A multi-faceted comparative approach on sentiment analysis on review text

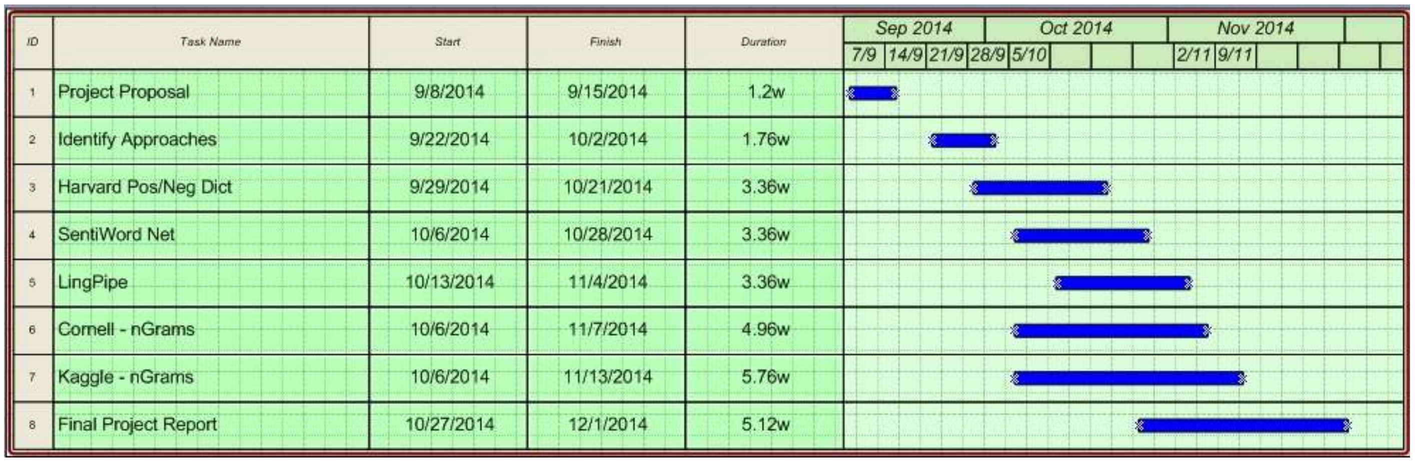
SENTIMENT PREDICTION

IDS – 594 Big Data Analytics

Prathima, Saurabh and Vignesh

Table of Contents

MILESTONES



ID	Task Name	Start	Finish	Duration
1	Project Proposal	9/8/2014	9/15/2014	1.2w
2	Identify Approaches	9/22/2014	10/2/2014	1.76w
3	Harvard Pos/Neg Dict	9/29/2014	10/21/2014	3.36w
4	SentiWord Net	10/6/2014	10/28/2014	3.36w
5	LingPipe	10/13/2014	11/4/2014	3.36w
6	Cornell - nGrams	10/6/2014	11/7/2014	4.96w
7	Kaggle - nGrams	10/6/2014	11/13/2014	5.76w
8	Final Project Report	10/27/2014	12/1/2014	5.12w

INTRODUCTION

1. BACKGROUND

Sentiment Analysis, as the term indicates, is the process of analyzing the sentiments or the contextual meaning of words or combination of words from subjective materials. It refers to the use of natural language processing, text analysis and computational linguistics to identify and extract subjective information in source materials. Therefore, it is also known as opinion mining.

Also termed as polarity prediction, it is the most researched topic around social networking datasets and has many popular models, all of which prove to be effective in a specific context or on a specific type of dataset.

Popular methods are:

- Keyword Spotting
- Lexical Affinity
- Statistical Methods
- Concept-Level Techniques

While the definitions and approaches differ under each method, the following are deemed necessary for improving accuracy of the sentiment analysis process.

1. Stop Words Removal
   In computing, stop words are words which are filtered out before or after processing of natural language data (text). There is not one definite list of stop words which all tools use and such a filter is not always used. Some tools specifically avoid removing them to support phrase search.

   Any group of words can be chosen as the stop words for a given purpose. For some search engines, these are some of the most common, short function words, such as the, is, at, which, and on. Other search engines remove some of the most common words—including lexical words, such as "want"—from a query in order to improve performance.

2. POS Tagging
   In corpus linguistics, part-of-speech tagging (POS tagging or POST), also called grammatical tagging or word-category disambiguation, is the process of marking up a word in a text (corpus) as corresponding to a particular part of speech, based on both its definition, as well as its context—i.e. relationship with adjacent and related words in a phrase, sentence, or paragraph. A simplified form of this is commonly taught to school-age children, in the identification of words as nouns, verbs, adjectives, adverbs, etc.

3. Stemming
   Stemming is the term used in linguistic morphology and information retrieval to describe the process for reducing inflected (or sometimes derived) words to their word stem, base or root form—generally a written word form. The stem needs not to be identical to the morphological root of the word; it is usually sufficient that related words map to the same stem, even if this stem is not in itself a valid root. Algorithms for stemming have been studied in computer science since the 1960s. Many search engines treat words with the same stem as synonyms as a kind of query expansion, a process called conflation.
4. Predefined sentiment / emotion affinity
   Many words that cannot be matched with polarities are updated with their proximity or affinity to an emotion that can be quantified with a polarity. This is done mostly using Latent Dirichlet Association or Similarity Matrices methods.

A variety of tools and frameworks make way for a large number of combinations to work with.

