CSE484 - HW03 - Turkish ‘-de’ and ‘-ki’ Suffix Classifier

Emircan Demirel – 1901042674

Introduction

In this project, we aimed to develop a classifier that determines whether the Turkish suffixes "de" and "ki" should be separated or not within a given sentence. This task is crucial for disambiguating sentences in Turkish, where the placement of these suffixes can significantly affect the meaning. For instance, in the sentence "Öğrenciler de geldi" (Students also came), "de" is separated from the preceding word, while in "Öğrencilerde gelişme var" (There is progress in students), "de" is attached to the preceding word.

Data Collection

For training and testing our classifier, we collected a dataset from web sources, primarily utilizing the Wikipedia dump of the Turkish language. This dataset contains a diverse range of sentences where the "de" and "ki" suffixes appear both separated and unified. You can access the dataset from there: [Turkish Wikipedia Dump (kaggle.com)]

Data Preprocessing

In this section, we will discuss the steps taken to prepare and preprocess the dataset for training our Turkish "de" and "ki" suffix classifier. Proper data preprocessing is crucial for ensuring that the model learns effectively from the input data.

Text Processing

The collected text data goes through a series of preprocessing steps to make it suitable for training. The preprocessing steps include:

Filtering Lines: The text is initially split into lines, and lines starting with </doc> or <doc> are filtered out. These lines are typically metadata and not relevant to our task.

Sentence Splitting: The filtered text is then split into sentences. Sentences are split using the period ('.') as the delimiter. This step allows us to process each sentence individually.

Sentence Level Labelling: Each sentence is processed separately. We determine whether the suffixes "de" and "ki" should be separated or not in each sentence. If either " de " or " ki " (with spaces) are found as separate words, the label is set to 1 (indicating separation). Otherwise, it is set to 0 (indicating no separation). This labelling is crucial for supervised learning.

Text Concatenation: After processing each sentence, the sentences are reassembled into a single string for further processing. The labelled sentences and corresponding labels are stored for training and evaluation.

Data Splitting

The pre-processed data is split into two sets:

Training Data: Approximately 95% of the data is allocated for training the model. This larger portion helps the model learn patterns and relationships in the data.

Testing Data: The remaining 5% of the data is reserved for testing and evaluating the model's performance. This data is used to assess how well the model generalizes to new, unseen examples.

Tokenization and Padding

To facilitate training, we employ tokenization and padding techniques. The steps involved in this process include:

Tokenizer Training: We use the Keras Tokenizer to tokenize the training data. The tokenizer is fitted on the processed training data, allowing it to learn the vocabulary and mapping between words and numerical indices.

Converting Sentences to Sequences: The tokenized sequences of words are generated for both the training and testing data. These sequences serve as the input data for our model.

Padding Sequences: To ensure uniform input dimensions, we pad the sequences with zeros to match the length of the longest sequence observed during training. Padding guarantees that all input sequences have the same shape and can be fed into the model without issues.

Label Formatting

The labels indicating whether the suffixes "de" and "ki" should be separated are formatted as numpy arrays. This format ensures compatibility with the model's output and simplifies the evaluation process.

This meticulous data preprocessing pipeline lays the foundation for training a robust and accurate Turkish suffix classifier. The prepared data is ready for ingestion into the neural network model, enabling it to learn and make informed predictions.

Model Training

In this section, we describe the process of training our Turkish "de" and "ki" suffix classifier using the prepared dataset and the specified deep learning model architecture.

Model Architecture

Before training the model, we designed an artificial neural network architecture that is well-suited for the given classification task. The architecture is as follows:

Embedding Layer: This layer is responsible for converting words into numerical vectors. It has an input dimension equal to the vocabulary size and an output dimension of 100. These settings allow the model to learn meaningful word representations.

LSTM Layer: We use a Long Short-Term Memory (LSTM) layer to capture sequential dependencies in the input data. This layer helps the model understand the contextual information within sentences.

