the context of short-text messages, emphasizing the limitations imposed by Turkish language's morphologically complex and agglutinative structure. Their proposed architecture for unsupervised text normalization incorporates a range of techniques, including lexical similarity and n-gram language modeling. The techniques employed develop from basic to more sophisticated solutions. Their approach, which integrates both lexical and semantic similarity for Turkish text normalization, is demonstrated to be effective through rigorous

evaluation using high-quality corpora, morphological parsers, and dictionaries. (Yildirim and Yildiz, 2015).
Buyuk, Erden, and Arslan present a sequence-to-sequence deep neural network approach to emphasize the impact of context on the correctness of correction. The authors introduce a foundational system that handles misspelled and reference words in an isolated manner, without considering their surrounding context. To improve the system's effectiveness, they incorporate both the context before and after misspelled words. The authors report a significant absolute enhancement in rectification performance by employing a context-dependent model, utilizing a substantial text corpus, and introducing substitution, deletion, and insertion errors (Buyuk, Erden, and Arslan, 2019).
Aydan and Sakar have made a noteworthy advancement by introducing a two-step deep learning model that can accurately identify and repair misspelled words. The researchers have incorporated a false positive reduction model to mitigate errors arising from foreign terminology and acronyms frequently prevalent in online platforms. The study shows the remarkable efficacy of the proposed Bi-LSTM-based model when used with the BPE tokenizer and various tokenization techniques, including character-based, syllable-based, and byte-pair encoding. The study also highlights the success of their innovative approach in improving precision without sacrificing recall in identifying misspelled words, utilizing LSTM and Bi-directional Bi-LSTM networks (Aydan and Sakar, 2023). Arikan, Gungor, and Uskudarli provide significant contributions, specifically focusing on the difficulties associated with clitic errors in Turkish language. The authors propose a neural sequence tagger model to address the constraints of vocabulary-based methods and the inadequacy of existing tools for languages with limited resources. The model's objective is to identify and rectify errors related to using "de/da" clitic. Their approach, which involves using an artificially created dataset, demonstrates a remarkable F1 score of 86.67%, outperforming existing spelling correction algorithms on a meticulously chosen dataset of challenging samples (Arikan, Gungor, and Uskudarli, 2019). Ozge, Bozal, and Ozge propose a novel method for rectifying diacritics in Turkish, acknowledging the importance of precise diacritic utilization in disentangling word meanings in natural language processing tasks. Instead of using traditional approaches for restoring diacritics in one direction, the authors propose a character-level sequence-to-sequence model that considers the context and translates diacritical letters to their ASCII equivalents in both forward and backward directions. Significantly, their language-agnostic model, which exclusively utilizes word embeddings, obtains an impressive 4.7% enhancement in F1 score for terms with multiple meanings and an overall improvement of 1.24% when assessed against a benchmark dataset of Turkish tweets (Ozge, Bozal, and Ozge, 2022).
Aydogan and Karci emphasize crucial significance of text processing, specifically pointing out the lack of language-specific research, especially for Turkish, in the fields of word embedding and deep neural networks. The authors offer significant contributions by constructing two Turkish datasets and training word vectors on a large unlabeled corpus using the Word2Vec methodology. The study uses advanced deep neural network designs, such as CNN, RNN, LSTM, and GRU. It concludes that the GRU and LSTM techniques surpass the others, leading to a notable improvement of 5% to 7% in accuracy when utilizing pre-trained word vectors (Aydogan and Karci, 2020). Goker and Can discuss the challenge of text normalization, a necessary pretreatment step for successful natural language processing, within the context of

the ever-changing and casual character of social media content. The authors propose two innovative alternatives: contextual normalization employing distributed word representations, and sequence-to-sequence normalization utilizing neural encoder-decoder models (Goker and Can, 2018).
Ince emphasizes the significance of incorporating morphological analysis and mathematical foundations into software development to effectively tackle the specific challenges posed by Turkish in spell checking and error correction. The project is around Turkish language and presents an application that utilizes the nZemberek tool, a Turkish corpus, and morphological structure to provide efficient spell checking and error correction. The program demonstrates the importance of language-specific features in developing effective spell-checking systems for Turkish. It attains a 95% success rate in spell checking and an 86% success rate in suggesting precise corrections (Ince, 2017).

3 Research Methodology

The research methodology from Preliminary Literature Analysis and Initial Proposal to Model and Experiments Implementation and Thesis Writeup is illustrated in Figure 2.

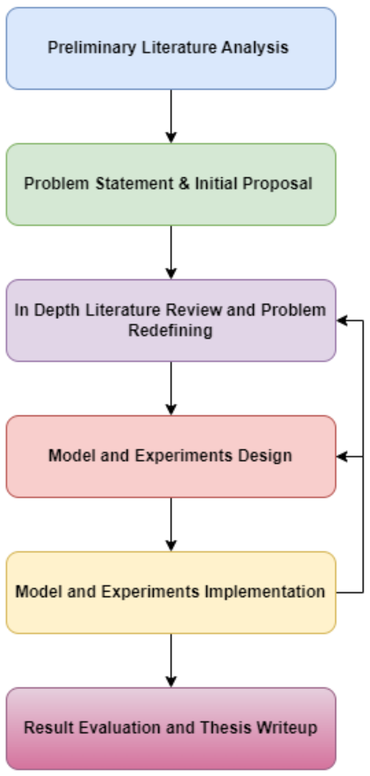

Preliminary Literature Analysis
Problem Statement & Initial Proposal
In Depth Literature Review and Problem Redefining
Model and Experiments Design
Model and Experiments Implementation
Result Evaluation and Thesis Writeup

Figure 2. Diagram of Research Process

The research methodology consists of six main steps: data loading and preprocessing, tokenization, padding sequences, model building, training, and evaluation, model prediction and text classification, and solution as illustrated in Figure 3.

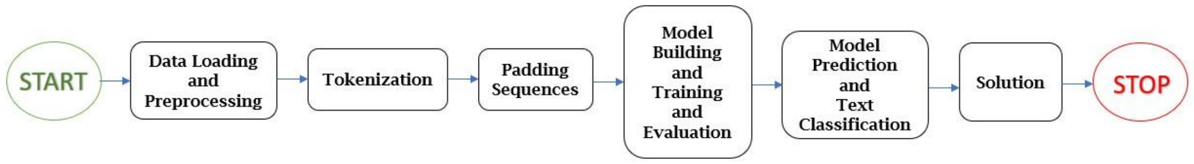

START
Data Loading and Preprocessing
Tokenization
Padding Sequences
Model Building and Training and Evaluation
Model Prediction and Text Classification
Solution
STOP

Figure 3. Research Methodology

Data Loading and Preprocessing is the initial stage where the dataset is loaded for research project purposes. Subsequently, the dataset undergoes preprocessing to eliminate any extraneous information. These steps may involve eliminating stop words, punctuation, and converting all text to lowercase. In this project, this phase is employed Pandas package in Python to import and read two CSV files ('Formal_Data.csv' and 'Informal_twitter_data.csv') into distinct data frames. Each dataset is limited to one million rows. Afterward, the two data frames are merged into a new data frame named data_new. The 'text' column is retrieved as the features, while the 'label' column is used as the targets. The data is later combined into pairs of features and targets, then randomly shuffled, and split into training and testing sets using a ratio of 80-20. This method produced x_train, y_train, x_test, and y_test variables that can be used in the text classification models in the study.
Throughout the time of Tokenization process, the preprocessed text data is divided into smaller units called tokens. Tokens are the fundamental units of text, representing individual words. Keras Tokenizer in Python is utilized to transform the text data into sequences of numerical tokens. The variable num_words variable is assigned a value of 10,000, which signifies the upper limit for the number of words to retain depending on their frequency. The tokenizer is applied to the training data, and the resulting tokenizer is serialized and stored using the pickle module. The previously saved tokenizer is reloaded into the script in due course.
At Padding Sequences step, the text data entries exhibited varying lengths. However, when it comes to modeling, it is crucial to provide data that has the same shape. Consequently, shorter sequences are augmented with zeros to align with the length of the longest sequence. Once the text data has been divided into sequences using Keras Tokenizer, the sequences are adjusted to have a consistent length for input into the model by either adding padding or truncating. It is crucial to perform this preparatory step to train deep learning models that require input of a specific size.
Model Building and Training step consisted of the model that develops the capacity to associate unique token patterns with preprocessed data and distinct classes. The procedure commenced with an embedding layer that transforms the input sequences, represented as integers, into compact vectors of a predetermined embedding_size=100. The proposed model layer, comprising 32 units, is employed to collect contextual information from both the forward and backward orientations in the input sequences. To address the issue of overfitting, a dropout layer is included with a dropout rate of 0.2. The last layer comprised a dense layer with a single unit and utilizes a sigmoid activation function, which is well-suited for addressing binary classification tasks. The model is trained using the binary cross entropy loss function and the Adam optimizer with a learning rate of 1e-3. The summary of the model architecture is shown, and the training is performed using the fit method with a batch size of 256, for a total of 5 epochs, using the provided training and validation data. The training history is stored in the

