

Kathmandu University
School of Engineering
Department of Computer Science & Engineering

Mini Project Report

COMP 484
(Machine Learning)



Review Analyzer

A **fourth year first semester mini-project report** submitted in partial fulfillment
of the requirements for the degree of
Bachelor of Engineering

Team Members

Depenti Karki (27)
Pratiksha Shrestha (53)

Fourth Year, First semester
Department of Computer Science and Engineering (DoCSE)

22nd December, 2018

Table of Contents

1. Project idea	1
2. Related Works	2
3. Dataset	2
4. Software and tools	3
4.1. Programming Language: Python	3
4.2. Natural Language Toolkit (NLTK)	3
4.2.1. word_tokenize()	3
4.2.2. Stopword	3
4.3. Scikit-learn	3
4.4. Numeric Python or Numerical Python (Numpy)	4
5. Technique Used	4
5.1. Naive Bayes Classifier	4
Multinomial Naive Bayes	5
5.2 Support Vector Machines (SVMs)	5
Linear SVM	5
5.3. K-fold cross validation	6
6. Methodology and Design	7
7. Testing and Evaluation	10
8. Conclusion	11
References	11

1. Project idea

Reviews can improve the visibility of business on Google and other search engines. Ratings and reviews are some of the most important triggers in app discoverability and installs, and are still core metrics app marketers use to gauge success. Reviews help people all over the world determine which services and products are best. Having a lot of positive reviews can help boost online authority and lead to more conversions and sales. Before the purchase, customer might want to view either a positive or negative review, or check if the product has high amount of positive review or not. The basic idea of the project **“Review Analyzer”** is to build a software tool to analyze review and sort it out as either positive or negative review.

The project is based on sentiment analysis which makes use of natural language processing, text analysis and computational linguistics to identify the sentiment or attitude behind an online mention. Sentiment essentially relates to feelings, attitudes, emotions and opinions whereas sentiment analysis refers to the practice of applying natural language processing and text analysis techniques to identify and extract subjective information from a piece of text. A person’s opinion or feelings are for the most part subjective and not facts which means to accurately analyze an individual’s opinion or mood from a piece of text can be extremely difficult. With sentiment analysis from a text analytics point of view, we are essentially looking to get an understanding of the attitude of a writer with respect to a topic in a piece of text and its polarity; whether it’s positive, negative or neutral. The given data containing words and sentences will be tokenized and then using techniques such as Naive Bayes Classifier, each tokenized word will be classified into either positive or negative. Based on this result, the review will be classified into either positive or negative.

“Review Analyzer” specifically focuses on movie review analysis whether a movie is a rotten tomato or a brilliant work of art, if people are watching it, it's worth critiquing. A decent movie review should entertain, persuade and inform, providing an original opinion without giving away too much of the plot. A *great* movie review can be a work of art in its own right. Review analyzer will train itself from data of movie reviews and incorporate it to find whether a review is positive or negative (i.e. Performance, Task and Experience(PTE) from movie reviews).

2. Related Works

Sentiment analysis is used across a variety of applications. It may be performed on Twitter to determine overall opinion on a particular trending topic. Companies and brands often utilize sentiment analysis to monitor brand reputation across social media platforms or across the web as a whole.

Sentiment analysis is widely used in monitoring call center and customer support performance. As companies seek to keep a finger on the pulse of their audiences, sentiment analysis is increasingly utilized for overall brand monitoring purposes.

Sentiment analysis has been used by political candidates and administrations to monitor overall opinions about policy changes and campaign announcements, enabling them to fine-tune their approach and messaging to better relate to voters and constituents. In brand reputation management applications, overall trends in sentiment analysis enables brands to identify peaks and valleys in overall brand sentiment or shifts in attitudes about products or services, thus enabling companies to make improvements perfectly in-tune with customer demands.

3. Dataset

The dataset is taken from the nltk corpora movie_reviews.

There are total 2000 processed down-cased data sets as text files which are divided into two subdirectories: pos and neg. the names of the two subdirectories indicate the true classification (sentiment) of the component files according to the automatic rating classifier.

Pos: It consists of 1000 reviews to various movies labeled as positive.

Neg: It consists of 1000 reviews to various movies labeled as negative.

The rating information as well as the other details to the movies have been removed from the text file. Only sentiment along with text review is stored.

4. Software and tools

4.1. Programming Language: Python

Python is a general purpose interpreted, object-oriented high level programming language with large number of libraries for machine learning and natural language processing. It is great for backend web development, data analysis, artificial intelligence, and scientific computing.

4.2. Natural Language Toolkit (NLTK)

The Natural Language Toolkit (NLTK), is a suite of libraries and programs for symbolic and statistical Natural Language Processing (NLP) for English written in the Python programming language. It contains text processing libraries for tokenization, parsing, classification, stemming, tagging and semantic reasoning. It also includes graphical demonstrations and sample data sets.

