Configuration Manual

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………………………………………………………………….………
Lecturer:	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:	1095…………………………… **Page Count:** 8…………………………….…….………

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: ………………………………………………………………………………………………

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):	

Configuration Manual

Onur Bayram
Student ID: x22186662

1 Introduction

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 deep learning are not enough to consider the unique characteristics of Turkish language, especially for informal written texts, leading to poor precision. In this research, the project proposes to develop and discuss a sequential deep learning models to aim informal text classification, and 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); these models are all types of neural network architectures, especially designed for handling sequential data. They benefit for informal Turkish-specific tasks. The proposed models are trained and tested equal to two million rows dataset, consisting of both formal and informal Turkish sentences and labels from Turkish news, Wikipedia, and Twitter. Each of the models has an accuracy of 97%. Detailed results of the 5 proposed models are presented in the 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 informal Turkish text classification. The research also analyses and displays misspelled words for the implemented informal written Turkish texts with 5 text experiments, one case study for each of the proposed models, in the implementation section. These experiments are effective to show spelling and grammatical error detection in informal Turkish texts.

This configuration manual contains fine-tune instructions on how to use a deep learning model to reproduce the experimental setup for a text classification project. In this study, a model is trained to distinguish between formal and informal Turkish text. TensorFlow and Keras, two well-known deep learning packages, are used in this project's Python code.

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 Python programming language 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).

2 System Requirements and Experimental Setup

See that the following prerequisites are installed before trying to replicate the experimental setup:
Python Environment: Python (3.6 or higher), Anaconda Distribution (latest version).
Required Libraries: TensorFlow (2.0 or higher), Keras (2.3 or higher), NumPy (latest version), Pandas (latest version), Matplotlib (latest version), Seaborn (latest version), Jupyter Notebook (for running the code interactively).
The design specification approach for this study project and literature 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. You can find all the related files in the ICT Solution Artefact's code and datasets folders.
You can open and check the Formal_Data.csv and Informal_twitter_data.csv from the datasets folder. "Formal_Data.csv" and "Informal_twitter_data.csv" are the two datasets used in the project. The files "Formal_Data.csv" and "Informal_twitter_data.csv" are where the formal and informal datasets are loaded, respectively. Make sure these datasets are accessible and formatted correctly. Using Pandas library, the code reads the data and preprocesses it to produce a composite dataset for testing and training.
You should be able to successfully complete the text classification project and duplicate the experimental setup by following these guidelines.

3 Instructions

These are a step-by-step explanation and instructions of the research project's Python code and the replication.
You can run all the steps via Turkish_Informal_Text_Analysis_Spelling_Error_Detection final Python code files by using Anaconda Navigator's Jupyter Notebook, respectively.