Dense Layers: Two dense (fully connected) layers follow the LSTM layer. The first dense layer has 64 units and uses the ReLU activation function. The second dense layer has a single unit with a sigmoid activation function, which is suitable for binary classification.

Compilation: The model is compiled with the Adam optimizer, which is an effective optimization algorithm for training neural networks. The loss function is set to binary crossentropy, which is suitable for binary classification tasks. Additionally, we monitor the training accuracy as a metric.

Training Process

The training process involves feeding the preprocessed data into the model and iteratively updating the model's weights to minimize the loss function. Here are the key training details:

Training Data: We split the dataset into training and validation sets. The training set contains a majority of the data (95%), while the validation set is used to monitor the model's performance during training.

Batch Size: We use a batch size of 32, which determines the number of examples processed in each training iteration. Batch training helps speed up the training process and can lead to more stable convergence.

Epochs: The model is trained for 10 epochs, meaning it goes through the entire training dataset 10 times. This number of epochs was chosen based on experimentation and can be adjusted as needed.

Training Progress

Throughout the training process, the model's performance metrics are monitored. These metrics include the training loss and accuracy, as well as the validation loss and accuracy. These metrics help us assess how well the model is learning and whether it is overfitting or underfitting.

```
Epoch 1/10
4835/4835 [==============================] - 156s 32ms/step - loss: 3.6199e-05 - accuracy: 1.0000 - val_loss: 0.1101 - val_accuracy: 0.9905
Epoch 2/10
4835/4835 [==============================] - 154s 32ms/step - loss: 3.3135e-06 - accuracy: 1.0000 - val_loss: 0.1463 - val_accuracy: 0.9864
Epoch 3/10
4835/4835 [==============================] - 156s 32ms/step - loss: 2.5517e-08 - accuracy: 1.0000 - val_loss: 0.1573 - val_accuracy: 0.9870
Epoch 4/10
4835/4835 [==============================] - 156s 32ms/step - loss: 3.6695e-09 - accuracy: 1.0000 - val_loss: 0.1680 - val_accuracy: 0.9874
Epoch 5/10
4835/4835 [==============================] - 158s 33ms/step - loss: 7.8013e-10 - accuracy: 1.0000 - val_loss: 0.1765 - val_accuracy: 0.9878
Epoch 6/10
4835/4835 [==============================] - 157s 32ms/step - loss: 2.9483e-10 - accuracy: 1.0000 - val_loss: 0.1824 - val_accuracy: 0.9879
Epoch 7/10
4835/4835 [==============================] - 155s 32ms/step - loss: 1.7490e-10 - accuracy: 1.0000 - val_loss: 0.1860 - val_accuracy: 0.9880
Epoch 8/10
4835/4835 [==============================] - 155s 32ms/step - loss: 1.2439e-10 - accuracy: 1.0000 - val_loss: 0.1886 - val_accuracy: 0.9880
Epoch 9/10
4835/4835 [==============================] - 154s 32ms/step - loss: 9.7778e-11 - accuracy: 1.0000 - val_loss: 0.1907 - val_accuracy: 0.9880
Epoch 10/10
4835/4835 [==============================] - 154s 32ms/step - loss: 8.0345e-11 - accuracy: 1.0000 - val_loss: 0.1923 - val_accuracy: 0.9879
```

Performance Evaluation

In this section, we present the results of evaluating our Turkish "de" and "ki" suffix classifier using a dedicated testing dataset. The primary goal is to assess the model's ability to make accurate predictions on previously unseen data.

Test Data Preparation

Before evaluating the model, we preprocess the testing data in the same way as the training and validation data. This includes tokenizing the test sentences, padding the sequences to ensure uniform length, and formatting the test labels.

Tokenization: We tokenize the test sentences using the same tokenizer that was used during training. This ensures consistency in the encoding of words into numerical vectors.

