SENTIMENT ANALYSIS OF TURKISH TEXT

PROJECT REPORT

ADVISOR: SUZAN ÜSKÜDARLI

ASLI ÖZTÜRK | SWE 599 | 2019-05-20

Contents

1 INTRODUCTION & MOTIVATION..2
2 REQUIREMENTS 2
3 SYSTEM DESIGN..3
3.1 MOCKUPS 3
3.2 USE CASE DIAGRAM 4
4 IMPLEMENTATION 5
4.1 MODEL IMPLEMENTATION 5
4.2 WEBSITE IMPLEMENTATION 7
5 CONCLUSION...9
6 REFERENCES...9

Figure 1: Sentiment Analysis Webpage Mockup..3
Figure 2: Use Case Diagram of System...4
Figure 3: Model Structure Step by Step...5
Figure 4: Model Summary 6
Figure 5: Website Implementation 7
Figure 6: Positive Demo of Analysis..8
Figure 7: Negative Demo of Analysis...8

1 INTRODUCTION & MOTIVATION

Natural Language Processing is a computational method for making human language understandable to computers. Sentiment analysis which is a subset of natural language processing, is about analyzing people's opinions or emotions towards particular products or brands. It helps measuring positive or negative emotions of customers so that companies may have an opinion about their products or services' social perception.

Nowadays, deep learning models have been applied in the field of NLP, especially in sentiment analysis. We will also use deep learning techniques to obtain more accurate results.

We will use customer comments from hepsiburada.com website. There is star-based rating system and we will accept 1 or 2 stars as negative (0), 4 or 5 stars as positive (1), and we will not use 3 starred comments because it is neither positive nor negative.

2 REQUIREMENTS

1. The user shall be able to analyze Turkish text as positive or negative.
   1.1. The user shall be able to enter Turkish text to input area.
   1.2. The user shall be able to see a graphical illustration of the text to analyze how likely it is used positive or negative.
2. The system shall be user friendly.
3. The system shall be to use Deep Learning to analyze texts.
   3.1. The system shall be able to use Recurrent Neural Network (RNN) algorithm.
      3.1.1. The system shall be able to use Keras to create RNN model.
4. The system shall be able to use 80% of data as Training Data, 20% of data as Testing Data.
5. The system shall be able to use trained model on web.
6. The system shall be able to tag outcome as positive if the result is greater or equal than 0.5, otherwise it will be tagged as negative.