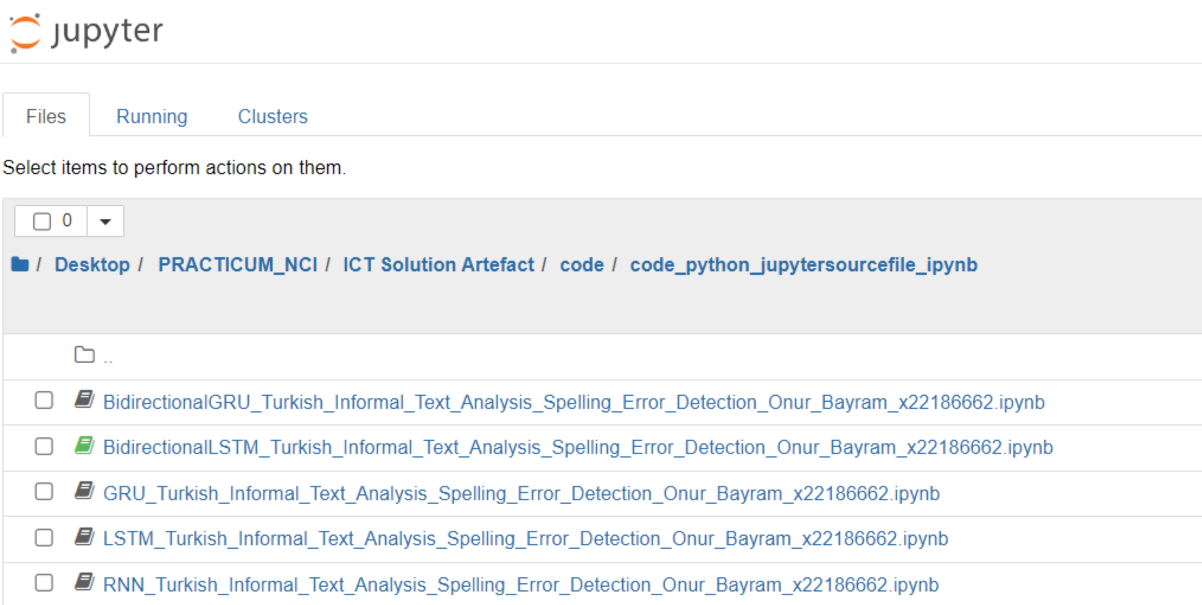


Figure 1. Jupyter Notebook http://localhost:8889/tree

Step 1: Import Anaconda Navigator Distribution, Create and Activate Conda Environment, Install and Import Python Libraries

By using Jupyter Notebook, the code starts by importing the necessary python libraries. These libraries will be used for data processing, model training, and evaluation.

jupyter BidirectionalLSTM_Turkish_Informal_Text_Analysis_Spelling_Error_Detection_Onur_Bayra... (autosaved) Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3 (ipykernel)

In [1]:
```
import numpy as np
import pandas as pd
import tensorflow as tf
from keras.preprocessing import sequence
from keras.models import Sequential
from keras.layers import Dense, Embedding, Bidirectional, LSTM,  Dropout
from keras.preprocessing.text import Tokenizer
from keras_preprocessing.sequence import pad_sequences
```

Figure 2. Jupyter Notebook Bidirectional LSTM code example 1

Step 2: Load and Combine Data

The code loads two datasets, Formal_Data.csv and Informal_twitter_data.csv, containing formal and informal text examples, respectively. It concatenates these datasets into a single DataFrame and splits the data into training and testing sets.

jupyter BidirectionalLSTM_Turkish_Informal_Text_Analysis_Spelling_Error_Detection_Onur_Bayra... (autosaved) Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3 (ipykernel)

In [2]:
```
data1 = pd.read_csv('Formal_Data.csv', encoding='utf-8')
data1=data1.drop(["id"], axis=1)
data1 = data1[:1000000]
data1.tail()
```

Out[2]:

	text	label
999995	1961'de inşa edilen Berlin Duvarı ile Doğu Alm...	1
999996	Berlin Duvarı, Soğuk Savaş'ın simgesi haline g...	1
999997	Her nasılsa Batı ve Doğu Almanya arasındaki ge...	1
999998	Bu yeniden birleşmeyi hızlandırdı	1
999999	Sonuçta iki ülke birleşti	1

In [3]:
```
data2 = pd.read_csv('Informal_twitter_data.csv', encoding='utf-8',low_memory=False)
data2.drop('id',axis=1,inplace=True)
data2 = data2.drop(columns="static_link")
data2 = data2.drop(columns="retweets")
data2 = data2.drop(columns="favorites")
data2 = data2.drop(columns="class")
data2 = data2.drop(columns="created_at")
data2 = data2.drop(columns="user_name")
data2["label"]=0
data2 = data2[:1000000]
data2.tail()
```

Out[3]:

	text	label
999995	Bence süper dizi olur ahahahah :)))\n\nKurtulu...	0
999996	@GumushAerospace 'e pas attım ki ASELSAT ve Kl...	0
999997	Sen mutluysan sorun yok o zaman keyfini çıkar ...	0
999998	müziksiz yaşayamam mı sandı akrepler :)))))))....	0
999999	Yani yani..:)))	0