variable 'history' for further analysis and evaluation. During the Evaluation phase, following the completion of model training, it is imperative to assess its performance. Usually, this is accomplished by using a validation set that the model has not previously experienced. To evaluate the model's performance, the predictions are compared to the actual values. After training the deep learning model, its performance is evaluated using the evaluate technique, which measures the loss and accuracy on the test set. The results are stored in the variable named 'result'. Following this, the model that has undergone training generates predictions on the test dataset. The predictions are transformed into binary predictions by applying a threshold of 0.5. Afterwards, the confusion matrix and classification report functions are employed. This robust evaluation allowed for the assessment of the model's effectiveness in classifying textual data into the specified categories.
In Model Prediction and Text Classification step, a highly trained model can be used to generate precise predictions on new, unseen data. The model categorized the text using the information it has gained via training. The trained model is saved on the disk in the HDF5 format using the save approach. Following that, the script reloaded the stored model and the tokenizer that was previously saved using the pickle module. The loaded tokenizer is used to tokenize a provided sample text, which is later padded to conform with the model's anticipated input size. Afterwards, the pre-trained model is used to predict the likelihood of each category for the provided input text. The prediction is discretized using a threshold of 0.5, and depending on the result, the text is classified as either "informal" or "formal." The casual text is stored in a file named 'epassage.txt'.
The proposed methodology also involved a solution incorporating Turkish spelling error detection into the text classification project. The solution process is created a program to read a dictionary file with correctly spelled words and a text file with informal texts that needed to be spell and grammar checked.
Every one of these steps is crucial in the process for researching text classification projects. It should be emphasized that the precise particulars, parameters, preferences and methods employed in each stage may vary based on the research criteria.

4 Design Specification

The design specification approach for this research project has been fully accomplished with an Intel(R) Core (TM) i5-10210U CPU @ 1.60GHz 2.11 GHz DELL personal computer running on Windows 10 Pro. Deep learning models are built on Anaconda Navigator’s Jupyter notebook utilizing Python programming language, and its libraries and frameworks such as Tensorflow, Keras, and Sci-kit learn.
The project utilized two datasets, namely "Formal_Data" and "Informal_twitter_data," which were obtained from Github.com (Fixy-TR, 2020). The dataset files consist of more than 1,000,000 input data points, each including columns converted and named "label" and "text." The column labelled "label" indicates the class of the sentences, with 1 being formal sentences and 0 representing informal sentences. The column labeled "Text" gives a description of Turkish sentences written in the target language.

Table 1. Sample text and label examples from the dataset

Text	Label
Clay Matematik Enstitüsü ilk doğru çözüme 1 milyon dolar vadetmişti ancak Perelman ödülü kabul etmedi (English: The Clay Mathematical Institute had promised $1 million for the first correct solution, but Perelman declined the prize)	1 (Formal)
Gel beni izle diyor resmeeen :) (English: It's likeee she's saying come and watch me :))	0 (Informal)

The datasets comprise Turkish texts obtained from Turkish news sources, Wikipedia, and Twitter. Each used dataset has precisely 1,000,000 sentences as data points. Figure 4 shows the sentence count for each category, providing a visual depiction of the datasets with an equal distribution of 1,000,000 texts per category.

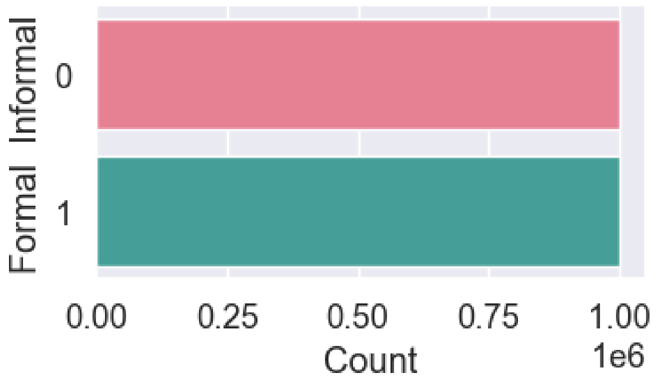


Figure 4. Count of Informal and Formal in final dataset

This guarantees that the models are trained on a diverse range of Turkish texts, enhancing their capacity to accurately predict informal language usage and morphological patterns inside Turkish sentences.

4.1 Deep Learning Text Classification Models

Deep learning enables the creation and use of many models to extract significant insights and predictions from complex datasets. The objective of this project is to evaluate the performance and effectiveness of suitably created the deep learning text classification models. This section examines five well-known deep learning models that selected for this study: RNN, LSTM, GRU, Bi-LSTM, and Bi-GRU.

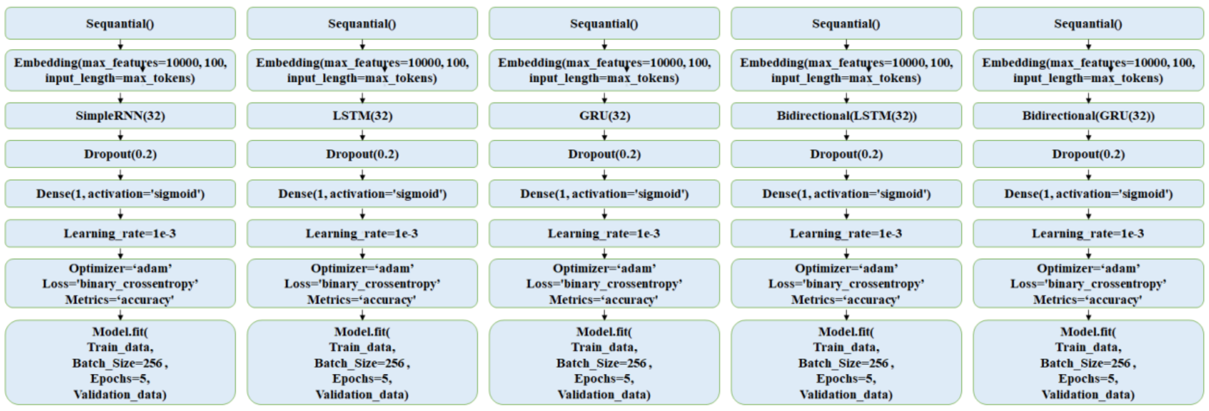
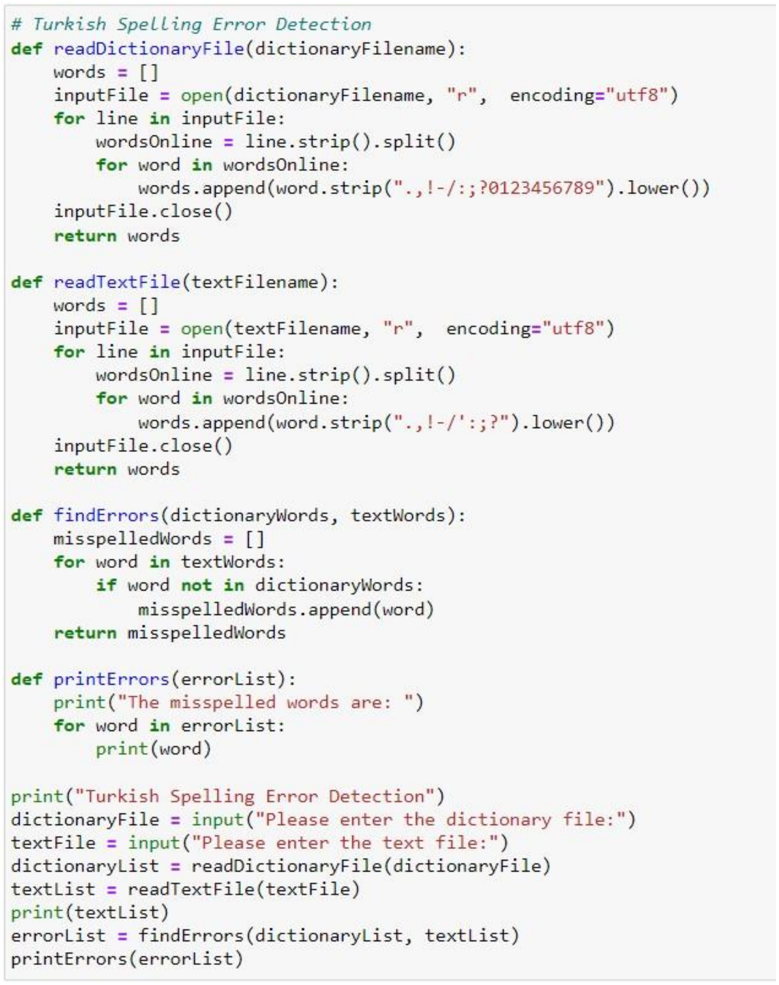