Some of the popular tools for data preparation are:

1. Rapid Miner – Stop Words Removal, POS Tagging and Stemming
2. Open NLP – POS Tagging and Stemming
3. NLTK – Stemming and Phrase Identification

Some of the frameworks that provide a sound platform or means for performing the increased accuracy of results are:

1. Apache Hadoop
2. Mahout
3. Apache PIG
4. Apache Hive
5. HBase

2. MOTIVATION

The following were the factors that caught our eye and served as our motivation to proceed with a thorough research on Sentiment Analysis methods that will help get the closest to the actual sentiment.

1. The Data Prediction Competition hosted by Kaggle:

   http://www.kaggle.com/c/sentiment-analysis-on-movie-reviews/data

2. The variety of methods with varying levels of accuracy available for sentiment analysis and
3. The dependency on effective opinion or sentiment analysis in the social networking age that we are in.

3. PROBLEM/CHALLENGES

Research on Sentiment Analysis although looked at as a Text Classification task, is more complicated when dealt with as an opinion mining exercise. Once looked into as an exercise to identify the actual sentiment of the text, the factor of Subjectivity, whether the comment is Subjective (talks about the Features of a Product or a topic / function in an Application) or Objective (calls out on opinions on a Product or an Application).

Classifications vary based on causal variables like:

1. Who the opinion holder is- EXAMPLE: Opinions differ when the opinion holder is part of the Product organization under review
2. Orientation of an opinion- EXAMPLE: If opinions are already classified as Positive and Negative, further classification and predictions will be influenced by the predetermined orientation
3. The Model of the Object under review- EXAMPLE: If the object is divided into features with individual positive and negative comments on each, a collective opinion cannot be derived on the object as such.
4. Model of the Opinionated Document- EXAMPLE: A direct opinion can be used as such to derive an opinion or sentiment whereas a comparative opinion that grades an object based on another object will get us a comparative sentiment influenced by the object that it is compared with and the opinion of the opinion holder on the compared object.

In order to address most of these variations, we have performed a short variety of sentiment analysis approaches that will provide a comparative overlook on the ability of each method to predict or come close to analyzing a reasonable sentiment off the text under review.

4. OBJECTIVE

The objective of this Project submission is to provide a variety of implementations that intend to encompass all the challenges perceived in Sentiment Analysis.

1. Opinionated Sentiment Analysis – Harvard Positive / Negative List Comparison Model
2. Model of the Document (POS Tags) – SentiwordNet Model
3. Opinion Orientation – LingPipe Sequential Model
4. Object Feature Granularity – N-Grams Model - Bi-grams and Tri-grams
5. Live Review Data; Supervised Learning – Feature Extraction - N-Grams Model - Bi-grams and Tri-grams

APPROACH 1 – Using Harvard Pos/Neg Dictionary

1. Data Collection and Processing

Harvard Positive and Negative Word Lists used in this approach are from a parallel implementation from Data Mining Course at UIC.

DATA COLLECTION	DATA SOURCE	PROCESSING
BASELINE	Harvard Positive and Negative Word List	Retain as Text Files
INPUT	User entered comment / review text as 1 comment per text file	Stop Words Removal

2. Implementation

Step 1: Remove Stop Words and Punctuation

Step 2: Compare individual words to the Harvard positive and Harvard negative word lists.

Step 3: If found in positive list then score word as +2 and if found in negative list then score word as -2.

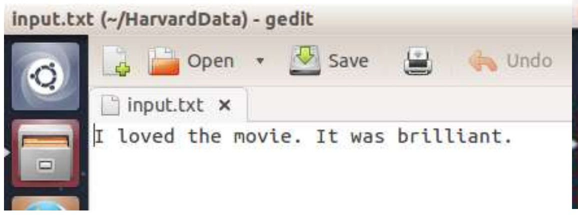
Step 4: If no match found, a relatedness confidence is found; if confidence is above +0.6 in relation to the positive or negative list, 1 or -1 is assigned to such words, respectively.

Step 5: The total score is calculated per comment.

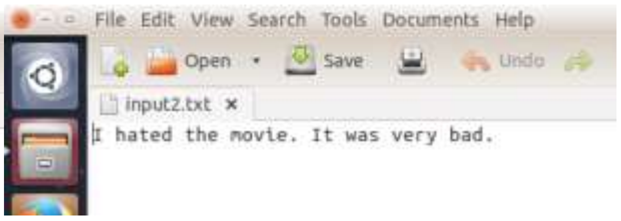
Score > 0: POSITIVE; Score < 0: NEGATIVE; Score = 0: NEUTRAL.

3. Result

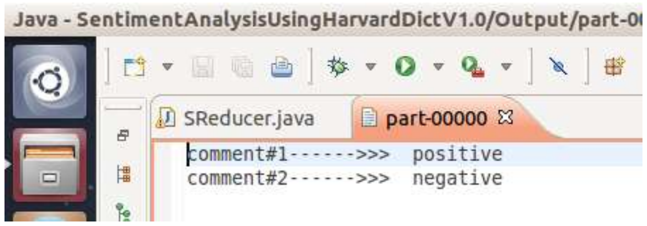
Result Set	Text	Expected	Result
Comment 1	I loved the movie. It was brilliant.	Positive	Positive
Comment 2	I hated the movie. It was very bad.	Negative	Negative



```
I loved the movie. It was brilliant.
```



```
I hated the movie. It was very bad.
```



```
comment#1------>>>  positive
comment#2------>>>  negative
```

4. Inference

- Simple comments without any irony or sarcasm can be easily deduced using Approach 1 - Harvard Positive and Negative Word List.
- Multiple comments related to a product or object can be given in order to deduce the overall polarity of review – positive / negative.