Figure 3. Jupyter Notebook Bidirectional LSTM code example 2

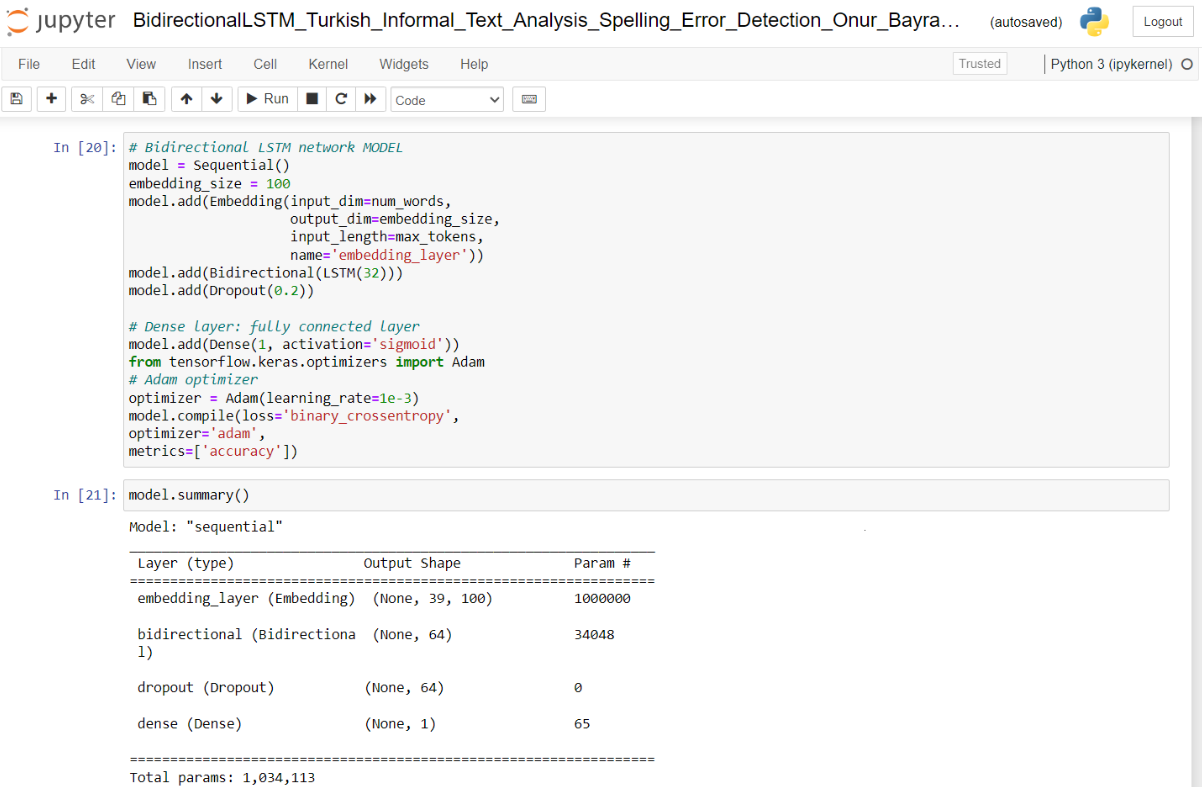
Step 3: Data Preprocessing

The code creates a tokenizer object to process the text data. It converts each text example into a sequence of integers representing the corresponding words in the vocabulary. The maximum length of the sequences is also determined to ensure consistent representation.

Figure 4. Jupyter Notebook Bidirectional LSTM code example 3

Step 4: Model Definition

The code defines a deep learning model using Keras. The model consists of an embedding layer, a simple deep learning model layer, a dropout layer, and a final dense layer with a sigmoid activation function to predict the formality of the text.



jupyter BidirectionalLSTM_Turkish_Informal_Text_Analysis_Spelling_Error_Detection_Onur_Bayra... (autosaved) Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3 (ipykernel)

