International Burch University
Faculty of Engineering and IT
Information Technologies Department
CEN 221 Object Oriented Programming

Sentiment Analysis of Movie Reviews

Abstract

Sentiment analysis is one of the most active fields in machine learning research which is actually classifying sentiments from some kind of text. It is usually done on online reviews, which are becoming more and more useful source of information every day. In this paper I explained results of different NLP methods applied on online movie reviews (whether the opinions are positive or negative as well as performance measurement). Python NLTK is used as a language and GUI is also made for the project for easier usage and application.

Introduction

Because of the Web flourish, more and more websites such as IMDb or Amazon are encouraging posting reviews in order to make improvements. However, there are a thousands and thousands of comments and reviews on sites like these, and it is hard to keep track of useful information that can be extracted from them. That is solved by the sentiment analysis approach in natural language processing which classifies those reviews in certain number of groups (very often two - positive and negative reviews) by using some machine learning algorithms implemented in some specific NLP tools.

In this paper, focus is on sentiment analysis of movie reviews on data taken from IMDb. The problem is challenging because it is sometimes hard to classify whether something is positive or negative even to humans. I used Python NLTK as a platform where I implemented some different classifiers like logistic regression, SVM, naive Bayes and so on and measure the performance.

First part of the paper is short summarization of some previous works and publications, and discussion about their methods and results for this problem. After that comes description of my approach, methods, language and data used, steps and challenges in implementation of those steps and in the end results and conclusion with suggestions for further improvements.

Previous Work

Considering the fact that nowadays sentiment analysis is very popular area in machine learning, and that a lot of money is invested in movie industry which depends on people's rating and reviews, it is no wonder that there were a lot of previous works based on this topic.

In paper "Sentiment Analysis on Movie Reviews Based on Combined Approach" A. Mulkalwar and K. Kelkar are, as the name says, using so called Combined Approach - they were using SVM (Support Vector Machine) and Hidden Markov Model and combined results by combine rule. In results of the approach it is stated that this combination performed much better than any other combination of algorithms, so it was a good sentiment analysis with high accuracy.

In paper "Analyzing Sentiment of Movie Review Data using Naive Bayes Neural Classifier" L. Dhande and G. Patnaik combined Naive Bayes (NB) and Neural Networks for unigram feature in order to increase performance up to 80.65 % (1613 of 2000 correct samples), which is much higher than independent results of those two approaches.
M. Govindarajan had similar idea in his work [3] except that he combined NB with Genetic Algorithm where the results showed very good 93.8% accuracy.

In [4] NB was compared to proposed MapReduce framework in Apache Hadoop (by using Apache Pig) where the proposed method showed much better results and therefore it is suggested.

For the publication [5] they also tested SVM and NB but this time on reviews taken from Twitter comments. Sentiment analysis on social networks brings some more challenges, like certain number of maximum characters, way too often misspellings, repeated characters and other mistakes as well as enormous usage of slang. The results were 75% for SVM and 65% for NB.

SVM was tested in [6] as well as random forest, logistic regression and recursive neural tensor network (RNTN) and compared.

K.Yessenov and S. Misailović also compared and experimented with a lot of classifiers in their paper "Sentiment Analysis of Movie Review Comments" like NB, Decision Trees, Maximum-Entropy and K-means clustering and made different kinds of experiments whose results were aggregated in graphs.

These were all previous approaches and in the next chapter I will present mine.

Approach

The dataset which I used is "**Large Movie Review Dataset**" used on Stanford University with 50000 reviews taken from IMDb, which are divided in two equal groups for training and test set of data. It is assumed that reviews are either positive or negative. Based on that, reviews were supposed to be classified in those two groups.