3 SYSTEM DESIGN

3.1 MOCKUPS

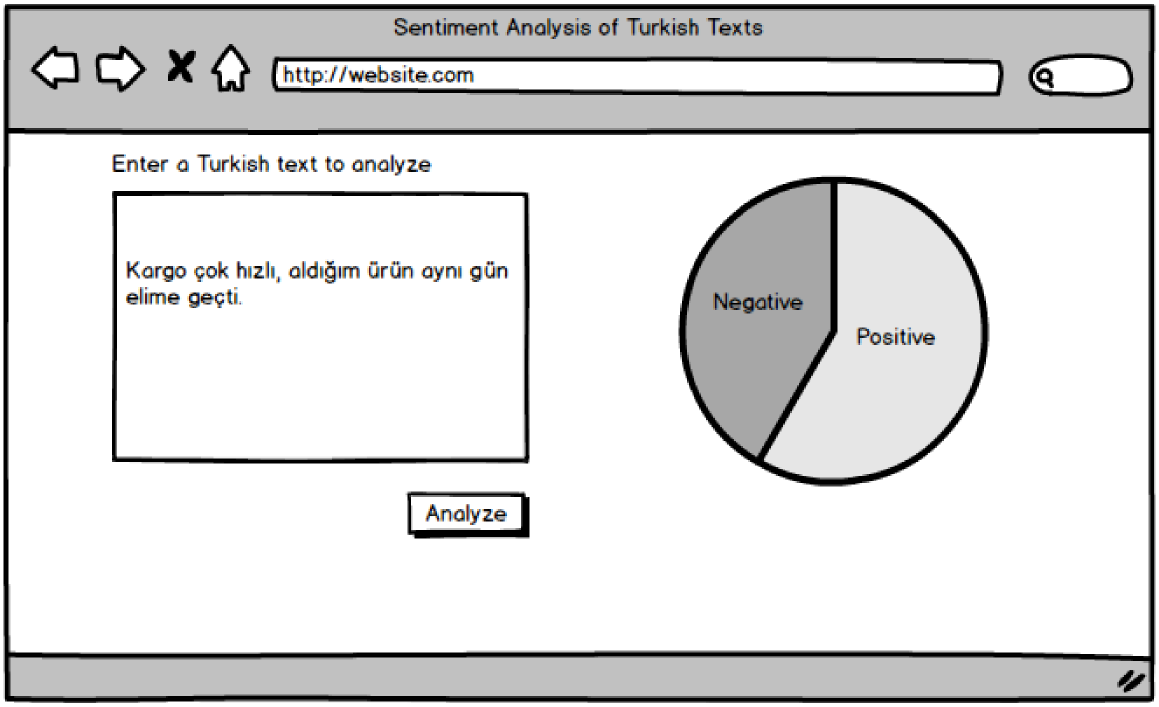

Figure 1: Sentiment Analysis Webpage Mockup

When user enters a text and clicks on “Analyze” button, pie chart will be generated automatically to show percentages of positive and negative perception.