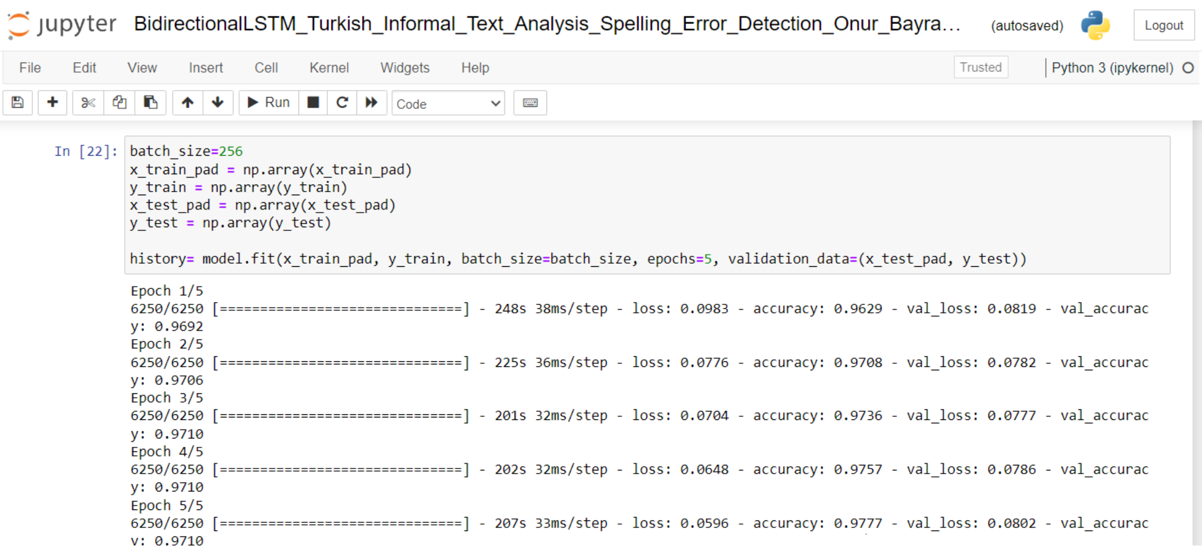
```
In [20]: # Bidirectional LSTM network MODEL
model = Sequential()
embedding_size = 100
model.add(Embedding(input_dim=num_words,
                    output_dim=embedding_size,
                    input_length=max_tokens,
                    name='embedding_layer'))
model.add(Bidirectional(LSTM(32)))
model.add(Dropout(0.2))

# Dense layer: fully connected layer
model.add(Dense(1, activation='sigmoid'))
from tensorflow.keras.optimizers import Adam
# Adam optimizer
optimizer = Adam(learning_rate=1e-3)
model.compile(loss='binary_crossentropy',
optimizer='adam',
metrics=['accuracy'])
```

```
In [21]: model.summary()

Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding_layer (Embedding)  (None, 39, 100)          1000000

 bidirectional (Bidirectiona  (None, 64)               34048
 l)

 dropout (Dropout)           (None, 64)                0

 dense (Dense)               (None, 1)                 65

=================================================================
Total params: 1,034,113
```

Figure 5. Jupyter Notebook Bidirectional LSTM code example 4

Step 5: Model Compilation and Training

The code compiles the deep learning model using the Adam optimizer and binary cross-entropy loss function. It trains the model for 5 epochs with a batch size of 256 on the training set.



jupyter BidirectionalLSTM_Turkish_Informal_Text_Analysis_Spelling_Error_Detection_Onur_Bayra... (autosaved) Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3 (ipykernel)

```
In [22]: batch_size=256
x_train_pad = np.array(x_train_pad)
y_train = np.array(y_train)
x_test_pad = np.array(x_test_pad)
y_test = np.array(y_test)

history= model.fit(x_train_pad, y_train, batch_size=batch_size, epochs=5, validation_data=(x_test_pad, y_test))

Epoch 1/5
6250/6250 [==============================] - 248s 38ms/step - loss: 0.0983 - accuracy: 0.9629 - val_loss: 0.0819 - val_accurac
y: 0.9692
Epoch 2/5
6250/6250 [==============================] - 225s 36ms/step - loss: 0.0776 - accuracy: 0.9708 - val_loss: 0.0782 - val_accurac
y: 0.9706
Epoch 3/5
6250/6250 [==============================] - 201s 32ms/step - loss: 0.0704 - accuracy: 0.9736 - val_loss: 0.0777 - val_accurac
y: 0.9710
Epoch 4/5
6250/6250 [==============================] - 202s 32ms/step - loss: 0.0648 - accuracy: 0.9757 - val_loss: 0.0786 - val_accurac
y: 0.9710
Epoch 5/5
6250/6250 [==============================] - 207s 33ms/step - loss: 0.0596 - accuracy: 0.9777 - val_loss: 0.0802 - val_accurac
y: 0.9710
```

Figure 6. Jupyter Notebook Bidirectional LSTM code example 5

Step 6: Model Evaluation

The code evaluates the trained deep learning model on the testing set and calculates the accuracy and loss metrics and results showcase classification report and confusion matrix.

