**Keyword Extraction and Sentiment Analysis on Amazon Review Dataset**

**Project Report**

**CMPE 256: Large-Scale Analytics**

**Submitted to:**

**Dr. Magdalini Eirinaki**

****

**Submitted by:**

**Team Name - DataAnalyzers**

|  |  |
| --- | --- |
| Student Name | SJSU ID |
| Abhishek Yadav | 012468417 |
| Suraj Khurana | 011464427 |
| Sreedeep Katragadda | 011545222 |

**Github -** [**https://github.com/khurana3773/ExtractKeywordsFromAmazonReviews**](https://github.com/khurana3773/ExtractKeywordsFromAmazonReviews)

**INDEX**

[Chapter 1 - Introduction](#_keqpasybz984)

[Chapter 2 - System Design & Implementation details](#_93s9rqifktvs)

[2.1 Algorithm(s) considered/selected](#_57338tsahsoy)

[2.2 Technologies & Tools used](#_5g3my98724q4)

[2.3 System design/architecture/data flow/workflows as applicable (you may use diagrams with some supportive text)](#_huq5nfwzfdcn)

[Chapter 3 - Experiments / Proof of concept evaluation](#_x2yc0z8iok4i)

[3.1 Dataset used](#_9j9zq3xptyua)

[3.2 Data preprocessing decisions](#_ws1mo1vq3xf5)

[3.3 Methodology followed](#_73aznd5tynni)

[Chapter 4 - Discussion and Conclusions](#_pf572n4y63tp)

[4.1 Decisions made](#_x4qvqzl9c3au)

[4.2 Difficulties faced](#_ly3y4n3imank)

[4.3 Things that worked](#_9d9resvmji7g)

[4.4 Things that didn’t work well](#_z4rboa8gyve8)

[4.5 Conclusion](#_q6r9j8h5ee0p)

[Chapter 5 - Project Plan / Task Distribution](#_uzxdrbb7f236)

[5.1 Who was assigned to what task](#_gt1or9kb5eqi)

[5.2 Who ended up doing what task](#_z1nravo67iyo)

[References](#_deivf3b1bacq)

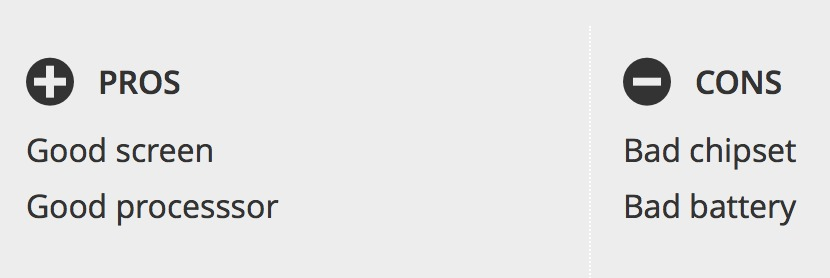
# 

# 

# 

# Chapter 1 - Introduction

Motivation - Let's say you are an Amazon shopper, who reads through reviews before buying a phone(any product). What you are looking for are pros and cons of the product. ie. what features of the product are good and what are bad. With this Project, we are trying to concisely sum up all the reviews and show information what matters to You(A Potential Buyer). Below Image is a sample of our Output



Objective - Technically speaking, the project is based on a combination of 2 parts. 1)Feature extraction and 2) sentiment analysis of features.

In the First part, our objective is to extract components or ‘features’ of the product. For example, for a mobile phone, key components are its Battery, Screen, Processor, etc.

In the second part, our objective is to find the sentiment associated with each component we extracted in part 1.

Note: Following words will be used interchangeably.

Part 1 = Keyword Extraction

Part 2 = Sentiment Analysis of Key

# Chapter 2 - System Design & Implementation details

## 2.1 Algorithm(s) considered/selected

In part 1, we are extracting key components from existing reviews of the product.. For example, for a mobile phone, key components are its Battery, Screen, Processor, etc. Based on the **Assumption**, reviewers usually write about these key components in reviews, our key components are high Frequency words. So we have written our custom(or one can say highly used brute force) Algorithm to find the high frequency words or Key Components.

Steps followed:

* Take all reviews for a product and consider it a big input text..
* Pass this big input to the tool : rake-nltk. Rake-NLTK tokenize text into individual words, and return an array of ‘word, and its frequency’ in the input.
* From this array, we then remove stop words. Stop words are words we use very frequently in sentences, and do not convey about Subject Or Object of the sentence. Examples : a, the , they, place , take , Though, although,etc (we could have let rake remove stop words too, but decided to stick with what we did)
* From the remaining array, we retained only those words which were Nouns, and removed other words from the array. The thought behind this is that, key components like Battery, Screen,etc are nouns and not adjectives or verbs or adverbs. NLTK provides a big set of words which are nouns, which was used to filter out words.
* Then we Lemmatized/Stemmed each words and aggregated ie Stemming/Lemmatization is bring different tenses and forms of a word to a base form. Example, Lemmatized form of Batteries is Battery. The reviewer could have used any form of the word in the review. By lemmatizing and aggregating, we get a higher(and actual) frequency of the Battery.
* From the remaining words,we consider only the top 20 highest frequency words.

