CSE 484 NLP Report

Yavuz Selim İkizler

1901042617

Introductıon

I have 2 source codes. One of them is the Turkish check the correctness of turkish suffix “de” “da”. The other part is to check the correctness of the use of the Turkish suffix “ki”. Each code includes 3 parts as preprocessing, embedding and training. I tried to find the optimal solution for each phase.

Program Structure

Preprocessing Step:

I used 2 different sources for the dataset of the model: wikipedia dump and tr corpus (from Kaggel). Using my own script program, I trimmed the data to include only the sentences containing the necessary suffixes and kept them in separate txt files.

“de” Preprocess:

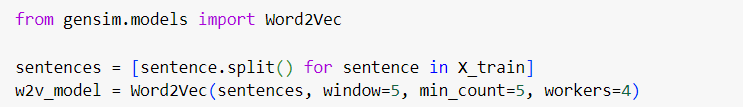
The data I collected from the datasets for the de suffix are the dataset and label dataset in which de s are used correctly; I divided it into the dataset where the tags were used incorrectly and the tag dataset, and kept them in separate txt files. I used a script I wrote to separate the sentences from the entire dataset. The script only extracts sentences with 10 characters and a single suffix. I also used the script for sentences with incorrectly used syllables.

“ki” Preprocess:

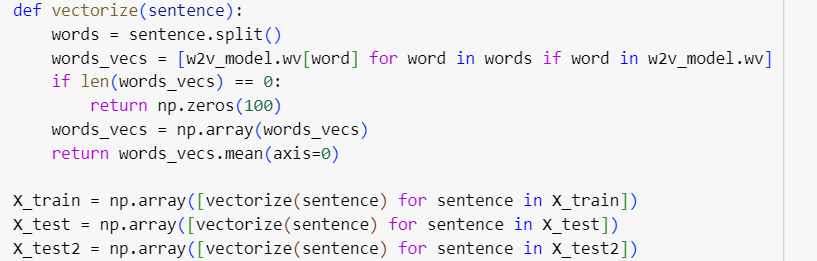
Using a script for the ki suffix, I separated the sentences with fewer than 15 words from the datasets. I labeled the separate ki suffixes in the sentences as 0 and the compound ki suffixes as 1 and wrote them in the label files.

Embedding Step:

I used the Word2Vec model in the gensim library to convert sentences into vectors. Word2Vec, specifically, is used to learn distributed representations of words based on their co-occurrence patterns in a large corpus of text. After the words are converted to vector representation. They send to the sequential models.



Convert the training, test and validation sentences into vectors one by one.



Training step:

‘de’ suffix program training:

I used 1181105 sentences for training data, 185118 sentences for Validation and 110160 sentences for test data. Training data is from wikipedia dump and tr corpus. Validation data consists of tr corpus and test data consists of wikipedia dump.

Word\_data: Wikipedia dump preprocessed dataset

Word\_data2: tr corpus preprocessed dataset

Sequential Model:

Fully connected sequential layer with 64 neurons with relu activation function as hidden layer.

1 neron layer which has activation function sigmoid as output layer.

Train the model by using epoches as 5 and batch size as 32.

Used Validation data for training mentioned above.



I used accuracy metric for calculating the models performance by using test data and also I used my test data which includes 20 sentences to see the models unseen data performance.

‘ki’ suffix program training:

I used 64510 sentences for training data, 11461 sentences for Validation and 4667 sentences for test data. Training data is from wikipedia dump and tr corpus. Validation data consists of tr corpus and test data consists of wikipedia dump.

Word\_data6: Wikipedia dump preprocessed dataset

Word\_data5: tr corpus preprocessed dataset

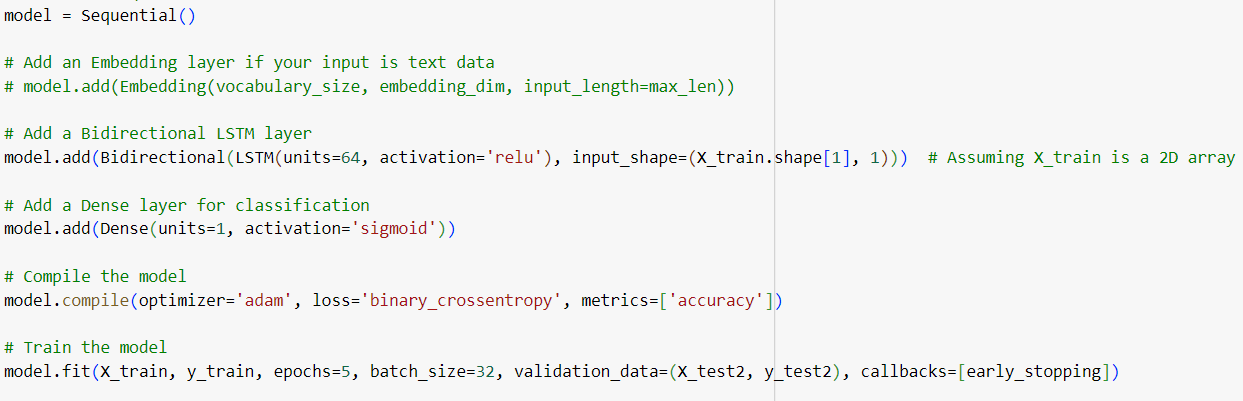
Sequential Model:

Bidirectional sequential layer with 64 neurons with relu activation function as hidden layer.