Padding: The sequences are padded to match the maximum sequence length observed during training. Padding ensures that all input sequences have the same dimensions, allowing them to be fed into the model.

Label Format: The test labels are converted into the correct format, represented as numpy arrays. These labels indicate whether the suffixes in the test sentences should be separated (1) or not separated (0).

Evaluation Metrics

To measure the performance of our classifier, we employ two key metrics:

Loss: The loss is a measure of how well the model's predictions match the actual labels in the testing dataset. It quantifies the error between the predicted and true values.

Accuracy: Accuracy represents the proportion of correctly classified instances in the testing dataset. It provides a clear indication of the model's overall performance in terms of separating and not separating suffixes.

Results

Upon evaluating the model using the testing dataset, we obtained the following results:

Loss: The model achieved a loss value of approximately 0.1822. This value indicates the average error in the model's predictions.

Accuracy: The accuracy of the model is approximately 98.82%. This high accuracy suggests that the model is proficient at distinguishing between Turkish suffixes that should be separated and those that should not.

```
276/276 [==============================] - 4s 13ms/step - loss: 0.1822 - accuracy: 0.9882
[0.18216589093208313, 0.9882139563560486]
```

Results

```
1/1 [==============================] - 0s 31ms/step
Sentence: 'Kitap masanın üstünde duruyordu.'
Prediction: Should be unified

1/1 [==============================] - 0s 27ms/step
Sentence: 'Arkadaşlar da gelmiş.'
Prediction: Should be unified

1/1 [==============================] - 0s 24ms/step
Sentence: 'O günki hava çok güzeldi.'
Prediction: Should be unified

1/1 [==============================] - 0s 27ms/step
Sentence: 'Derslerinde başarılı bir öğrenciydi.'
Prediction: Should be unified

1/1 [==============================] - 0s 28ms/step
```

```
Sentence: 'Herkesin de bir hikayesi var.'
Prediction: Should be separated

1/1 [==============================] - 0s 25ms/step
Sentence: 'Kapıdaki kimdi?'
Prediction: Should be unified

1/1 [==============================] - 0s 25ms/step
Sentence: 'Yazdıklarını okudum da çok beğendim.'
Prediction: Should be unified

1/1 [==============================] - 0s 27ms/step
Sentence: 'Bu işin sonu nereye varacak bilmiyorum ki.'
Prediction: Should be unified

1/1 [==============================] - 0s 25ms/step
Sentence: 'Olanlar olmuştu artık, yapacak bir şey yoktu.'
Prediction: Should be unified

1/1 [==============================] - 0s 26ms/step
Sentence: 'Sen de mi Brutus?'
Prediction: Should be separated

1/1 [==============================] - 0s 25ms/step
Sentence: 'Annesi de buradaydı.'
Prediction: Should be separated

1/1 [==============================] - 0s 23ms/step
Sentence: 'Bize de haber verir misin?'
Prediction: Should be separated

1/1 [==============================] - 0s 30ms/step
Sentence: 'Kitaplarda masanın üstündeydi.'
Prediction: Should be unified

1/1 [==============================] - 0s 24ms/step
Sentence: 'Yemekler hazırlandı ki.'
Prediction: Should be unified

1/1 [==============================] - 0s 24ms/step
Sentence: 'Evimizde duruyor.'
Prediction: Should be unified

1/1 [==============================] - 0s 36ms/step
Sentence: 'O günde çok güzeldi.'
Prediction: Should be unified

1/1 [==============================] - 0s 28ms/step
Sentence: 'Yarın da gelecekler.'
Prediction: Should be unified

1/1 [==============================] - 0s 26ms/step
Sentence: 'Gömlekteki leke çıktı.'
Prediction: Should be unified

1/1 [==============================] - 0s 26ms/step
Sentence: 'Dün de aynıydı.'
Prediction: Should be separated
```

```
1/1 [==============================] - 0s 28ms/step
Sentence: 'Herkesinki farklı'
Prediction: Should be unified
```