3.2 USE CASE DIAGRAM

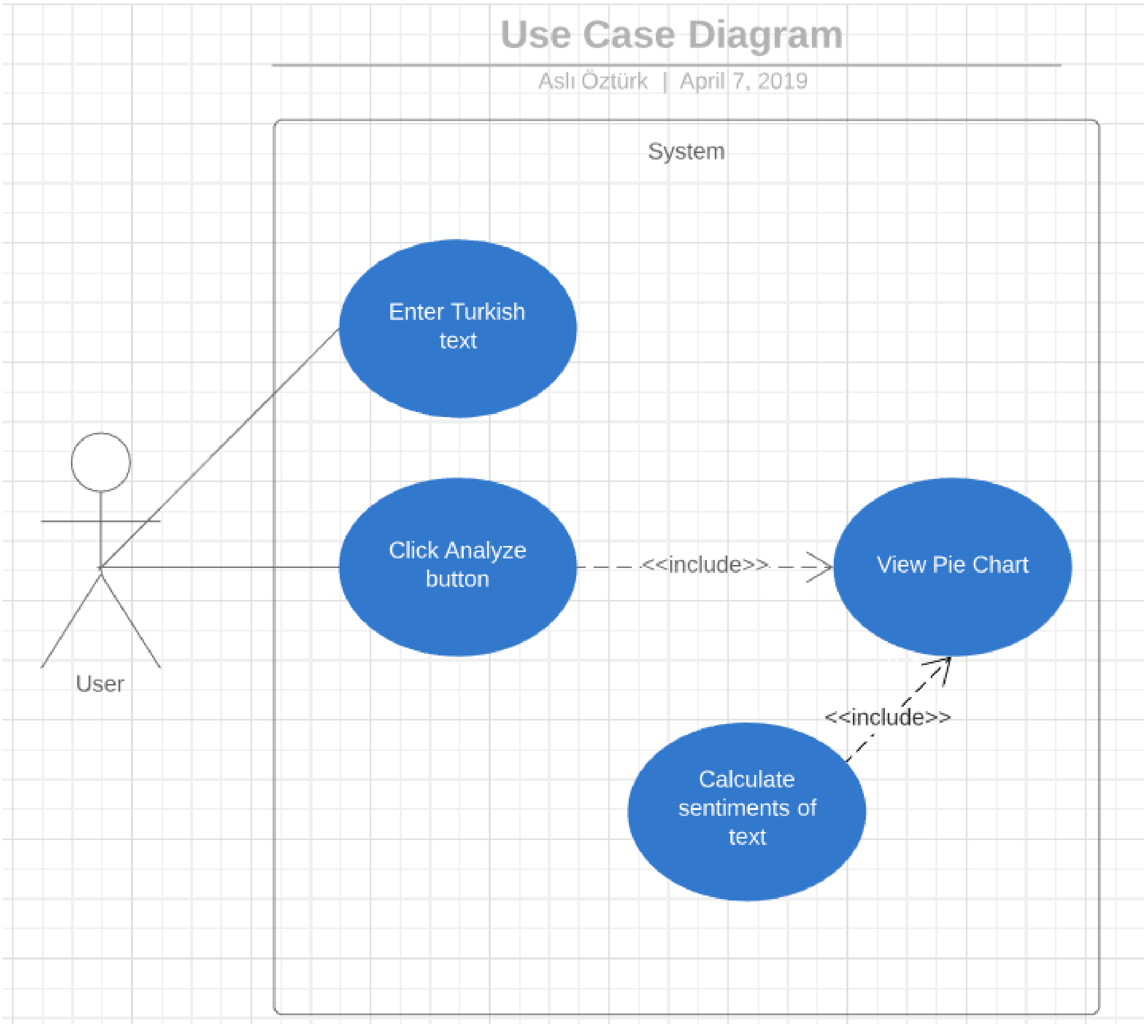

Figure 2: Use Case Diagram of System

4 IMPLEMENTATION

4.1 MODEL IMPLEMENTATION

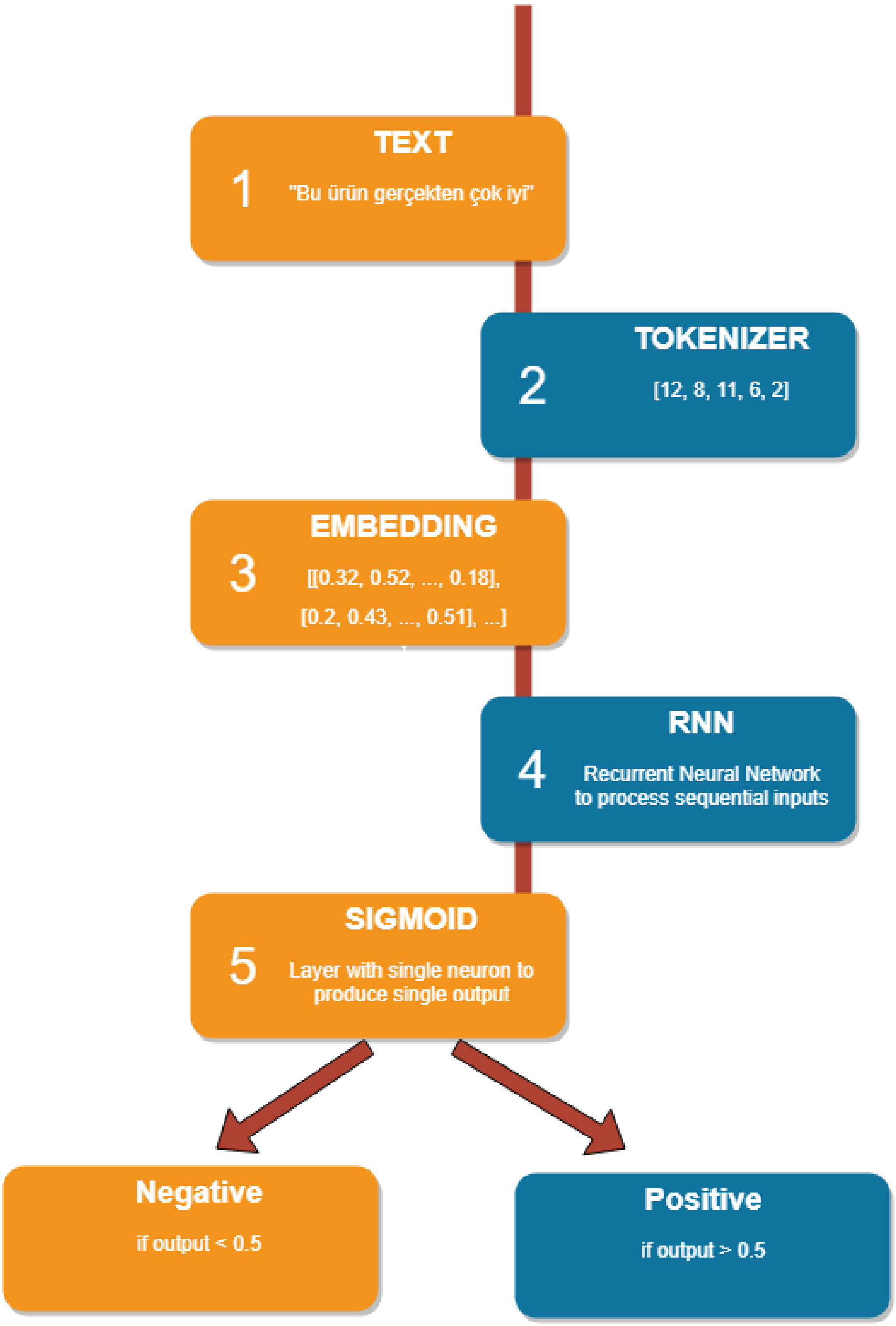

Figure 3: Model Structure Step by Step

Recurrent Neural Network (RNN) has been proved to be a good solution for Natural Language Processing tasks. However, there are some limitations like vanishing gradient problem in Basic RNN, so different RNN types have been developed like LSTM and GRU. We used LSTM in our model, created with Keras.

Before we create the model, we need to preprocess our text. First, we take the text and tokenize it with integer numbers starting from 1. And then, we need to create an Embedding Layer. In this layer, we create a word vector for every token (word). When this step is completed, we build our RNN model with Keras. Our RNN model consists of 3-Layered LSTM architecture. Then, we add a Dense Layer which consists of a single neuron has a Sigmoid activation so that we can produce a single output, either positive or negative.

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_layer (Embedding)  (None, 59, 50)            500000
_________________________________________________________________
lstm_1 (LSTM)                (None, 59, 16)            4288
_________________________________________________________________
dropout_1 (Dropout)          (None, 59, 16)            0
_________________________________________________________________
lstm_2 (LSTM)                (None, 59, 8)             800
_________________________________________________________________
dropout_2 (Dropout)          (None, 59, 8)             0
_________________________________________________________________
lstm_3 (LSTM)                (None, 4)                 208
_________________________________________________________________
dropout_3 (Dropout)          (None, 4)                 0
