National Research University
Higher School of economics
Faculty of Computer Science

Bachelor’s Programme “HSE and University of London Double Degree Programme in Data Science and Business Analytics”

Software project report on the topic: Sentiment Analysis based on Glamping Reviews from TripAdvisors: Developing a Decision - Making Assistant”

Student: Morozova Milena Artemovna

Supervisor: Chuvulina Anna Andreevna

Date: 28.05.2022

Contents

1 Introduction

1.1 Abstract

Over the past two years, the glamping industry in Russia has made a sharp jump. Glampings are luxury camp sites that are built in nature or even in surrounding places. But ecotourism, which is also called green tourism, involves a complete disconnection from the noise of civilization — a rejection of the cult of comfort and mass communications. Most people, interested in glamping in Russia, are trying to find reviews on the Internet for places they would like to visit. However, reading a thousand reviews is a very long and labor-intensive process that not all people like.

Sentiment analysis allows you to determine the emotional color of a multi-line review in a matter of minutes, which greatly simplifies the choice of a glamping site for many Russian tourists. As part of our project, we use sentimental analysis to calculate the approximate rating of the place depending on the number of reviews and their emotional coloring.

1.2 My role in the software project

My work was dedicated to development of a graphical user interface application, which would be as simple and understandable as possible to each user. The development of the interface includes: creating application windows and minimal application design.

Moreover, in order for the glamping rating to be reflected not only by a number, we decided to conduct a frequency analysis of reviews of each place and make a ”word cloud” reflecting the most frequently used words in the texts of particular reviews. Performing frequency analysis, creating word clouds and adding them to the app was also part of my job.

As a part of collective work we decided to construct a high accuracy mathematical model, which analyses reviews and divides them into 2 categories: positive and negative. For increasing the accuracy of this model, my groupmate performed work on cleaning, tokenization, lemmatization of text, as well as filtering and removing words that do not give emotional color to the text.

1.3 Instruments

1. Anaconda, Jupiter Notebook, Pycharm
2. Python libraries: numpy, pandas, matplotlib, sklearn, scipy,wordCloud
3. Microsoft Excel
4. PyQt, Qt Designer
5. TextBlob
6. TripAdvisor

1.4 Main goal

The main goal of our application is to perform the sentiment analysis and to calculate rating of glamping places in Russia in order to help tourists with their choice.

2 Review and comparative analysis

2.1 Review of abstracts which you used for project and review of analogues

The project is dedicated to learning how to filter the text and use it in different purposes, also we have learned how to increase the accuracy of mathematical models. To create the app we need to use our knowledge of GUI development and some kind of design. We have not found any exact analogues to the application we created. Of course, there are many applications that compare, for example, hotels or air tickets at a price, but the purpose of our application is to assess a specific place and acquaint the user with the words and phrases that visitors of glamping most often used to describe a particular resort. Thanks to this approach, the user will be able to quickly and accurately determine what place he will like without rereading hundreds of reviews on the Internet. And also, since glamping is a relatively new direction for Russia in the field of tourism, there are few reviews for glamping, they are scattered throughout the Internet, and our application allows you to structure all reviews so that the user can find out all the information he needs in one place just by clicking on the button. Now I will compare our application with an application called "Twitter Sentiment analyzer". It is a distant analogue of our application and analyzes the emotional coloring of statements on Twitter.

Characteristics	**Twitter Sentiment Analyzer**	**a Decision-Making Assistant**
Paid/Free	Paid	Free
Visualization	Short video clips	Calculated rating and word clouds
Used for	Analyzing the product reviews	Analyzing glamping places

2.2 List of key words

`Sentiment Analysis` — determining the polarity of emotional assessments in the text under study, which contains opinions, judgments, emotions, the author's attitude to entities, personalities, questions, events, topics and their attributes.

`A machine learning model` — a file that is trained to recognize certain types of patterns. You train a model based on a dataset, providing it with an algorithm that it can use to analyze and learn from that data.

`Graphical User Interface(GUI)` — a system of tools for user interaction with electronic devices, based on the presentation of all system objects and functions available to the user in the form of graphic components of the screen (windows, icons, menus, buttons, lists, etc.).

`A recurrent neural network(RNN)` — is a class of artificial neural networks in which connections between nodes form a directed or undirected graph along a time sequence. This allows her to exhibit temporary dynamic behavior. Derived from direct communication neural networks, RNNs can use their internal state (memory) to process sequences of variable-length input data.

`Dataset for machine learning` — processed and structured information in tabular form.

