Classifying Turkish “de” and “ki” Suffixes Using NLP Techniques

Ayşe Begüm Nur

January 29, 2024

Abstract

This project focuses on the application of natural language processing techniques to address a specific challenge in the Turkish language: the classification of ”de” and ”ki” as either separate words or suffixes in sentences. Turkish, known for its agglutinative nature and rich morphology, presents unique challenges in computational linguistics. The primary objective was to develop an automated classifier using deep learning models to accurately distinguish between the separated and non-separated usage of these particles. Leveraging a dataset extracted from the Turkish Wikipedia dump, the project employed preprocessing techniques including tokenization, padding, and class balancing, followed by training a neural network model using TensorFlow and Keras. The model showed promising results, achieving a significant level of accuracy in classification tasks. The findings of this project contribute to the growing field of Turkish text processing and open avenues for further research in improving the model’s accuracy and adaptability to different text sources.

1 Introduction

1.1 Background

Natural Language Processing (NLP) has become an essential part of understanding and processing human languages. With the advent of machine learning and AI, NLP techniques have seen significant advancements. This project focuses on the Turkish language, which presents unique challenges due to its agglutinative nature and rich morphology.

1.2 Project Significance

The project aims to address a specific challenge in Turkish text processing: classifying the usage of "de" and "ki". These particles can either be separate words or suffixes in Turkish sentences, and their correct classification is crucial for accurate language understanding and processing.

2 Problem Definition

In Turkish, "de" and "ki" can be used in various forms, which often leads to ambiguities in natural language processing tasks. The core problem this project addresses is the classification of these particles—whether they should be separate or attached to a word. This issue is particularly challenging due to the flexible and context-dependent nature of these particles in Turkish grammar. The project's goal is to develop a reliable method to automate their correct classification, which is a significant step in processing Turkish text effectively.

3 Methodology

This project employs a machine learning approach using TensorFlow and Keras, popular libraries in the field of deep learning. The choice of these technologies is motivated by their flexibility, ease of use, and wide support for various neural network architectures.

3.1 Model Architecture

The model architecture is designed to efficiently process text data and make accurate classifications. It includes the following key components:

- An Embedding layer to convert input sequences into dense vectors of fixed size.

- LSTM (Long Short-Term Memory) layers to capture the sequential nature and dependencies of the text.
- Dense layers for classification purposes.

Each of these layers plays a crucial role in the model's ability to learn from the textual data effectively.

4 Data Preprocessing

The primary data source for this project is the Turkish Wikipedia dump, which provides a rich and diverse text corpus in the Turkish language.

4.1 Preprocessing Steps

The preprocessing of the data involved several key steps:

1. **Sentence Extraction and Initial Labeling:** Sentences containing "de" and "ki" were extracted from the corpus, and initial labeling was done based on their usage as separate words or suffixes.
2. **Merging Separated Suffixes:** For sentences where "de" and "ki" were used as separate words, these suffixes were merged with the preceding word. This step was crucial for maintaining contextual integrity in the dataset.
3. **Handling Suffixes:** Special attention was given to correctly handle "de" and "ki" when used as suffixes.
4. **Tokenization and Padding:** The Tokenizer from Keras was used to convert text data into sequences, which were then padded to ensure uniform length.
5. **Class Balancing:** Due to the imbalance in the dataset, undersampling techniques were applied to ensure an even distribution of classes.

This preprocessing strategy was key in preparing the dataset for effective learning by the neural network model.

5 Model Implementation

The model architecture was meticulously designed to address the specific challenges posed by the Turkish language, particularly in the context of "de" and "ki" suffix separation.

5.1 Model Architecture

The model is built using the Keras library with the following layers:

```
model = Sequential([
    Embedding(VOCAB_SIZE, 32, input_length=MAX_SEQUENCE_LENGTH),
    Bidirectional(LSTM(64, return_sequences=True)),
    Dropout(0.5),
    Flatten(),
    Dense(64, activation='relu'),
    Dropout(0.5),
    Dense(1, activation='sigmoid')
])
```

Figure 1: Model Architecture

- **Embedding Layer:** The first layer is an Embedding layer with a vocabulary size of `VOCAB_SIZE` set to 20000 and an output dimension of 32. It processes the input sequences of maximum length `MAX_SEQUENCE_LENGTH`, set to 100.
- **Bidirectional LSTM Layer:** The next layer is a Bidirectional LSTM with 64 units. The bidirectional approach is crucial for understanding the context from both directions in a sentence, enhancing the model's ability to capture nuances in language patterns.
- **Dropout Layers:** Two Dropout layers, each with a dropout rate of 0.5, are added following the Bidirectional LSTM and the first Dense layer. These layers help in preventing overfitting by randomly setting a fraction of input units to 0 during training.
- **Flatten Layer:** A Flatten layer follows, converting the multi-dimensional output of the previous layers into a one-dimensional array for input into the dense layers.
- **Dense Layers:** After flattening, the model includes two Dense layers. The first Dense layer has 64 units with a 'relu' activation function, and the final output layer has a single unit with a 'sigmoid' activation function for binary classification.

