Sentiment Analysis: Amazon Product Movie Review

**Bharath Kumar S,1MS12CS018,**

Department of Computer Science and Engineering,

M.S.R.I.T, Bangalore, India

**Ganapati R Hegde,1MS12CS032,**

Department of Computer Science and Engineering,

M.S.R.I.T, Bangalore, India

**Jivitesh Mopuri,1MS12CS042,**

Department of Computer Science and Engineering,

M.S.R.I.T, Bangalore, India

**Keshava R,1MS12CS045,**

Department of Computer Science and Engineering,

M.S.R.I.T, Bangalore, India

Abstract: Sentiment analysis is language processing task that uses a computational approach to identify opinionated content and categorize it as positive or negative. The unstructured textual data on the Web often carries expression of opinions of users. Sentiment analysis tries to identify the expressions of opinion and mood of writers. A simple sentiment analysis algorithm attempts to classify a document as ‘positive’ or ‘negative’, based on the opinion expressed in it. The document-level sentiment analysis problem is essentially as follows: Given a set of documents, a sentiment analysis algorithm classifies each document into one of the two classes, positive and negative. Positive label denotes that the document expresses a positive opinion and negative label means that expresses a negative opinion of the user. More sophisticated algorithms try to identify the sentiment at sentence-level, feature-level or entity-level.

There are broadly three types of approaches for sentiment classification of texts: (a) using a machine learning based text classifier -such as Naïve Bayes, SVM or kNN - with suitable feature selection scheme; (b) using the unsupervised semantic orientation scheme of extracting relevant n-grams of the text and then labeling them either as positive or negative and consequentially the document; and (c) using the SentiWordNet based publicly available library that provides positive, negative and neutral scores for words.

So this analysis is done on the datasets that contains movie reviews from the Amazon product dump. This helps the user to better decisions whilst watching/picking an appropriate movie(s).

1. **INTRODUCTION**

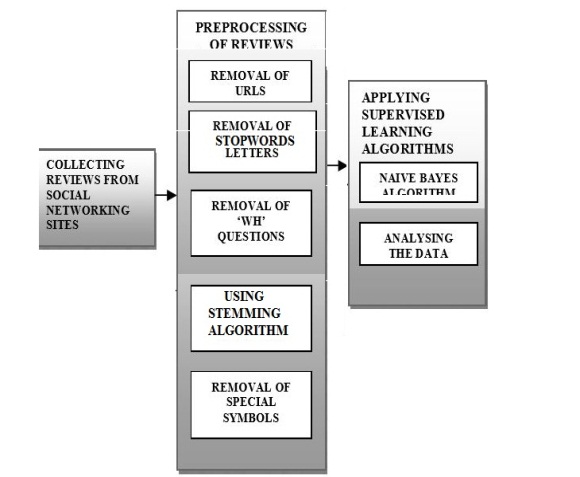
This project is a movie review mining using machine learning and semantic orientation. Supervised classification and text classification techniques are used in the proposed machine learning approach to classify the movie review. A corpus is formed to represent the data in the documents and all the classifiers are trained using this corpus.

Thus, the proposed technique is more efficient. Though, the machine learning approach uses supervised learning, the proposed semantic orientation approach uses “unsupervised learning” because it does not require prior training in order to mine the data.

Thus, the study concludes that the supervised machine learning is more efficient but requires a considerable amount of time to train the model. On the other hand, the semantic orientation approach is slightly less accurate but is more efficient to use in real time applications. The results confirm that it is practicable to automatically mine opinions from unstructured data.

The project used machine learning techniques to investigate the effectiveness of classification of documents by overall sentiment. Experiments demonstrated that the machine learning techniques are better than human produced baseline for sentiment analysis on movie review data. The experimental setup consists of movie-review corpus with randomly selected 8million sentiment reviews. Learning methods Naïve Bayes, was employed. The machine learning techniques are better than human baselines for sentiment classification. Whereas the accuracy achieved in sentiment classification is much lower when compared to topic based categorization. Sentiment Analyzer to extract opinions about a subject from online data documents. Sentiment analyzer uses natural language processing techniques. The Sentiment analyzer finds out all the references on the subject and sentiment polarity of each reference is determined. The sentiment analysis conducted by the researchers utilized the sentiment lexicon and sentiment pattern database for extraction and association purposes. Online product review articles for digital camera and music were analyzed using the system with good results.