`Long short-term memory (LSTM)` — a special type of recurrent neural network architecture capable of learning long-term dependencies.

3 Selection of methods, algorithms, and models for project implementation

3.1 Information about the given data

Firstly, I got unfiltered data called 'data.csv' in the format of .csv which I opened in Microsoft Excel. The dataset contains data from "TripAdvisor" — the website, containing the reviews about hotels, restaurants, glamping etc. In the original unfiltered csv file, we have two columns: the name of the place and the feedback about it. Working with the unfiltered data set, we receive information that we need later for work, namely: the number of places for glamping and their names and the number of reviews for each specific place.

3.2 Purifying data for analysis and processing

Any text data in its raw material form cannot be analyzed by NLP libraries. This data must be cleaned using various data processing techniques. This part of work was done by my groupmate, so I will describe it quite briefly in my report and focus on describing my work.In this project, we used standard processing methods used for sentimental analysis, namely:

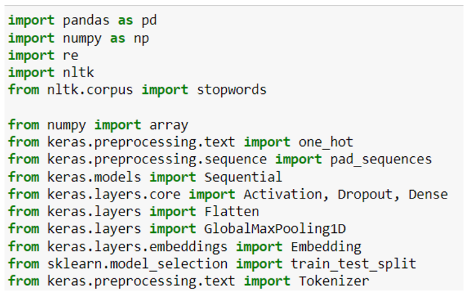
- `Eliminate HTML Tags:` Unstructured Text Contains a lot of "noise" and therefore we need to remove HTML tags, if any.
- `Eliminate special characters:` Any non-alphanumeric characters in the text must be removed.
- `Deleting stop words:` Stop words are commonly used words (i.e., "the," "a," "an") that do not add meaning to a sentence and can be ignored without significantly affecting the meaning of the sentence.

- `Lemmatization:` We implemented lemmatization with Spacy so that we could count the appearance of each word. Lemmatization removes grammatical time and transforms each word into its original form. Another way to convert words to its original form is called stemming. Although stemming takes the language root of the word, lemmatization takes the word into its original lemma. For example, if we did contrary to the word ”apples,” then we would get ”appl,” while lemmatization would give us ”apple.” Therefore, we used lemmatization instead of stemming, as it is much easier to interpret.

3.3 Preparing training data for the future model

To prepare the data and to build the model we are using Keras. Keras is a deep learning library, which is a high-level API written in Python and capable of running on top of TensorFlow, Theano or CNTK. It was designed with the expectation of rapid learning. To train the model, we will use two sets of data that we will later combine - one contains only positive statements, the second — negative ones.

Let’s start by importing the necessary dependencies to preprocess the data and build the model:



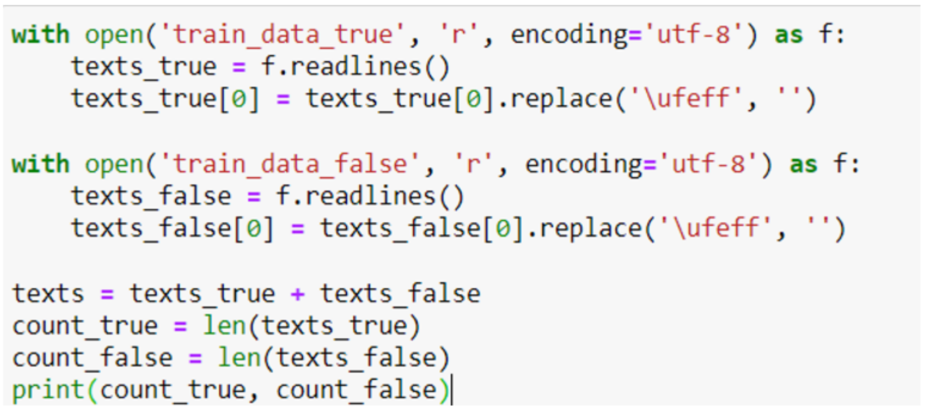
```python
import pandas as pd
import numpy as np
import re
import nltk
from nltk.corpus import stopwords

from numpy import array
from keras.preprocessing.text import one_hot
from keras.preprocessing.sequence import pad_sequences
from keras.models import Sequential
from keras.layers.core import Activation, Dropout, Dense
from keras.layers import Flatten
from keras.layers import GlobalMaxPooling1D
from keras.layers.embeddings import Embedding
from sklearn.model_selection import train_test_split
from keras.preprocessing.text import Tokenizer
```

Now let’s download texts with positive and negative statements (they will be in separate files: train_data_true and train_data_false). Next, we will immediately combine the lists of statements into a single list and calculate the length of each of them(Look at).Then we need to break these statements into words. To do this, we will use the `Tokenizer` tool and assume that the maximum number of words will be 2000.

