CHAPTER 1

INTRODUCTION

1.1 Machine Learning

Machine learning is a method of data analysis that automates analytical model building. Using algorithms that iteratively learn from data, machine learning allows computers to find hidden insights without being explicitly programmed where to look.

Machine learning is a subfield of computer science that evolved from the study of pattern recognition and computational learning theory in artificial intelligence. Machine learning is defined as a "Field of study that gives computers the ability to learn without being explicitly programmed". Machine learning explores the study and construction of algorithms that can learn from and make predictions on data. Such algorithms operate by building a model from example inputs in order to make data-driven predictions or decisions, rather than following strictly static program instructions.

Machine learning is closely related to (and often overlaps with) computational statistics; a discipline which also focuses in prediction-making through the use of computers. It has strong ties to mathematical optimization, which delivers methods, theory and application domains to the field. Machine learning is employed in a range of computing tasks where designing and programming explicit algorithms is unfeasible.

Example applications include spam filtering, optical character recognition (OCR), search engines and computer vision. Within the field of data analytics, machine learning is a method used to devise complex models and algorithms that lend themselves to prediction - in commercial use this is known as predictive analytics. These analytical models allow researchers, data scientists, engineers, and analysts to "produce reliable, repeatable decisions and results" and uncover "hidden insights" through learning from historical relationships and trends in the data.

1.2 Evolution of Machine Learning

Because of new computing technologies, machine learning today is not like machine learning of the past. It was born from pattern recognition and the theory that computers can learn without being programmed to perform specific tasks; researchers interested in artificial intelligence wanted to see if computers could learn from data. The iterative aspect of machine learning is important because as models are exposed to new data, they are able to independently adapt. They learn from previous computations to produce reliable, repeatable decisions and results. It's a science that's not new – but one that's gaining fresh momentum.

While many machine learning algorithms have been around for a long time, the ability to automatically apply complex mathematical calculations to big data – over and over, faster and faster – is a recent development. Here are a few widely publicized examples of machine learning applications you may be familiar with:

- The heavily hyped, self-driving Google car? The essence of machine learning.
- Online recommendation offers such as those from Amazon and Netflix? Machine learning applications for everyday life.
- Knowing what customers are saying about you on Twitter? Machine learning combined with linguistic rule creation.
- Fraud detection? One of the more obvious, important uses in our world today.

1.3 How it works?

To get the most value from machine learning, you have to know how to pair the best algorithms with the right tools and processes. Statistical Analysis System combines rich, sophisticated heritage in statistics and data mining with new architectural advances to ensure your models run as fast as possible – even in huge enterprise environments.

Algorithms: SAS graphical user interfaces help you build machine learning models and implement an iterative machine learning process. You don't have to be an advanced statistician. Our comprehensive selection of machine learning algorithms can help you quickly get value from your big data and are included in many SAS products.

SAS machine learning algorithms include:

- Neural networks
- Decision trees
- Random forests
- Associations and sequence discovery
- Gradient boosting and bagging
- Support vector machines
- Nearest-neighbor mapping
- k-means clustering
- Local search optimization techniques (e.g., genetic algorithms)
- Expectation maximization
- Multivariate adaptive regression splines
- Bayesian networks
- Kernel density estimation
- Principal component analysis
- Singular value decomposition
- Gaussian mixture models
- Sequential covering rule building

Tools and Processes: As we know by now, it’s not just the algorithms. Ultimately, the secret to getting the most value from your big data lies in pairing the best algorithms for the task at hand with:

- Comprehensive data quality and management.
- GUIs for building models and process flows.
- Interactive data exploration and visualization of model results.
- Comparisons of different machine learning models to quickly identify the best one.
- Automated ensemble model evaluation to identify the best performer.
- Easy deployment so you can get repeatable, reliable results quickly.