Sequantial()
Embedding(max_features=10000, 100, input_length=max_tokens)
SimpleRNN(32)
LSTM(32)
GRU(32)
Bidirectional(LSTM(32))
Bidirectional(GRU(32))
Dropout(0.2)
Dense(1, activation='sigmoid')
Learning_rate=1e-3
Optimizer='adam' Loss='binary_crossentropy' Metrics='accuracy'
Model.fit(Train_data, Batch_Size=256, Epochs=5, Validation_data)

Figure 5. The Basic Architecture of the proposed deep learning models

RNN, a specialized type of neural network created to process sequential input, exhibits excellent performance in natural language processing's classification tasks. RNNs possess a feedback loop that enables them to store information from prior inputs. This allows them to efficiently document and comprehend the chronological connections within the material. GRU, an adapted variant of the conventional RNN, incorporates gating mechanisms to control the flow of information within the network. The design of GRU architecture is especially aimed at addressing the issue of vanishing gradients that occur in conventional RNNs. GRUs enhance the model's ability to grasp distant dependencies in sequential data, while still assuring effective computing efficiency. LSTM is a type of RNN that tackles the problem of the vanishing gradient by integrating a more intricate memory cell structure. LSTMs possess the capacity to retain information throughout extended sequences, rendering them exceptionally efficient in scenarios were understanding context and preserving associations over prolonged durations is paramount. Bi-GRU architecture improves upon the GRU model by analyzing input sequences in both the forward and backward directions. The bidirectional technique enhances the model's ability to capture dependencies from both preceding and succeeding contexts, enabling a more comprehensive understanding of sequential patterns, and improving performance in classification tasks that require context-aware processing. Bi-LSTM, like Bi-GRU, augments the capabilities of LSTM by examining input sequences in both the forward and backward directions. This model's bidirectional structure enables it to gather dependencies from both preceding and subsequent elements in the sequence, and this helps understanding of the context and temporal connections within the data. This improves its efficiency in the tasks.

4.2 Turkish Spelling and Grammatical Error Detection

Misspelled words can be categorized into two main groups: non-word errors and real-word errors. Non-word errors arise when an erroneous word is not acknowledged as a legitimate component of the language's dictionary. For instance, the incorrect spelling of "cement" as "sement" or "chicken" as "chiken" exemplifies this specific type of mistake. To address these problems algorithmically, one can employ string distance metrics to compare the misspelled words with the language's word corpus, facilitating their straightforward correction. On the other hand, faults that occur in real-world situations pose a more challenging barrier when it comes to detection. These faults comprise words that are grammatically correct but thematically unsuitable for the given context. In the given phrase "The men power required for this job...", the term "manpower" has been mistakenly replaced with "men". Considering that both "man" and "men" are valid words in the dictionary, it is crucial to thoroughly analyses the sentence's context to identify and rectify this error (El Gayar et al., 2022).



```python
# Turkish Spelling Error Detection
def readDictionaryFile(dictionaryFilename):
    words = []
    inputFile = open(dictionaryFilename, "r",  encoding="utf8")
    for line in inputFile:
        wordsOnline = line.strip().split()
        for word in wordsOnline:
            words.append(word.strip(".,!-/:;?0123456789").lower())
    inputFile.close()
    return words

def readTextFile(textFilename):
    words = []
    inputFile = open(textFilename, "r",  encoding="utf8")
    for line in inputFile:
        wordsOnline = line.strip().split()
        for word in wordsOnline:
            words.append(word.strip(".,!-/':;?").lower())
    inputFile.close()
    return words

def findErrors(dictionaryWords, textWords):
    misspelledWords = []
    for word in textWords:
        if word not in dictionaryWords:
            misspelledWords.append(word)
    return misspelledWords

def printErrors(errorList):
    print("The misspelled words are: ")
    for word in errorList:
        print(word)

print("Turkish Spelling Error Detection")
dictionaryFile = input("Please enter the dictionary file:")
textFile = input("Please enter the text file:")
dictionaryList = readDictionaryFile(dictionaryFile)
textList = readTextFile(textFile)
print(textList)
errorList = findErrors(dictionaryList, textList)
printErrors(errorList)
```

Figure 6. Python code of the Turkish dictionary-based spelling error detection

In this project, there is a straightforward and efficient implementation of a spelling error detection system as illustrated in Figure 6 that relies on a Turkish dictionary text file. The code is organized into multiple functions, each serving a distinct purpose, along with a primary execution block. The function readDictionaryFile(dictionaryFilename) is used to read a file that includes a dictionary. Each line of the file represents a single word. The function readTextFile(textFilename) is analogous to the readDictionaryFile function. It reads a text file, analyses each line, and provides a collection of words in the form of a list. The distinction is in the specific characters that are removed from each word, except numerical digits. The function findErrors(dictionaryWords, textWords) accepts two lists of words as input, one derived from the dictionary and the other from the text file. It verifies each word in the text file by comparing it to the dictionary. The function printErrors(errorList) accepts a list of incorrectly spelled words as its input and proceeds to display them individually. The main code block prompts the user to provide the filenames for the dictionary file and the text file.

This method represents a direct approach to implementing a spelling error detection system. Nevertheless, it is important to acknowledge that the current version is rudimentary and there exist numerous possibilities for its expansion.

5 Implementation and Evaluation

Python is the predominant programming language for artificial intelligence and machine learning development because of its user-friendly nature, clarity, and vast collection of modules and packages. The generation of each model involved the development of complex code modules, using the versatile programming language Python along with a variety of specialized machine learning libraries. To efficiently handle and preserve several separate Python environments, along with their corresponding packages, the project utilized the widely accepted Anaconda distribution of Python (Silberztein et al., 2018).

The performance of the proposed deep learning models is evaluated throughout both the training and validation stages. The loss graphs illustrated the model's loss with time, with the x-axis representing the number of epochs and the y-axis representing the loss value. Both lines should exhibit a decline with time, signifying that the model is acquiring knowledge and enhancing its forecasts, while the accuracy graphs illustrated the model's precision as time progresses. Both lines should have an upward trend, signifying that the model is progressively generating more accurate predictions as it acquires knowledge. Both graphs depict the training phase with a blue line and the validation phase with an orange line. The training phase encompasses the process in which the model acquires knowledge from the given train data, while the validation stage evaluates the model's performance using previously unseen test data.

A confusion matrix (an error matrix) is a structured table that provides a visual representation of the effectiveness of an algorithm, usually one that involves supervised learning. The confusion matrix is a valuable tool since it provides both an understanding of the mistakes made by your classifier and the specific types of mistakes produced. Such an elevated level of detail enables the implementation of more refined optimization and model selection procedures. The confusion matrixes are a tabular form with dimensions of 2x2, which include four combinations of predicted and actual values. True Positives (TP) are occasions where the model correctly predicted the positive class, showing that the actual class was positive, and the model appropriately classified it as positive. True Negatives (TN) are instances where the model correctly predicted the negative class, showing that the actual class was negative, and the model appropriately classified it as negative. False Positives (FP) arise when the model incorrectly predicts that instances are positive when they are negative. This is typically known as a Type I error. False Negatives (FN) occur when the model incorrectly classifies a positive instance as negative. This occurrence is frequently denoted as a Type II error.

The following figures show a comparison of the five deep learning models based on their accuracy and loss. The loss and accuracy plots offer a distinct representation of the model's performance throughout each optimization iteration (epoch). Consequently, a model is considered superior when it has a lesser loss. The results of the project demonstrate the potential of each model, with the highest maximum average accuracy of 97% and the lowest minimum average loss function of 8% for informal Turkish text classification task, derived from the calculated values.

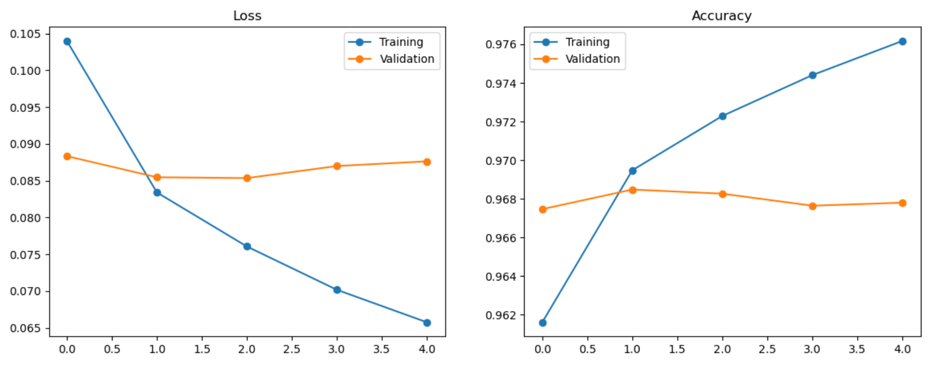


Figure 7. Loss and Accuracy Plots of the proposed RNN model

After 5 epochs, the results of the proposed RNN model during its training and validation phases, the calculated accuracy is 0.967 and the calculated loss of 0.087 for informal Turkish text task based on the datasets.

