**Report on Assignment №2**

«Naïve Bayes Text classification»

# Proposed text classification method

One of the best Python libraries to work with Naïve Bayes Classifier is a NLTK, because it has various methods that significantly ease building a classification model and applying some other NLP techniques.

For example, the main methods of NLTK applied in my programme are enlisted in the table below.

Table 1 – Main methods of NLTK library

|  |  |
| --- | --- |
| The name of a method | The purpose of a method |
| nltk.FreqDist(all\_words) | Creates a distribution of words from movie reviews |
| nltk.NaiveBayesClassifier.train(training\_feature\_set) | Conducts training of Naïve Bayes Classifier model on a training feature set |
| classifier.classify\_many(testing\_set\_content) | Conducts classifying reviews from the testing set |
| nltk.ConfusionMatrix(golden\_label, tested\_label) | Creates a confusion matrix with different classes of predictions |
| nltk.classify.accuracy(classifier, testing\_feature\_set) | Calculates a final accuracy of classification |
| classifier.show\_most\_informative\_features(50) | Outputs words that had the biggest difference in frequency in opposite classes of reviews and promoted efficient classification |

# Pre-processing methods such as removing stop words, punctuations.

The first step of conducting pre-processing of movie reviews’ corpus is changing all words’ case to a lower one in order to apply a lemmatisation.

Lemmatisation helped to decrease amount of different words but with the same lemma, thereby it preserved their common sense.

For this task the WordNetLemmatiser from NLTK was applied, because of its main feature - it is based on a huge dataset of words connected with each other by semantic relationships. As the consequence this tool managed to simplify a larger share of words in reviews.

After that, all stop-words were eliminated using a list of stop-words from Spacy (another rather useful Python NLP library) Along with that, all words that contain digits (0-9) or have length less than two letters were removed as well. By the way, considering that initially all reviews in the corpus had been divided into single elements, this also allowed to delete even some sequential punctuation signs using that condition above. In general, the whole result may be seen on the figure 1 and figure 2 below.

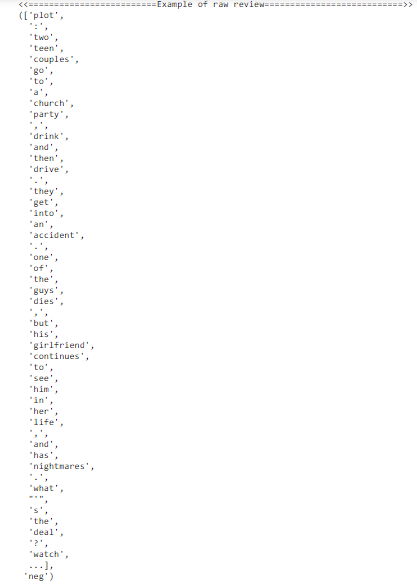


Figure 1 – The example of a raw review

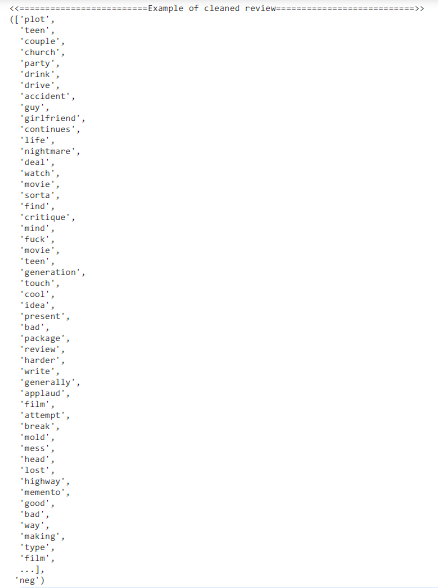


Figure 2 – The example of a cleaned review

Considering the figures above, a general quality and value of information has been significantly enhanced. Therefore, it would become easier for the classifier to distinguish positive and negative classes, because words that occur in both classes have been removed.

