Project Report
On
Sentiment Analysis

Submitted by

170001007 Ankit

170001008 Ankit Gehlot

Computer Science and Engineering

2nd year

Under the guidance of

Dr. Kapil Ahuja

Department of Computer Science and Engineering

Indian Institute of Technology Indore

Spring 2019

Introduction

If you are considering to buy a product, you might go to a search engine and enter the query “Product review”. Search Engine (Google) may report around a thousand matches. It would be useful to know what fraction of these matches recommend that product is good. This is the motivation to find an algorithm to automatically classify a review as “thumbs up” or “thumbs down”.
The main approaches of sentiment analysis are - Unsupervised - Lexicon Based Analysis, PMI-IR, and Supervised - Naive Bayes approach.

Abbreviations

SO	-	Semantic Orientation
PMI	-	Pointwise Mutual Information
IR	-	Information Retrieval

1. Lexicon Based Approach

Lexicon-based sentiment analysis is typically based on lists of words with a predetermined emotional score of an individual word.

Given a document d, the algorithm detects all words that belong to the emotional dictionary and extracts their polarity and intensity. We modify the initial term scores with addition to the neighborhood of every word that is present in the text and is also in the lexicon dictionary in order to check the whole sentence sentiment, for e.g., "I think this product isn't too good.....," neighborhood is the area of 6 words before and after the emotional term.
The algorithm checks in an area of the word present in the lexicon dictionary and does the following-

- If an intensifier or diminisher word is found, then the original emotional value of the word is modified by the respective modifier score which is either added (in case of an intensifier) or subtracted (in case of a diminisher) to the absolute value of the term. For example, if “good” has an initial value of +3 then “very good” would be modified to +4 and if "bad" has an initial value of -3 then "very bad" would have -4.

- If a negation term is found then the absolute value of the emotional term is decreased by 1 and its polarity is reversed. For example "bad" has -3 then “not bad” would be +2.
- Last, for the capitalization detection, if a word, larger than two characters, that is written fully in capital letters is detected within the neighborhood of an emotional word, including the actual emotional word, then the score of the word is modified to +1 of the original score.

Input : Reviews
Output : Sentiment Score

```
NL : Negation List
IL : Intensifier List
Senti_Score(Reviews) :
    # Tasks
    -- 1. if a word is in NL, then reverse the polarity of next word
    -- 2. if a word is in IL, then modify the polarity of next word
    -- 3. if all letters in the word are in the upper case then add
            fraction to word score, i.e., That Movie was AWESOME.
    -- 4. enhance word score if it contains repeated letters, i.e.,
            So Goooood.
```

Algorithm :

```
ptext = preprocessor(Reviews)    # Removing stop words (a, in..)
tokens = tokenize(ptext)         # Breaking Sentence into words

For word in tokens :
    if word in emoticons then
        score = emoticon score
    else
        if word in Lexicon then
            score = lexicon score
            do task 1 to 4
        else if
            search its synonym and antonym and assign
                                                 score
```

```
                do task 1 to 4
            else
                score = 0                         # not found
        endif
    review_score += score
```

2. Pointwise Mutual Information

It is an unsupervised learning algorithm to classify a review as recommending or not recommending a product. Taking a review as input, this algorithm produces a classification as output.
Steps :

1. Use a Part-of-Speech Tagger to extract phrases in the input text that contain adjectives or adverbs
2. Estimate the Semantic Orientation (SO) of each extracted phrase A phrase has a positive SO when it has good associations (e.g., “Best Mixer”) and a negative SO when it has bad associations (e.g., “terrible mixer”).
3. Assign the given review to a class, recommended or not recommended, based on the average SO of the phrases extracted from the review

PMI-IR uses Pointwise Mutual Information (PMI) and Information Retrieval (IR) to measure the similarity of pairs of words or phrases. PMI is a measure of association between pair of outcomes (i.e., x and y) of some variables (i.e, X and Y). PMI gives a score about the association between the probability of coincidence of these outcomes.

Unlike previously discussed Lexicon-based approach, the polarity of two consecutive words following a certain pattern is considered, the first word is an adjective or adverb and the second word provide context.