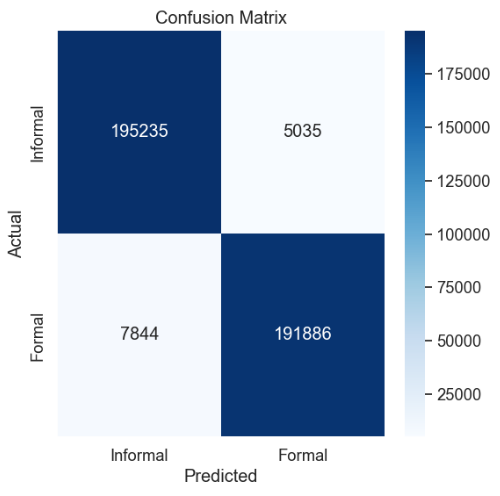


Figure 8. Confusion Matrix of the proposed RNN model

The confusion matrix of the proposed RNN model is presented, the value of TP is 195,235, indicating that the model accurately identified 195,235 data points belonging to informal class, the value of TN is 191,886, indicating that the model accurately identified 191,886 data points belonging to formal class. FP value is 5,035, indicating that the model erroneously identified 5,035 data points from formal class as belonging to informal class. FN value is 7,844, indicating that the model mistakenly identified 7,844 data points from informal class as belonging to formal class. RNN classifier demonstrated exceptional performance on the dataset, as indicated by the numerous instances of true positive and true negative values. This indicates that the model proficiently differentiates between accurate and inaccurate forms.

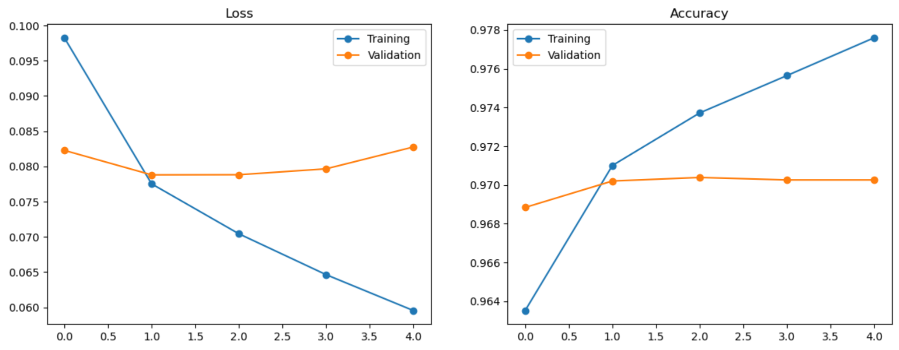


Figure 9. Loss and Accuracy Plots of the proposed LSTM model

After 5 epochs, the results of the proposed LSTM model during its training and validation phases, the calculated accuracy is 0.970 and the calculated loss of 0.082 for informal Turkish text task based on the datasets.

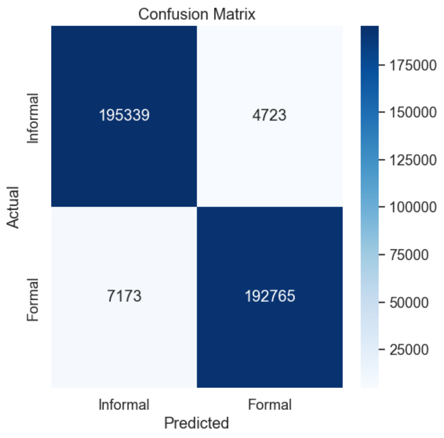


Figure 10. Confusion Matrix of the proposed LSTM model

The confusion matrix of the proposed LSTM model is presented, TP value is 195,339, indicating that the model accurately identified 195,339 data points belonging to informal class. The value of TN is 192,765, indicating that the model accurately identified 192,765 data points belonging to formal class. FP value is 4,723, indicating that the model erroneously identified 4,723 data points from formal class as belonging to informal class. FN value is 7,173, indicating that the model erroneously identified 7,173 data points from informal class as belonging to formal class. LSTM classifier had excellent performance on the dataset, as indicated by the considerable number of true positive and true negative values. This indicates that the model proficiently differentiates between accurate and inaccurate forms.

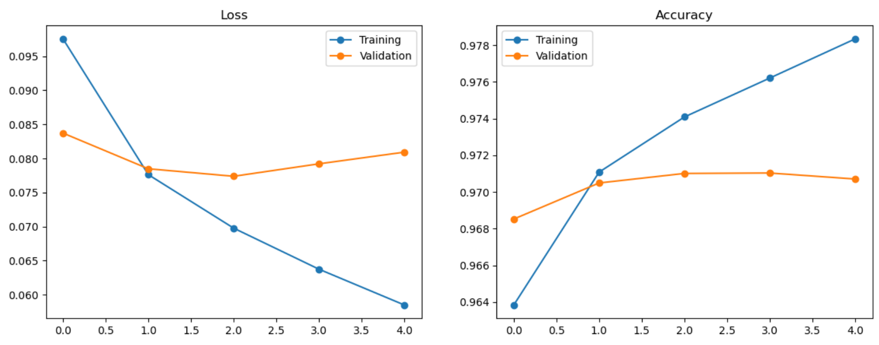


Figure 11. Loss and Accuracy Plots of the proposed GRU model

After 5 epochs, the results of the proposed GRU model during its training and validation phases, the calculated accuracy is 0.970 and the calculated loss of 0.080 for informal Turkish text task based on the datasets.

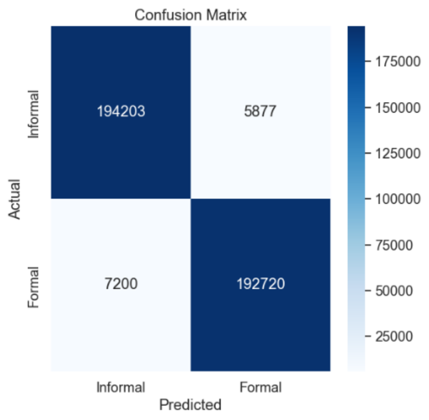


Figure 12. Confusion Matrix of the proposed GRU model

The confusion matrix of the proposed GRU model is presented, TP value is 194,203, indicating that the model accurately identified 194,203 data points belonging to informal class. The value of TN is 192,720, indicating that the model accurately identified 192,720 data points belonging to formal class. FP value is 5,877, indicating that the model erroneously identified 5,877 data points from formal class as belonging to informal class. The value of FN is 7,200, indicating that the model misclassified 7,200 data points from informal class as belonging to formal class. GRU classifier had excellent performance on the dataset, as indicated by the high number of true positive and true negative values. This indicates that the model proficiently differentiates between accurate and inaccurate forms.

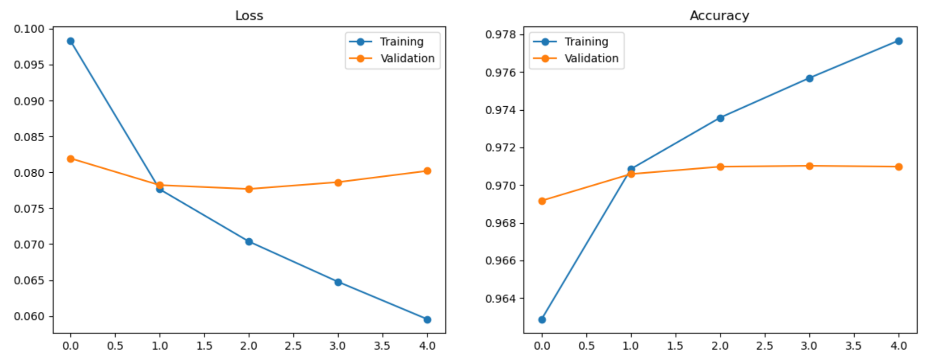


Figure 13. Loss and Accuracy Plots of the proposed Bi-LSTM model

After 5 epochs, the results of the proposed Bi-LSTM model during its training and validation phases, the calculated accuracy is 0.970 and the calculated loss of 0.080 for informal Turkish text task based on the datasets.

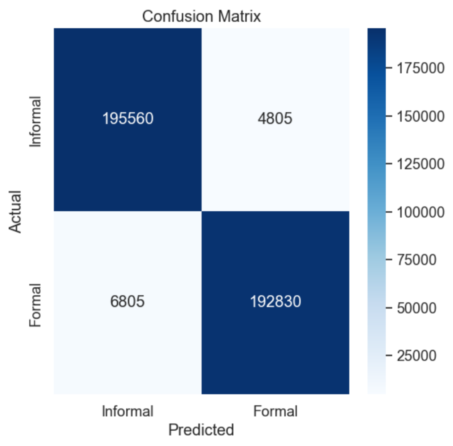


Figure 14. Confusion Matrix of the proposed Bi-LSTM model

The confusion matrix of the proposed Bi-LSTM model is presented, the value of TP is 195,560, indicating that the model accurately identified 195,560 data points belonging to informal class. The value of TN is 192,830, indicating that the model accurately identified 192,830 data points belonging to formal class. FP value is 4,805, indicating that the model erroneously identified 4,805 data points from formal class as belonging to informal class. FN value is 6,805, indicating that the model erroneously identified 6,805 data points from informal class as belonging to formal class. Bi-LSTM classifier had excellent performance on the dataset, as indicated by the high number of true positive and true negative values. This indicates that the model proficiently differentiates between accurate and inaccurate forms.