# Feature selection methods such as selection the 3000 most important words.

There are two feature selection methods implemented in this assignment:

1. Selecting top-frequent words
2. Selecting word features using Part-Of-Speech Tagging

The first method is based on a theory that the most important words are those that have the biggest frequency in the training set of movie reviews. Thus, if we train a classifier with these words we could obtain a rather high prediction accuracy, because it would be easier for a classifier to deal with top-frequent words.

To find the top-frequent words we should apply a “nltk.FreqDist(all\_words)” NLTK method and take from it a “most\_common(word\_features\_number)” property. The example of top - 25 words from the 3000 most frequent words is on the figure 3 below.

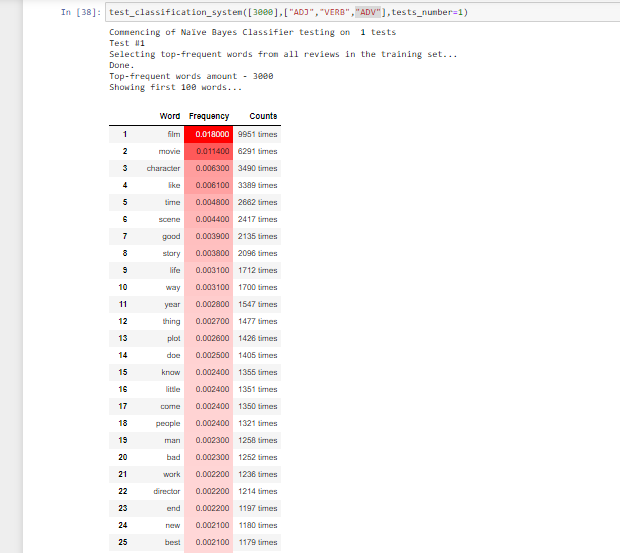


Figure 3 – The example of top-25 words among the most frequent 3000 words.

As we can see from the figure 3, the most frequent word in the entire training dataset is a word “film”.

The second method is mostly based on the first one. We use Part-of-Speech Tagger to select some specific words from the top-frequent ones for our needs. For example, we may consider that the most influential words are adjectives because they are regularly used to describe something.

In this assignment, we used a Spacy library to load its “en\_core\_web\_sm” English language model and to apply its model to determine what a certain word is using command “docs = nlp(word[0])”, where “word[0]” is a word itself.

After this, we should use the following command “docs[0].pos\_” to retrieve a certain tag to that word. Finally, we select only those words what we need from a predefined list of required tags such as “["ADJ","VERB","ADV"]”. Eventually, we may obtain the next data table that can be seen on the figure 4 below.

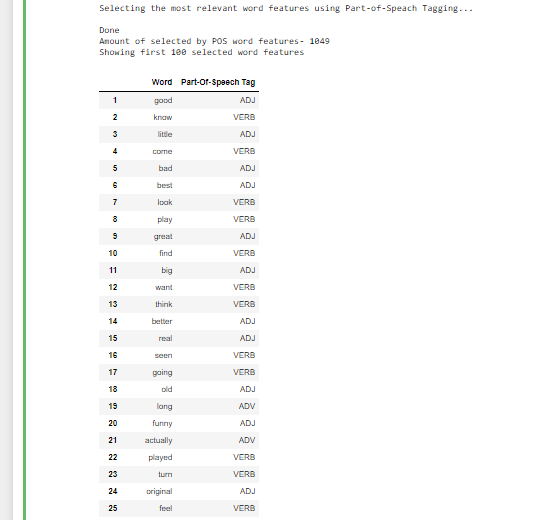


Figure 4 – The example of a data table with words and their tags.

# Model evaluation

The model evaluation is mainly conducted by methods “train”, “classify\_many”, “nltk.ConfusionMatrix”,“nltk.classify.accuracy”, “classifier.show\_most\_informative\_features” mentioned in the table 1 and one another method – “calculate\_metrics(cm)”, which is purposed to calculate Recall, Precision, Accuracy and F1 Score of the built model.