APPROACH 2 – Using SentiWordNet

1. Data Collection and Processing

DATA COLLECTION	DATA SOURCE	PROCESSING
BASELINE	Stanford CoreNLP – POS Tagging	Retain as Text File
INPUT	User entered comment / review text	Stop Words Removal

2. Implementation

Step 1: Remove Stop Words and Punctuation

Step 2: Perform POS Tagging and Lemmatization using Stanford CoreNLP.

EXAMPLE:

NN	Noun, singular or mass
NNS	Noun, plural
NNP	Proper noun, singular
NNPS	Proper noun, plural

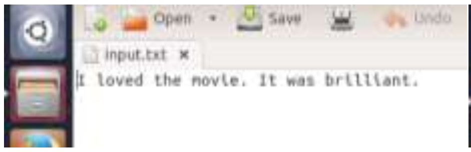
Step 3: Score each tagged word using its POS Tag by the Sentiwordnet approach.

Step 4: If no match found, a relatedness confidence is found; if confidence is above +0.6 in relation to the positive or negative list, 1 or -1 is assigned to such words, respectively.

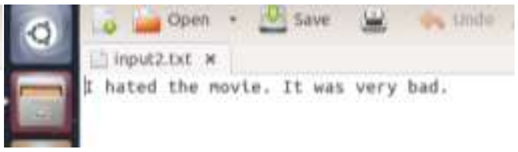
Step 5: The total score is calculated per comment. Score > 0: POSITIVE; Score < 0: NEGATIVE; Score = 0: NEUTRAL.

3. Result

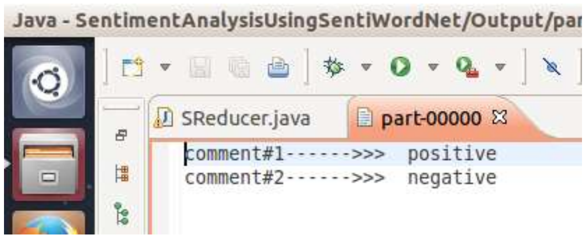
Result Set	Text	Expected	Result
Comment 1	I loved the movie. It was brilliant.	Positive	Positive
Comment 2	I hated the movie. It was very bad.	Negative	Negative



```
I loved the movie. It was brilliant.
```



```
I hated the movie. It was very bad.
```



```
comment#1------>>>  positive
comment#2------>>>  negative
```

4. Inference

- Simple comments without any irony or sarcasm can be easily deduced using Approach 1 - Harvard Positive and Negative Word List.
- Multiple comments related to a product or object can be given in order to deduce the overall polarity of review – positive / negative.
- The sentiwordnet approach is restricted to the accuracy of the POS Tag associated with each word; if the POS tagging is of poor accuracy and quality, the predictions will be skewed accordingly.

APPROACH 3 – Using Lingpipe (Sequential Program)

1. Data Collection and Processing

This approach was performed for verifying our understanding on the Model creation on Cornell Dataset. This was performed in preparation for Approach 4.

DATA COLLECTION	DATA SOURCE	PROCESSING
BASELINE	Cornell Dataset	Retain as Text File
INPUT	User entered comment / review text	None

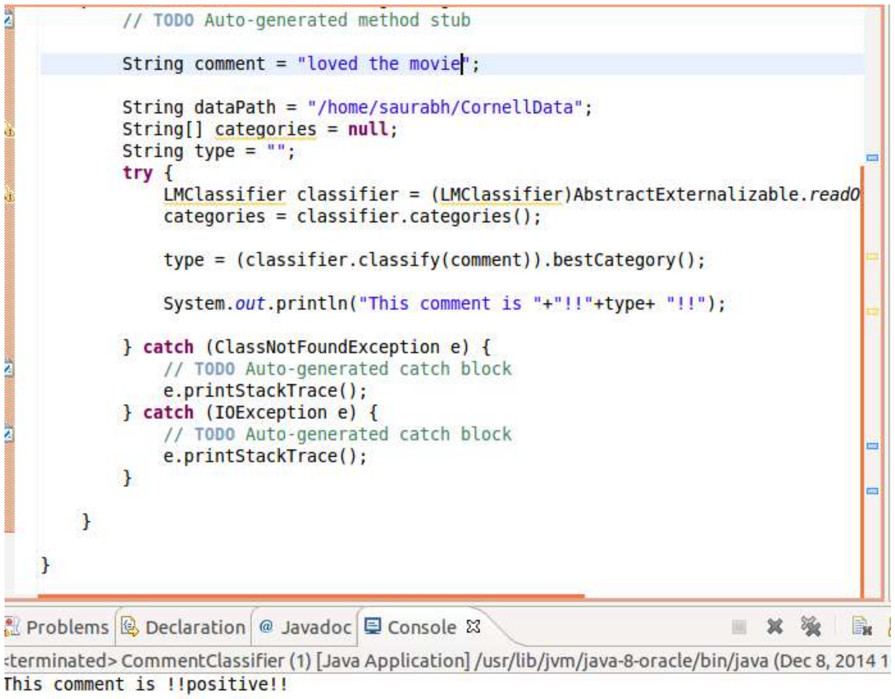
2. Implementation

Step 1: Trained the model on Cornell movie review data to form a model and create a classification category file.

Step 2: Apply the model on the input to identify a sentiment per comment.