Note : We could see battery and screen in these top 20 words, but there was also some noise.

Hence, for part 2, we hard coded features in the Program. Currently version on github, has Screen, Battery and Sound as key components for a phone.

For Part 2, we used these Extracted (or hard coded) key components from Part 1, and try to find a general sentiment for this component from the reviews.We then classify this sentiment as good or bad and display component as (+) good Component or (-) bad Component. The algorithm we used for this is as follows.

* From all reviews of the Product, extract all sentences with a key components present. Ie extract all sentences containing any of the words battery , screen or sound.
* Pass each sentence to a classifier, to classify it a good or a bad sentiment.
* Count a score of good and bad sentiments for a component. If score of good> score of bad, show result as a Good Component. Ie after running analysis, the program would show (+) Good Battery And Vice Versa. It would show such output for each component.

As mentioned in Algorithm of Part 2, we use classifier to classify sentiment as good or bad. We have used 2 classifiers in our code: 1) VADER Sentiment Analyser. 2) Naive Bayes Classifier.

VADER is a pretrained where as we have to train Naive Bayes Classifier.

Following are the steps we used to train Naive Bayes Classifier:

* Obtained Labelled data of good and bad sentiments from nltk.corpus.pros\_cons
* Tokenize words, Lemmatize words and create an document of words and its frequencies, and consider only top 5000 highest frequency words important.
* Then for each original Labelled sentence, extract words which are present in the top 5000 word, and save these extracted words with label, to a SentimentSet.
* Use this SentimentSet to train Naive Bayes Classifier

Note: We tried using other listed classifiers too, but avoided them due to high computation time. They had similar training steps as described above in Naive Bayes Classifier.

1. MultinomialNB
2. BernoulliNB
3. LogisticRegression
4. SGDClassifier
5. LinearSVC
6. NuSVC

Note - We were only able to train NaiveBayesClassifier with huge data and run accuracy scory. We attempted N-Fold validation and accuracy, F-Score for the algorithm.

## 2.2 Technologies & Tools used

1. Rake-NLTK library
2. VADER classifier from NLKT
3. Naive Bayes from NLTK
4. Libraries of nouns and verbs, word and sentence tokenizer from NLTK.
5. And Lastly, Anaconda Jupyter Notebook

## 2.3 System design/architecture/data flow/workflows as applicable (you may use diagrams with some supportive text)

For Part 1,



For Training Naive Bayes Classifier and Part 2,

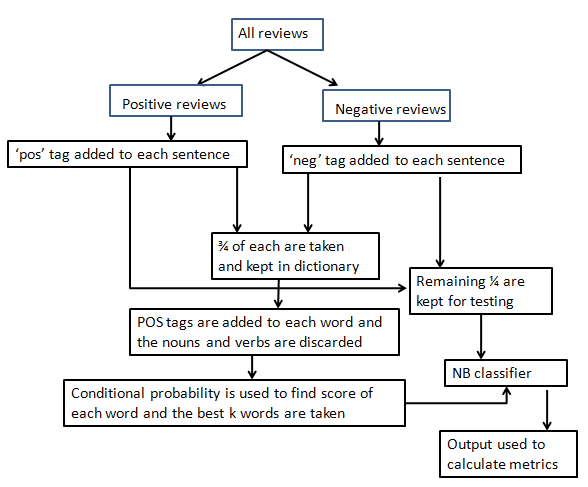
****

Fig 1 - Naïve Bayes‘ flowchart [1]

# Chapter 3 - Experiments / Proof of concept evaluation

## 3.1 Dataset used

The dataset is obtained from DAIS

http://sifaka.cs.uiuc.edu/~wang296/Data/LARA/Amazon/AmazonReviews.zip

Its size is 960,530,364 bytes (1.01 GB on disk) for 20,828 items and it contains -

Amazon product reviews under six categories: camera, mobile phone, TV, laptop, tablet and video surveillance system. Each product contains the attributes of: product ID (unique), name, product features, listing price and image URL.

Each review contains the attributes of: review ID (unique), author, title, content, overall rating, date.

Note: This was going to the original dataset to be used train and evaluate algorithm. But Labelling data was a big problem. Hence we used labelled dataset from NLKT.corpora.pros\_cons was used for training model and evaluation metrics. The above dataset from amazon, was used only to extract components of product.

## 3.2 Data preprocessing decisions

The project was built without doing any preprocessing on the dataset

1. No way of interpolating missing values
2. No way of normalizing reviews given by a user
3. Too time consuming for labelling.