This parameter has one significant plus: of all the words found, we leave 1999 most frequently encountered, that is, we discard rare words that are not particularly needed when learning the neural network.

For example, with a large training sample, a good choice would be a value within 15000-20000. While splitting the text into words, we remove all punctuation signs and unnecessary characters, and also bring all our words to the lower case.

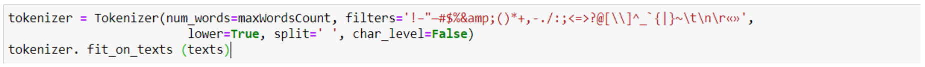


```
with open('train_data_true', 'r', encoding='utf-8') as f:
    texts_true = f.readlines()
    texts_true[0] = texts_true[0].replace('\ufeff', '')

with open('train_data_false', 'r', encoding='utf-8') as f:
    texts_false = f.readlines()
    texts_false[0] = texts_false[0].replace('\ufeff', '')

texts = texts_true + texts_false
count_true = len(texts_true)
count_false = len(texts_false)
print(count_true, count_false)
```

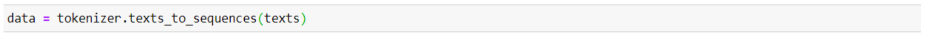
Figure 1: Downloading texts



```
tokenizer = Tokenizer(num_words=maxWordsCount, filters='!–"—#$%&amp;()*+,-./:;<=>?@[\\]^_`{|}~\t\n\r«»',
                      lower=True, split=' ', char_level=False)
tokenizer. fit_on_texts (texts)
```

Figure 2: Breaking into words

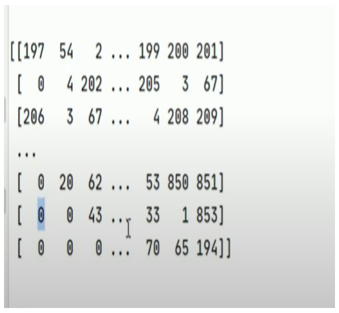
Next, we convert the text into a sequence of numbers in accordance with the resulting dictionary. To do this, I used a special method of the Tokenizer class:



```
data = tokenizer.texts_to_sequences(texts)
```

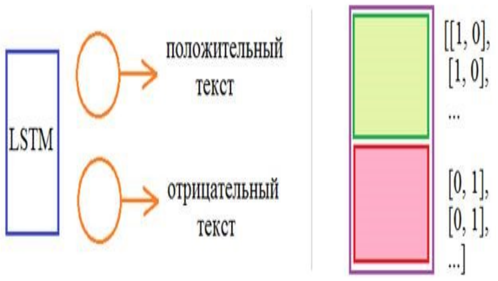
At the output, we get a two-dimensional array of numbers of the numpy object. Now, we need to align all these vectors to a length equal to the maximum length of the text. To do this, another built-in pad_sequences method is used, which trims the data array to a length of max_text_len and adds zeros for short vectors.(Look at 3a)

As we can see, for too short phrases zeros were added at the beginning of the vectors. We obtained a two-dimensional training sample tensor. Let's form another tensor for the required output values of the network. We will encode the answers according to the following rule(Picture 3b)



```
[[197  54   2 ... 199 200 201]
 [  0   4 202 ... 205   3  67]
 [206   3  67 ...   4 208 209]
 ...
 [  0  20  62 ...  53 850 851]
 [  0   0  43 ...  33   1 853]
 [  0   0   0 ...  70  65 194]]
```

(a) Vectorized text

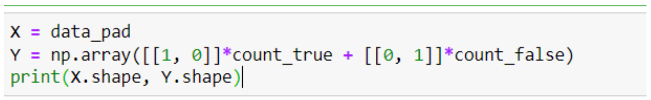


(b) Encoding rule

Figure 3: Text coding and vectorization

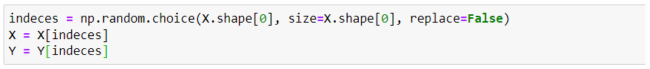
Our neural network will have 2 neurons at the output - the upper neuron will be responsible for the positive text, the lower - for the negative. For positive statements at the output, we will require the vector [1.0], respectively for negative - [0.1]. For example, we know that in our collection data_pad first all positive statements go, and then all negative ones, and we can also find out their number using the count function. Using this number, we will form the output vector we need.