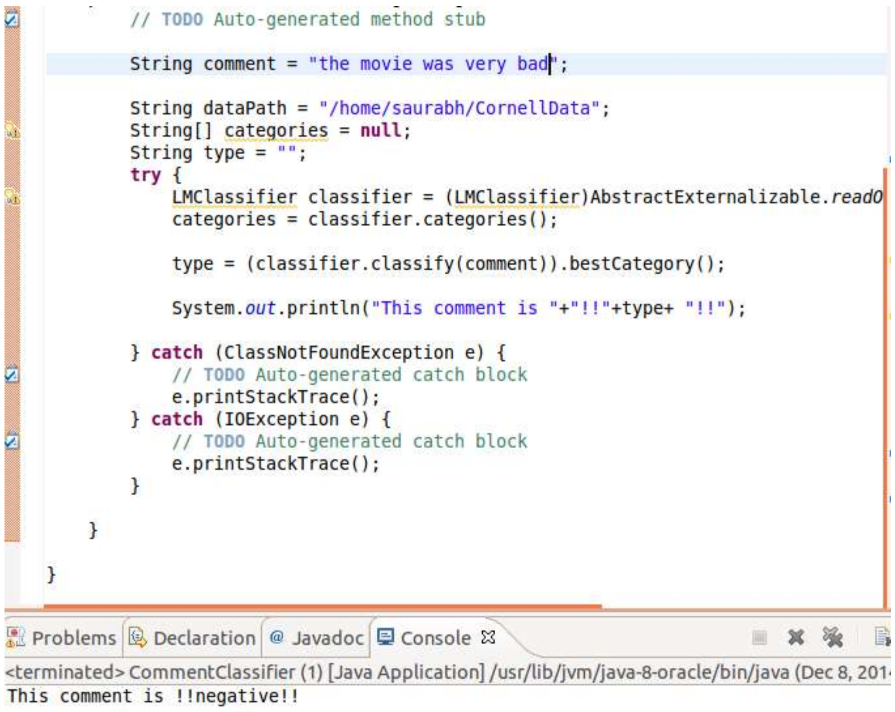
3. Result

Result Set	Text	Expected	Result
Comment 1	Loved the movie	Positive	Positive
Comment 2	The movie was very bad	Negative	Negative
Comment 3	The movie was somewhat good but very lengthy	Positive	Negative



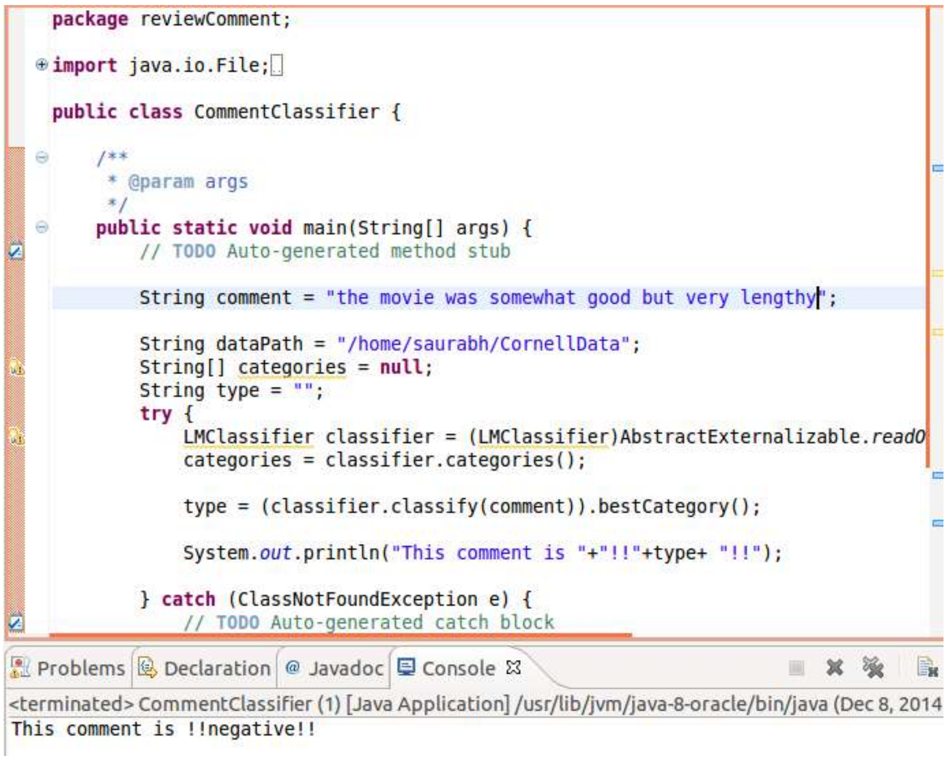
```
        // TODO Auto-generated method stub

        String comment = "loved the movie";

        String dataPath = "/home/saurabh/CornellData";
        String[] categories = null;
        String type = "";
        try {
            LMClassifier classifier = (LMClassifier)AbstractExternalizable.readO
            categories = classifier.categories();

            type = (classifier.classify(comment)).bestCategory();

            System.out.println("This comment is "+"!!"+type+ "!!");

        } catch (ClassNotFoundException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        } catch (IOException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }

    }

}
```

```
<terminated> CommentClassifier (1) [Java Application] /usr/lib/jvm/java-8-oracle/bin/java (Dec 8, 2014 1
This comment is !!positive!!
```



```
        // TODO Auto-generated method stub

        String comment = "the movie was very bad";

        String dataPath = "/home/saurabh/CornellData";
        String[] categories = null;
        String type = "";
        try {
            LMClassifier classifier = (LMClassifier)AbstractExternalizable.readO
            categories = classifier.categories();

            type = (classifier.classify(comment)).bestCategory();

            System.out.println("This comment is "+"!!"+type+ "!!");

        } catch (ClassNotFoundException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        } catch (IOException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }

    }

}
```

```
<terminated> CommentClassifier (1) [Java Application] /usr/lib/jvm/java-8-oracle/bin/java (Dec 8, 201
This comment is !!negative!!
```



```
package reviewComment;

import java.io.File;

public class CommentClassifier {

    /**
     * @param args
     */
    public static void main(String[] args) {
        // TODO Auto-generated method stub

        String comment = "the movie was somewhat good but very lengthy";

        String dataPath = "/home/saurabh/CornellData";
        String[] categories = null;
        String type = "";
        try {
            LMClassifier classifier = (LMClassifier)AbstractExternalizable.readO
            categories = classifier.categories();

            type = (classifier.classify(comment)).bestCategory();

            System.out.println("This comment is "+"!!"+type+ "!!");

        } catch (ClassNotFoundException e) {
            // TODO Auto-generated catch block
```

```
<terminated> CommentClassifier (1) [Java Application] /usr/lib/jvm/java-8-oracle/bin/java (Dec 8, 2014
This comment is !!negative!!
```