Accepted Pattern and Example :

First Word	Second Word
JJ	NN
RB	JJ
JJ	JJ
NN	JJ
RB	VB

Extracted Phrase	Part-of-speech Tag	Semantic Orientation (SO)
Low fees	JJ NN	0.33
Inconveniently located	RB VB	-1.54
True service	JJ NN	0.73

Algorithm :

1. Tokenization of sentence
   assigning Part-of-speech tags to words
2. Using PMI formula

$$PMI(word\,1, word\,2) = \log \frac{P(word\,1 \wedge word\,2)}{P(word\,1)\,P(word\,2)}$$

P(word1 & word2) = Probability of co-occurrence of two words
P(word1) = Probability of occurrence of word1
P(word2) = Probability of occurrence of word2

Semantic Orientation :

$$SO(phrase) = PMI(phrase, \text{excellent}) - PMI(phrase, \text{poor})$$

SO is positive when the phrase is strongly associated with “excellent” and SO is negative when the phrase is strongly associated with “poor”.

$$SO(phrase) = \log \frac{hits(phrase\ AROUND\ \text{excellent})\, hits(poor)}{hits(phrase\ AROUND\ \text{poor})\, hits(excellent)}$$

hits(phrase) = number of hits when “phrase” is searched in custom search engine

3. Calculate average SO (adding SO of all extracted phrases in a review)

3. NAIVES BAYES

Naive Bayes is a supervised classification method which can be utilized for text classification. First, review text goes through pre-processing (i.e., removing stop words like a, that). Next, these pre-processed texts are transformed to 'feature vector'.

In order to perform classification, feature vector is generated based on the vocabulary which is generated from the training dataset (reviews), and the vocabulary doesn't have any duplicate words. The size of the feature vector is the size of this vocabulary.
This vocabulary is defined by selecting features from the dataset (reviews). For text classification, feature vector is the most important structure during the training and classification process.

Generally, order of words in the document is ignored in text classification. Instead, presence or absence of the single word is considered, for instance, whether a word in the vocabulary is included in the document or not. This is called a bag of words. It is like we throw all words in a bag and they could be in any order in the bag.

In Naives Bayes implementation, for a sentence like a review, a feature vector is initialized with all elements equal to zero. Then each word in the vocabulary is checked to see if the word exists in that review. If it exists, then mark the corresponding element in the feature vector to n (frequency of the word).
In that way, every single review can be represented using a feature vector with n's (frequency of words in the review).

Every feature vector can be thought of as an n-dimension point in an n-dimensional coordinate system, where n is the size of the vocabulary. For the training data, we can treat it as several classes of a bunch of n-dimension points.

Then the text classification problem becomes traditional points classification problem, though the dimension might be very large.
For example, if we have a document A, and we have classes set B{+,-}, which contains some classes. Then in order to get which class that document A belongs to, we just need to compute the conditional probability P(A|B), and choose the largest one.

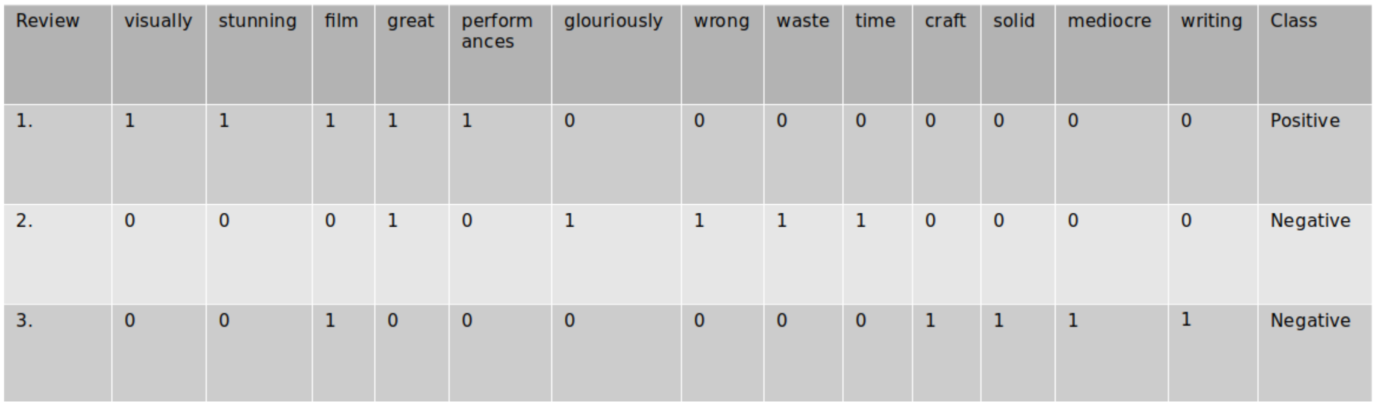
P(A | B) can be computed by Bayes' Theorem:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

P(A | B) is conditional probability: the likelihood of event A occurring given that B is true
P(B | A) is conditional probability: the likelihood of event B occurring given that A is true
P(A) and P(B) are the probabilities of independently A and B being true

Example :

1. A visually stunning film with great performances.
2. How gloriously wrong a film can be, waste of time.
3. You can't craft a solid film with mediocre writing.