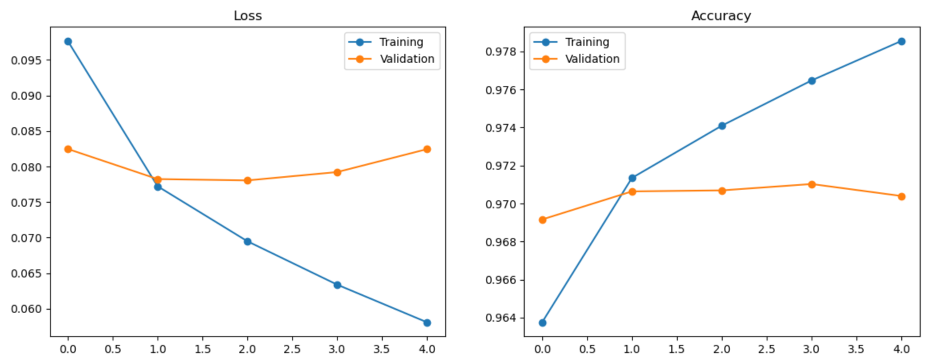


Figure 15. Loss and Accuracy Plots of the proposed Bi-GRU model

After 5 epochs, the results of the proposed Bi-GRU model during its training and validation phases, the calculated accuracy is 0.970 and the calculated loss of 0.082 for informal Turkish text task based on the datasets.

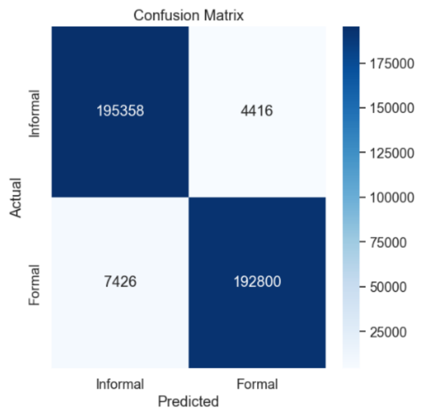
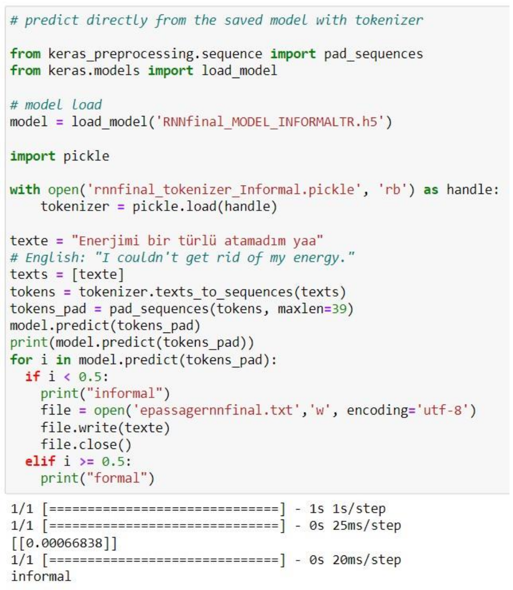


Figure 16. Confusion Matrix of the proposed Bi-GRU model

The confusion matrix of the proposed Bi-GRU model is presented, TP value is 195,358, indicating that it accurately identified 195,358 data points belonging to informal class. The value of TN is 192,800, indicating that the model accurately identified 192,800 data points belonging to formal class. FP value is 4,416, indicating that the model erroneously identified 4,416 data points from formal class as belonging to informal class. FN value is 7,426, indicating that the model erroneously identified 7,426 data points from informal class as belonging to formal class. Bi-GRU classifier had excellent performance on the dataset, as indicated by the high number of true positive and true negative values. This indicates that the model proficiently differentiates between accurate and inaccurate forms.

5.1 RNN Informal Text Experiment / Case Study 1

This is a text classification experiment that uses a pre-trained proposed RNN model to classify a given text as either formal or informal.

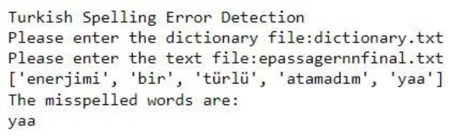


```
# predict directly from the saved model with tokenizer

from keras_preprocessing.sequence import pad_sequences
from keras.models import load_model

# model load
model = load_model('RNNfinal_MODEL_INFORMALTR.h5')

import pickle

with open('rnnfinal_tokenizer_Informal.pickle', 'rb') as handle:
    tokenizer = pickle.load(handle)

texte = "Enerjimi bir türlü atamadım yaa"
# English: "I couldn't get rid of my energy."
texts = [texte]
tokens = tokenizer.texts_to_sequences(texts)
tokens_pad = pad_sequences(tokens, maxlen=39)
model.predict(tokens_pad)
print(model.predict(tokens_pad))
for i in model.predict(tokens_pad):
  if i < 0.5:
    print("informal")
    file = open('epassagernnfinal.txt','w', encoding='utf-8')
    file.write(texte)
    file.close()
  elif i >= 0.5:
    print("formal")
```

```
1/1 [==============================] - 1s 1s/step
1/1 [==============================] - 0s 25ms/step
[[0.00066838]]
1/1 [==============================] - 0s 20ms/step
informal
```

Figure 17. Result of the proposed RNN model's Informal Text Experiment

The pre-trained RNN model is loaded from a file named 'RNNfinal_MODEL_INFORMALTR.h5'. The file named 'rnnfinal_tokenizer_Informal.pickle' is utilized to transform the input text into sequences of numbers, which can then be inputted into the model. The input text "Enerjimi bir türlü atamadım yaa" is segmented into tokens using the tokenizer that has been loaded. Subsequently, the sequence of numbers is extended to a length of 39, which corresponds to the model's maximum expected length. The predict method of the RNN model is invoked using the padded sequence as input. This function generates a forecast for the provided input text. The prediction is printed to the console. If the predicted value is below 0.5, the text is categorized as informal, and the input text is saved in a file named 'epassagernnfinal.txt'. If the forecast is equal to or above 0.5, the text is categorized as formal. The outcome demonstrates that RNN model possesses the ability to differentiate between formal and informal text, relying on the training it has undergone. This has the potential to be applied in the field of Turkish spelling and grammatical error detection. However, additional experiments should be conducted to assess the model's efficacy by testing it on various sorts of text.



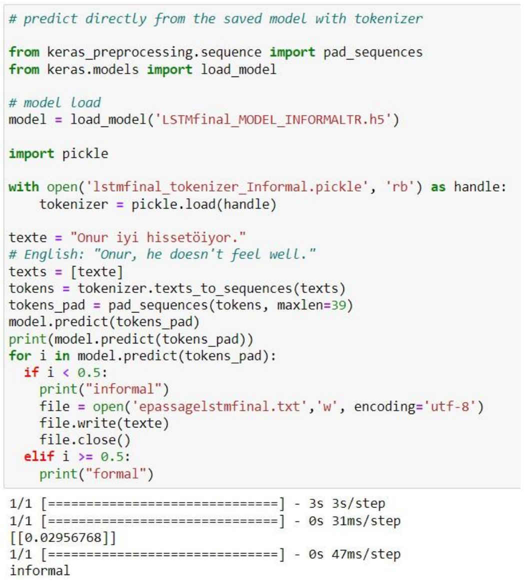
```
Turkish Spelling Error Detection
Please enter the dictionary file:dictionary.txt
Please enter the text file:epassagernnfinal.txt
['enerjimi', 'bir', 'türlü', 'atamadım', 'yaa']
The misspelled words are:
yaa
```

Figure 18. Result of Spelling Error Detection for Informal Text Experiment Case Study 1

Turkish spelling error detection tool for this project verifies an informal text by comparing it to a dictionary of accurately written words. The application prompts the user to input a dictionary file, which we have designated as 'dictionary.txt'. This file comprises a compilation of accurately spelled Turkish words. Subsequently, the program prompts the user to provide a text file for the purpose of detecting spelling problems. We have provided the file 'epassagernnfinal.txt', which was used in the prior Python code. The program parses the text file and dissects it into discrete words, also known as tokens. Subsequently, the algorithm verifies each token against the dictionary. When a token is not included in the dictionary, it is classified as a misspelled word. In this instance, the term 'yaa' is absent from the dictionary and is thus identified as an erroneous word. This type of detection can be highly advantageous for natural language processing tasks, such as context analysis and grammar correction. 'Yaa' is an informal Turkish interjection that may not be included in an verifiable dictionary. These sentences need attention.