1.4 Classification of Machine learning

1.4.1 Supervised Learning

This algorithm consists of a target / outcome variable (or dependent variable) which is to be predicted from a given set of predictors (independent variables). Using these set of variables, we generate a function that map inputs to desired outputs. The training process continues until the model achieves a desired level of accuracy on the training data. Examples of Supervised Learning: Regression, Decision Tree, Random Forest, KNN, Logistic Regression etc.

1.4.2 Unsupervised Learning

In this algorithm, we do not have any target or outcome variable to predict / estimate. It is used for clustering population in different groups, which is widely used for segmenting customers in different groups for specific intervention. Examples of Unsupervised Learning: Apriori algorithm, K-means.

1.4.3 Reinforcement Learning

Using this algorithm, the machine is trained to make specific decisions. It works this way: the machine is exposed to an environment where it trains itself continually using trial and error. This machine learns from past experience and tries to capture the best possible knowledge to make accurate business decisions. Example of Reinforcement Learning: Markov Decision Process.

1.5 List of Common Machine Learning Algorithms

Here is the list of commonly used machine learning algorithms. These algorithms can be applied to almost any data problem:

1. Linear Regression
2. Logistic Regression

3. Decision Tree
4. SVM
5. Naive Bayes
6. KNN
7. K-Means
8. Random Forest
9. Dimensionality Reduction Algorithms
10. Gradient Boost & Adaboost

1.6 Importance of Machine learning

Resurging interest in machine learning is due to the same factors that have made data mining and Bayesian analysis more popular than ever. Things like growing volumes and varieties of available data, computational processing that is cheaper and more powerful, and affordable data storage.

All of these things mean it's possible to quickly and automatically produce models that can analyze bigger, more complex data and deliver faster, more accurate results – even on a very large scale. And by building precise models, an organization has a better chance of identifying profitable opportunities – or avoiding unknown risks.

What's required to create good machine learning systems?

- Data preparation capabilities.
- Algorithms – basic and advanced.
- Automation and iterative processes.
- Scalability.
- Ensemble modeling

1.7 Applications of Machine Learning

- **Opportunities and challenges for machine learning in business**

  This O'Reilly white paper provides a practical guide to implementing machine-learning applications in your organization.
- **Machine learning + wearable medical devices**

  How can these technologies work together to help patients? Find out machine learning and wearable devices can diagnose patients faster and provide care at a lower cost.
- **Principles and practices of machine learning**

  In this on-demand webinar, SAS data scientist Patrick Hall discusses the basics of machine learning as well as factors to consider when putting it into practice.
- **Applying machine learning to IoT**

  Machine learning can be used to achieve higher levels of efficiency, particularly when applied to the Internet of Things. This Insights article explores the topic.

1.8 Problems solved by Machine Learning

1.8.1 Manual data entry

Inaccuracy and duplication of data are major business problems for an organization wanting to automate its processes. Machines learning (ML) algorithms and predictive modeling algorithms can significantly improve the situation. ML programs use the discovered data to improve the process as more calculations are made. Thus machines can learn to perform time-intensive documentation and data entry tasks. Also, knowledge workers can now spend more time on higher-value problem-solving tasks. Aria, an AI based firm has developed a natural language processing technology which scans texts and determines the relationship between concepts to write reports.

1.8.2 Detecting Spam

Spam detection is the earliest problem solved by ML. Four years ago, email service providers used pre-existing rule-based techniques to remove spam. But now the spam filters create new rules themselves using ML. Thanks to ‘neural networks’ in its spam filters, Google now boasts of 0.1 percent of spam rate. Brain-like “neural networks” in its spam filters can learn to recognize junk mail and phishing messages by analyzing rules across an enormous collection of computers. In addition to spam detection, social media websites are using ML as a way to identify and filter abuse.

1.8.3 Product recommendation

Unsupervised learning enables a product based recommendation system. Given a purchase history for a customer and a large inventory of products, ML models can identify those products in which that customer will be interested and likely to purchase. The algorithm identifies hidden pattern among items and focuses on grouping similar products into clusters. A model of this decision process would allow a program to make recommendations to a customer and motivate product purchases. E-Commerce businesses such as Amazon have this capability. Unsupervised learning along with location detail is used by Facebook to recommend users to connect with others users.

