Sentiment Analysis on Book Reviews using Text Mining

CS4089 Project

Final Evaluation

Aakifah Rahman, Arun John Kuruvilla, Theertha Gireesan
Guided By: Anu Mary Chacko

April 3, 2016

Outline

Introduction

- Sentiment analysis using text mining characterizes the sentiment content of a piece of text and detects its contextual polarity.
- The project has been divided into two phases, out of which Phase I was completed in the previous semester and Phase II completed in this semester.
- We implemented Phase I as a Maven project using OpenNLP library on a Java platform.
- We have implemented Phase II as a Python project using NLTK library (Natural Language Tool Kit), on Spark cluster.

Introduction

- Phase I
  - SentiWordNet:
    - SentiWordNet is a lexical resource for opinion mining. It contains opinion information on terms extracted from the WordNet database - by using a semi supervised learning method.
  - Apache OpenNLP:
    - The Apache OpenNLP library is a machine learning based toolkit for the processing of natural language text.
  - Apache Maven:
    - Apache Maven is a software project management and comprehension tool.

Introduction

- Phase II
  - Machine-learning based method:
    - Algorithm is given a set of data to infer information about the properties of the data and this allows it to make predictions about unseen data.
  - Apache Spark:
    - An open source cluster computing framework, well-suited for running machine-learning algorithms. [1]
  - Natural Language Tool Kit:
    - A python platform to work with human language data which provides a suite of text processing libraries. [2]
  - Naive Bayes Method:
    - A multiclass classification algorithm with the assumption of independence between every pair of features.
  - Support Vector Machine Method:
    - Support Vector Machines (SVM) are supervised learning models with associated learning algorithms that analyze data used for classification and regression analysis.

Problem Statement

- To develop an interactive dashboard to visualize results of Sentiment Analysis of book reviews using text mining.
  To evaluate the various approaches of sentiment analysis using performance metrics.

Literature Survey

- Bing Liu [3] presents the basics of sentiment analysis and document sentiment classification. The fundamentals of sentiment analysis were comprehended from this textbook.
- Basics of Apache Spark set-up, installation and running was investigated from the Spark official website, so as to use it as a platform for text mining [1].
- The usage of OpenNLP toolkit for text mining was surveyed and studied from the OpenNLP documentation [4]. The various text mining tasks supported by OpenNLP, such as Sentence Detector, Tokenizer, POS Tagger etc. was also studied from the manual.
- Using SentiWordNet for sentiment retrieval of English words was understood from a Slideshare tutorial on Sentiment analysis [5]. Vocabularies used in Sentiment analysis as well as the Summation-on-Reviews method for Sentiment score calculation was studied from this presentation.

Literature Survey

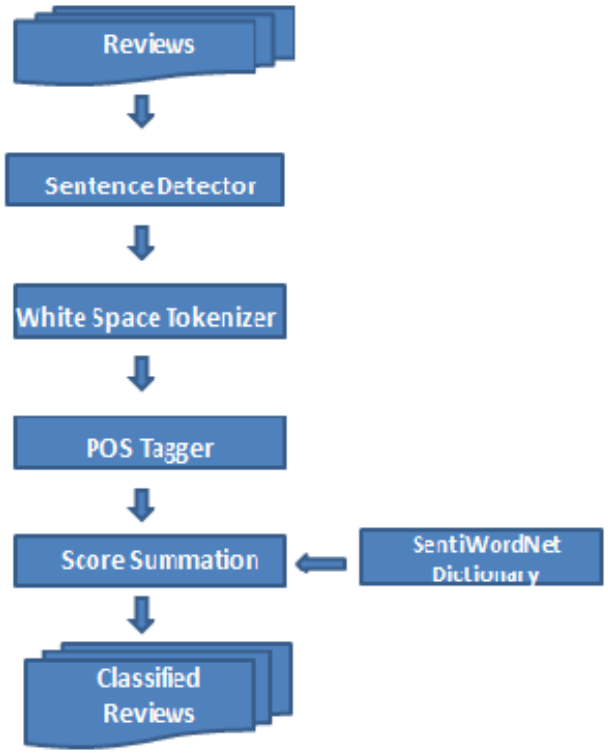
- Review classification using SentiWordNet Lexicon was also understood from a paper by Hamouda and Rohaim [6]. The paper proved to be a significant contribution towards the understanding of the use of SentiWordNet as a dictionary for opinion mining.
- The fundamental concepts of Naive Bayes classifier was learnt from the NLTK website chapters on text classification [7]. The chapters covered classification tasks in general, and supervised learning in particular. Implementations of Naive Bayes for document sentiment classification was understood from here.
- Feature extraction and Transformation using Spark was understood from The Spark documentation [8]. The documentation presented the various feature extractors available as Spark modules.
- Machine learning and feature selection were further comprehended from the Clips technical report by Sarah Schrauwen [9].