5.2 LSTM Informal Text Experiment / Case Study 2

This is a text classification experiment that uses a pre-trained proposed LSTM model to classify a given text as either formal or informal.

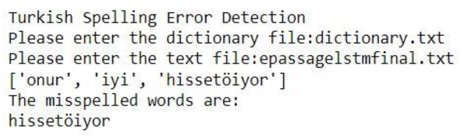


```python
# predict directly from the saved model with tokenizer

from keras_preprocessing.sequence import pad_sequences
from keras.models import load_model

# model load
model = load_model('LSTMfinal_MODEL_INFORMALTR.h5')

import pickle

with open('lstmfinal_tokenizer_Informal.pickle', 'rb') as handle:
    tokenizer = pickle.load(handle)

texte = "Onur iyi hissetöiyor."
# English: "Onur, he doesn't feel well."
texts = [texte]
tokens = tokenizer.texts_to_sequences(texts)
tokens_pad = pad_sequences(tokens, maxlen=39)
model.predict(tokens_pad)
print(model.predict(tokens_pad))
for i in model.predict(tokens_pad):
  if i < 0.5:
    print("informal")
    file = open('epassagelstmfinal.txt','w', encoding='utf-8')
    file.write(texte)
    file.close()
  elif i >= 0.5:
    print("formal")
```

```
1/1 [==============================] - 3s 3s/step
1/1 [==============================] - 0s 31ms/step
[[0.02956768]]
1/1 [==============================] - 0s 47ms/step
informal
```

Figure 19. Result of the proposed LSTM model’s Informal Text Experiment

The pre-trained LSTM model is imported from the file named 'LSTMfinal_MODEL_INFORMALTR.h5'. The file 'lstmfinal_tokenizer_Informal.pickle' is utilized to convert the input text into sequences of integers, which are then inputted into the model. The input text "Onur iyi hissetöiyor" is segmented into tokens using the loaded tokenizer. The content is categorized as informal, and the input text is saved to a file entitled 'epassagelstmfinal.txt'. The outcome demonstrates that LSTM model possesses the ability to differentiate between formal and informal writing, relying on the training it has undergone. However, additional experiments need be conducted to assess the model's efficacy by testing it on various text genres.



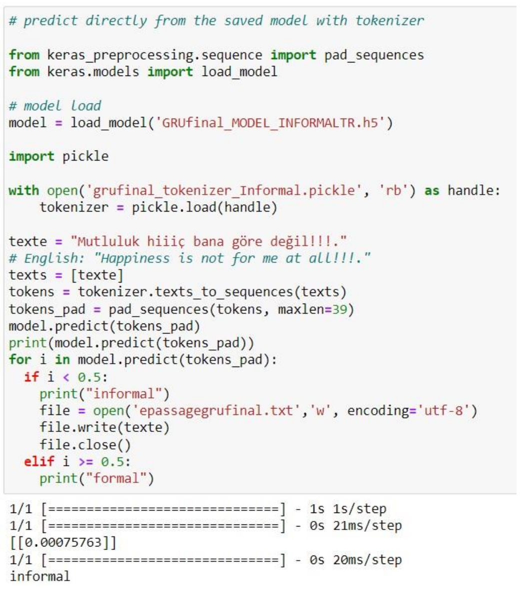
```
Turkish Spelling Error Detection
Please enter the dictionary file:dictionary.txt
Please enter the text file:epassagelstmfinal.txt
['onur', 'iyi', 'hissetöiyor']
The misspelled words are:
hissetöiyor
```

Figure 20. Result of Spelling Error Detection for Informal Text Experiment Case Study 2

We have included the file 'epassagelstmfinal.txt', which was previously written to in the Python code that was shared. The program parses the text file and dissects it into discrete words, also known as tokens. Subsequently, the algorithm verifies each token against the dictionary. When a token is not included in the dictionary, it is classified as a misspelled word. In this instance, the term 'hissetöiyor' is absent from the dictionary, resulting in its identification as an erroneous word.

5.3 GRU Informal Text Experiment / Case Study 3

This is a text classification experiment that uses a pre-trained proposed GRU model to classify a given text as either formal or informal.

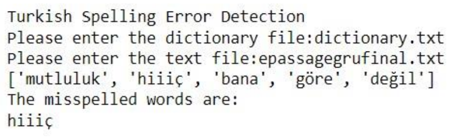


```
# predict directly from the saved model with tokenizer

from keras_preprocessing.sequence import pad_sequences
from keras.models import load_model

# model load
model = load_model('GRUfinal_MODEL_INFORMALTR.h5')

import pickle

with open('grufinal_tokenizer_Informal.pickle', 'rb') as handle:
    tokenizer = pickle.load(handle)

texte = "Mutluluk hiiiç bana göre değil!!!."
# English: "Happiness is not for me at all!!!."
texts = [texte]
tokens = tokenizer.texts_to_sequences(texts)
tokens_pad = pad_sequences(tokens, maxlen=39)
model.predict(tokens_pad)
print(model.predict(tokens_pad))
for i in model.predict(tokens_pad):
  if i < 0.5:
    print("informal")
    file = open('epassagegrufinal.txt','w', encoding='utf-8')
    file.write(texte)
    file.close()
  elif i >= 0.5:
    print("formal")
```

```
1/1 [==============================] - 1s 1s/step
1/1 [==============================] - 0s 21ms/step
[[0.00075763]]
1/1 [==============================] - 0s 20ms/step
informal
```

Figure 21. Result of the proposed GRU model’s Informal Text Experiment

GRU model that has been pre-trained is loaded from the file named 'GRUfinal_MODEL_INFORMALTR.h5'. The file 'grufinal_tokenizer_Informal.pickle' is utilized to transform the input text into sequences of numbers that can be inputted into the model. The loaded tokenizer is used to tokenize the input text "Mutluluk hiiiç bana göre değil!!!". The content is categorized as informal, and the input text is saved to a file entitled 'epassagegrufinal.txt'. The outcome demonstrates that GRU model possesses the ability to differentiate between formal and informal writing, relying on the training it has undergone. However, additional experiments need be conducted to assess the model's efficacy by testing it on various text genres.



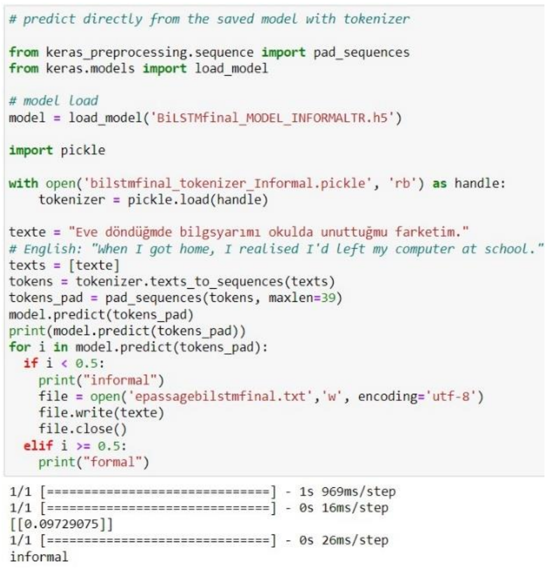
```
Turkish Spelling Error Detection
Please enter the dictionary file:dictionary.txt
Please enter the text file:epassagegrufinal.txt
['mutluluk', 'hiiiç', 'bana', 'göre', 'değil']
The misspelled words are:
hiiiç
```

Figure 22. Result of Spelling Error Detection for Informal Text Experiment Case Study 3

We have included the file 'epassagegrufinal.txt', which was previously written to in the Python code that was published. In this instance, the term 'hiiiç' is not present in the dictionary and is consequently identified as an erroneous word.

5.4 Bi-LSTM Informal Text Experiment / Case Study 4

This is a text classification experiment that uses a pre-trained proposed Bi-LSTM model to classify a given text as either formal or informal.