4.2.1. word_tokenize()

A sentence or data can be split into words using the method `word_tokenize()`.

4.2.2. Stopword

Stopwords are the words we want to filter out before training the classifier. These are usually high frequency words that aren't giving any additional information to labeling. In fact, they actually confuse our classifier. In English, they could be *the*, *is*, *at*, *which*, and *on*.

4.3. Scikit-learn

Scikit-learn is a free software machine learning library for the Python programming language. It features various classification, regression and clustering algorithms including support vector machines, random forests, gradient boosting. It is designed to interoperate with the Python numerical and scientific libraries NumPy and SciPy.

4.4. Numeric Python or Numerical Python (Numpy)

NumPy is an open source extension module for Python, which provides fast precompiled functions for mathematical and numerical routines. Furthermore, NumPy enriches the programming language Python with powerful data structures for efficient computation of multi-dimensional arrays and matrices.

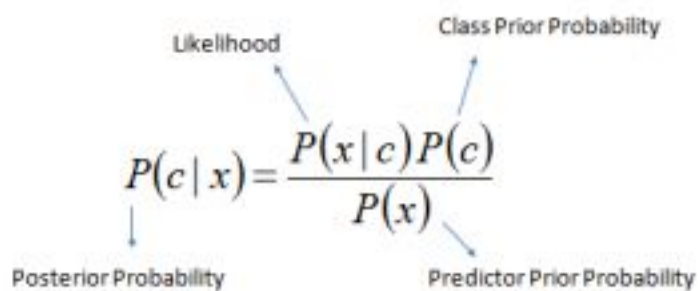
SciPy (Scientific Python) is often mentioned in the same breath with NumPy. SciPy extends the capabilities of NumPy with further useful functions for minimization, regression, Fourier-transformation and many others.

5. Technique Used

5.1. Naive Bayes Classifier

Naive Bayes Classifier is a classification technique based on Bayes' Theorem that has strong independence assumptions between the predictors. In simple terms, a Naive Bayes classifier assumes that the presence of a particular feature in a class is unrelated to the presence of any other feature. Naive Bayes model is easy to build and particularly useful for very large data sets. Even though being simple, Naive Bayes Classifier can outperform even highly sophisticated classification methods.

Bayes theorem provides a way of calculating posterior probability $P(c|x)$ from $P(c)$, $P(x)$ and $P(x|c)$.



The diagram shows the formula for Bayes' Theorem: $P(c|x) = \frac{P(x|c)P(c)}{P(x)}$. Arrows point from labels to the corresponding parts of the formula: 'Likelihood' points to $P(x|c)$, 'Class Prior Probability' points to $P(c)$, 'Posterior Probability' points to $P(c|x)$, and 'Predictor Prior Probability' points to $P(x)$.

$$P(c | X) = P(x_1 | c) \times P(x_2 | c) \times \dots \times P(x_n | c) \times P(c)$$

Above,

- c is the class for occurring event
- x is the predictor event that has already occurred
- $P(c)$ is the prior probability of class.
- $P(x)$ is the prior ability of predictor
- $P(x|c)$ is the likelihood which is the probability of predictor given class.
- $P(x)$ is the prior probability of predictor.

Naive Bayes Classifier assumes that the presence or absence of a particular event of a class is independent to the presence or absence of other events. That is, on a given class c , the predictor x is conditionally independent of each other.

- **Multinomial Naive Bayes**

Multinomial Naive Bayes is a specialized version of Naive Bayes that is designed more for text documents. It is suitable for classification with discrete features (e.g., word counts for text classification). The multinomial distribution normally requires integer feature counts. However, in practice, fractional counts such as tf-idf may also work.

5.2 Support Vector Machines (SVMs)

Support Vector Machines (SVMs, also support vector network) are supervised learning models with associated learning algorithms that analyze data used for classification and regression analysis. Given a set of training examples, each marked as belonging to one or the other of two categories, an SVM training algorithm builds a model that assigns new examples to one category or the other, making it a non-probabilistic binary linear classifier.

- **Linear SVM**

Linear SVM is the newest extremely fast machine learning (data mining) algorithm for solving multiclass classification problems from ultra large data sets that implements an original proprietary version of a cutting plane algorithm for designing a linear support vector machine.

Linear SVM is a linearly scalable routine meaning that it creates an SVM model in a CPU time which scales linearly with the size of the training data set.

5.3. K-fold cross validation

K-fold cross validation function of scikit-learn is to be used to optimize the accuracy of Naive Bayes Classifier model. In K-fold cross validation, the dataset is split into K equal partitions (or folds). The first fold is used as the testing set and the union of the other folds as the training set. The test accuracy is calculated. The same process is repeated for each fold leading to K-iterations. The average testing accuracy is used as the estimate of out-of-sample accuracy. The main advantage of using this validation function is that the accuracy is better than the simple Naive Bayes classification and the data are used more efficiently as every observation is used for both training and testing.