Review	visually	stunning	film	great	performances	glouriously	wrong	waste	time	craft	solid	mediocre	writing	Class
1.	1	1	1	1	1	0	0	0	0	0	0	0	0	Positive
2.	0	0	0	1	0	1	1	1	1	0	0	0	0	Negative
3.	0	0	1	0	0	0	0	0	0	1	1	1	1	Negative

P(+) = 0.33

P(-) = 0.67

To find the probability of a word being in a class, Bayes Formula is modified as

$$P(word|class) = \frac{f(word, class) + 1}{f(class) + vocabulary}$$

P(word | class) = probability of occurrence of a word in a given class

f(word, class) = occurrence of a word in a given class

f(class) = occurrence of all words in a given class

Vocabulary = total words in the dataset

4. Cut-Based Classification (Optimization for Naive Bayes Algorithm)

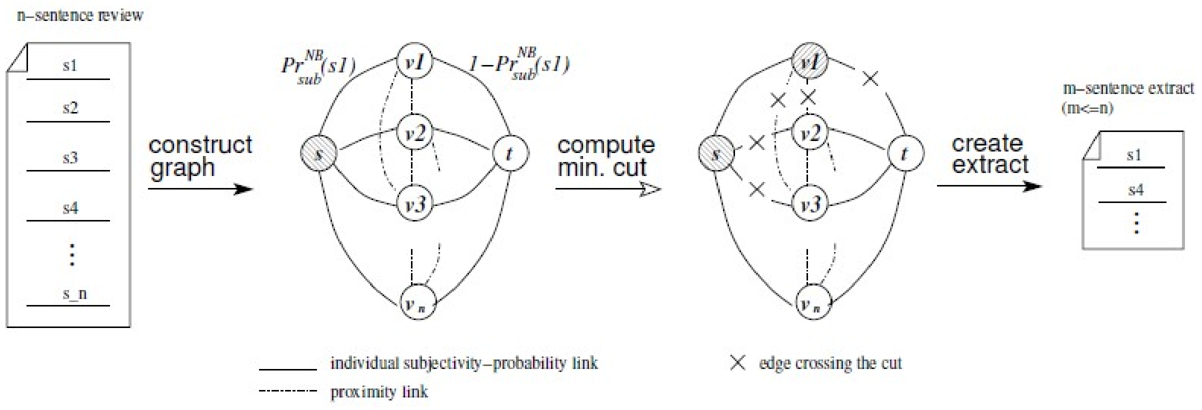
We have n sentences X1, X2, X3,...Xn to divide into two classes Subjective(S) or Objective(O) and we have two types of weights:

- Individual weight ind(j)(Xi): non-negative weights for Xi being in class j (j is either S or O).
- Association weight assoc(Xi)(Xj): non-negative weight for Xi and Xj being in the same class.

We will divide the sentences into subjective and objective class such that the subjective class will have all the necessary sentences that are required for further sentiment score calculations.
The individual weight and association weights are calculated using Naives Bayes.

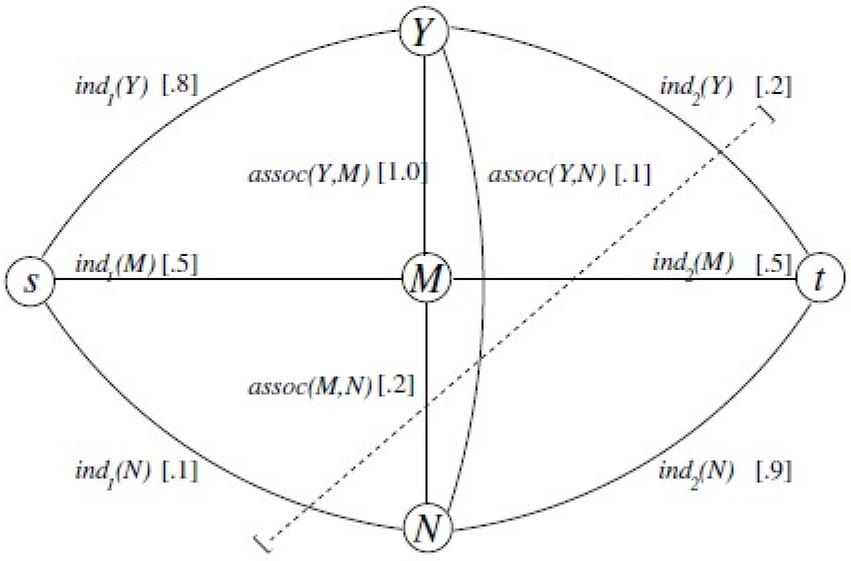
Input : n sentences
C1 = subjective class
C2 = objective class
For each sentence, we have individual score for C1 and C2.
ind{i}(x) , C{i} score of sentence x
assoc(x{i}, x{j}), association score for x{i} and x{j} being in same class

Build an undirected graph G with vertices {x1, x2, ..., s, t}
Add n edges (s, x{i}) each having weight ind{1}(x{i})
Add n edges (t, x{i}) each having weight ind{2}(x{i})
Add nC2 edges (x{i}, x{j}) each with weight assoc(x{i}, x{j})