_________________________________________________________________
dense_1 (Dense)              (None, 1)                 5
=================================================================
Total params: 505,301
Trainable params: 505,301
Non-trainable params: 0
_________________________________________________________________
```

Figure 4: Model Summary

We can see the model summary in Figure 4. With this architecture, we train the model with 10 epochs and 256 batch size. After training is completed, we have nearly 95.11% accuracy.

4.2 WEBSITE IMPLEMENTATION

Trained model is integrated to website using Flask as a web server.

```python
from flask import Flask, render_template, request
import tensorflow as tf
from keras.models import load_model
from tensorflow.python.keras.preprocessing.text import Tokenizer
from tensorflow.python.keras.preprocessing.sequence import pad_sequences
from keras.backend import clear_session
import pandas as pd

clear_session()

app = Flask(__name__)

# load the Model from file
nlp_model = load_model('lstm_nlp1.h5')

global graph
graph = tf.get_default_graph()

dataset = pd.read_csv('ecommercereviews.csv')
data = dataset['Review'].values.tolist()


def predict(texts):
    tokenizer = Tokenizer(num_words=10000)
    tokenizer.fit_on_texts(data)
    tokens = tokenizer.texts_to_sequences(texts)
    tokens_pad = pad_sequences(tokens, maxlen=59)
    with graph.as_default():
        prediction = nlp_model.predict(tokens_pad)[0][0]
    return prediction


@app.route('/', methods=['GET', 'POST'])
def home():
    in_text = request.values.get('text_input')
    arr = [in_text]
    # if input is provided process else show default page
    if request.method == 'POST':
        result = predict(arr)
        return render_template('home.html', result=result, text=in_text)
    else:
        return render_template('home.html')