NLTK, which stands for **Natural Language Toolkit**, is a very well known platform for natural language processing, which can be helpful for anything is needed to perform it like splitting sentences, words, distinguishing part of speech for the words and helping the machine to understand the text. It supports all the variety of algorithms, and here are the ones that are implemented in my work:

Naive Bayes is a simple probabilistic classifier which is based on Bayes' Theorem. It is has strong independence assumption works well for text classification. In this method, all features are conditionally independent to one another. There is a certain number of models in which algorithm can be implemented. The ones used for this paper are Multinomial and Bernoulli NB. **Multinomial NB** is usually used in text classifications and implements the naive Bayes algorithm for multinomially distributed data which are typically discrete values represented as word vector counts. **Bernoulli NB** implements the naive Bayes training and classification algorithms for data that is distributed according to multivariate Bernoulli distributions. There may be multiple features which are assumed to be a binary-valued variable. f that is not the case Bernouly NB will binarize the input data.

SVM (Support vector machine) is supervised learning model with learning algorithms which analyze data for classification and regression. It is mostly used for text, image classification as well as permutation tests used in biological and other natural sciences. SVM algorithms that are implemented here are Support Vector Classification-SVC, Nu - SVC and Linear SVC. **SVC** and **Nu SVC** are similar, but accept slightly different parameters and have a bit different mathematical formulas. **Linear SVC** is another implementation of SVC for the case of a linear kernel. Input: array X [n_samp., n_feat.] with training samples, and array y of class labels [n_samp.]:

```
>>> from sklearn import svm

>>> X = [[0, 0], [1, 1]]

>>> y = [0, 1]

>>> clf = svm.SVC()

>>> clf.fit(X, y)

SVC(C=1.0, cache_size=200, class_weight=None, coef0=0.0,

    decision_function_shape=None, degree=3, gamma='auto', kernel='rbf',

    max_iter=-1, probability=False, random_state=None, shrinking=True,

    tol=0.001, verbose=False)
```

Logistic Regression is regression model which takes only two values, which represent outputs like pass/fail, win/lose, alive/dead or healthy/sick. That makes it perfect for this kind of sentiment analysis where outcomes are "positive" and "negative" review. If there are more than two categories, we talk about multinomial logistic regression.

Stochastic Gradient Descent (SGD) is a simple and efficient approach to discriminative learning of linear classifiers. SGD has been successfully applied to large-scale machine learning problems often based on text classification and natural language processing. This model easily scale problems with more than 10000 tr. examples and features. Its main task is to minimize the cost of some linear classifiers. SGD requires some parametres like regularization parameter and the number of iterations. It is sensitive to feature scaling. **SGDClassifier** supports multi-class classification by combining multiple binary classifiers in a "one versus all" scheme. For each of the N classes, a binary classifier discriminates between one of them and all other N-1 classes. Then, the confidence score is computed for each classifier and the class with the highest confidence is chosen.

It is important to mention that all of these method are implemented as part of scikit-learn algorithm. The steps that are done in their implementation and their performance and results are described in the rest of the document.

Methodology

The project was done in two parts. First part is training the data with algorithms I mentioned before. The second one was making the graphical user interface for training as well as applying the algorithms on new examples taken as an input.

Training part is consisted of next steps:

- Collecting the data to be trained
- Tokenizing - splitting sentences and words from the body of text and handling punctuation marks
- Finding unwanted words and stop words
- Distinguishing stemming words
- Finding synonyms - for reducing tokens
- Recognizing name entities like names of places, people, locations and so on
- Lemmatization - reducing the words into their original form
- Using WordNet - lexical database for English language used in NLTK for all needed comparisons
- Performing textual classification on this data into lists (arrays) so that each token is classified as positive or negative (by comparing the words to the feature lists of words - most common words considered as positive or negative)
- Performing the Naive - Bayes Classification
- Importing Pickle library and saving the classifier
- Importing Scikit - Learn for easier implementation of algorithms

- Using Scikit - Learn for implementing all the previously mentioned algorithms
- Training data, testing data and measuring performance (accuracy) in percentage for comparison.