5.2 Parameters and Hyperparameters

The model uses the Adam optimizer and binary cross-entropy as the loss function. The model was trained for 10 epochs with a validation split of 20%.

5.3 Training Process

The model's training involved feeding the balanced training dataset, which underwent the preprocessing steps mentioned earlier. The training process aimed to optimize the model's parameters to accurately classify whether "de" and "ki" should be separated in given Turkish sentences.

6 Results and Accuracy

6.1 Model Performance

The model's performance was evaluated based on its ability to accurately classify the test dataset. The primary metrics used for evaluation were accuracy, precision, recall, and F1-score. The results were as follows:

```
Model Evaluation:
Accuracy: 0.7212, Precision: 0.7196, Recall: 0.7196, F1-Score: 0.7196
```

Figure 2: Model Accuracy Results

6.2 Interpretation of Results

The results indicate a competent level of performance by the model in distinguishing between the separated and non-separated usage of "de" and "ki" in Turkish sentences. While the accuracy is substantial, there is still room for improvement, especially in terms of recall. The balance between precision and recall, as indicated by the F1-score, shows the model's reasonable effectiveness in handling both types of classification errors.

```python
test_sentences = [
    "Okuldan eve dönünce derslerinide yapacak.",
    "Bu kitapları okuduktan sonra seninkileride alacağım.",
    "Arkadaşlarıyla parkta oynadı ve yorulduklarında eve de döndüler.",
    "Anneside bu durumdan memnun kalmıştı.",
    "Oyun oynarken zamanın nasıl geçtiğinide anlamadılar.",
    "Öğretmende sorunun cevabını bilmiyordu.",
    "Kedileride yanlarına alarak yola çıktılar.",
    "Sende mi geleceksin?",
    "Bu konudaki görüşlerini merak ediyorum.",
    "Dün gördüğümüz filmdeki karakter çok etkileyiciydi.",
    "Karşıdaki evdeki ışıklar hâlâ yanıyor.",
    "Bu konudaki düşüncelerini değiştirebilir miyim?",
    "Toplantıdaki herkes fikrini açıkça belirtti.",
    "Okuldaki sınavlarımız haftaya başlayacak.",
    "Yoldaki engelleri aşarak hedefine ulaştı.",
    "Dolaptaki elbiseleri değiştirmek istiyorum.",
    "Bu akşamki planlarınızı iptal etmek zorunda kaldık.",
    "Balkondaki çiçekler susuz kalmış.",
    "Kütüphanedeki kitaplar çok eski.",
    "Bahçedeki ağaçların altında piknik yapalım."
]
```

Figure 3: Example Sentences For Testing

```
1/1 [==============================] - 0s 441ms/step
Sentence: Okuldan eve dönünce derslerinide yapacak. - Not Separate
Sentence: Bu kitapları okuduktan sonra seninkileride alacağım. - Not Separate
Sentence: Arkadaşlarıyla parkta oynadı ve yorulduklarında eve de döndüler. - Separate
Sentence: Anneside bu durumdan memnun kalmıştı. - Separate
Sentence: Oyun oynarken zamanın nasıl geçtiğinide anlamadılar. - Not Separate
Sentence: Öğretmende sorunun cevabını bilmiyordu. - Separate
Sentence: Kedileride yanlarına alarak yola çıktılar. - Separate
Sentence: Sende mi geleceksin? - Separate
Sentence: Bu konudaki görüşlerini merak ediyorum. - Not Separate
Sentence: Dün gördüğümüz filmdeki karakter çok etkileyiciydi. - Separate
Sentence: Karşıdaki evdeki ışıklar hâlâ yanıyor. - Not Separate
Sentence: Bu konudaki düşüncelerini değiştirebilir miyim? - Separate
Sentence: Toplantıdaki herkes fikrini açıkça belirtti. - Not Separate
Sentence: Okuldaki sınavlarımız haftaya başlayacak. - Not Separate
Sentence: Yoldaki engelleri aşarak hedefine ulaştı. - Not Separate
Sentence: Dolaptaki elbiseleri değiştirmek istiyorum. - Not Separate
Sentence: Bu akşamki planlarınızı iptal etmek zorunda kaldık. - Separate
Sentence: Balkondaki çiçekler susuz kalmış. - Separate
Sentence: Kütüphanedeki kitaplar çok eski. - Not Separate
Sentence: Bahçedeki ağaçların altında piknik yapalım. - Separate
```

Figure 4: Test Cases Results

6.3 Model Limitations and Future Work

Though the model demonstrates promising results, it is not without limitations. Future work could focus on further tuning the model, experimenting with different architectures, or incorporating a larger and more diverse dataset to enhance its accuracy and generalizability. Additionally, exploring advanced natural language processing techniques could further refine the model's performance.