1. **DESIGN**



1. *Naïve Bayes Classifier*

Naive Bayes is a simple technique for constructing classifiers: models that assign class labels to problem instances, represented as vectors of feature values, where the class labels are drawn from some finite set. It is not a single algorithm for training such classifiers, but a family of algorithms based on a common principle: all naive Bayes classifiers assume that the value of a particular feature is independent of the value of any other feature, given the class variable.

With a multinomial event model, samples (feature vectors) represent the frequencies with which certain events have been generated by a multinomial (p_1, \dots, p_n)where p_i is the probability that event i occurs (or K such multinomials in the multiclass case). A feature vector \mathbf{x} = (x_1, \dots, x_n) is then a histogram, with x_i counting the number of times event i was observed in a particular instance. This is the event model typically used for document classification, with events representing the occurrence of a word in a single document (see bag of words assumption). The likelihood of observing a histogram x is given by


p(\mathbf{x} \vert C_k) = \frac{(\sum_i x_i)!}{\prod_i x_i !} \prod_i {p_{ki}}^{x_i}


The multinomial naive Bayes classifier becomes a linear classifier when expressed in log-space


\begin{align}
\log p(C_k|\mathbf{x}) & \varpropto \log \left( p(C_k) \prod_{i=1}^n {p_{ki}}^{x_i} \right) \\
                       & = \log p(C_k) + \sum_{i=1}^n x_i \cdot \log p_{ki}                 \\
                       & = b + \mathbf{w}_k^\top \mathbf{x}
\end{align}


Where b = \log p(C_k) and w_{ki} = \log p_{ki}

If a given class and feature value never occurs together in the training data, then the frequency-based probability estimate will be zero. This is problematic because it will wipe out all information in the other probabilities when they are multiplied. Therefore, it is often desirable to incorporate a small-sample correction, called pseudocount, in all probability estimates such that no probability is ever set to be exactly zero. This way of regularizing naive Bayes is called Laplace smoothing when the pseudocount is one, and Lidstone smoothing in the general case.

Here is a worked example of naive Bayesian classification to the document classification problem. Consider the problem of classifying documents by their content, for example into spam and non-spam e-mails. Imagine that documents are drawn from a number of classes of documents which can be modelled as sets of words where the (independent) probability that the i-th word of a given document occurs in a document from class C can be written as

p(w_i \vert C)\,

Then the probability that a given document D contains all of the words w_i, given a class C, is

p(D\vert C)=\prod_i p(w_i \vert C)\,

The question that we desire to answer is: "what is the probability that a given document D belongs to a given class C?" In other words, what is p(C \vert D)\,?

Now by definition

p(D\vert C)={p(D\cap C)\over p(C)}and p(C\vert D)={p(D\cap C)\over p(D)}

Bayes' theorem manipulates these into a statement of probability in terms of likelihood.

p(C\vert D)={p(C)\over p(D)}\,p(D\vert C)

Assume for the moment that there are only two mutually exclusive classes, S and ¬S (e.g. positive and negative review), such that every element is in either one or the other;

p(D\vert S)=\prod_i p(w_i \vert S)\, and p(D\vert\neg S)=\prod_i p(w_i\vert\neg S)\,

Using the Bayesian result above, we can write:

p(S\vert D)={p(S)\over p(D)}\,\prod_i p(w_i \vert S) and p(\neg S\vert D)={p(\neg S)\over p(D)}\,\prod_i p(w_i \vert\neg S)

Dividing one by the other gives:

{p(S\vert D)\over p(\neg S\vert D)}={p(S)\over p(\neg S)}\,\prod_i {p(w_i \vert S)\over p(w_i \vert\neg S)}

Thus, the probability ratio p(S | D) / p(¬S | D) can be expressed in terms of a series of likelihood ratios. The actual probability p(S | D) can be easily computed from log (p(S | D) / p(¬S | D)) based on the observation that p(S | D) + p(¬S | D) = 1

Taking the logarithm of all these ratios, we have:

\ln{p(S\vert D)\over p(\neg S\vert D)}=\ln{p(S)\over p(\neg S)}+\sum_i \ln{p(w_i\vert S)\over p(w_i\vert\neg S)}

Finally, the document can be classified as follows. It is positive review if p(S\vert D) > p(\neg S\vert D)(i.e., \ln{p(S\vert D)\over p(\neg S\vert D)} > 0), otherwise it is a negative review.