Besides that, the model evaluation is divided into carrying out multiple tests with different amount of top-frequent words and various POS-tags. As the result, we test our model several times on a certain set of input parameters to define average values of all metrics. Thus, we can estimate our model with a better accuracy.

In this project, the Naïve Bayes Classifier was tested on the following sets of input parameters and returned the following results repesented in the table 2.

Table 2 – Table of tests and obtained results

|  |  |
| --- | --- |
| Single launch of the Naїve Bayes Classifier with 3000 top-frequent words | |
| [3000],  ["ADJ","VERB","ADV"],  tests\_number=1 | {'3000 top-frequent words': {  'Average recall': '83.908%',  'Average Precision': '73.000%',  'Average accuracy': '79.500%',  'Average F1 score': '78.075%'},  'Tests quantity': 1,  'POS-tags': ['ADJ', 'VERB', 'ADV']} |
| Experiments with different amount of top-frequent words for feature selection and training the Naїve Bayes Classifier | |
| ([1000,2000,3000,4000,8000,  10000,12000, 15000, 18000, 20000],  ["ADJ","VERB","ADV"],  tests\_number=3 | {'1000 top-frequent words': {  'Average recall': '79.682%',  'Average Precision': '79.333%',  'Average accuracy': '79.500%',  'Average F1 score': '79.381%'},  'Tests quantity': 3,  'POS-tags': ['ADJ', 'VERB', 'ADV']  }  {'2000 top-frequent words': {  'Average recall': '83.451%',  'Average Precision': '76.000%',  'Average accuracy': '80.333%',  'Average F1 score': '79.330%'},  'Tests quantity': 3,  'POS-tags': ['ADJ', 'VERB', 'ADV']  }  {'3000 top-frequent words': {  'Average recall': '84.634%',  'Average Precision': '78.000%',  'Average accuracy': '81.833%',  'Average F1 score': '81.141%'},  'Tests quantity': 3,  'POS-tags': ['ADJ', 'VERB', 'ADV']  }  {'4000 top-frequent words': {  'Average recall': '83.044%',  'Average Precision': '75.000%',  'Average accuracy': '79.833%',  'Average F1 score': '78.697%'},  'Tests quantity': 3,  'POS-tags': ['ADJ', 'VERB', 'ADV']  }  {'8000 top-frequent words': {  'Average recall': '82.439%',  'Average Precision': '71.667%',  'Average accuracy': '78.167%',  'Average F1 score': '76.598%'},  'Tests quantity': 3,  'POS-tags': ['ADJ', 'VERB', 'ADV']  }  {'10000 top-frequent words': {  'Average recall': '82.970%',  'Average Precision': '73.333%',  'Average accuracy': '79.167%',  'Average F1 score': '77.761%'},  'Tests quantity': 3,  'POS-tags': ['ADJ', 'VERB', 'ADV']  }  {'12000 top-frequent words': {  'Average recall': '80.448%',  'Average Precision': '73.333%',  'Average accuracy': '77.667%',  'Average F1 score': '76.661%'},  'Tests quantity': 3,  'POS-tags': ['ADJ', 'VERB', 'ADV']  }  {'15000 top-frequent words': {  'Average recall': '82.337%',  'Average Precision': '72.667%',  'Average accuracy': '78.500%',  'Average F1 score': '77.174%'},  'Tests quantity': 3,  'POS-tags': ['ADJ', 'VERB', 'ADV']  }  {'18000 top-frequent words': {  'Average recall': '82.937%',  'Average Precision': '71.333%',  'Average accuracy': '78.333%',  'Average F1 score': '76.678%'},  'Tests quantity': 3,  'POS-tags': ['ADJ', 'VERB', 'ADV']}  {'20000 top-frequent words': {  'Average recall': '82.900%',  'Average Precision': '73.667%',  'Average accuracy': '79.333%',  'Average F1 score': '77.957%'},  'Tests quantity': 3,  'POS-tags': ['ADJ', 'VERB', 'ADV']} |
| Experiments with some combinations of Part-Of-Speach tags | |
