Sentiment Analysis for Movie Reviews using NLP
Prachi Shevate (Team 5)
California State University, Sacramento
prachishevate@csus.edu

Abstract
Sentiment Analysis is an important aspect of Natural Language processing as depicted in the below diagram.

The field of Natural language processing converts text to signals which are understandable by the machine. It allows machines to understand how human speaks. Real-world applications include automatic text summarization, sentiment analysis, topic extraction, named entity recognition, parts-of-speech tagging, relationship extraction and stemming. We use NLP for text mining, machine translation, and automated question answering. In this project, we focus on Sentiment Analysis that helps us analyze people’s sentiments, emotions, and evaluations from written language. The implementation of NLP models to identify a given set of words as positive sentiment or negative sentiment.

Keywords: Sentiments, Natural Language Processing.

1. Background
As we know, sentiment analysis of natural language texts is a vast and growing field. Our previous work particularly relevant to our task falls naturally in two groups. The first relates to techniques to automatically generate sentiment lexicons. The second relates to systems that analyze sentiment on a global or local basis for entire documents.

We should go down a bit. As any software engineer knows, there is a major distinction between the way people speak with each other and the way we “talk” with computers. When composing programs, we need to utilize precisely grammar and structure, yet when chatting with other individuals, we take a great deal of freedoms. We make short sentences. We make longer sentences, we layer in additional importance. We make numerous sentences with a similar importance. We discover numerous approaches to state a similar thing. You get the thought. It's convoluted!

2. Method
Analysis of the sentiments on Movie reviews from a platform like IMDB. By using Natural Language Processing, we will make the computer truly understand more than just the objective definitions of the words. This analysis will help us segregate the data that has good as well as bad movie reviews. It includes using Bag of words model which is a way of extracting features from the text for use in modeling. Not only that but also using Classifier module to identify whether a given piece of text is positive or negative. In this case, we are using Random Forest as our classifier using decision trees. The scikit library will help the algorithm learn with a faster curve, helper class to clean our data, Pandas will help us read our CSV files and NTLK removes unnecessary data from the dataset.
Below diagram defines the process for Sentiment Analysis process:

3. **Experiments**
We evaluate our model with document-level and sentence-level categorization tasks in the domain of online movie reviews. For document categorization, we compare our method to previously published results on a standard dataset and introduce a new dataset for the task. In both tasks, we compare our model's word representations with several bags of words weighting methods and alternative approaches to word vector induction.

3.1. **Architecture**

The above architecture diagram represents the process that takes place throughout the Sentiment Analysis process. **Steps are as follows:**
1. Input files are fed into the Algorithm.
2. To extract features, from the input file and train the dataset.
3. Apply the bag-of-words model to prepare a bag with a vocabulary of words.
4. Use the Random Forest Classifier to train the dataset.
5. The classifier will divide the data into either positive or negative reviews.
6. Finally, the output will be captured as a CSV file with sentiments captured as positive as well as negative.

3.2. **Approach**

3.2.1. Data exploration:
- The uploading of files is done using pandas libraries.
- A conversion of the variable is done to different data types.
- Creation of plots will be done using matplotlib.
- Using a bag of words to store the data by splitting it into words. This is a Natural Language processing model that uses NLTK.
- Using Scikit-learn for using Random Forest Classifier.

3.2.2. Model construction:
- Natural Language Processing - Bag of words model
- Decision Trees using Random Forest Classifier
- NLTK – Natural Language Processing Toolkit

3.2.3. Performance Evaluation:
- A separate CSV file is created which will store the positive and negative reviews of a particular movie.
- A comparison will then be based on the actual data and the training data that we have available for testing.

3.3. **Models**

3.3.1. Bag of Words Model
We can define the bag-of-words model as a simplifying representation used in natural language processing and information retrieval. It is also known as the Vector Space Model. In this model, a text such as a sentence or a document is represented as the bag of its words by disregarding grammatical errors and even word order. We maintain the multiplicity in case of word order. For example, consider two sentences,

**Sentence 1:** The cat is moving towards the dog.
**Sentence 2:** The dog is eating food.

The list of words that are formed from the above two sentences will be:

```
{cat, moving, towards, dog, eating, food}
```

This list is stored in a bag and that forms a “Bag of words”.