A cut (S, T) of G is a partition of its nodes into sets S = {s} U S' and T = {t} U T', where s does not belong to S', t does not belong to T'. Its cost cost(S, T) is the sum of the weights of all edges crossing from S to T. A minimum cut of G is one of minimum cost.

$$Partition\,Cost = \sum_{x \in C1} ind2(x) + \sum_{x \in C2} ind1(x) + \sum_{x(i) \in C1, x(j) \in C2} assoc(x(i), x(j))$$



Input : Graph G(V, E)

```
Algorithm :
Create a residual graph R_G with G
While there is a path p from s to t :
flow =  minimum residual_capacity in p
For every R_G edge in p :
                    weight of edge(u,v) -= flow
                    weight of edge(v,u) += flow
Vertices for which there doesn't exist any path from s to reach them,
are added in T and rest (V – T) are added in S.
```

Code :

```
#include <bits/stdc++.h>
using namespace std;

#define V 1000

/* Returns true if there is a path from source 's' to sink 't' in
residual graph. Also fills parent[] to store the path */
int bfs(int rGraph[V][V], int s, int t, int parent[])
{

	bool visited[V];
	memset(visited, 0, sizeof(visited));

	queue <int> q;
	q.push(s);
	visited[s] = true;
	parent[s] = -1;

	while (!q.empty())
	{
		int u = q.front();
		q.pop();

		for (int v=0; v<V; v++)
		{
			if (visited[v]==false && rGraph[u][v] > 0)
			{
				q.push(v);
				parent[v] = u;
				visited[v] = true;
			}
		}
	}

	// If we reached sink in BFS starting from source, then return
	// true, else false
	return (visited[t] == true);
```

```
}

void dfs(int rGraph[V][V], int s, bool visited[])
{
        visited[s] = true;
        for (int i = 0; i < V; i++)
        if (rGraph[s][i] && !visited[i])
                dfs(rGraph, i, visited);
}

// Prints the minimum s-t cut
void minCut(int graph[V][V], int s, int t)
{
        int u, v;
        int rGraph[V][V];   // residual graph
        for (u = 0; u < V; u++)
                for (v = 0; v < V; v++)
                        rGraph[u][v] = graph[u][v];

        int parent[V];

        while (bfs(rGraph, s, t, parent))
        {
                int path_flow = INT_MAX;
                for (v=t; v!=s; v=parent[v])
                {
                        u = parent[v];
                        path_flow = min(path_flow, rGraph[u][v]);
                }

                for (v=t; v != s; v=parent[v])
                {
                        u = parent[v];
                        rGraph[u][v] -= path_flow;
                        rGraph[v][u] += path_flow;
                }
        }
        bool visited[V];
        memset(visited, false, sizeof(visited));
        dfs(rGraph, s, visited);

        // Print all edges that are from a reachable vertex to non-
reachable vertex in the original graph
        for (int i = 0; i < V; i++)
        for (int j = 0; j < V; j++)
                if (visited[i] && !visited[j] && graph[i][j])
                        cout << i << " - " << j << endl;

        return;
}

int main()
{

        int graph[V][V];
        for(int i = 0; i < V; i++){
                for(int j = 0; j < V; j++){
                        cin >> graph[i][j];
                }
        }
        minCut(graph, 0, 5);
        return 0;
}
```

Time Complexity = O({max flow}*{edges})

Conclusion

This paper briefly explains some unsupervised sentiment analysis approaches like lexicon-based approach. In PMI-IR approach, instead of taking single words and their assigned polarity, consecutive words were taken and their polarity was calculated according to hits we get from search engine but we got irrelevant results. And this problem was overcome using the Google custom search engine taking some specific sites in consideration and got slightly improved results.

Then this paper move to supervised Naive Bayes approach and proposes optimization using Min-Cut algorithm by forming an undirected graph taking Naive Bayes training dataset input. Supervised learning has its limitations that training dataset is never complete. Since it requires training and thus can't be readily applied to a wide selection of environments.

References

- Twitter, MySpace, Digg: Unsupervised Sentiment Analysis in Social Media
  GEORGIOS PALTOGLOU and MIKE THELWALL, University of Wolverhampton
  2012

- Thumbs Up or Thumbs Down? Semantic Orientation Applied to Unsupervised Classification of Reviews
  Peter D. Turney
  2002

- Sentiment Detection, Recognition and Aspect Identification
  Salma Al-Asmari, Mohamed Yehia Dahab
  2017

- A Sentimental Education: Sentiment Analysis Using Subjectivity Summarization Based on Minimum Cuts
  Bo Pang and Lillian Lee

- Lexicon-Based Sentiment Analysis in the Social Web
  Fazal Masud Kundi, Aurangzeb Khan, Shakeel Ahmad, Muhammad Zubair Asghar