| [8000],None,tests\_number=1 | {'8000 top-frequent words': {'Average recall': '82.353%', 'Average Precision': '70.000%', 'Average accuracy': '77.500%', 'Average F1 score': '75.676%'}, 'Tests quantity': 1, 'POS-tags': None} |
| [8000],None,tests\_number=5 | {'8000 top-frequent words': {'Average recall': '82.281%', 'Average Precision': '69.600%', 'Average accuracy': '77.400%', 'Average F1 score': '75.266%'}, 'Tests quantity': 5, 'POS-tags': None} |
| [8000],["ADJ"],tests\_number=1 | {'8000 top-frequent words': {'Average recall': '87.640%', 'Average Precision': '78.000%', 'Average accuracy': '83.500%', 'Average F1 score': '82.540%'}, 'Tests quantity': 1, 'POS-tags': ['ADJ'] |
| [8000],["ADJ"],tests\_number=5 | {'8000 top-frequent words': {'Average recall': '80.111%', 'Average Precision': '73.600%', 'Average accuracy': '77.600%', 'Average F1 score': '76.544%'}, 'Tests quantity': 5, 'POS-tags': ['ADJ']} |
| [8000],["ADJ","NOUN",],tests\_number=1 | {'8000 top-frequent words': {'Average recall': '86.420%', 'Average Precision': '70.000%', 'Average accuracy': '79.500%', 'Average F1 score': '77.348%'}, 'Tests quantity': 1, 'POS-tags': ['ADJ', 'NOUN']} |
| [8000],["ADJ","NOUN",],tests\_number=5 | {'8000 top-frequent words': {'Average recall': '82.666%', 'Average Precision': '69.400%', 'Average accuracy': '77.400%', 'Average F1 score': '75.368%'}, 'Tests quantity': 5, 'POS-tags': ['ADJ', 'NOUN']} |
| [8000],["ADJ","NOUN","VERB"],tests\_number=1 | {'8000 top-frequent words': {'Average recall': '82.895%', 'Average Precision': '63.000%', 'Average accuracy': '75.000%', 'Average F1 score': '71.591%'}, 'Tests quantity': 1, 'POS-tags': ['ADJ', 'NOUN', 'VERB'] |
| [8000],["ADJ","NOUN","VERB"],tests\_number=5 | {'8000 top-frequent words': {'Average recall': '82.189%', 'Average Precision': '73.400%', 'Average accuracy': '78.700%', 'Average F1 score': '77.518%'}, 'Tests quantity': 5, 'POS-tags': ['ADJ', 'NOUN', 'VERB']} |
| [8000],["ADJ","NOUN","VERB","ADV"],tests\_number=1 | {'8000 top-frequent words': {'Average recall': '81.818%', 'Average Precision': '72.000%', 'Average accuracy': '78.000%', 'Average F1 score': '76.596%'}, 'Tests quantity': 1, 'POS-tags': ['ADJ', 'NOUN', 'VERB', 'ADV'] |
| [8000],["ADJ","NOUN","VERB","ADV"],tests\_number=5 | {'8000 top-frequent words': {'Average recall': '83.813%', 'Average Precision': '74.200%', 'Average accuracy': '79.900%', 'Average F1 score': '78.665%'}, 'Tests quantity': 5, 'POS-tags': ['ADJ', 'NOUN', 'VERB', 'ADV'] |

As we can see from the second experiment with different amounts of word features, the best result is reached with 3000 of words features: all its metrics (except from the precision which is less than a precision from the first test) have the highest value among values from other tests. Thus, it should be concluded that an increasing of amount of word features does not always mean increasing of classification accuracy.

Besides this, if we take a look at the last set of experiments with POS tags we can notice that the following combination of tags such as “["ADJ","NOUN","VERB","ADV"],” has provided the highest values of all metrics on 5 tests. This means that these particular parts of speech have the greatest influence on a text’s sentiment.

All in all, in this assignment there were represented four stages of conducting classification of movie reviews using Naïve Bayes Classifier and revealed various results of carried out experiments that show us an importance of deeper research such essential field of Data Science as NLE.