3.3.2. Natural Language Tool Kit (NLTK)
Finally, we need to decide how to deal with frequently occurring words that don't carry much meaning. Such words are called "stop words"; in English, they include words such as "a", "and", "is", and "the". Conveniently, there are Python packages that come with stop word lists.
built in. Let's import a stop word list from the Python Natural Language Toolkit (NLTK). The following commands to use the library:

```python
import nltk
nltk.download()
```

### 3.3.3. Word2Vec Utility

Word2vec, published by Google in 2013, is a neural network implementation that learns distributed representations for words. Other deep or recurrent neural network architectures had been proposed for learning word representations prior to this, but the major problem with these was a long time required to train the models. Word2Vec learns quickly relative to other models.

Word2Vec does not need labels in order to create meaningful representations. This is useful since most data in the real world is unlabeled. If the network is given enough training data (tens of billions of words), it produces word vectors with intriguing characteristics. Words with similar meanings appear in clusters, and clusters are spaced such that some word relationships, such as analogies, can be reproduced using vector math. The famous example is that, with highly trained word vectors, "king - man + woman = queen."

**Problem using this model:** The accuracy level using Word2Vec model is very low. We can say around 0.75. And the main focus of our program would be to get an accuracy that is higher than Word2Vec model.

### 3.4. IMDB Review Dataset

The dataset is from IMDB, which a famous movie online database of information related to world movies and other entertainment sources. A collection of 50,000 reviews from IMDB, where it allows no more than 30 reviews per movie. The constructed dataset contains an even number of positive and negative reviews, so randomly guessing yields 50% accuracy.

Two important conditions are considered while computing the sentiments:

1. A negative review has a score \( \leq 4 \) out of 10,
2. A positive review has a score \( \geq 7 \) out of 10.

The dataset is divided into two sets, namely training and test sets. The training set has 25,000 labeled reviews used to induce word vectors. The 25,000 review labeled training set does not include any of the same movies as the 25,000 review test set. In addition, there are another 50,000 IMDB reviews provided without any rating labels. Differences in accuracy are small, but, because the test set contains 25,000 examples, the variance of the performance estimate is quite low. For example, an accuracy increase of 0.1% corresponds to correctly classifying an additional 25 reviews [2].

### 3.5. Word Representation Learning

We initiate word portrayals with our model utilizing 25,000 motion picture surveys from IMDB. Since a few films get significantly a greater number of audits than others, we constrained ourselves to include at most 30 surveys from any motion picture in the gathering. We fabricate a settled word reference of the 5,000 most continuous tokens, however, overlook the 50 most regular terms from the first full vocabulary. Conventional stop word evacuation was not utilized in light of the fact that specific stop words (e.g. nullifying words) are characteristic of estimation. Stemming was not connected on the grounds that the model learns comparable portrayals for expressions of a similar stem when the information recommends it. Moreover, on the grounds that specific non-word tokens (e.g. "!" and ":-") are characteristic of conclusion, we permit them in our vocabulary. Evaluations on IMDB are given as star esteem \( (\in \{1, 2, ..., 10\}) \), which we directly guide to \([0, 1]\) to use as report marks when preparing our model. The semantic segment of our model does not require archive marks. We prepare a variation of our model which utilizes 50,000 unlabeled surveys notwithstanding the marked arrangement of 25,000 audits. The unlabeled arrangement of audits contains unbiased surveys and in addition those which are energized as found in the named set. Preparing the model with extra unlabeled information catches a typical situation where the measure of named information is little in respect to the measure of unlabeled information accessible. For all word vector models, we utilize 50-dimensional vectors.

### 3.6. Capturing Word Sentiment

The model introduced so far does not catch sentiment information. Applying this calculation to reports will deliver portrayals where words that happen together in archives have comparative portrayals. Nonetheless, this unsupervised approach has no unequivocal method for catching which words are prescient of slant instead of substance-related. Much past work in natural language processing accomplishes better portrayals by gaining from numerous undertakings (Collobert and Weston, 2008; Finkel and Manning, 2009). Following this topic, we acquit a moment assignment with use marked reports to enhance our model's statement portrayals [6]. The slant is a complex, multi-dimensional idea. Contingent upon which parts of feeling we wish to catch, we can give some collection of content a sentiment label which can be categorical, continuous, or multi-dimensional.