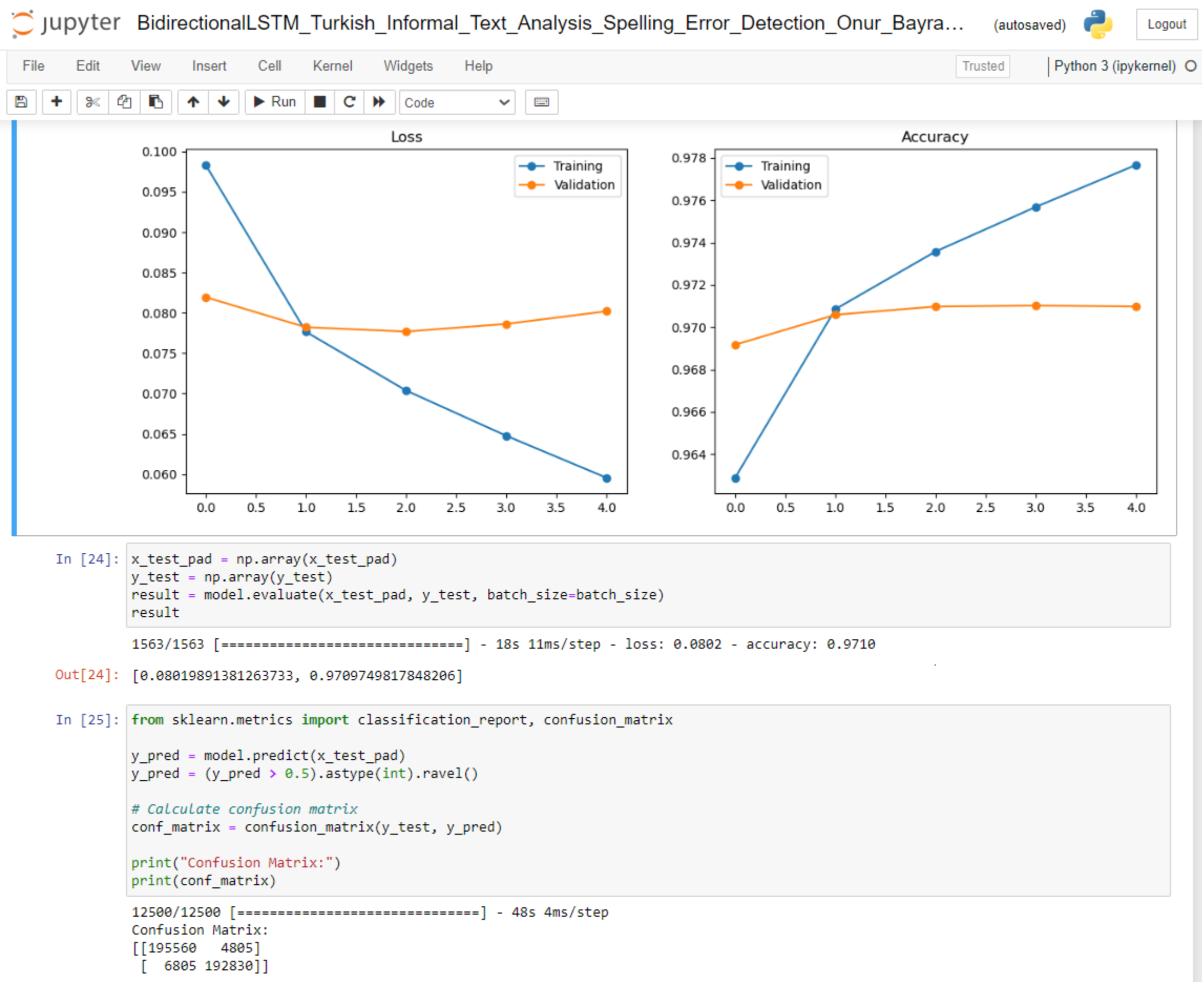


Figure 7. Jupyter Notebook Bidirectional LSTM code example 6

Step 7: Model Saving and Loading

The code saves the trained deep learning model and tokenizer to disk using Pickle to enable future use without retraining.