1 neron layer which has activation function sigmoid as output layer.

Train the model by using epoches as 5 and batch size as 32.

Used Validation data for training mentioned above.

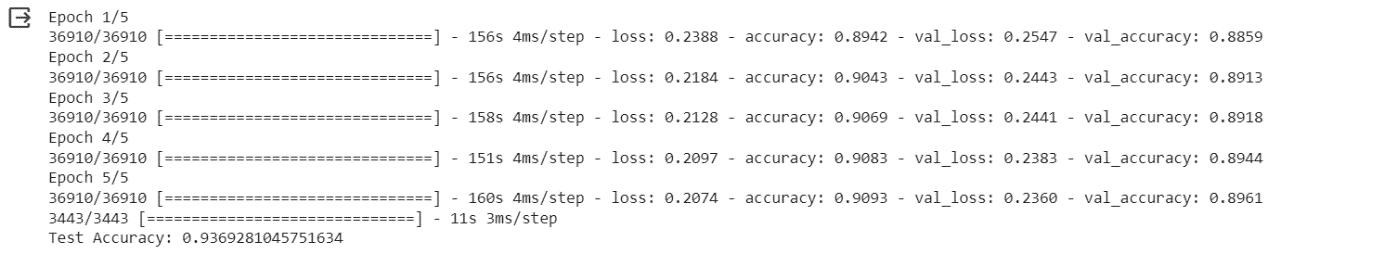


I tried to train the modhel kind of hyper parametre tunnings, learning rates and epoches. Also I train the model using kind of sequentail neural networks like RNN’s until finding an optimal solution for each suffix checkers.

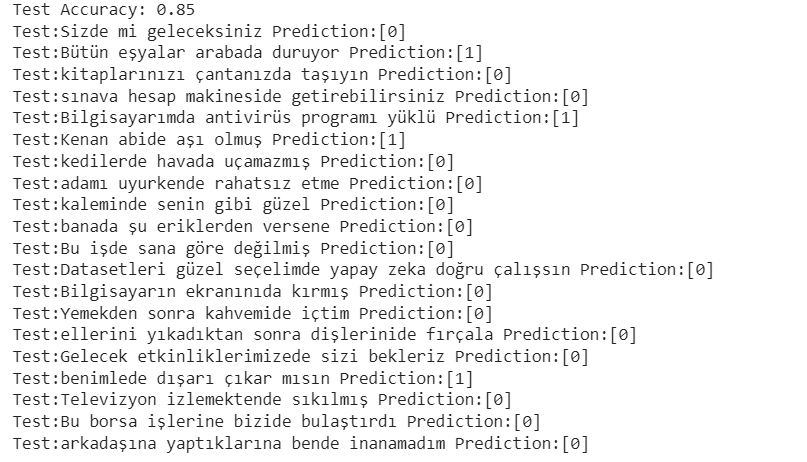
Output and Results

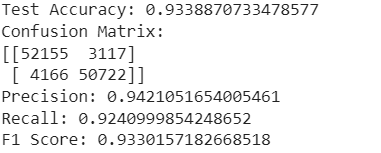
‘de’ suffix program output:

Test Performance using test data.



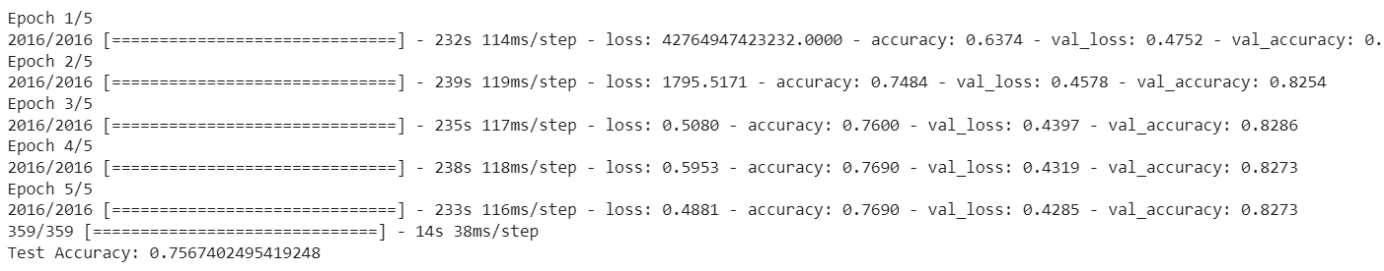
Test performance of using my sentences.





‘ki’ suffix program output:

Test Results using test data

  
Test Results using my sentences