10-fold cross validation is used in **Review Analyzer** that means the dataset is split into 10 equal partitions from which the first fold is used as the testing set and the union of the other folds as the training sets and the same process is repeated for each fold leading to 10-iterations.

6. Methodology and Design

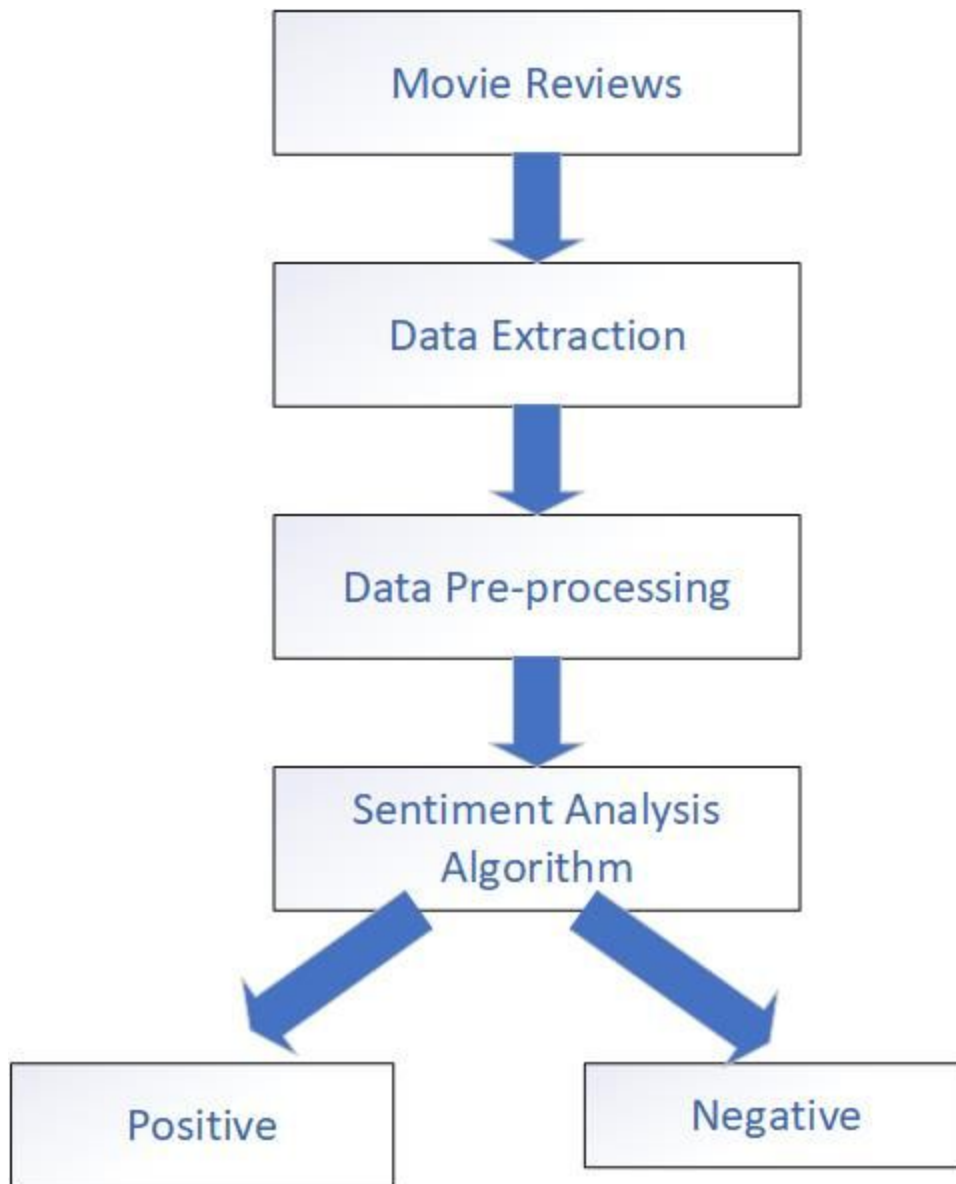


Fig 7.1: Methodology of Sentiment Analyzer

The experiments are done in JupyterLab. The necessary NLTK and sklearn libraries are imported. We have experimented on nltk corpus movie_reviews. At first, the necessary data is extracted from the movie reviews source. These data are then loaded. In pre-processing phase, all the stopwords (i.e. unnecessary words) are removed, all the text are converted to lowercase

and for each remaining word, a dictionary is created. This is the format NLTK expects as input. The movie reviews text is then decomposed into numerical data and the data is splitted randomly into training and testing sets.