1. **IMPLEMENTATION**

There are 4 modules used in this project for the analysis

* Module for retrieving the datasets
* Module for preprocessing the data
* Sentiment analysis module
* Filtering or classification module

1. *Implementation of the Modules*.

The 4 modules can be described as follows:

* Retrieving the datasets: The datasets that has been fetched is in the raw format, we have to retrieve the datasets in the proper format. The attributes required for the analysis should extracted so that so they can be further used.
* Preprocessing of the data: The datasets or the reviews that are extracted have to be processing before being used for analysis. Preprocessing can done by

1. Removing the stop words.
2. Stemming of the reviews.

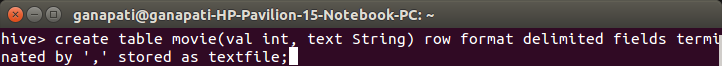
* Sentiment analysis module: The reviews are categorized and word count is done using the HPC tool like Hadoop and categorizing is done on the reviews and divided into positive or negative words.

1.Counting words using hive

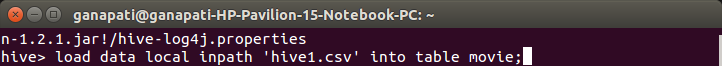
To analyse the reviews we need to count the no of positive and negative words in the review which can be done by counting the no of occurrences of each word. We use hive for this purpose .

The dataset containing reviews is converted into a csv file so that it can be easily processed in hive.The input file to hive is a csv file which is stored in hdfs.

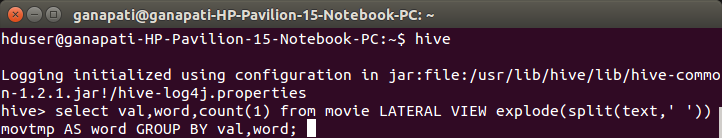
We create a table in hive using a query which is a sql type of query as shown below.

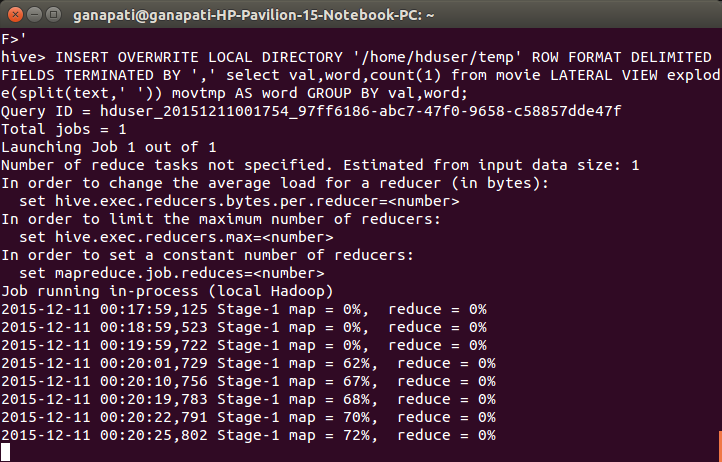


After creating the table, input file in hdfs is loaded into the table in hive. Created input file is loaded into hive as a table using the query as shown below.