Design

- Phase I
  - Environment Setup and Installations
    - Build manager Maven was set-up to run OpenNLP as a Maven project. JDK 8 was also installed to run OpenNLP in Java. Eclipse Mars was set-up as a user-friendly interface to run the Maven Java project.
  - Database Creation
    - We created the database manually from Amazon and Goodreads book reviews. The database was organized as a set of 20 folders with each folder named as the respective book, containing positive and negative reviews as plain text files. For each book, there were 4-10 reviews, each of which contained 10-400 words.

Design

- Phase I
  - System and Algorithm



input : Bookname B
output: The book is well worth a read or not

dictionary *D*;
bookScore = 0;
foreach review *R* **in** *B*
    reviewScore = 0;
    foreach sentence *S* **in** *R*
        sentenceScore = 0;
        foreach word *W* **in** *S* **do**
            pos = getPOS(*W*, *S*);
            score = getScore(*W*, *pos*, *D*);
            sentenceScore = *sentenceScore* + *score*;
        end
        reviewScore = *reviewScore* + *sentenceScore*;
    end
    bookScore = *bookScore* + *reviewScore*;
end
if (*bookScore* >0)
    print "The book is well worth a read";
else
    print "The book is not a good read";

Design

- Phase II
  - Environment Setup and Installations
    - Spark was setup as the platform for Phase II. NLTK library was imported to run the NLP tasks.
  - Database Creation
    - We have used the Amazon Product Dataset, obtained from a research project in University of California, San Diego [10]. From the dataset we collected the first 50,000 reviews of 50 MB. Each book had an average of 13 reviews. Also each review averaged to 400 words, ranging from 150 words to 650 words.