We generate a set of such outputs as follows:



```
X = data_pad
Y = np.array([[1, 0]]*count_true + [[0, 1]]*count_false)
print(X.shape, Y.shape)
```

Now we have a training sample and the required output values. For better training, we mix all these statements so that mixed and positive and negative are submitted to the entrance. So the network will be better trained:



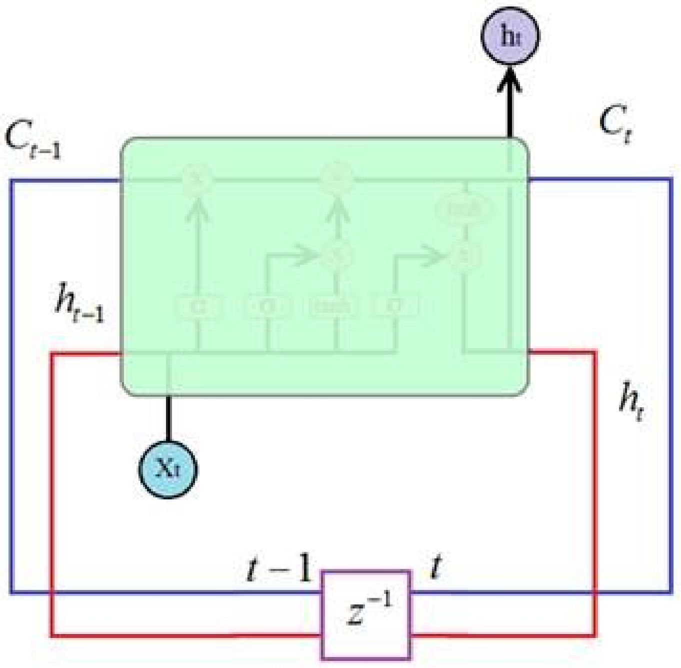
```
indeces = np.random.choice(X.shape[0], size=X.shape[0], replace=False)
X = X[indeces]
Y = Y[indeces]
```

3.4 Creating a Reccurence Network Model

To create a recurrent network model, we will use the LSTM recurrent layer. To create it in Keras, we use the class:

keras.layers.LSTM(units,. . .)

As the first parameter (units), the number of neurons in each fully connected layer inside the LSTM cell is indicated:

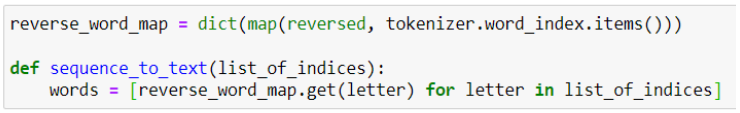


They will also form the dimension of the output vector. For example, let's create an LSTM layer with 64 neurons. Then the dimension of the output vector will be 64 elements. Next, let's add another same layer with 32 neurons. We will get a stack of two recurrent layers.

At the output, we will put a fully connected layer with two neurons and the softmax activation function. Let's define Adam optimization with a convergence step of 0.0001.

After we have started the learning process,to check whether our model works, we will form some text and convert it to the input format of our network.

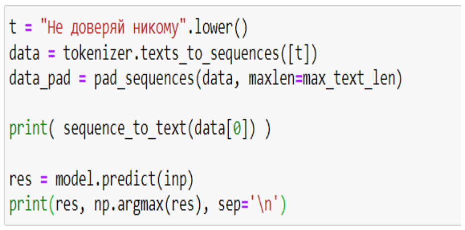
Firstly, we add an auxiliary function. When preparing the text, we formed a dictionary in which indexes first go, and then words. With this dictionary, our function sequence_to_text converts a sequence of indexes back into words:


```
reverse_word_map = dict(map(reversed, tokenizer.word_index.items()))

def sequence_to_text(list_of_indices):
    words = [reverse_word_map.get(letter) for letter in list_of_indices]
```

Then we test the model(Picture 4a) and finally obtain the result(Picture 4b)

It can be noted that we have the greatest value in the resulting vector on the second neuron, and the vector itself is close to the vector [0.1], which we demanded precisely for negative statements, therefore the expression has a negative emotional color.