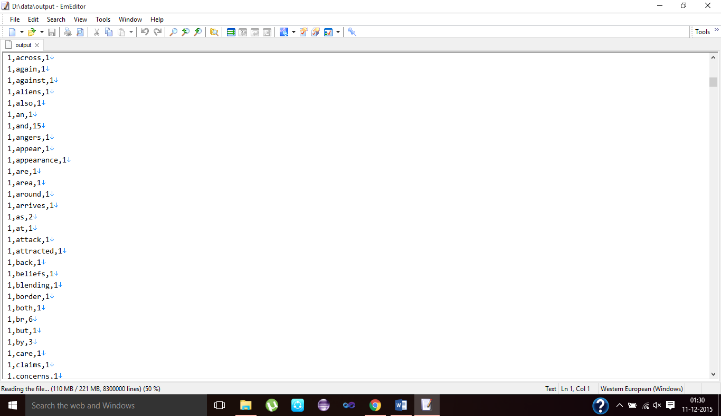


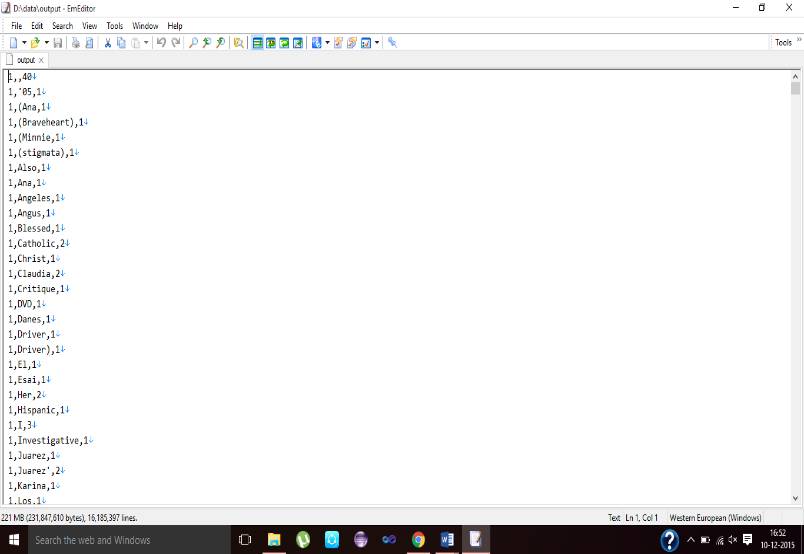
After loading the input file into hive in the form of table we query the table by creating the lateral view and grouping words and their count for each review as shown below.





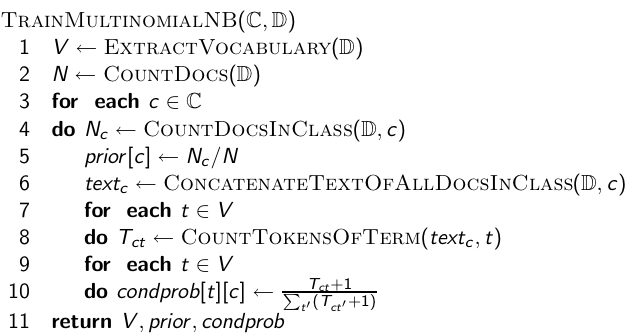
The output of the hive looks like,which is stored in a file in hdfs.



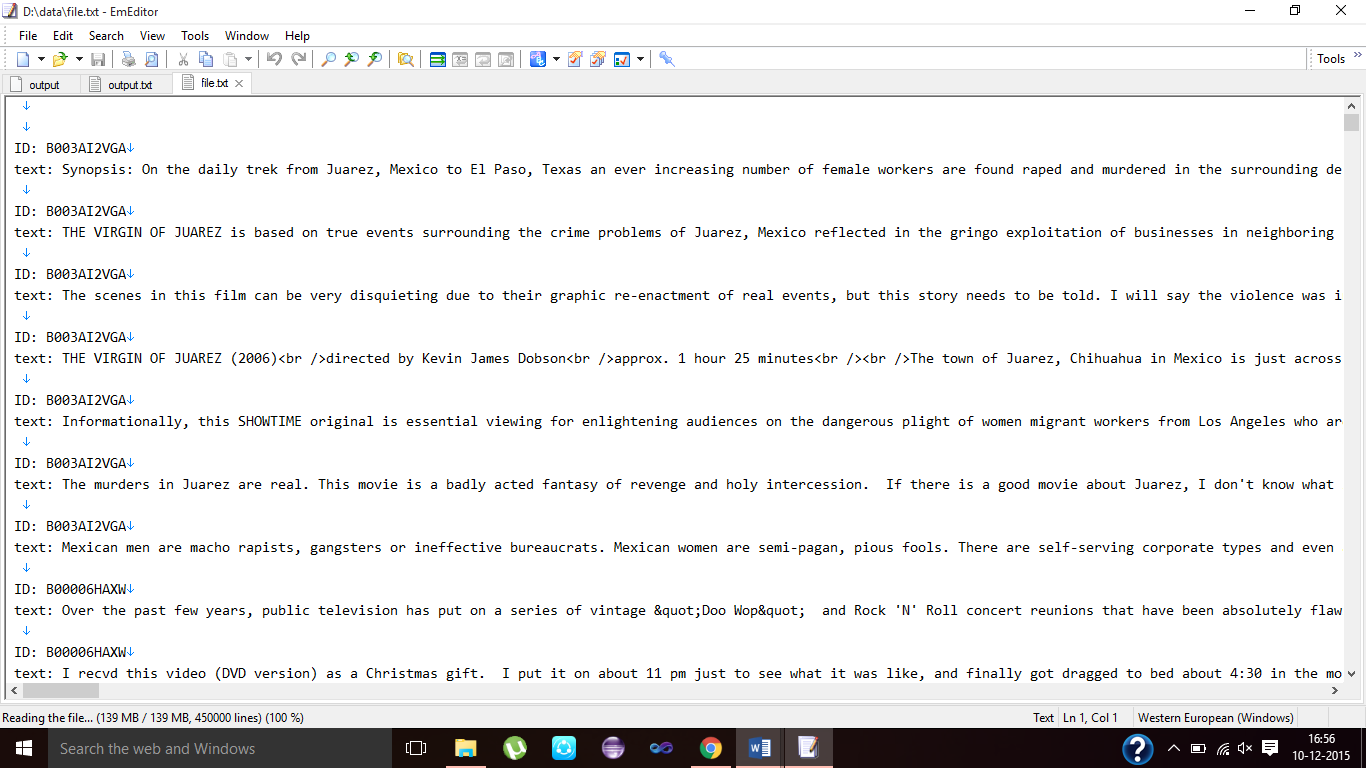
1. Categorizing is done using a java program which counts positive and negative words for each review using files having positive and negative words.