Design

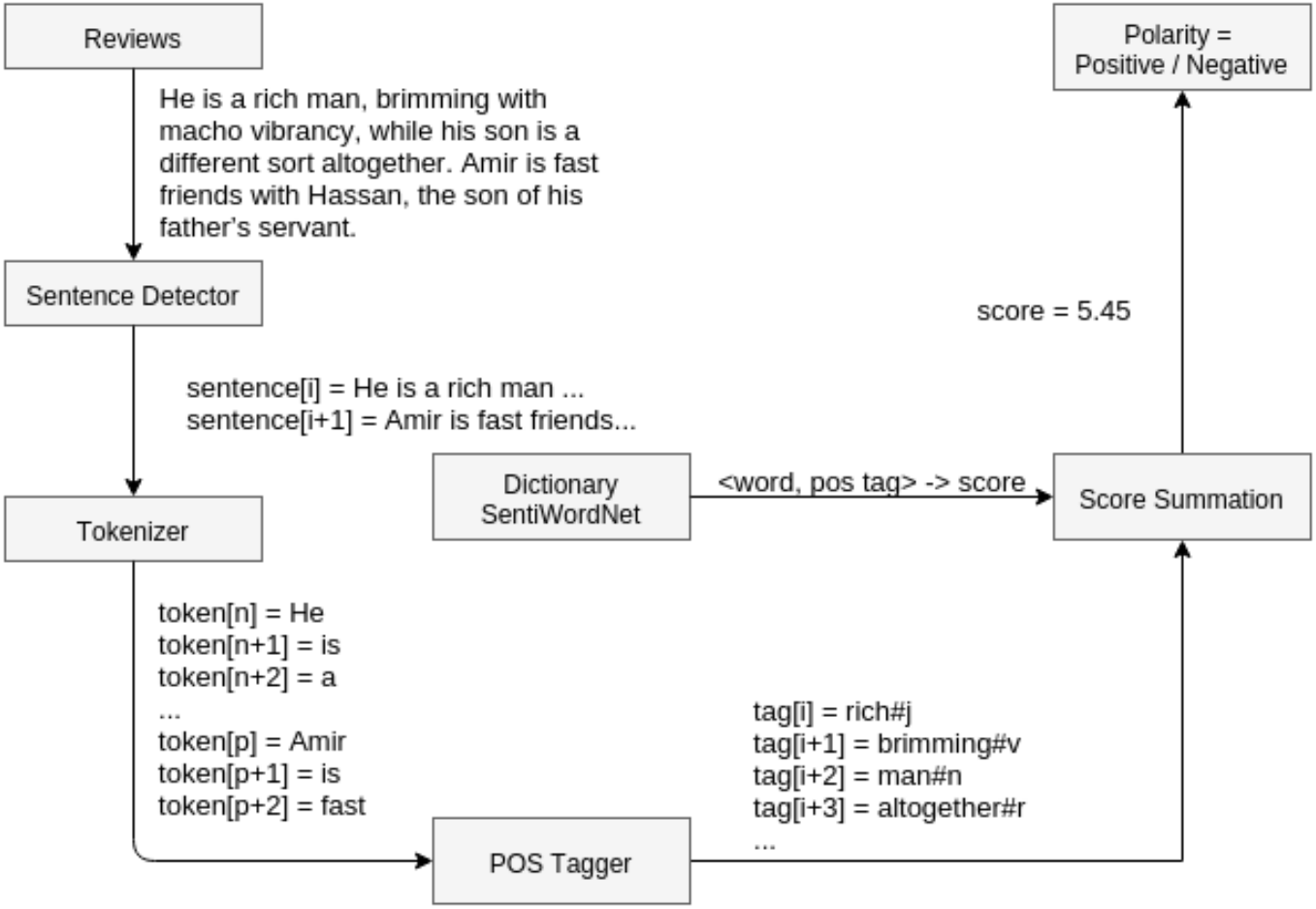
- Phase II
  - Database Creation


```
{
  "reviewerID": "A2SUAM1J3GNN3B",
  "asin": "0000013714",
  "reviewerName": "J. McDonald",
  "helpful": [
    2,
    3
  ],
  "reviewText": "I bought this for my husband
    who plays the piano.  He is having a
    wonderful time playing these old hymns.",
  "overall": 5,
  "summary": "Heavenly Highway Hymns",
  "unixReviewTime": 1252800000,
  "reviewTime": "09 13, 2009"
}
```

Implementation

- Phase I:
  - Implemented the project as a Maven project using JDK 8, supported by the OpenNLP toolkit.
  - The reviews related to a book are loaded into the system. Each review is first run through a Sentence Detector, to split the entire review paragraph into sentences.
  - Each sentence is then run through a Tokenizer. Each token is then tagged with a POS using the POS Tagger. Each token-POS pair is then assigned a score obtained from the dictionary, populated from SentiWordNet.
  - The dictionary is created by taking the weighted average of the Positive and Negative scores of each word-POS pair in the SentiWordNet. The Score Summation method is then used, to sum final scores of words and sentences.
  - The Sum-on-Review method is then implemented, to obtain a final book score. The positivity or negativity of the final score decides whether the book is a good read or not.