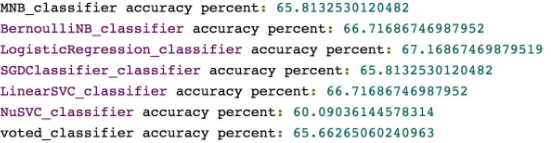
As mentioned above, a labelled dataset from NLKT.corpora.pros\_cons was used for training model and evaluation metrics.

However, we did perform some data preprocessing steps like

1. Tokenizing words
2. Remove stop words from reviews
3. Lemmatization of words
4. Creating a document of word,and its frequency

## 3.3 Methodology followed

* We used Machine learned classifiers, and the following evaluations metrics are going to talk about them only.
* Our Dataset consisted of approx 44000 good and bad labelled sentences.
* We split ¾ th part of dataset into Training set. Approx 34500 and remaining ¼ th part in testing set for Single Fold
* We only Thoroughly trained Naive Bayes Classifier(NBC). For NBC, we used single fold and n-fold approach. For N-fold, we used n=5.
* On a small training sample the comparison between 6 algorithms looked like follow



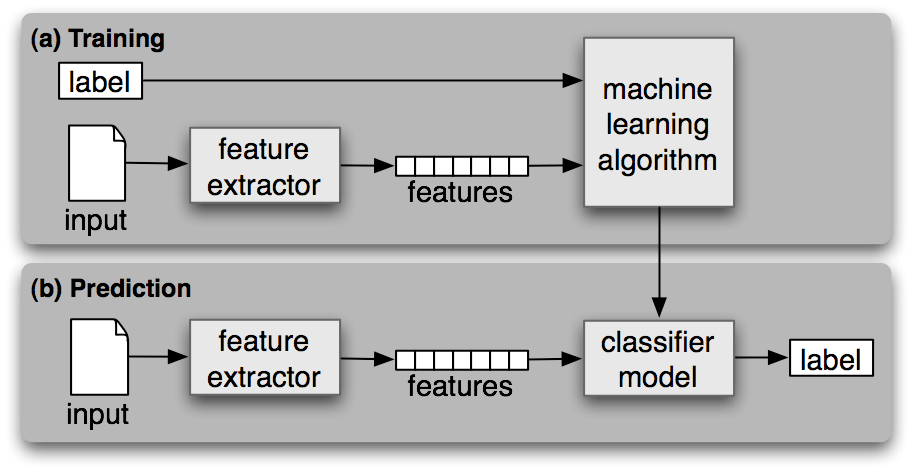


Fig 2 - Diagram showing the use of Training Data and Testing Data

# Chapter 4 - Discussion and Conclusions

## 4.1 Decisions made

Major decisions involved in the making of this project are as follows -

1. Using a large preprocessed dataset of positive.:
2. Because of time constraint, we avoided the preprocessing of the dataset. NLTK corpus has an inbuilt pros and cons list
3. As the extracted components from Part 1 had noise, we hard coded components for Part 2.
4. We trained naive bayes classifier only once , and reused the classifier as training was taking too long.

## 4.2 Difficulties faced

The major difficulty for the project was in dealing the large dataset with multiple features. Json does not contain info about most helpful reviews. Training of models took a long time, and was time consuming to fine tune different parameters.

## 4.3 Things that worked

Binary classification of a feature for a device as good or bad. Example: correctly classifying features like battery, camera, performance as Good Battery, Good Camera, Bad Performance from Phone reviews.

## 4.4 Things that didn’t work well

Extracting desired features for a device. Example: correctly extracting features like battery, camera, performance from Phone reviews. Many of unnecessary words were extracted with features.

## 4.5 Conclusion

Successfully classified features as Good or Bad, given a hard coded Feature List, and an extensive training model from pros cons from NLTK corpus. Additionally, a ‘complementary-recommendation’ system was planned, where item recommendations for a phone with Bad Battery will be phones with Good Battery.

# Chapter 5 - Project Plan / Task Distribution

## 5.1 Who was assigned to what task

Everyone were working on the same task to achieve the objective of the project.

## 5.2 Who ended up doing what task

1. Suraj Khurana -
   1. Writing algorithm for extracting keywords and preprocessing data.
   2. Writing code to use Classifier and dataset.
   3. Planning and implementing most of the project.
   4. Meaningful contributions to presentation and report. >50%
2. Abhishek Yadav -
   1. Building classifier and building training dataset
   2. Made Project Report and Presentation Slides
   3. Reading relevant materials for keyword extractions and semantic analysis through internet
3. Sreedeep Katragadda -
   1. Building classifier models, classification of keywords using classifiers
   2. Contributed to the report and ppt

# References

[1] https://www.researchgate.net/file.PostFileLoader.html?id=58a8166b217e2038266d4654&assetKey=AS%3A463048591843328%401487410795205

[2] <http://www.nltk.org/book/ch06.html>

[3] http://blog.chapagain.com.np/machine-learning-sentiment-analysis-text-classification-using-python-nltk/