4. Inference

- While lingpipe does take care of simple comments, comments that have a skewed polarity are being classified wrong, as seen in Comment 3 above. Therefore, using the lingpipe model on cornell data set is not one of the perfect solutions for all review comments.

APPROACH 4 – Cornell dataset - nGrams

1. Data Collection

As suggested by you during our meetings about this project, that we can use Cornell dataset for our project. We went ahead with that idea and found a collection of movie-review documents. We found different versions of following three types of datasets -

- Sentiment Polarity Dataset - labeled with respect to their overall sentiment polarity (positive or negative)
- Sentiment Scale Dataset - labeled with respect to their subjective rating (e.g., "two and a half stars")
- Subjectivity Dataset - sentences labeled with respect to their subjectivity status (subjective or objective)

Out of these 3 we selected the polarity dataset v2.0 (3.0Mb) which includes 1000 positive and 1000 negative processed reviews. *Introduced in Pang/Lee ACL 2004. Released June 2004.*

2. Processing

The reviews can have some valuable info about it's sentiment but the rest of the words may or may not really help in determining the review's sentiment. Hence, it makes sense to preprocess the movie reviews.

- Removing Special Characters – we are removing all the special characters mentioned below in the following code

```
//special characters
 text = text.replaceAll("\\'|\"|[()]|[0-
 9]|\\/|\\_|\\[|\\]|\\#|\\$|\\*|\\=|\\`|\\>|\\<|\\?|\\&|\\@|\\+|\\|\\%|\\^|\\+", "");
```

- Replacing Punctuations – here we are replacing all the punctuations mentioned below in the following code with space

```
 //for punctuations
text = text.replaceAll("\\.|\\,|\\?|\\!|\\:|\\;|\t|\\-", " ");
```

- Replacing White spaces – we are replacing extra white spaces between the words with a single space as mentioned below in the code

```
text = text.replaceAll("\\s+", " ");
```

- Stop Words Removal
- Convert to Lower case
- Creating tokens of words by splitting them using space ("\\s+") as the delimiter.

3. Implementation

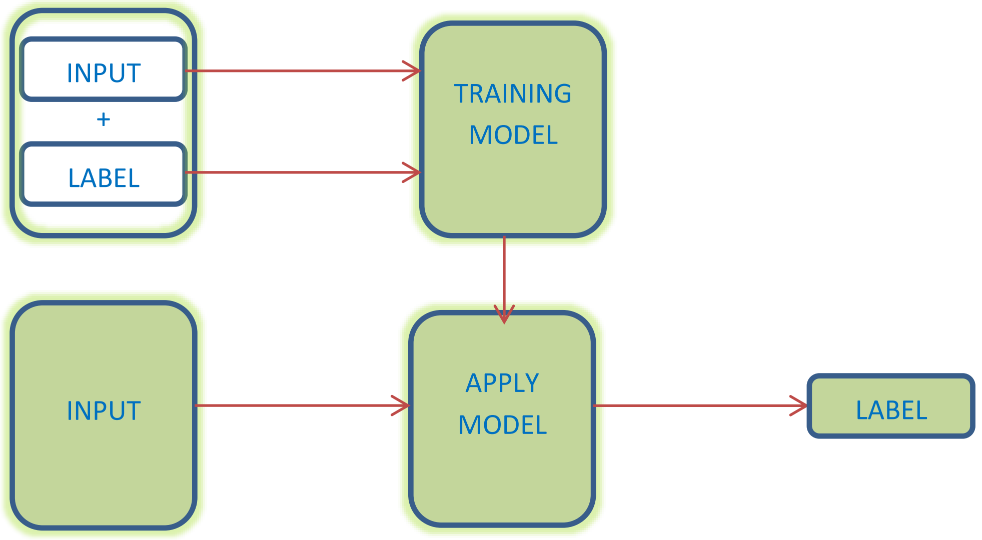


Figure 1: Cornell Implementation Method

<u>MAPPER</u>

- We are using the already positive and negative classified reviews to train the classifier
- In the training data, consisting of positive and negative review, we are filtering some unwanted words and punctuations as mentioned in the processing section above. We are then splitting each review into words
- After splitting the words we are forming bi-grams (e.g. “not good”) and tri-grams (e.g. “not very bad”) and checking if the label is positive or if negative, we are adding them to their respective lists named positive and negative
- As we process, each of the reviews, we keep adding words to the lists and ignoring other words
- Now, if we give an unseen data/review, we do the processing and tokenize the review and form bi-grams and tri-grams

<u>REDUCER</u>

- After this, we match each of them against out positive and negative list of words. If we find a match we do the following
    - Positive tri-gram : add score +3
    - Positive bi-gram : add score +2
    - Negative tri-gram : add score -3
    - Negative bi-gram : add score -2