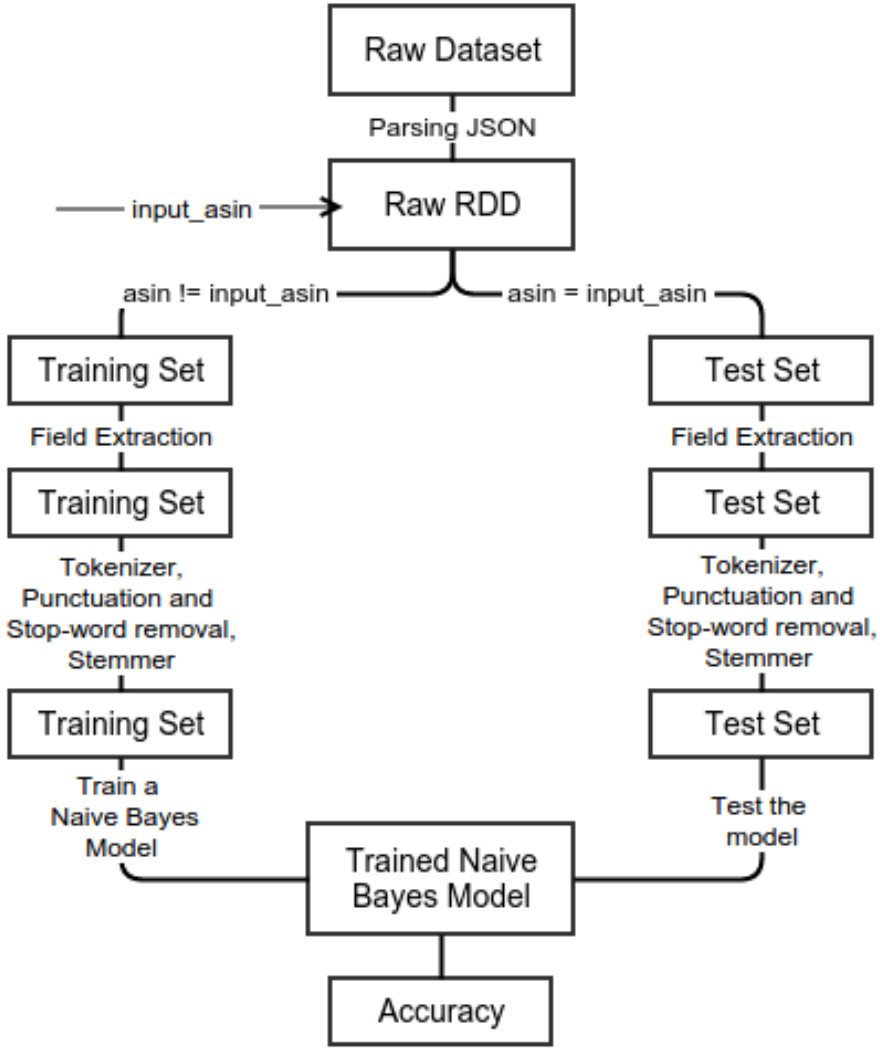
Implementation



Implementation

- Phase II Part A: Naive Bayes Method
  - The entire RDD is split into two RDDS, one for training and one for testing based on 'asin' value entered by the user.
  - The relevant fields required for sentiment classification are then extracted from the sets. The overall fields of both sets were then mapped as negative ($\{1,2,3\} \rightarrow 0$) or positive ($\{4,5\} \rightarrow 1$).
  - Each review text was tokenized, by first converting the text into tokens, changing all the letters to lowercase, removing punctuation and stopwords, and then stemming using Porter algorithm.
  - A BoW(Bag of Words) feature vector, using term frequency of each word, was created for both datasets.
  - A Naive Bayes model was trained using training dataset and was then used to predict the sentiment label of the test dataset.
  - The predicted labels were then compared with the original labels and the accuracy of the model of the reviews of that particular book was calculated.