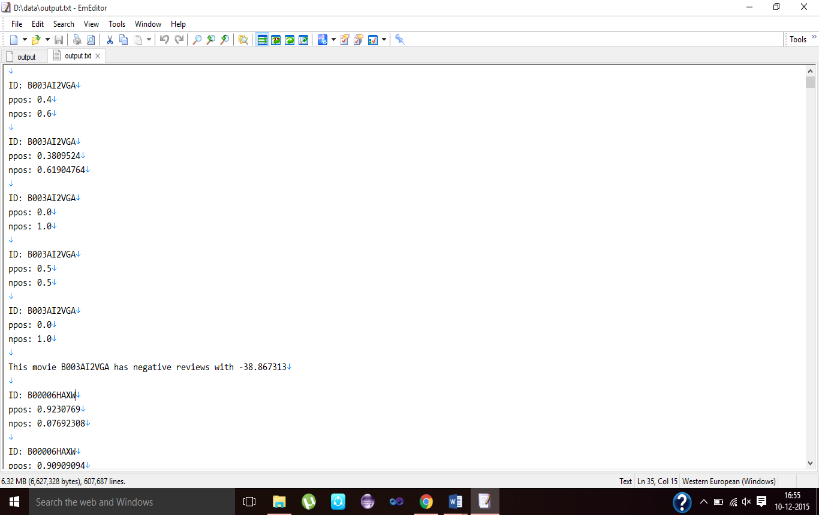
* Classifying and filtering module: The classifications of the reviews are done using the Naïve Bayes classifier and the final output is retrieved.

1. *Algorithm design*
2. Porter’s Algorithm (stemming)
3. Gets rid of plurals and -ed or -ing suffixes.
4. Turns terminal y to i when there is another vowel in the stem.
5. Maps double suffixes to single ones: -ization, -ational, etc.
6. Deals with suffixes, -full, -ness etc.
7. Takes off -ant, -ence, etc.
8. Removes a final –e.
9. Naive bayes algorithm (classification)



1. **RESULTS**





1. **SCOPE AND FUTURE WORK**

The scope of this project aims to satisfy the user community to make their own analysis of this classified data and also can be used for further research topic.

Creating more and new path of exploration in the field of entertainment in this case movies. This expands the horizons of these current fields which were not possible before this.

1. **CONCLUSION**

Our basis of the project is roughly based on the above techniques discussed to build a custom database of these movie reviews with the score on the positive and negative scale. So, that the user community can have these somewhat processed raw data for this dataset to be expanded to produce different products in this topic.

**REFERENCE**

1. https://snap.stanford.edu/data/
2. https://en.wikipedia.org/wiki/Sentiment\_analysis
3. http://www.bogotobogo.com/Hadoop/BigData\_hadoop\_Install\_on\_ubuntu\_single\_node\_cluster.php
4. https://Hadoop.apache.org/
5. https://hive.apache.org