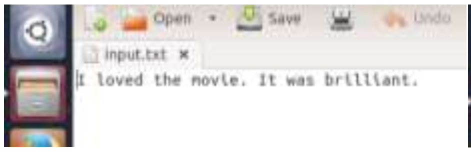
- We are not adding unigrams to the list as it won't give an accurate performance and rather will increase the size of our list of positive and negative words and hence leading to an increased time of execution of the unseen data

- We add the score and if the

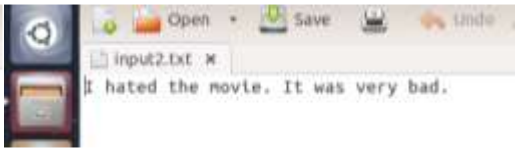
    - Score > 0 : predicted label = positive
    - Score < 0 : predicted label = negative

4. Result

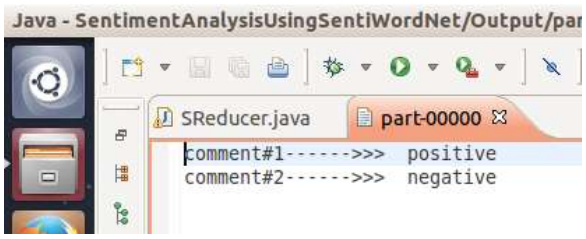
Result Set	Text	Expected	Result
Comment 1	I loved the movie. It was brilliant.	Positive	Positive
Comment 2	I hated the movie. It was very bad.	Negative	Negative



```
I loved the movie. It was brilliant.
```



```
I hated the movie. It was very bad.
```



```
comment#1------>>>  positive
comment#2------>>>  negative
```

5. Inference

- Simple comments without any irony or sarcasm can be easily deduced using Approach 1 - Harvard Positive and Negative Word List.
- Multiple comments related to a product or object can be given in order to deduce the overall polarity of review – positive / negative.
- Comments with sarcasm are also being identified correctly as the cornell dataset is extensive and performing the bigrams and trigrams are aiding in the accuracy of prediction.

APPROACH 5 – Kaggle dataset - nGrams

1. Data Collection

We came across this while looking for the project proposal for IDS 594 – Big Data Analytics and found it very interesting. This is a competition titled "Classify the sentiment of sentences from the Rotten Tomatoes dataset"

- *The Rotten Tomatoes movie review dataset is a corpus of movie reviews used for sentiment analysis, originally collected by Pang and Lee [1]. In their work on sentiment treebanks, Socher et al. [2] used Amazon's Mechanical Turk to create fine-grained labels for all parsed phrases in the corpus.*

- The dataset is comprised of tab-separated files with phrases from the Rotten Tomatoes dataset. The train/test split has been preserved for the purposes of benchmarking, but the sentences have been shuffled from their original order. Each Sentence has been parsed into many phrases by the Stanford parser. Each phrase has a PhraseId. Each sentence has a SentenceId. Phrases that are repeated (such as short/common words) are only included once in the data.
    - Train.tsv contains the phrases and their associated sentiment labels. We have additionally provided a SentenceId so that you can track which phrases belong to a single sentence.
    - test.tsv contains just phrases. You must assign a sentiment label to each phrase.

The sentiment labels are:

- 0 - negative
- 1 - somewhat negative
- 2 - neutral
- 3 - somewhat positive
- 4 - positive

2. Processing

- The train.tsv file is in the following format, we are using the PhraseId SentenceId and Sentiment for the analysis

```
PhraseId    SentenceId  Phrase      Sentiment
1     1     A series of escapades demonstrating the adage that what is
good for the goose is also good for the gander , some of which
occasionally amuses but none of which amounts to much of a story .   1
2     1     A series of escapades demonstrating the adage that what is
good for the goose      2
3     1     A series    2
4     1     A     2
5     1     series      2
```

- While going through the train.tsv data we saw a lot of characters like ("'s"), hence removing that

```
// removing 's
line = line.replaceAll("'s", "");
```

- Removing all the special characters mentioned below in the following code

```
// remove special characters
line = line.replaceAll(
"\\'|\"|[()]|\\/|\\_|\\[|\\]|\\#|\\$|\\*|\\=|\\`|\\>|\\<|\\?|\\&|\\@|\\+|\\|\\%|\\^|\\+","");
```

- Replacing all the punctuations with white space

```
// remove punctuations
line = line.replaceAll("\\.|\\,|\\?|\\!|\\:|\\;|\t|\\-", " ");
```

- Replacing all extra spaces and tabs with

```
// replace all spaces
line = line.replaceAll("\\s+|\t|\n", " ");
```