1.8.4 Medical Diagnosis

Machine Learning in the medical field will improve patient’s health with minimum costs. Use cases of ML are making near perfect diagnoses, recommend best medicines, predict readmissions and identify high-risk patients. These predictions are based on the dataset of anonymized patient records and symptoms exhibited by a patient. Adoption of ML is happening at a rapid pace despite many hurdles, which can be overcome by practitioners and consultants who know the legal, technical, and medical obstacles.

1.8.5 Customer segmentation and Lifetime value prediction

Customer segmentation, churn prediction and customer lifetime value (LTV) prediction are the main challenges faced by any marketer. Businesses have a huge amount of marketing relevant data from various sources such as email campaign, website visitors and lead data. Using data mining and machine learning, an accurate prediction for individual marketing offers and incentives can be achieved. Using ML, savvy marketers can eliminate guesswork involved in data-driven marketing. For example, given the pattern of behavior by a user during a trial period and the past behaviors of all users, identifying chances of conversion to paid version can be predicted. A model of this decision problem would allow a program to trigger customer interventions to persuade the customer to convert early or better engage in the trial.

1.8.6 Financial analysis

Due to large volume of data, quantitative nature and accurate historical data, machine learning can be used in financial analysis. Present use cases of ML in finance include algorithmic trading, portfolio management, fraud detection and loan underwriting. According to Ernst and Young report on 'The future of underwriting' – Machine learning will enable continual assessments of data for detection and analysis of anomalies and nuances to improve the precision of models and rules. And machines will replace a large no. of underwriting positions. Future applications of ML in finance include Chabot and conversational interfaces for customer service, security and sentiment analysis.

1.8.7 Predictive maintenance

Manufacturing industry can use artificial intelligence (AI) and ML to discover meaningful patterns in factory data. The corrective and preventive maintenance practices costs and efficiency is shown in Figure 1.1. Corrective and preventive maintenance practices are costly and inefficient. Whereas predictive maintenance minimizes the risk of unexpected failures and reduces the amount of unnecessary preventive maintenance activities.