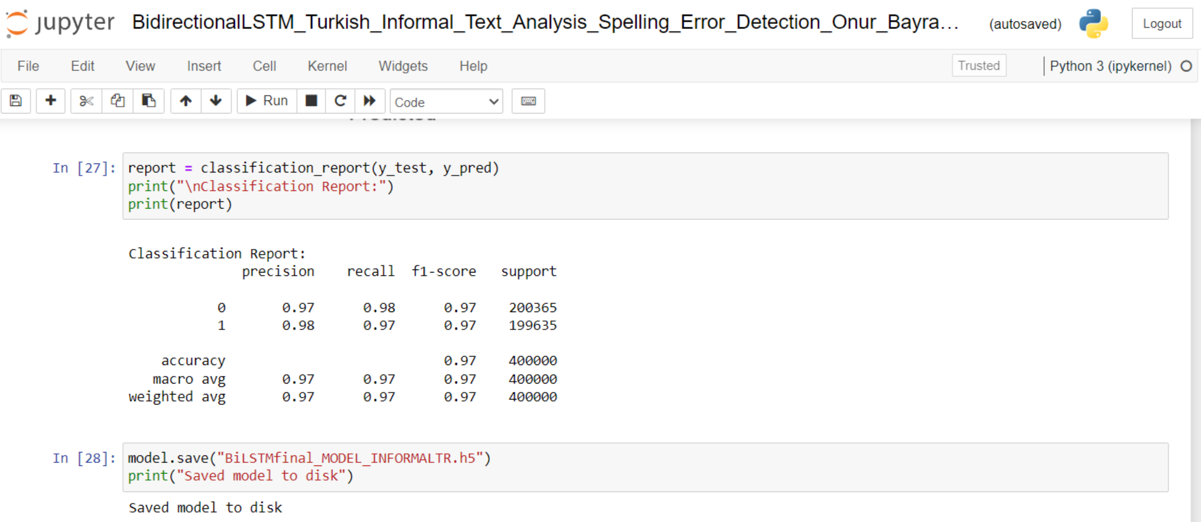


Figure 8. Jupyter Notebook Bidirectional LSTM code example 7

Step 8: Model Prediction

The code loads the saved model and tokenizer and demonstrates the model's ability to classify new text examples as formal or informal. You can change ‘textexample’ variable with new sentences, then you can give new predictions directly.

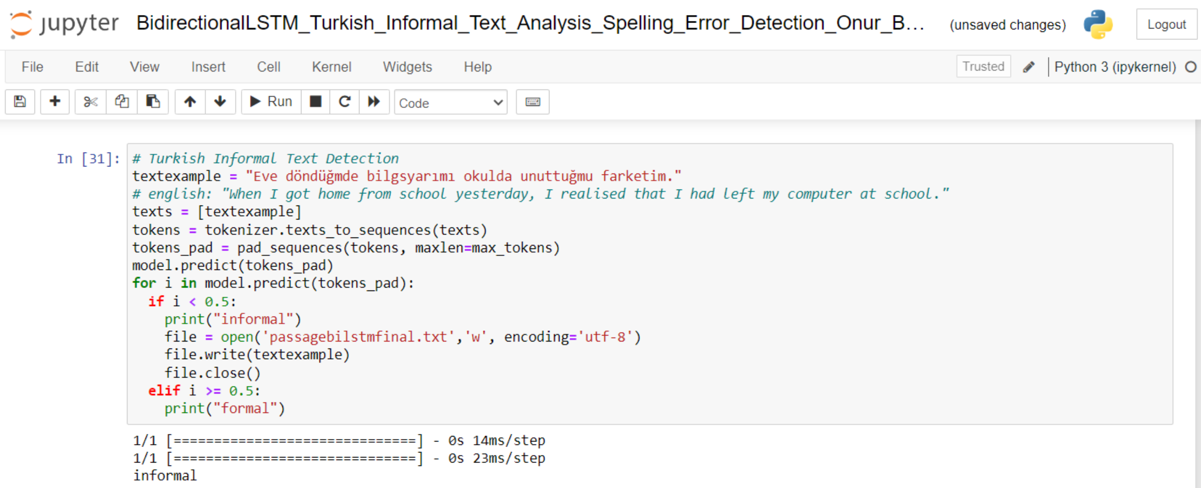


Figure 9. Jupyter Notebook Bidirectional LSTM code example 8

Step 9: Spelling Error Detection Implementation

The code defines functions to load a dictionary of Turkish words and a text file, extract words from the text, find misspelled words using the dictionary, and print the list of misspelled words. You can write the inputs ‘dictionary.txt’ and any text file in your computer, then you can check the misspelled Turkish words in the text file directly.