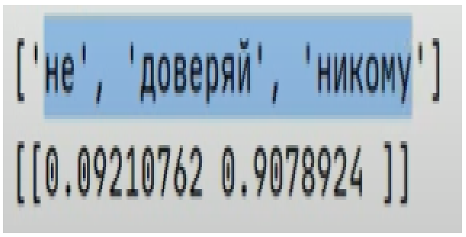


```
t = "Не доверяй никому".lower()
data = tokenizer.texts_to_sequences([t])
data_pad = pad_sequences(data, maxlen=max_text_len)

print( sequence_to_text(data[0]) )

res = model.predict(inp)
print(res, np.argmax(res), sep='\n')
```

(a) Testing the model



```
['не', 'доверяй', 'никому']
[[0.09210762 0.9078924 ]]
```

(b) Result

Figure 4: Final stage

3.5 Word Clouds

As we have already purified out data, then it is quite easy to create a word cloud for every particular place.

To do this, we use the Wordloud package. The Wordloud package in Python helps us find out the frequency of a word in text content using rendering. We use this visualization because the cloud shows the most popular words of the text, which is useful for quickly evaluating it. I have tested the WordCloud on a random filtered dataset and obtained the following result:

good
Better
food
residents
activities
great
like
think
Get
meals
care
need
help
time
Keep
staff
Nothing
choices
cook
variety
shopping
items
weekends
transportation
service
community
music
people
meal

3.6 GUI Application

The next step is creating the graphical user interface with the help of PyQt and QtDesigner. Instead of using fixed positions and element sizes in the application, we use layouts, since fixed positions and dimensions will only look good until the window is resized, and with fixed positions we cannot be sure that everything will be the same on other machines and/or operating systems. Layouts are containers for widgets that will keep them in a certain position relative to other elements. Therefore, when changing the size of the window, the size of widgets will also change.

In our application we have:

1. The Icon

2. The main window, where we have the name of the application, the string in which the dataset should be entered and the clickable button to start the application.

3. After the user clicks the button, a new window appears on the screen, which on each new line contains the name of the glamping place available in the dataset. The name of each place has a button near it so that the user can choose a specific place, information about which he wants to find out.

4. When the user clicks on one of the names, a new window opens containing information about this place, namely:

   (a) The total number of reviews.

   (b) The number of positive reviews.

   (c) Rating of this place calculated by the program.

   (d) Cloud of words showing the most common words in reviews.

All additional code can be found here:
https://github.com/MorozovaMilena/Project-2-course

3.7 Formulas

I do not have any formulas in my project, so these are arbitrary formulas from chemistry, physics etc.

$$E_p = E_0 - \frac{RT}{nF} \ln \left(\frac{C(K_4[Fe(CN)_6])}{K_4[Fe(CN)_6]} \right)$$

$$FWHM = \sqrt{(\frac{0.88\lambda_{em}}{n-\sqrt{(n^2-NA^2)}})^2 + (\frac{nPH\sqrt{2}}{NA^2})^2}$$

$$p_\theta(x) = \frac{\theta^\beta x^{\beta-1}}{\Gamma(\beta)} e^{-\theta x}$$

4 Results and Conclusion

In the software project I have significantly improved my GUI development skills, got acquainted with many useful libraries, such as, for example, WordCloud, TextBlob, Keras and Scipy. Moreover, by observing and participating a little in the part of the work that was devoted to cleaning text, I have learned many new methods and algorithms that make it much easier to work with data. Global work has been done with mathematical models. Models and classifiers based on theorems that we studied in different subjects of the 2nd course (Mathematical statistics, Discrete mathematics, and so on) were studied. In the process of studying mathematical models, were identified several factors that affect their accuracy, which was very useful to learn in order to be able to independently improve the accuracy of these models.

5 Bibliography

- Machine Learning, Neural and Statistical Classification

  https://www1.maths.leeds.ac.uk/ charles/statlog/

- Cleaning Preprocessing Text Data for Sentiment Analysis

  https://towardsdatascience.com/cleaning-preprocessing-text-data-for-sentiment-analysis-382a41f150d6

- Generating WordClouds in Python Tutorial

  https://www.datacamp.com/tutorial/wordcloud-python

- TripAdvisor

  https://www.tripadvisor.ru/

- Sentiment Analysis of Review Datasets Using Naïve Bayes‘ and K-NN Classifier

  https://www.researchgate.net/publication

- Implementing Naive Bayes for Sentiment Analysis in Python

  https://medium.datadriveninvestor.com/implementing-naive-bayes-for-sentiment-analysis-in-python-951fa8dcd928

- Keras Documentation

  https://ru-keras.com/recurrent-layers/

- Python GUI(PyQt and Qt Designer)

  https://tproger.ru/translations/python-gui-pyqt/