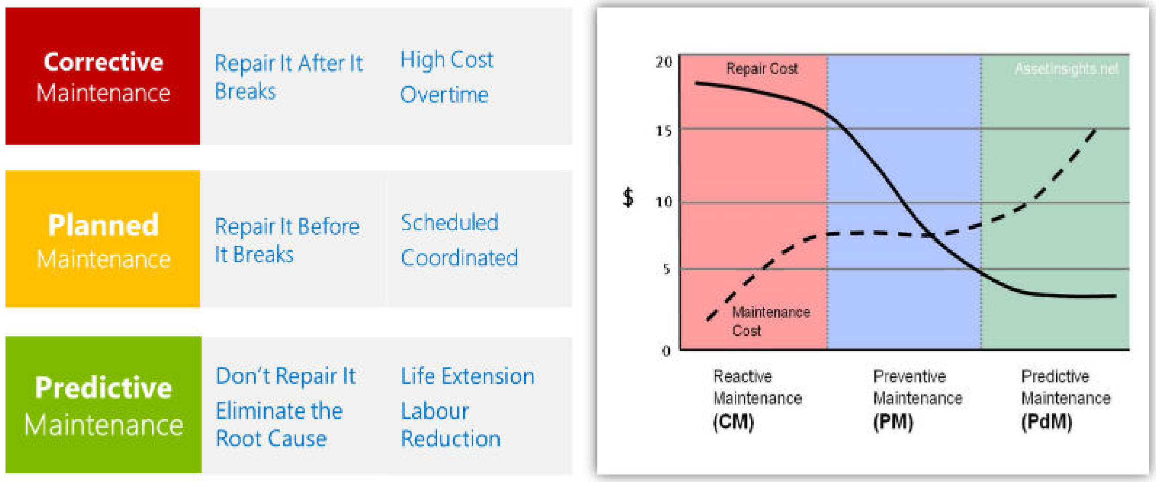


Figure 1.1: Corrective, Preventive and Predictive Maintenance

For predictive maintenance, ML architecture can be built which consists of historical device data, flexible analysis environment, and workflow visualization tool and operations feedback loop. Azure ML platform provides an example of simulated aircraft engine run-to-failure events to demonstrate the predictive maintenance modeling process. The asset is assumed to have a progressing degradation pattern. This pattern is reflected in asset's sensor measurement. In order to predict future failures, ML algorithm learns the relationship between sensor value and changes in sensor values to historical failures.

1.8.8 Image recognition (Computer Vision)

Computer vision produces numerical or symbolic information from images and high-dimensional data. It involves machine learning, data mining, and database knowledge discovery and pattern recognition. Figure 1.2 shows the image recognition problem solved by ML. Potential business uses of image recognition technology are found in healthcare, automobiles – driverless cars, marketing campaigns, etc. Baidu has developed a prototype of DuLight for visually impaired which incorporates computer vision technology to capture surrounding and narrate the interpretation through an earpiece. Image recognition based marketing campaigns such as Makeup Genius by L'Oreal drive social sharing and user engagement.

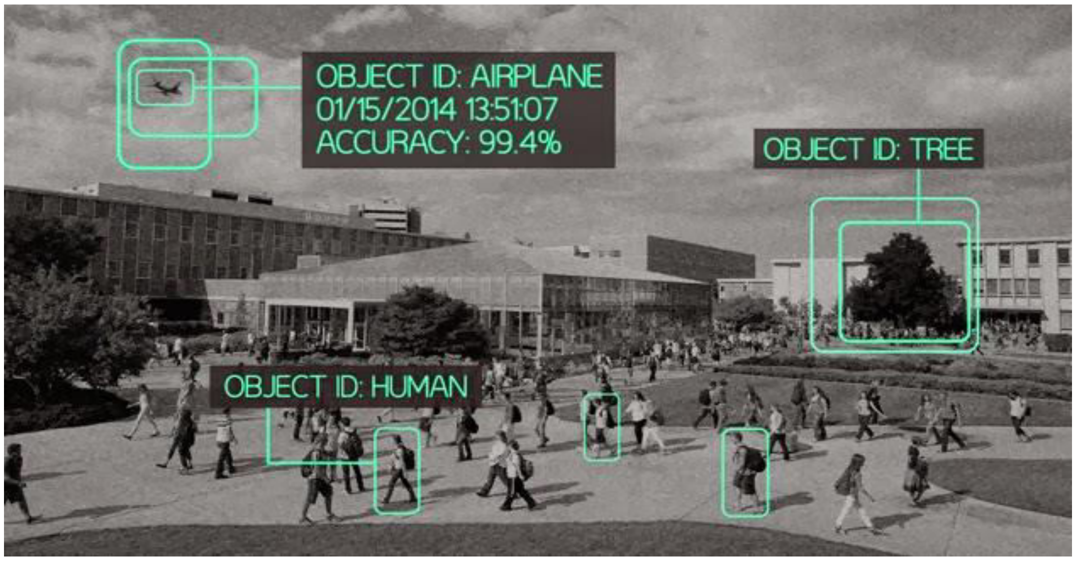


Figure 1.2: Image Recognition problem solved by ML

Most of the above use cases are based on an industry-specific problem which may be difficult to replicate for your industry. This customization requires highly qualified data scientists or ML consultants. The machine learning platforms will no doubt speed up the analysis part, helping businesses detect risks and deliver better service. But the quality of data is the main stumbling block for many enterprises. Thus apart from knowledge of ML algorithms, businesses need to structure the data before using ML data models.

CHAPTER 2

LITERATURE SURVEY

A literature survey is the effective evaluation of selected documents on a research topic. A review may form an essential part of the research process or may constitute a research project itself. In the context of a research paper or thesis, the literature review is a critical synthesis of previous research. The evaluation of the literature leads logically to the research question.