In classification phase, different classification algorithms are used which classified training set as input during its initialization. With each review in training set, the classifier learns how each feature impacts the sentiment of a reviewer. For each review in test set, it predicts what the sentiment should be based on the text of the review.

CODE SNIPPETS

The stopwords are identified and the words that do not identify with those stopwords are changed to lowercase.

```
useless_words1 = ['.', ',', '-', '_', '(', ')', ':', '?', ';', '"']
words = []
for word in movie_reviews.words():
    if word not in useless_words and word not in useless_words1:
        words.append(word.lower())
```

The remaining words are then appended to the list featuresets.

```
featuresets = []
for (review, category) in documents:
    words = set([word for word in review if word not in useless_words and word not in useless_words1])
    features = {}

    for w in word_features:
        features[w] = w in words
    featuresets.append([features, category])
```

The dataset is splitted into training and test sets and Naive Bayes Classifier is implemented

```
training_set = featuresets[:1400]
testing_set = featuresets[1400:]
```

```
classifier = nltk.NaiveBayesClassifier.train(training_set)
```

```
nltk.classify.accuracy(classifier, testing_set)
```

```
0.8066666666666666
```

Implementation of 10-fold cross validation

```
kf = KFold(n_splits=10)
sum = 0
for train, test in kf.split(featuresets):
    train_data = np.array(featuresets)[train]
    test_data = np.array(featuresets)[test]
    classifier = nltk.NaiveBayesClassifier.train(train_data)
    sum += nltk.classify.accuracy(classifier, test_data)
average = sum/10
```

The user input for review is taken whose sentiment is determined.

```
try_rev = '''Amazing piece of art. Worldclass. Awesome. Worth it.A bit shady.'''
```

```
try_word = word_tokenize(try_rev)
```

```
try_word = create_word_features(try_word)
```

```
classifier.classify(try_word)
```

```
'pos'
```

7. Testing and Evaluation

The application is tested over 2000 movie review datasets. The Naive Bayes Classifier obtained an accuracy of **80.66%**. Using 10-fold cross-validation technique, the classifier obtained an accuracy of **81.9%** which is slightly better compared to the original Naive Bayes Classifier. Similarly, SVM classifier was compared with Naive Bayes classifier. Linear SVC is found to be slightly better than the Naive Bayes classifier. Different classifiers and their obtained accuracy are given below:

Classifier	Accuracy
Naive Bayes Classifier	80.66%
Naive Bayes Classifier with 10-fold Cross-Validation	81.9%
Multinomial Naive Bayes Classifier	81%
LinearSVC	82.16%

Fig 7.2: Accuracy obtained from various classifier

The most informative words were found to be:

Most Informative Features

sloppy = True	neg : pos = 14.9 : 1.0
stupidity = True	neg : pos = 14.2 : 1.0
seagal = True	neg : pos = 12.2 : 1.0
ludicrous = True	neg : pos = 11.3 : 1.0
unintentional = True	neg : pos = 10.2 : 1.0
3000 = True	neg : pos = 9.6 : 1.0
idiotic = True	neg : pos = 9.3 : 1.0
traveling = True	pos : neg = 9.1 : 1.0
hudson = True	neg : pos = 8.9 : 1.0
atrocious = True	neg : pos = 8.3 : 1.0

Unlabeled data were manually added to determine whether the given review is a positive or negative review.

8. Conclusion

With a variety of user applications, nowadays sentiment analysis is an evolving field. Naive Bayes machine learning algorithm helped to better understanding of natural language opinions and reported high accuracy for predicting sentiment of movie reviews. Due to the conditional independence assumptions, Naive Bayes classifiers are extremely fast to train and can scale over large datasets. Both Naive Bayes and SVM classifiers are proven to be good methods to learn sentiments from movie reviews.

References

Cordero, Lesley. "Posts By Stack." *Twilio Cloud Communications Blog*,

www.twilio.com/blog/2017/12/sentiment-analysis-scikit-learn.html.

"How to Prepare Movie Review Data for Sentiment Analysis." *Machine Learning Mastery*, 18 Dec. 2017,

machinelearningmastery.com/prepare-movie-review-data-sentiment-analysis/.

Kalyanarangan, Vivek, et al. "Opinion Mining - Extraction of Opinions from Free Text." *Dataconomy*, 24

Nov. 2016, dataconomy.com/2016/11/opinion-mining-extracting-opinions/.

"Naive Bayes and Text Classification." *Dr. Sebastian Raschka*, 4 Oct. 2014,

sebastianraschka.com/Articles/2014_naive_bayes_1.html.

"The Ultimate Guide to Sentiment Analysis Using Python!" *LinearData*, 14 Sept. 2017,

lineardata.net/the-ultimate-guide-to-sentiment-analysis-using-python/.