We are aware of the fact that “sentiment” is a multi-dimensional and complex concept. We can give a sentiment label “SL” depending on which aspects of sentiment we wish to capture. This can be defined in 3 categories: categorical, continuous, or multi-dimensional. To leverage such labels, we introduce an objective that the word vectors model should predict the sentiment label using some appropriate predictor.
In the above equation, an appropriate predictor function \( f(x) \) is used to map a word vector \( \phi_w \) to a predicted sentiment label \( SL^* \). We can then improve our word vector \( \phi_w \) to better predict the sentiment labels of contexts in which that word occurs.

3.7. Code snippets

There are 7 important steps performed to achieve an efficient accuracy. Steps are as follows:

**Step 1: Importing data files**

*Import:* import pandas as pd

*Syntax:*

```python
pd.read_csv(os.path.join(os.path.dirname(__file__),folder_name, file_name), header=0, delimiter=",", quoting=3)
```

**Step 2: Cleaning and parsing training set movie reviews**

*Syntax:*

```python
clean_train_reviews.append(".
```

**Step 3: Creating bag of words using CountVectorizer**

*Syntax:*

```python
CountVectorizer(analyzer="word", tokenizer=None, preprocessor=None, stop_words=None, max_features=<specify the limit for features>)
```

**Step 4: Initializing Random Forest with number of trees**

*Import:* from sklearn.ensemble import RandomForestClassifier

*Syntax:*

```python
RandomForestClassifier(n_estimators=<specify the number of trees>)
```

**Step 5: Fit the forest to the training set**

```python
forest.fit(train_data_features, train["sentiment"])
```

**Step 6: Make predictions using Random forest model**

```python
predict(<data features>)
```

**Step 7: Calculating the accuracy using mean absolute error**

```python
mean_abs_error = mean_absolute_error(<output>, <key>)
accuracy = (1 - mean_abs_error)
```

3.8. Accuracy Chart

- **Best work:**
  The best accuracy I have achieved so far using Bag of Words model of Natural Language processing is **92.34%** which is greater than other models by **3.17%**

**Snapshot of accuracy:**

Predicting test labels...

Wrote results to Bag_of_Words_model.csv

The absolute error rate is: 7.6533333333333 %
The accuracy percentage is: 92.3466666666666 %

**Snapshot of Data:**

- **Observations for worst model used in previous work:**
  As seen in above table, Word2Vec model gives an accuracy rate of just 75% whereas using the Bag-of-words models can provide us with an accuracy rate that is greater than 80% or above. So, this model cannot be used to predict a good accuracy.

4. Conclusion

Explored a model that Natural Language processing uses which is known as the “Bag of words model”. It learns word portrayals catching semantic and opinion data. The Bag of Words display takes in a vocabulary from the majority of the archives, at that point models each report by tallying the circumstances each word shows up. The unsupervised model was reached out to fuse notion data
and demonstrated how this expanded model can use the wealth of feeling named writings accessible online to yield word portrayals that catch both assumption and semantic relations. The utility of such portrayals is shown on two assignments of feeling arrangement, utilizing existing datasets and in addition a bigger one that would be accessible for future research. These undertakings include moderately straightforward estimation data, however the model is profoundly adaptable in such manner; it can be utilized to portray a wide assortment of explanations, and in this manner is comprehensively relevant in the developing zones of conclusion examination and recovery.

5. Lessons Learnt
- Data pre-processing is extremely necessary to get better accuracy. It helps us understand the model construction. In my case, it helped me to explore two important machine learning models: Bag of words model of Natural Language Processing and Decision Trees using Random Forest Classifier.
- It is extremely important to consider the previous work done in field we want to work. Research helps us understand about the achievements, as well as various optimal ways to find a better solution.

6. Future Work
In future, I plan to extend my work by implementing a model which will improve my accuracy more than 92%. Also to include comparisons with different machine learning models.

7. References