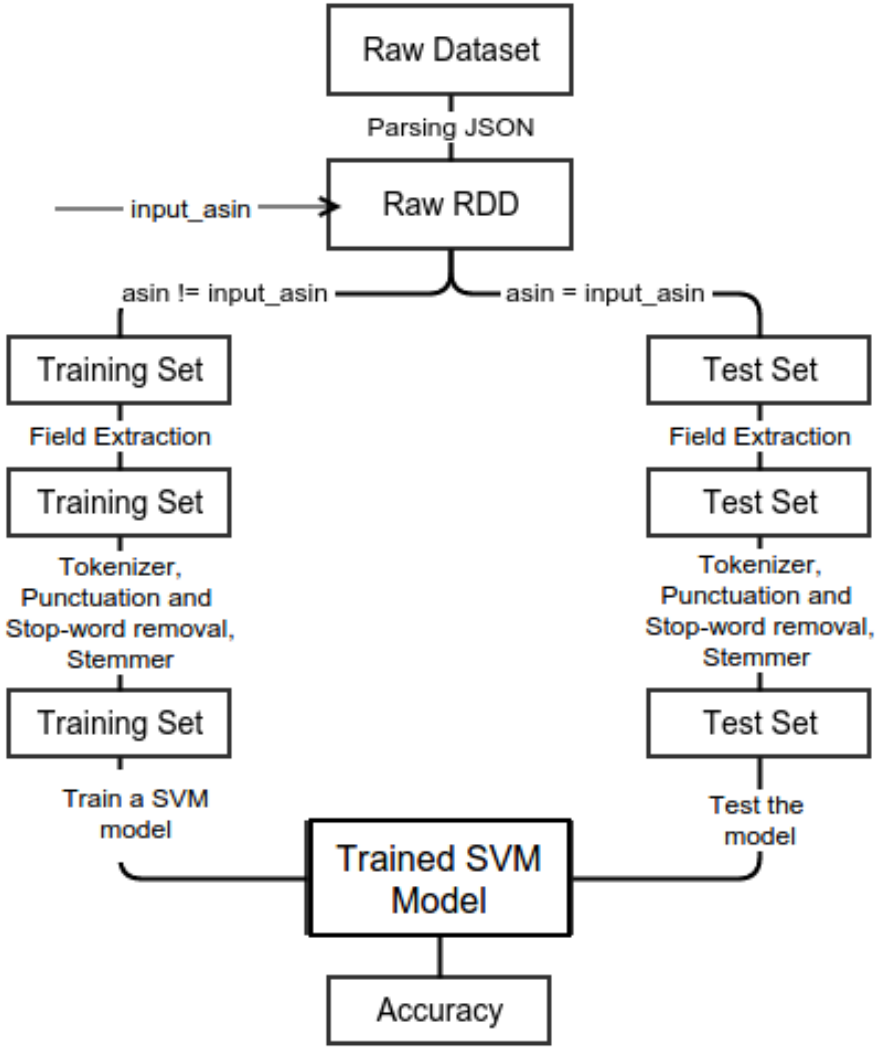
Implementation



Implementation

- Phase II Part B: Support Vector Machine Method
  - Data preparation and pre-processing steps were carried out as in Part A.
  - The model was then trained using Support Vector Machine, to predict labels of the testing set.

Implementation

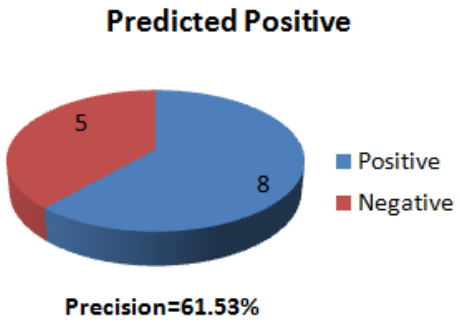


Results

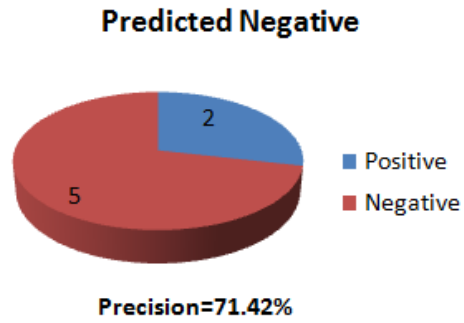
- Phase I:
  - In the Lexicon based analysis, the total score for each review was calculated from the score of positive and negative words, and then the sentiment polarity was determined by assigning the review to the respective class using the 'Sum On Review' method. This method yielded an overall accuracy of 65%, with precision and recall of the two classes, as detailed in the figures.