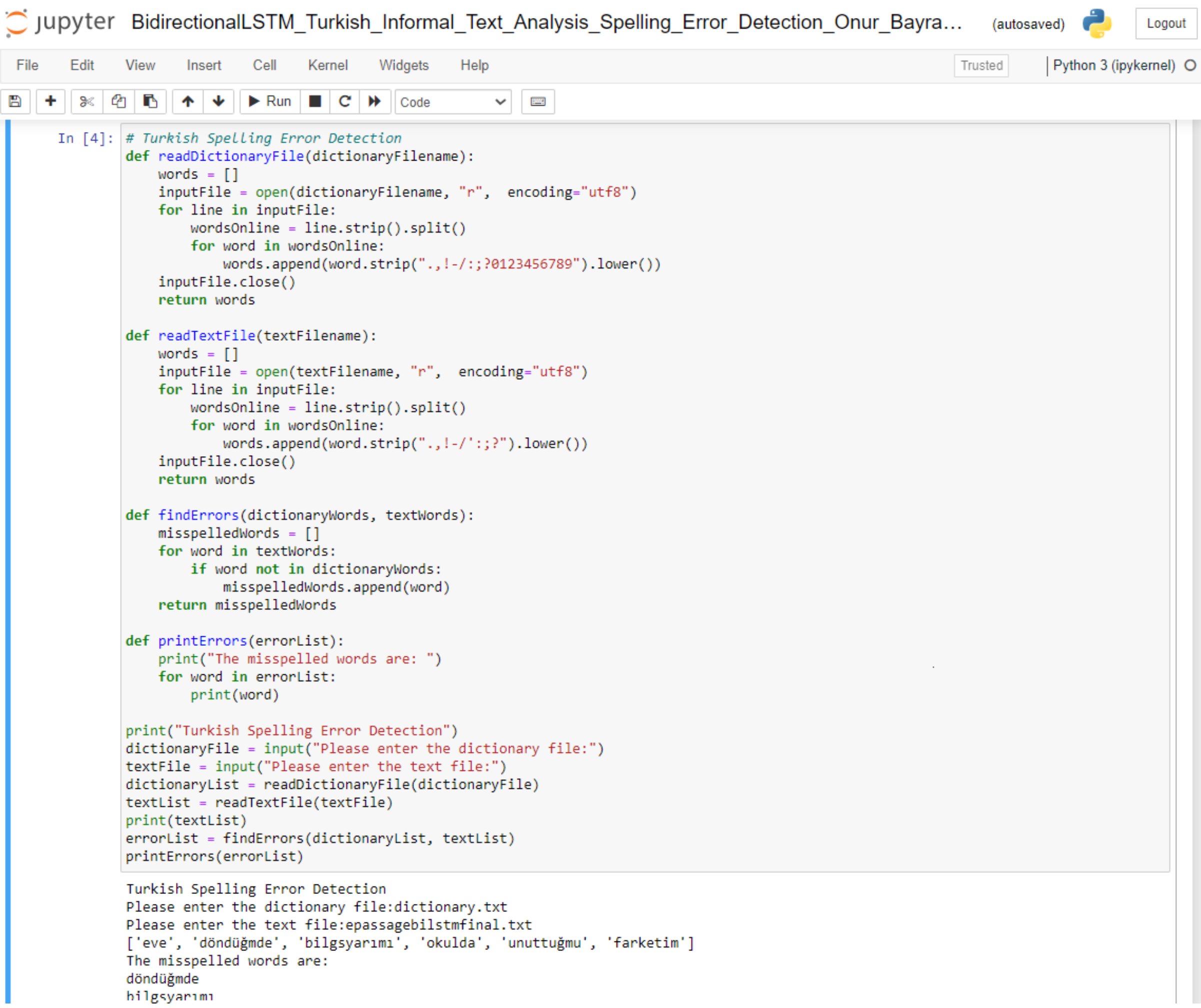


Figure 10. Jupyter Notebook Bidirectional LSTM code example 9

References

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.