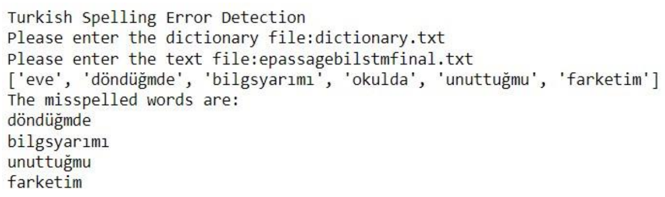


```
# predict directly from the saved model with tokenizer

from keras_preprocessing.sequence import pad_sequences
from keras.models import load_model

# model load
model = load_model('BiLSTMfinal_MODEL_INFORMALTR.h5')

import pickle

with open('bilstmfinal_tokenizer_Informal.pickle', 'rb') as handle:
    tokenizer = pickle.load(handle)

texte = "Eve döndüğmde bilgsyarımı okulda unuttuğmu farketim."
# English: "When I got home, I realised I'd left my computer at school."
texts = [texte]
tokens = tokenizer.texts_to_sequences(texts)
tokens_pad = pad_sequences(tokens, maxlen=39)
model.predict(tokens_pad)
print(model.predict(tokens_pad))
for i in model.predict(tokens_pad):
  if i < 0.5:
    print("informal")
    file = open('epassagebilstmfinal.txt','w', encoding='utf-8')
    file.write(texte)
    file.close()
  elif i >= 0.5:
    print("formal")
```

```
1/1 [==============================] - 1s 969ms/step
1/1 [==============================] - 0s 16ms/step
[[0.09729075]]
1/1 [==============================] - 0s 26ms/step
informal
```

Figure 23. Result of the proposed Bi-LSTM model's Informal Text Experiment

The pre-trained Bi-LSTM model is loaded from the file named 'BiLSTMfinal_MODEL_INFORMALTR.h5'. The file named 'bilstmfinal_tokenizer_Informal.pickle' is utilized to transform the input text into sequences of numbers, which can then be inputted into the model. The input text "Eve döndüğümde bilgisayarımı okulda unuttuğumu fark ettim" is tokenized using the loaded tokenizer. The text is categorized as informal, and the input text is saved to a file entitled 'epassagebilstmfinal.txt'. The outcome demonstrates that Bi-LSTM model can differentiate between formal and informal text, relying on the training it has undergone. Nevertheless, additional tests must be conducted to assess the model's efficacy across various text genres and appraise its performance.



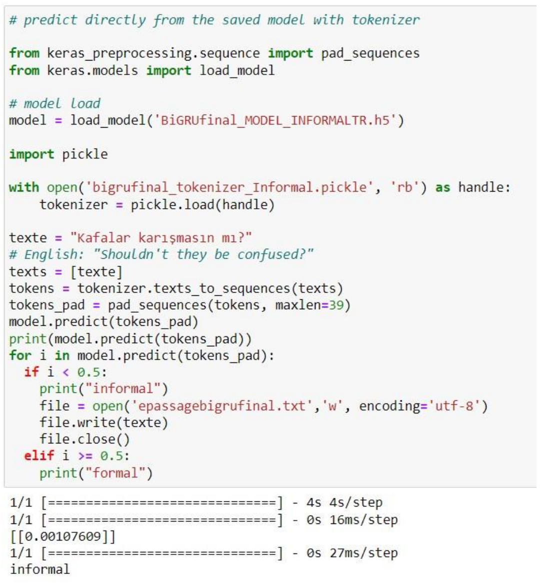
```
Turkish Spelling Error Detection
Please enter the dictionary file:dictionary.txt
Please enter the text file:epassagebilstmfinal.txt
['eve', 'döndüğmde', 'bilgsyarımı', 'okulda', 'unuttuğmu', 'farketim']
The misspelled words are:
döndüğmde
bilgsyarımı
unuttuğmu
farketim
```

Figure 24. Result of Spelling Error Detection for Informal Text Experiment Case Study 4

We have provided the file 'epassagebilstmfinal.txt', which was previously written in the shared Python code. In this instance, the words 'döndüğmde', 'bilgsyarımı', 'unuttuğmu', and 'farketim' are not included in the dictionary, therefore being identified as misspelled words.

5.5 Bi-GRU Informal Text Experiment / Case Study 5

This is a text classification experiment that uses a pre-trained proposed Bi-GRU model to classify a given text as either formal or informal.

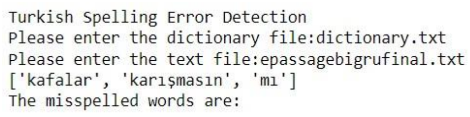


```
# predict directly from the saved model with tokenizer

from keras_preprocessing.sequence import pad_sequences
from keras.models import load_model

# model load
model = load_model('BiGRUfinal_MODEL_INFORMALTR.h5')

import pickle

with open('bigrufinal_tokenizer_Informal.pickle', 'rb') as handle:
    tokenizer = pickle.load(handle)

texte = "Kafalar karışmasın mı?"
# English: "Shouldn't they be confused?"
texts = [texte]
tokens = tokenizer.texts_to_sequences(texts)
tokens_pad = pad_sequences(tokens, maxlen=39)
model.predict(tokens_pad)
print(model.predict(tokens_pad))
for i in model.predict(tokens_pad):
  if i < 0.5:
    print("informal")
    file = open('epassagebigrufinal.txt','w', encoding='utf-8')
    file.write(texte)
    file.close()
  elif i >= 0.5:
    print("formal")
```

```
1/1 [==============================] - 4s 4s/step
1/1 [==============================] - 0s 16ms/step
[[0.00107609]]
1/1 [==============================] - 0s 27ms/step
informal
```

Figure 25. Result of the proposed Bi-GRU model's Informal Text Experiment

The pre-trained Bi-GRU model is loaded from the file named 'BiGRUfinal_MODEL_INFORMALTR.h5'. The file named 'bigrufinal_tokenizer_Informal.pickle' is utilized to transform the input text into sequences of numbers, which can then be inputted into the model. The input text "Kafalar karışmasın mı?" is segmented into tokens using the tokenizer that has been loaded. The content is categorized as informal, and the input text is saved in a file entitled 'epassagebigrufinal.txt'. The outcome demonstrates that Bi-GRU model possesses the ability to differentiate between formal and informal writing, because of the training it underwent. Nevertheless, additional experiments should be conducted to assess the model's efficacy by testing it on various text genres.



```
Turkish Spelling Error Detection
Please enter the dictionary file:dictionary.txt
Please enter the text file:epassagebigrufinal.txt
['kafalar', 'karışmasın', 'mı']
The misspelled words are:
```

Figure 26. Result of Spelling Error Detection for Informal Text Experiment Case Study 5

We have provided the file 'epassagebigrufinal.txt', which was used in the prior Python code. In this instance, each word inside the text is present in the dictionary, and there are no words that have been spelled incorrectly. Nevertheless, Bi-GRU classifier successfully identified the text as casual. When a native Turkish speaker enforces it, this sentence becomes an informal text in Turkish.

5.6 Results and Discussion

The project covered a complete presentation of the performance results using evaluation metrics such as classification reports, confusion matrices, and accuracy-loss plots. These metrics allowed for a thorough understanding of all the model’s performance. The proposed models aimed to address particularly focusing on improving the accuracy of detection informal Turkish texts. Additionally, the study conducted five text experiments with different Turkish sentences —one case study for each proposed model, to validate the models' effectiveness. The project also involved checking and displaying misspelled words in the dictionary-based implementation of informal text case studies. This practical plan added an extra layer of validation by applying the models to real-world scenarios. The project has created a structure that allows all the proposed models to be tested in an unlimited way with many different formal and informal Turkish texts.
The detailed results of five different models; RNN, LSTM, GRU, Bi-LSTM, and Bi-GRU are shown in Table 2 as a classification report.

Table 2. Results of classification report using the proposed deep learning models

Model	Accuracy	Class	Precision	Recall	F1-Score
RNN	0.97	Informal	0.96	0.97	0.97
		Formal	0.97	0.96	0.97
LSTM	0.97	Informal	0.96	0.98	0.97
		Formal	0.98	0.96	0.97
GRU	0.97	Informal	0.97	0.97	0.97
		Formal	0.97	0.97	0.97
Bi-LSTM	0.97	Informal	0.97	0.98	0.97
		Formal	0.98	0.97	0.97
Bi-GRU	0.97	Informal	0.96	0.98	0.97
		Formal	0.98	0.96	0.97

Table 2 highlights the robust capabilities of the proposed models within the scope of the job under investigation.
RNN model has a 0.97 accuracy, indicating that it accurately identified 97% of the occurrences. LSTM model exhibits a 0.97 accuracy, suggesting a comparable performance to RNN model. GRU model achieves a comparable level of performance to RNN and LSTM models, demonstrating an accuracy of 0.97. Bi-LSTM model maintains the trend by achieving an accuracy of 0.97. Bi-GRU model achieves a precision of 0.97. Nevertheless, RNN, LSTM and Bi-GRU exhibit a little reduced precision for the informal class in comparison to the other two models.
Bi-GRU model’s precision achieves 0.96 for informal class and 0.98 for formal class that means that when the model predicts a Turkish text as informal, it is correct 96% of the time,

