Y.P. Zybina, E.M. Bozhko
Ural Federal University named after the first President of Russia B.N. Yeltsin
Yekaterinburg, Russia

SENTIMENT ANALYSIS BASED ON MACHINE LEARNING ALGORITHMS

Abstract: This article reveals principles of sentiment analysis in text based on supervised learning algorithms. In particular, the Naive Bayes Classifier algorithm is chosen as an example. Also, the article considers steps of preparing data for the analysis.

Key words: sentiment analysis, computer linguistics, natural language processing, machine learning, supervised learning, naive Bayes classifier.

We live in the world where we can express opinions over the Internet. Social networks, news media sites, online shopping platforms and many other types of web resources give users the ability to share their thoughts on different subjects, such as events, services, movies, etc. Most of these data is open for public access.

The amount of such data is large. For example, each second about 6,000 new tweets are published in Twitter. Manual data processing would require too much human resources, so automated solutions were invented to help. There are many different ways to analyze texts on the basis of artificial intelligence. In this paper, we will concentrate on finding the opinion of the author of the text concerning the subject of the text, or *sentiment analysis*.

Sentiment analysis (SA), also known as opinion mining, is an area of computer linguistics that studies emotions in text documents. Sentiments are feelings (attitudes, emotions and opinions), they are subjective impressions and not facts. A basic task in SA is classifying opinions into two polarities, “negative” and “positive”.

SA can be a useful tool in a lot of different areas, and can help find answers to such questions as: “Is this product review positive or negative?” or “How do people react to specific events”?

One of the main challenges that automatic SA has to deal with is that people express opinions in complex ways. The content of the text expressing opinion can be misleading, topics can be changed, and people can speak ironically and negate things. So, before feeding the computer with data and making any conclusions, it's important to prepare data for analyzing.

At the intersection of information technologies and computer linguistics there is an area called *Natural Language Processing*. "Natural languages" are languages that people use for communication: Russian, English, etc.; they are hard to pin down with explicit rules. Natural Language Processing (or NLP for short), is a set of computer manipulations on language that gives useful responses about text or speech input.

So, what can be done with a raw text, so that a computer would be able to understand it? This process is called feature extraction, and here are several possible steps. Some of them are:

- *Stemming*, a process of extracting word's *stem* – a base part or root form. The stem doesn't have to be a morphological root, usually it's enough for the related words to have a similar stem. For example: for words "argue", "argued", "argues", "arguing", and "argus", the stem will be "argu";
- *Lemmatization*, a dictionary form of the word (*lemma*); for example, for words "argue", "arguing", "argued" lemma will be the word "argue";
- *Tokenization* - a task of chopping the text into pieces (tokens), each of them being able to represent a single word or a part of a complex word;
- *Trimming vocabulary* – removing very frequent "stop-words" such as "a", "the", "or", as well as unknown and very rare words;
- Defining *classes*, in which data will be categorized. In our case, they will be "Positive" and "Negative";
- Defining *feature set* for *feature vectors*. The common ways to do it are the following:
    - *bag-of-words*, when text is represented as a set of its words, grammar and word order being disregarded;
    - *N-grams* - words or collocations consisting on *N* words. For example, in a sentence "The movie is good", words "the",

"movie", "is", "good" will be *unigrams*, and collocations from two words "the movie", "movie is", "is good" will be *bigrams*.

But how to make a computer understand whether data can be classified as "positive" or "negative"? There are several methods to do so, and in this paper, we will dwell on one approach called *supervised learning*. Supervised learning is the machine learning task. *Machine learning* is a field of computer science that gives computers the ability to learn without being explicitly programmed. Supervised learning makes conclusions based on the labeled training data.

The process can be described as follows (Figure 1): a piece of raw text is processed by the feature extractor, which creates a feature set. In the process of training, correctly labeled data is given to the machine learning algorithms to generate a model. The difference between training and prediction is that in prediction the label is a result of the prediction process, while in training the label is already known.