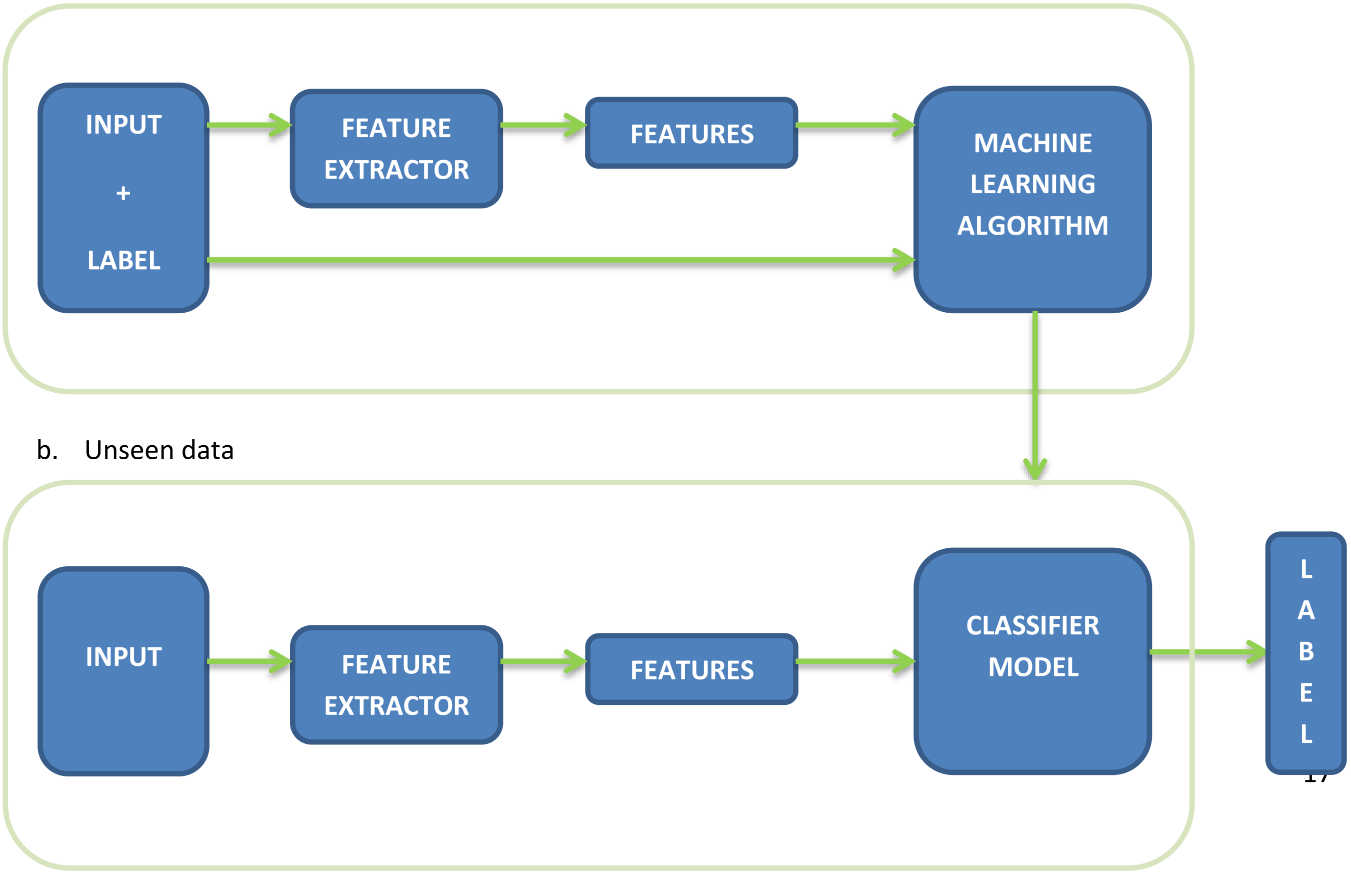
- Converting to lower case and tokenizing into words using space as the delimiter

```
splits = line.trim().toLowerCase().split("\\s+");
```

3. Implementation

a. Training

b. Unseen data



Training the Classifier

The classifiers need to be trained and to do that we are using the train.tsv and the following mentioned steps on that dataset

MAPPER

- We are using PhraseId as the key, then we are checking the length of the Phrase if the length of the Phrase is >2 then we are creating bigrams and trigrams for the respective Phrases and retaining all the phrases with length <= 2

- The key, value pair we are sending to the reducer is in the form of PhraseId as key and the "sentiment_bigram/trigram" as the value

CREATING Feature vector is the most important concept in implementing a classifier. A good feature vector directly determines how successful your classifier will be. The feature vector is used to build a model which the classifier learns from the training data and further can be used to classify previously unseen data.

REDUCER

- Here we are splitting the value using "_" as the delimiter and storing the bigram/trigram as the key and its respective sentiment rating in the value

Running The Test.tsv

- Our test.tsv is in the following format -

```
PhraseId    SentenceId  Phrase
156061      8545  An intermittently pleasing but mostly routine effort
.
156062      8545  An intermittently pleasing but mostly routine effort
156063      8545  An
156064      8545  intermittently pleasing but mostly routine effort
156065      8545  intermittently pleasing but mostly routine
156066      8545  intermittently pleasing but
```

- In mapper for the test dataset we are just using the PhraseId and the Phrase. We are using PhraseId as the key, then we are checking the length of the Phrase if the length of the Phrase is >2 then we are creating bigrams and trigrams for the respective Phrases and retaining all the phrases with length <= 2

- The key, value pair we are sending to the reducer is in the form of PhraseId as key and the "bigram/trigram" as the value

REDUCER

- Here we are reading our feature vector which contains the pre-processed bigrams/trigrams, their respective sentiment rating and storing them into a hash map whose key is the bigram and trigram and value is the sentiment rating

- After this we are using the *HashMap.getKey();* to check whether we find a match between the bigram/trigram sent from the mapper with the already existing training model

- Since we have 5 sentiment ratings we have created 5 counters each named Count0, Count1, Count2, Count3 and Count4 and every time we find a match we check the value from the HashMap and increase that respective rating's counter by +1