and when predicting a Turkish text as formal, it is correct 98% of the time. Bi-GRU model's recall achieves 0.98 for Informal class and 0.96 for Formal class that means that the model captures 98% of actual informal Turkish texts and 96% of actual formal Turkish texts. Bi-LSTM model's precision achieves 0.97 for informal class and 0.98 for formal class that means that when the model predicts a Turkish text as informal, it is correct 97% of the time, and when predicting a Turkish text as formal, it is correct 98% of the time. Bi-LSTM model's recall achieves 0.98 for Informal class and 0.97 for Formal class that means that the model captures 98% of actual informal Turkish texts and 97% of actual formal Turkish texts. GRU model's precision achieves 0.97 for informal class and 0.97 for formal class that means that when the model predicts a Turkish text as informal, it is correct 97% of the time, and when predicting a Turkish text as formal, it is correct 97% of the time. GRU model's recall achieves 0.97 for Informal class and 0.97 for Formal class that means that the model captures 97% of actual informal Turkish texts and 97% of actual formal Turkish texts. LSTM model's precision achieves 0.96 for informal class and 0.98 for formal class that means that when the model predicts a Turkish text as informal, it is correct 96% of the time, and when predicting a Turkish text as formal, it is correct 98% of the time. LSTM model's recall achieves 0.98 for Informal class and 0.96 for Formal class that means that the model captures 98% of actual informal Turkish texts and 96% of actual formal Turkish texts. RNN model's precision achieves 0.96 for informal class and 0.97 for formal class that means that when the model predicts a Turkish text as informal, it is correct 96% of the time, and when predicting a Turkish text as formal, it is correct 97% of the time. RNN model's recall achieves 0.97 for Informal class and 0.96 for Formal class that means that the model captures 97% of actual informal Turkish texts and 96% of actual formal Turkish texts. In terms of precision, all different models perform well, with scores ranging from 0.96 to 0.98. This indicates an important level of accuracy in identifying both informal and formal texts. In terms of recall, with scores ranging from 0.96 to 0.98, all the different models capture a massive portion of actual informal and formal texts. Bi-LSTM and GRU models exhibit a balanced performance with high precision and recall for both classes. LSTM, Bi-GRU, and RNN models also exhibit a still effective performance with high precision and recall. However, without additional context, it is difficult to draw definitive conclusions.

Precision, Recall, and F1-Score metrics indicate that all models are performing near perfect, demonstrating their ability to accurately classify text as either formal or informal. Nevertheless, in the absence of other context or evidence, it is challenging to reach conclusive inferences. Overall, the findings indicate that all models are exhibiting satisfactory performance. Still, the performance of the model should be repeatedly tested by conducting many different experiments.

In the context of the main challenges the project faced and addressed, the dataset is preprocessed to eliminate any extraneous information and converted "text" and "label" columns. The final dataset was limited my final dataset to two million rows, one million each, this size was enough for training reliable models, especially with the proposed deep learning.

models. The developed and trained models were resource-efficient and environmentally sustainable. The datasets combined Turkish texts from diverse sources, including Twitter. This combination introduced noise and inconsistencies. Before the research project started, we checked the overall quality of the datasets to lead to a more robust model in detail. Checking with informal text structure was hard. Words were spelled wrongly, or there were abbreviations also emojis. If not managed well, the model could have difficulty understanding and classifying informal text accurately. Another challenge was that there should not be a significant difference in the number of formal and informal sentences. If not balanced, the model could have gotten biased towards the more common class, making it less effective at identifying the less frequent class. Another challenge was deciding how many words to consider during tokenization. If set too low, important words could have been left out, affecting the model's ability to comprehend and learn from my final dataset. The study tried to find and set the optimum num_words and tokenizer.fit_on_texts. Turkish sentences can vary in length. The study also tried to adjust sentences (padding sequences), this impacted how well the model understands the text data. Preprocessing, tokenizing, handling led to gain of information or necessary additions, impacting the model's performance. The final dataset's diversity helped capture unique styles of language, enhancing the representativeness of the dataset. While the dataset is diverse, it is essential to control specific language characteristics and biases and ethical engagement. We checked most of Turkish sentences of the datasets to lead to a more unbiased and neutral model in detail. We also shuffled the final dataset after combining formal and informal texts to help mitigate potential biases introduced during data preparation and data target pairs. Then the dataset was split into training and testing sets, that helped the final dataset do not overly influence the model's training and evaluation. It can be advantageous to address potential biases to continue monitoring the model's performance on different datasets with new data sources and contemplating extra preprocessing processes tailored to informal language nuances.

6 Conclusion and Future Work

The research project investigated five specific deep learning models and their methodologies designed for classifying informal Turkish writing and detecting informal Turkish writing's spelling and grammatical errors. These methodologies incorporated deep learning models as well as a combination of dictionary-based implementation methods. To summarize, the results showed that the proposed models have strong skills in the given task, which confirms their potential as a powerful tool for conducting experiments and applications incorporating formal and informal Turkish texts.

We acknowledge the possibility of making additional improvements, such as automatically expanding the size of the training set, integrating self-training methods, and using weighted finite-state language and error models to tackle concerns related to memory consumption. In future research, it would be beneficial to explore the use of other languages, considering the language-agnostic nature of our models.

Although our text detection algorithms achieved an important level of accuracy when applied to Turkish social media data, there is still potential for further enhancement and expansion. Subsequent studies can examine the efficacy of neural normalization strategies in more agglutinative languages. The researchers can prioritize the enhancement of discretization and vowel tasks, by integrating sophisticated approaches such restricted Viterbi algorithms in the decoding phase. This work establishes the fundamental basis for future progress in spelling correction and text normalization, thereby making a valuable contribution to the wider domain of natural language processing.

References

Batmaz, S. (2022). A SEQ2SEQ Transformer Model for Turkish Spelling Correction, Master of Science thesis, Bogazici University.

Aytan, B. and Sakar, C.O., 2023. Deep learning-based Turkish spelling error detection with a multi-class false positive reduction model. Turkish Journal of Electrical Engineering and Computer Sciences, 31(3), pp.581-595.

Uz, H. and Eryigit, G., 2023, May. Towards Automatic Grammatical Error Type Classification for Turkish. In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics: Student Research* Workshop (pp. 134-142).

Eryigit, G. and Torunoglu-Selamet, D., 2017. Social media text normalization for Turkish. Natural Language Engineering, 23(6), pp.835-875.

Adali, K. and Eryiğit, G., 2014, April. Vowel and diacritic restoration for social media texts. In *Proceedings of the 5th Workshop on Language Analysis for Social Media (LASM)* (pp. 53-61).

Sonmez, C. and Ozgur, A., 2014, October. A graph-based approach for contextual text normalization. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)* (pp. 313-324).

Marsan, B., Yildiz, O.K., Kuzgun, A., Cesur, N., Yenice, A.B., Saniyar, E., Kuyrukcu, O., Arican, B.N. and Yildiz, O.T., 2022, June. A Learning-Based Dependency to Constituency Conversion Algorithm for the Turkish Language. In *Proceedings of the Thirteenth Language Resources and Evaluation Conference* (pp. 5054-5062).

Taylan, B., Satyanarayana, A. and Samb, S., 2019, November. A Writing Tool that Provides Real-Time Feedback to Students on their Grammar Using Deep Learning. In 2019 *Fall Mid Atlantic States Conference.*

Yildirim, S. and Yildiz, T., 2015. An unsupervised text normalization architecture for Turkish language. *Research in Computing Science*, 90, pp.183-194.

Buyuk, O., Erden, M. and Arslan, L.M., 2019, April. Context influence on sequence to sequence Turkish spelling correction. In 2019 *27th Signal Processing and Communications Applications Conference (SIU)* (pp. 1-4). IEEE.

Aytan, B. and Sakar, C.O., 2023. Deep learning-based Turkish spelling error detection with a multi-class false positive reduction model. *Turkish Journal of Electrical Engineering and Computer Sciences*, 31(3), pp.581-595.

Arikan, U., Gungor, O. and Uskudarli, S., 2019, September. Detecting clitics related orthographic errors in Turkish. In *Proceedings of the International Conference on Recent Advances in Natural Language Processing* (RANLP 2019) (pp. 71-76).

Aydogan, M. and Karci, A., 2020. Improving the accuracy using pre-trained word embeddings on deep neural networks for Turkish text classification. *Physica A: Statistical Mechanics and its Applications*, 541, p.123288.

Goker, S. and Can, B., 2018, September. Neural text normalization for turkish social media. In 2018 *3rd International Conference on Computer Science and Engineering* (UBMK) (pp. 161-166). IEEE.

El Gayar, N., Trentin, E., Ravanelli, M. and Abbas, H. eds., 2022. *Artificial Neural Networks in Pattern Recognition: 10th IAPR TC3 Workshop, ANNPR 2022, Dubai, United Arab Emirates, November 24–26, 2022, Proceedings* (Vol. 13739). Springer Nature.

Silberztein, M., Atigui, F., Kornyshova, E., Métais, E., & Meziane, F. (2018) *Natural language processing and information systems: 23rd International Conference on Applications of Natural Language to Information Systems, NLDB 2018, Paris, France, June 13-15, 2018, Proceedings,* in Silberztein, M., Atigui, F., Kornyshova, E., Métais, E., & Meziane, F. (eds.), Natural Language Processing and Information Systems, Switzerland: Springer.

Fixy-TR. (2020) Fixy. Available at: *https://github.com/Fixy-TR/fixy/tree/master/data* (Accessed: 10.09.2023 and 12.12.2023)