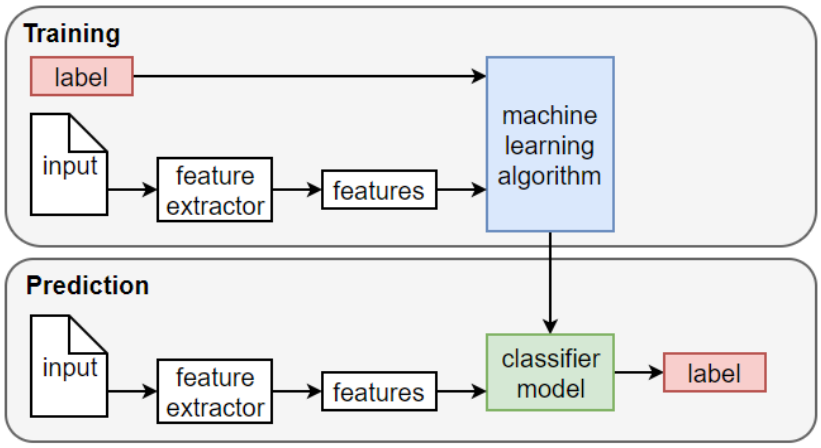

Figure 1- Text classification process for supervised learning

The purpose of a *classifier* is to take a single observation, to extract some useful features and to classify the observation into one class from the set of classes. The goal of machine learning is to make a classifier be able to map new text input (document) with its correct class. There are several ways to implement a classifier, and in this paper we shall look at *the naive Bayes classifier* (NBC).

Naive Bayes is a *probabilistic* classifier, which means that for each input document it calculates the probability of a possible class ("positive" or "negative", in our case) and returns the one with maximum probability. It is based on the Bayes theorem, which determines the probability of some event in case another related event happened. Why is the classifier

naive? We make an assumption that parameters are not related to each other, which is quite naive, but usually, there are dependencies among parameters. NBC results with calculating the most correct class *c* for a given text input considering a probability of class *P(c)* and a likelihood of a set of features for the class *P (f|c)*, where f is the features set.

Calculating *P(c)* and *P(f|c)* is called *training* the classifier. To learn these probabilities, we need to provide a training set, i.e. text inputs, that have already been labeled correctly by people (with so called *"gold labels"*). *P(c)* can be estimated as a ratio N_c/N_{doc}, where *Nc* is a number of text inputs from a training set with class *c*, and N_{doc} is total amount of documents.

To learn the probability $P(f_i|c)$, we can assume that the feature is an existence of a word in a bag-of-words, which had been created earlier. We will need $P(w_i|c)$, the fraction of times the word w_i appears among all words of all documents related to class *c*. First, we connect all documents with class *c*, and then use frequency of w_i to give a maximum likelihood estimate of the probability.

To estimate the accuracy of the classifier, such metrics as precision and recall are used. *Precision* measures the percentage of the items that the system detected as positive and are in fact positive (according to gold labels). *Recall* measures the percentage of items actually present in the input that were correctly identified by the system.

The advantages of using NBC are: relatively simple implementation, low computational costs. The disadvantages are: the algorithm requires preparation of training data.

To sum up, this article introduces essential concepts of sentiment analysis task, and prepares a theoretical base for programming implementation.

Е.П. Зыбина, Е.М. Божко
Уральский федеральный университет имени первого Президента России Б.Н. Ельцина
Екатеринбург, Россия

АНАЛИЗ ТОНАЛЬНОСТИ ТЕКСТА НА ОСНОВЕ АЛГОРИТМОВ МАШИННОГО ОБУЧЕНИЯ

Аннотация: в статье рассмотрены принципы анализа тональности текста на основе алгоритмов обучения с учителем. Выбранным алгоритмом является наивный классификатор Байеса. Также, рассмотрен процесс подготовки данных к анализу.

Ключевые слова: анализ тональности текста, компьютерная лингвистика, машинное обучение, обучение с учителем, наивный классификатор Байеса.

СПИСОК ЛИТЕРАТУРЫ:

1. Steven Bird. Natural Language Processing with Python / Steven Bird, Ewan Klein, Edward Loper / O'Reilly Media, 2009.
2. Персональный блог Digital Marketing consultant. URL: https://www.dsayce.com/social-media/tweets-day/ (дата обращения 19.01.18)
3. Хабрахабр [Электронный ресурс] Обучаем компьютер чувствам (sentiment analysis по-русски). — Режим доступа: https://habrahabr.ru/post/149605/ (дата обращения 25.12.17)
4. Хабрахабр [Электронный ресурс] Наивный Байесовский классификатор в 25 строк кода. — Режим доступа: https://habrahabr.ru/post/120194/ (дата обращения 05.01.18)
5. Хабрахабр [Электронный ресурс] Топ-10 data mining-алгоритмов простым языком. — Режим доступа: https://habrahabr.ru/company/iticapital/blog/262155/ (дата обращения 10.01.18)
6. Language & Communication Technologies [Электронный ресурс]. — Режим доступа:
https://lct-master.org/files/MullenSentimentCourseSlides.pdf (дата обращения 25.12.17)
7. MachineLearning.ru [Электронный ресурс]. — Режим доступа: http://www.machinelearning.ru (дата обращения 10.01.18)
8. Speech and Language Processing (3rd ed. draft) [Электронный ресурс]. — Режим доступа: https://web.stanford.edu/~jurafsky/slp3/ (дата обращения 12.01.18)