Second part was the graphical user interface. It is consisted of two parts:

1. For training data and measuring the performance in percentages:

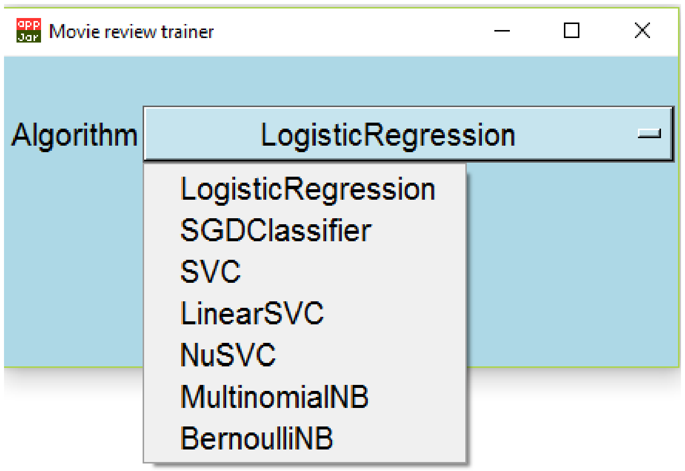
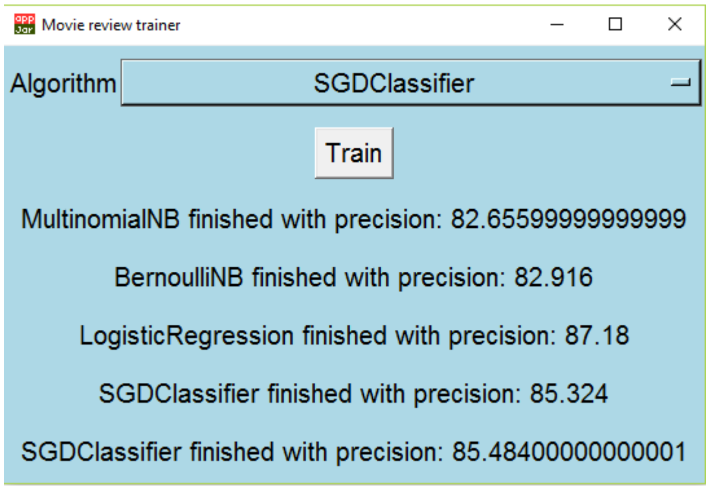


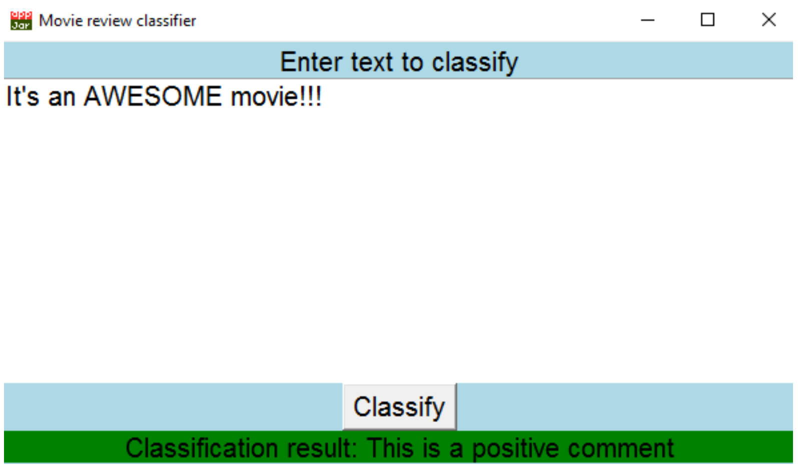
Figure 1 : Training Algorithms



```
MultinomialNB finished with precision: 82.65599999999999
BernoulliNB finished with precision: 82.916
LogisticRegression finished with precision: 87.18
SGDClassifier finished with precision: 85.324
SGDClassifier finished with precision: 85.48400000000001
```