2.1 Purpose of a Literature Survey

In the context of a research paper on a thesis, the literature review provides a background to the study being proposed. The background may consider one or more of the following aspects depending on the research question being posed.

1. Theoretical background – past, present or future
2. Clinical practice – previous or contemporary
3. Methodology and/or research methods
4. Previous findings
5. Rationale and/or relevance of the current study

2.2 Literature Survey

The following are some of the literatures surveyed for preparation of the report.

[1] Abinaya.R, Aishwaryaa.P, Baavana.S,ThamaraiSelvi.N.D,Automatic Sentiment Analysis of User Reviews, 2016.

Bag of Words model along with Dual Sentiment Analysis is used to classify the reviews as positive negative and neutral. However, the performance of it sometimes remains limited due to fundamental deficiencies in handling the polarity shift problem.

The projected approach uses a dictionary based classification for accurately classifying the reviews as positive negative and neutral.

[2] A.Jeyapriya and C.S.KanimozhiSelvi, Extracting Aspects and Mining Opinions in Product Reviews using Supervised Learning Algorithm, 2015.

Social media is emerging rapidly on the internet. This media knowledge helps people, company and organizations to analyze information for important decision making. Opinion mining involves in building a system to gather and examine opinions about the product made in reviews or tweets, blog posts on the web.

The projected system implements aspect extraction using frequent item set mining in customer product reviews and mining opinions whether it is positive or negative. It identifies sentiment orientation of each aspect by supervised learning algorithms in customer reviews.

[3] Kang Liu, LihengXu, and Jun Zhao, Co-Extracting Opinion Targets and Opinion Words from Online Reviews Based on the Word Alignment Model, 2015.

Mining opinion targets and opinion words from online reviews are important tasks for the firm-grained opinion mining, the key component of which involves detecting opinion relation among words.

This paper proposes a novel approach based on the partially-supervised alignment model, which regards identifying opinion relations as an alignment process. Then, a graph-based co-ranking algorithm is exploited to estimate the confidence of each candidate and finally, candidates with higher confidence are extracted as opinion targets or opinion words.

[4] Alexandra Cornian, ValentinSgarcian, Bogdan Martin, Sentiment analysis from product reviews using SentiWordNet as lexical resource, 2015.

In the social, technological and economic context, customers make their decisions based mostly on the opinion of other consumers. On the other side, companies need quick feedback from their customers in order to adapt to their needs in real time. The effective connection between these two aspects relies on opinion mining tools, which automatically process consumers' reviews and opinions about products or services.

This paper presents a semantic approach for a sentiment analysis application, which is based on using the SentiWordNet lexical resource. The experimental validation proved a 61% average rate of success of the application.

[5] AnkitaSrivastava, Dr. M.P.Singh, Prabhat Kumar, Supervised Sentiment Analysis of Product Reviews using K-NN Classifier, 2014.

The importance of product review is escalating exponentially. Most of the existing sentiment analysis system requires large training datasets and complex tools for implementation. This paper presents a Two-Parse algorithm with a training dataset for automatic product review analysis. The proposed classifier is capable of successfully classifying weakly and mildly polar views along with the highly polar views.

[6] Zheng-Jun Zha, Jianxing Yu, Meng Wang and Tat-Seng Chua, Product Aspect Ranking and Its Applications, 2013.

Numerous consumer reviews of products are now available on the internet. Consumer reviews contain rich and valuable knowledge for both firms and users. However, the reviews are often disorganized, leading to difficulties in information navigation and knowledge acquisition.

This article proposes a product aspect ranking framework, which automatically identifies the important aspects of products from online consumer reviews, aiming at improving the usability of the numerous reviews.

[7] David Garcia and Frank Schweitzer, Emotions in product reviews – Empirics and models, 2011.

Online communities provide internet users with means to overcome some information barriers and constraints, such as the difficulty to gather independent information about products and firms. Product review communities allow customers to share their opinions and emotions after the purchase of a product.