CLASS	POSITIVE	NEGATIVE
Predicted Positive	8	5
Predicted Negative	2	5
Total	10	10
Class Recall	80%	50%
Class Precision	61.53%	71.42%

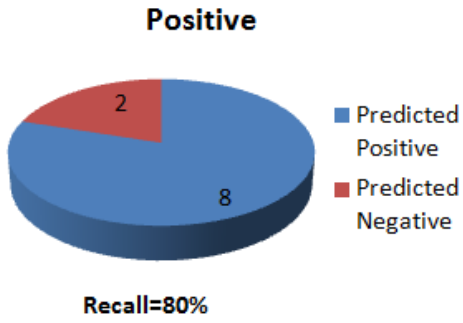
Results



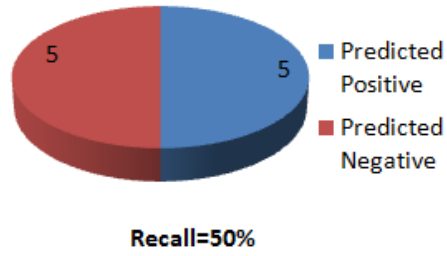
Precision=61.53%



Precision=71.42%



Recall=80%

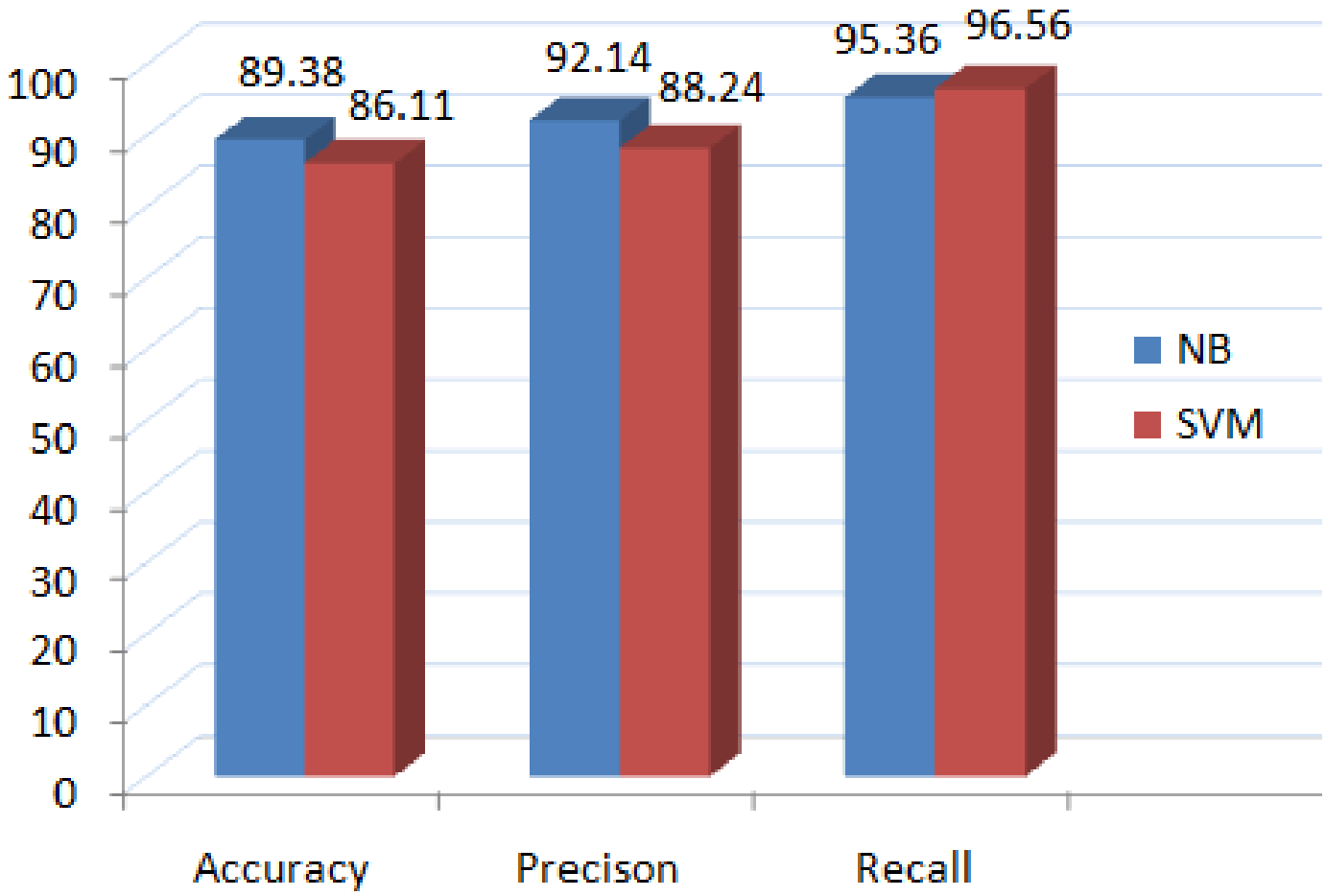


Recall=50%

Results

- Phase II:
  - The model accuracy varies when tested on the testing set according to the input book ‘asin‘ value. Thus, we selected 11 books with large number of reviews (ranging from 20 to 6400) and averaged their accuracy, precision and recall values, to obtain the final statistics for both the models.

Results


100
90
80
70
60
50
40
30
20
10
0
89.38
86.11
92.14
88.24
95.36
96.56
NB
SVM
Accuracy
Precison
Recall

Conclusions

- Phase I implemented the method of lexicon based classification, which produced an accuracy of 65%.
- The limitations faced in this phase included the presence of ambiguous words in reviews and disregard of negations. The lack of differentiation between negative words in the context of the book content, and negative words in the review, was another major challenge faced.
- Phase I of the project proved that a lexicon based analysis can provide a reasonable accuracy. But for large reviews, or books with large number of reviews, this method does not prove to be a feasible solution.

Conclusions