Figure 2 : Training results

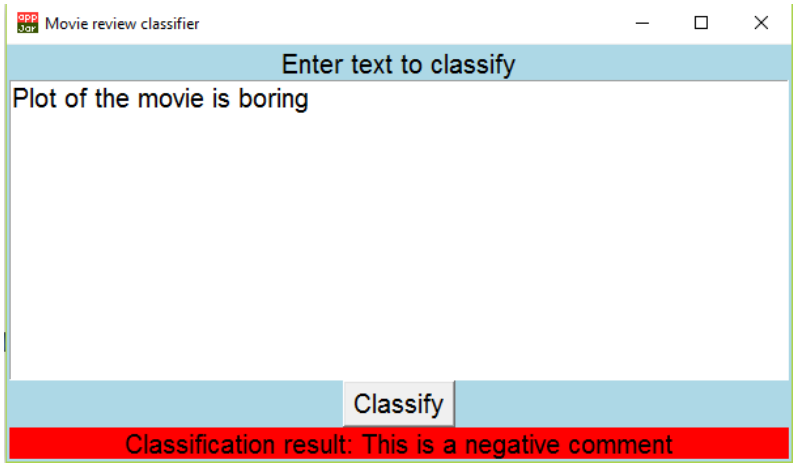
2. For testing on new training examples the lastly saved classifier from the part 1:



It's an AWESOME movie!!!

Classification result: This is a positive comment

Figure 3: Positive review



Plot of the movie is boring

Classification result: This is a negative comment

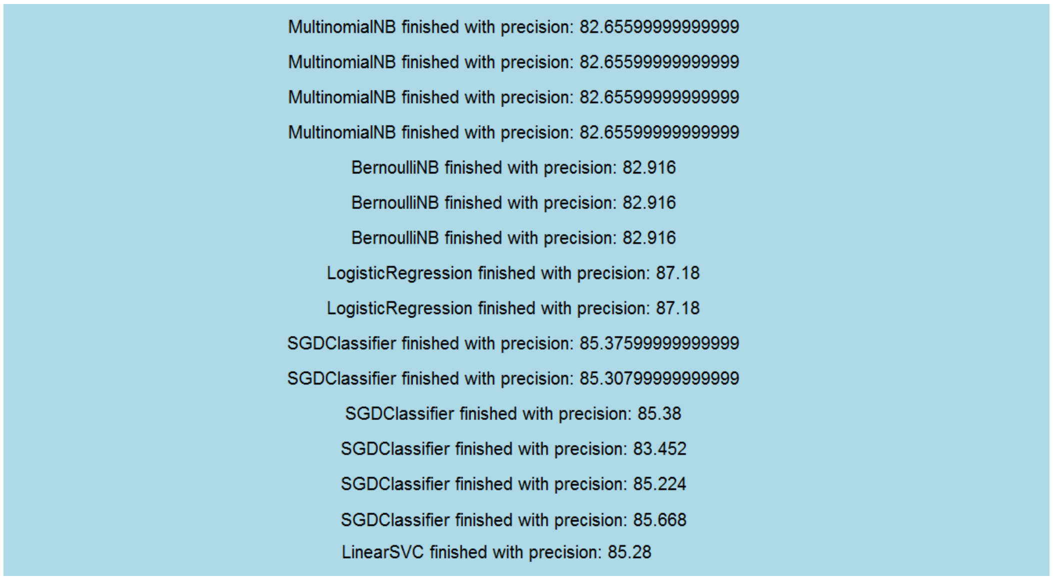
Figure 4: Negative review

Results

For Multinomial Naive Bayes result is always the same for this dataset : 82.655999% accuracy which is pretty good result. Bernoulli Naive Bayes has somewhat better performance: 82.916 % accuracy.