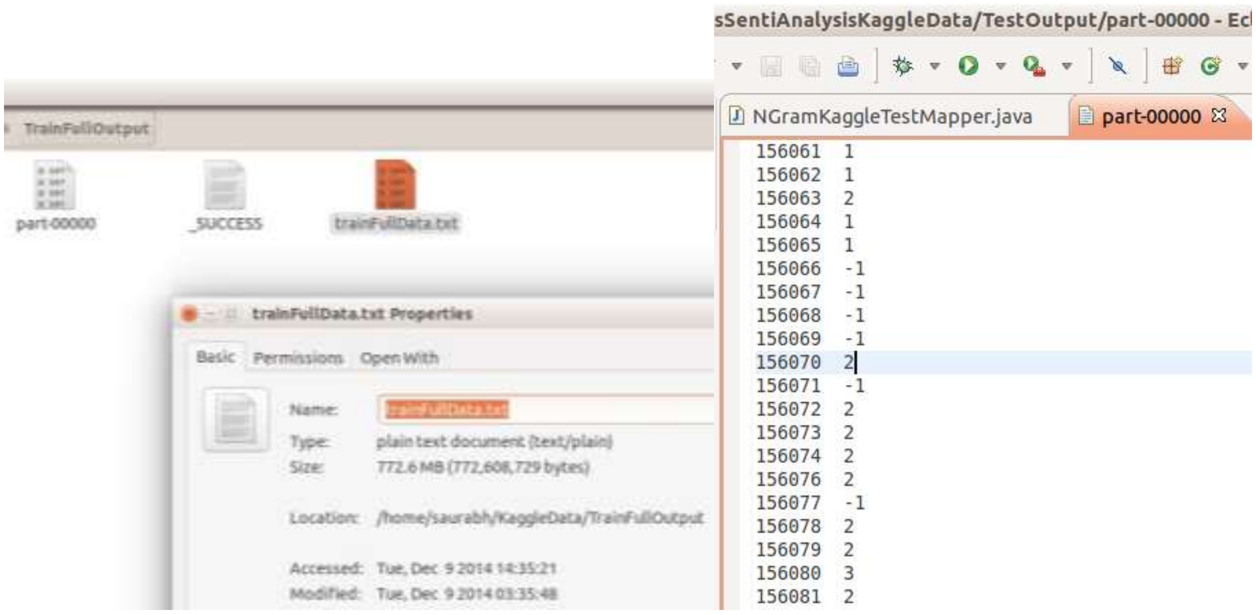
- After this, we are using a max function to find which rating's counter has the max value and then we assign that particular sentiment rating to that Phrase.

- If max = -1, it means no match was found anywhere and we assign -1 as the sentiment rating

- Our Reducer's output is in the form of (Key, Value) -> (PhraseId, Sentiment Rating)

4. Result

Text	Expected	Result
PhraseId-SentenceId-Phrase 156061 8545 An intermittently pleasing but mostly routine effort	Somewhat negative	Somewhat negative


TrainFullOutput
part-00000
_SUCCESS
trainFullData.txt
trainFullData.txt Properties
Basic
Permissions
Open With
sSentiAnalysisKaggleData/TestOutput/part-00000 - Ec
NGramKaggleTestMapper.java
part-00000

Name: trainFullData.txt
Type: plain text document (text/plain)
Size: 772.6 MB (772,608,729 bytes)
Location: /home/saurabh/KaggleData/TrainFullOutput
Accessed: Tue, Dec 9 2014 14:35:21
Modified: Tue, Dec 9 2014 03:35:48

```
156061  1
156062  1
156063  2
156064  1
156065  1
156066  -1
156067  -1
156068  -1
156069  -1
156070  2
156071  -1
156072  2
156073  2
156074  2
156076  2
156077  -1
156078  2
156079  2
156080  3
156081  2
```

5. Inference

- Based on all the preparatory approaches, we were able to run a successful model on the Kaggle Dataset (a subset of the overall dataset) that identifies the polarity of the comment and also identifies how positive and how negative the comment is, based on n-grams (bi and tri-grams).
- While underlying sarcasm identifiers and self-learning logic of adding predicted text with their polarity as a precursor dataset for comparison can increase the accuracy, the accuracy that is seen in the current output is pretty high and the prediction from the analysis is close to the actual sentiment in the review text.

FUTURE SCOPE

- Identifying the contextual sense of a word before assigning it a positive or negative polarity.
    - Looking into assigning each comment to a topic or category based on the comment or the product / object associated with the comment in order to narrow down
- Associating an appropriate sentiment by considering order of words / phrases and turn of phrases.
    - Looking into grouping consecutive words and comparing them to known list of phrases and word groups with appropriate polarity against each; in addition to this, a skewed polarity like 25% positive, 50% negative can be used.
- Based on the language used in the comment and any underlying sarcasm that reverses polarity, a semantic analysis needs to be performed
    - Populate a baseline model that identifies harsh / soft language pointers along with sarcasm hints in word groups.
    - Identify sentiment shifters in the input data using a baseline dataset that can refer to a history of comments and perform a comparison – Neural Networks.
    - Compare input comment against this model if the original prediction is not accurate.

REFERENCES

http://www.cs.uic.edu/~liub/FBS/IEEE-Intell-Sentiment-Analysis.pdf

http://en.wikipedia.org/wiki/Sentiment_analysis

http://en.wikipedia.org/wiki/Text_mining

http://searchbusinessanalytics.techtarget.com/definition/opinion-mining-sentiment-mining

http://sentiwordnet.isti.cnr.it/

http://www.cs.cornell.edu/people/pabo/movie-review-data/

http://ravikiranj.net/drupal/201205/code/machine-learning/

http://alias-i.com/lingpipe/demos/tutorial/sentiment/read-me.html

http://nlp.stanford.edu/software/corenlp.shtml

GROUP – EVALUATION

S.No.	APPROACH	PRATHIMA	SAURABH	VIGNESH
1	Harvard Pos/Neg Dict		✓	✓
2	SentiWord Net		✓	✓
3	LingPipe	✓	✓	
4	Cornell - nGrams	✓	✓	
5	Kaggle - nGrams	✓	✓	✓
6	Report	✓	✓	✓