if __name__ == '__main__':
    app.run(debug=False)
```

Figure 5: Website Implementation

Trained model is saved as Keras model, named as lstm_nlp1.h5. Text that comes from request, goes to predict() method that returns a value between 0 and 1. If the result value greater than 0.5, than it can be accepted as positive, else it is negative.

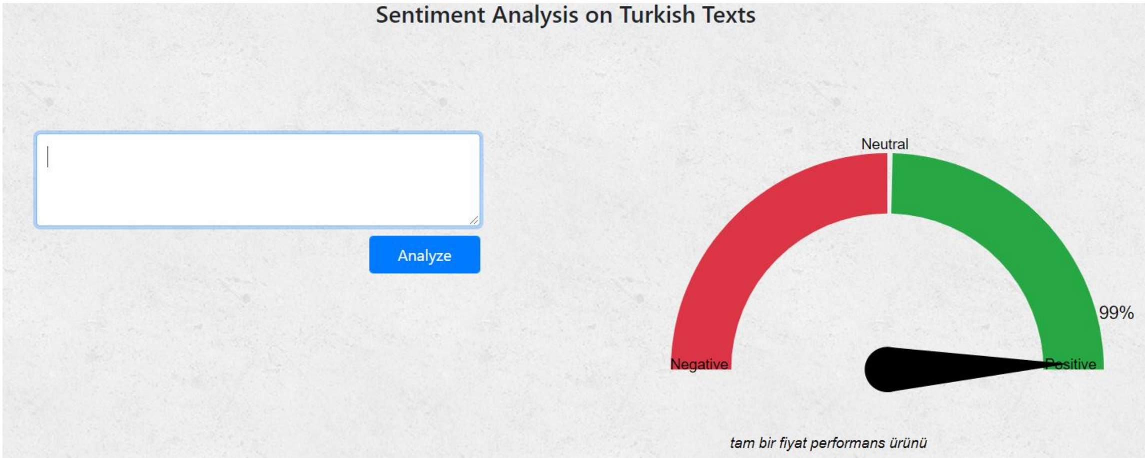

Figure 6: Positive Demo of Analysis

In Figure 6, we have tested our model with “tam bir fiyat performans ürünü” sentence. As a result, it returned 99% positive.

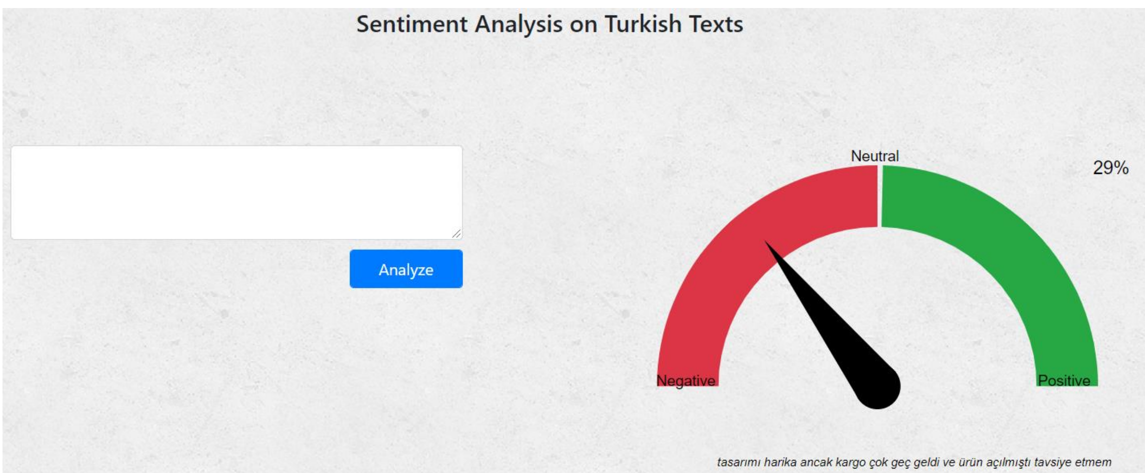

Figure 7: Negative Demo of Analysis

In Figure 7, we have tested our model with “tasarımı harika ancak kargo çok geç geldi ve ürün açılmıştı tavsiye etmem” sentence. As a result, it returned 29% negative.

5 CONCLUSION

NLP is currently very promising area in machine learning. From market analysis which contains opinions (sentiments) of customers to chatbots and neural machine translation, NLP can be applied to many contexts. In this study, I am happy to share my work on Sentiment Analysis on Turkish Texts. There are many researches on NLP, however, very few of them is about Turkish language.

For the future studies, incorrect results should be investigated and model should be trained with more data to increase accuracy and reliability.

6 REFERENCES

1. Eisenstein J. (2018) Natural Language Processing
2. Zhang L., Wang S., Liu B. Deep Learning for Sentiment Analysis: A Survey
3. Keras Tutorial: https://keras.io/
4. Oflazer K. Turkish Natural Language Processing