- In Phase II, we have implemented a machine learning based analysis technique using training data to learn Naive Bayes and Support Vector Machine models. Machine learning based method proves to guarantee higher accuracies, as the model is trained on real-world data. The high recall statistic depicts that an appreciable percentage of reviews are captured correctly - around 92.14% for Naive Bayes, and 88.24% for SVM.
- We have implemented a Unigram Naive Bayes Model. This can be further extended to Bigram and Trigram models, which can further reduce ambiguity and negations.
- Implementing Phase II on Spark posed certain constraints in terms of algorithms available for training Multi class classification models. Another platform could possibly allow using other algorithms, which could be more suitable for sentiment analysis on book reviews.

References I

[1] Apache Spark - lightning fast cluster computing. spark.apache.org

[2] NLTK - nltk.org/

[3] Bing Liu. *Sentiment Analysis and Opinion Mining*, Morgan and Claypool Publishers, 2012.

[4] OpenNLP Documentation. opennlp.apache.org/documentation/1.6.0/manual/opennlp.html

[5] Tutorial of Sentiment Analysis. slideshare.net/faigg/tutotial-of-sentiment-analysis

[6] Alaa Hamouda Mohamed Rohaim. *Reviews Classification Using SentiWordNet Lexicon*.

[7] Naive Bayes Classifier - http://www.nltk.org/book/ch06.html/

References II

[8] Feature Extraction and Transformation - spark.apache.org/docs/latest/mllib-feature-extraction.html

[9] Machine Learning and Feature selection - clips.ua.ac.be/sites/default/files/ctrs-001-small.pdf

[10] Inferring networks of substitutable and complementary products - J. McAuley, R. Pandey, J. Leskovec - *Knowledge Discovery and Data Mining*, 2015 - jmcauley.ucsd.edu/data/amazon/links.html