Linear models had much higher precision. Logistic regression has 87.18 % performance which almost leads to overfitting (what would be bad because of worse performance on new training examples). Scholastic Gradient Descent changed percentage of accuracy through many iterations like shown on figure below.

When we speak about support vector machine and its classifiers, it can be noticed that it takes them really a long time for training the data. Performance was always around 85 - 86%.



```
MultinomialNB finished with precision: 82.65599999999999
MultinomialNB finished with precision: 82.65599999999999
MultinomialNB finished with precision: 82.65599999999999
MultinomialNB finished with precision: 82.65599999999999
BernoulliNB finished with precision: 82.916
BernoulliNB finished with precision: 82.916
BernoulliNB finished with precision: 82.916
LogisticRegression finished with precision: 87.18
LogisticRegression finished with precision: 87.18
SGDClassifier finished with precision: 85.37599999999999
SGDClassifier finished with precision: 85.30799999999999
SGDClassifier finished with precision: 85.38
SGDClassifier finished with precision: 83.452
SGDClassifier finished with precision: 85.224
SGDClassifier finished with precision: 85.668
LinearSVC finished with precision: 85.28
```

Figure 5: Accuracy of Algorithms

Discussion

All of the methods I used seem to be with a very good performance, some of them even lean to overfitting (logistic regression), but all of them performed results between 82 and 88 %. As a next step and future work for this project, all of these algorithms can be tested with corresponding parameters as well as in combined approach (combining some or even all of them). For interface web would be much better option and next step. Anyway, all of this was great way to learn how several different machine learning algorithms work in practice, as well as sentiment analysis and natural language processing in general. NLTK and Python itself seem to have everything needed for this area of machine learning, and I recommend it to everyone.

Conclusion

To sum up, the purpose of the project was to see how sentiment analysis can be performed on some real-life example. In this case it was very popular movie review sentiment analysis on IMDB data implemented in Natural Language Toolkit in Python using several models their algorithms and measuring and comparing their performance. Everything was tested on new examples which are taken as an input through GUI made for this purpose. There is a lot of space left for improvements, and different ways to do all this (different platform, language, models ...), but even this simple one showed really good results. Maybe combined approach would be the next step, but even the individual algorithms which were used performed high accuracy and successful sentiment analysis.

References

1. A. Mulkalwar, K. Kelkar (2014, July) "*Sentiment Analysis on Movie Reviews Based on Combined Approach"* : Dept. of Computer Engineering, University of Mumbai, India
2. L. Dhande, G. Patnaik (2014, July-August) "*Analyzing Sentiment of Movie Review Data using Naive Bayes Neural Classifier*" : Department of Computer Engineering, SBBT's College of Engineering and Technology, Maharashtra, India
3. M. Govindarajan (2013, December), "*Sentiment Analysis of Movie Reviews using Hybrid Method of Naive Bayes and Genetic Algorithm"* : International Journal of Advanced Computer Research
4. B. Narendra, K. Uday Sai, G. Rajesh, K. Hemanth, M. V. C. Teja, K. D. Kumar (2016, August) "*Sentiment Analysis on Movie Reviews : A Comparative Study of Machine Learning algorithm and Open Source Technologies*": Sree Vidyanikethan Engineering College, Tirupathi
5. A. Amolik, N. Jivane, M. Bhandari, M. Venkatesan (2015, January) *"Twitter Sentiment Analysis of Movie Reviews using Machine Learning Techniques" :* School ofComputer Science and Engineering, VIT University, Tamilnadu, India
6. H. Pouransari, S. Ghili "*Deep learning for sentiment analysis of movie reviews"* : Stanford University
7. K. Yessenov, S. Misailović (2009) "*Sentiment Analysis of Movie Review Comments"*
8. H. Shyziya, G. Kavitha, R. Zaheer (2015, November) "*Text Categorization of Movie Reviews for Sentiment Analysis*" : International Journal of Innovative Research in Science, Engineering and Technology