This article quantifies the evidence of polemic review, i.e. some users find them helpful and others not. It identifies the correlation between rating and helpfulness. The collective expression of positive emotions in product reviews shows a strong bias towards positive values, while the negative content is more evenly distributed along its possible values.

[8] JantimaPolpinij and Aditya K. Ghose,An Ontology-based Sentiment Classification Methodology for Online Consumer Reviews, 2008.

Sentiment analysis also known as opinion mining, studies people's sentiments toward certain entities. Internet is a resourceful place with respect to sentiment information. From a user's perspective, people are able to post their own content through various social media such as forums or online social networking sites. From a researcher's perspective, many social media sites release their APIs, prompting data collection and analysis by researchers and developers.

This paper presents a method of ontology-based sentiment classification to classify the online product reviews. It implements and experiments with Super Vector Machine based on the lexical variation ontology.

[9] SilvanaAciar, Debbie Zhang, Simeon Simoff, JohnDebenham,Recommender System Based on Consumer Product Reviews, 2006.

This paper proposed a novel approach to create recommendations in recommender system which utilizes online consumer review comments.

A ranking mechanism for prioritizing the product quality with respect to the consumer level of expertise and the rating given to some features of the product has been developed.

CHAPTER 3

MAJOR CHALLENGES OF MACHINE LEARNING

3.1 Communication: Unclear questions and outcome metrics

A fundamental challenge facing data scientists has nothing to do with ensemble algorithms, optimization methods, or computing power. Communication – prior to any analysis or data engineering – is crucial to solving an ML problem quickly and painlessly.

There are many, many questions ML can solve: this is an incredibly powerful tool for making sense of the world around us. However, these questions have to be specific and formulaic in a way that the people responsible for identifying the problem, such as management or marketing, might be unfamiliar with.

Questions posed in a 'real-world' environment, while substantively useful for framing and approaching a business problem, are often too vague to translate directly into ML modeling. Because of this, it is crucial to communicate effectively between different branches within the organization: the 'small' question being solved by ML modeling has to match the 'big' question that constitutes the business problem itself.

3.2 Feature engineering: getting more information out of a data set

Feature engineering and feature selection are important parts of any ML task. Even with highly sophisticated estimation algorithms and powerful, cheap computing capabilities, the data scientist plays an important role in creating a model that is both accurate and efficient.

Significant time and energy can (and should!) be spent on looking over the data itself to try and identify additional information that may be 'hiding' in the features already included. It may be, for example, that the difference between two values (for example, length of time since a customer’s most recent transaction) matters more to predictive accuracy than either of the values themselves.

This means that feature engineering is a combination of subject matter expertise and general intuition: skilled feature engineers can pull the maximum amount of useful information out of a given set of input data, giving an ML model the most informative data set possible to work with.

3.3 Logistics: budgeting computational resources

Few things are more frustrating than putting in hours, days, or weeks of work on cleaning and preparing a data set for analysis, only to hit an ‘out of memory’ error when trying to build the finalized model. Budgeting computational resources for ML estimation can be tricky: over-budgeting on powerful computing systems can waste significant money, but under-budgeting can produce severe bottlenecks in model construction and deployment.

However, cloud computing has taken dramatic steps towards making computational pipelines more expandable. Using a system like Amazon’s AWS allows for the deployment of larger virtual machines (or greater numbers of machines, if working in parallel) with relatively low cost and high speed. This type of elastic-computing framework makes it much, much easier to budget appropriately when setting up an ML system, especially when working with very large data sets.

3.4 Generalizability: Conflation of training and testing data sets

Particularly for those who are first getting into data science, this can be an easy step to miss, but it is incredibly important. ML models are built for estimation: their purpose is to intake new data and generate values that can be used to guide future decisions. Because of this, it is absolutely crucial to separate ‘training’ data that is used to fit an original ML model from ‘testing’ data that is used to assess the model’s accuracy.

Failure to do some type of out-of-sample testing can result in a model that looks fantastic in terms of accuracy and fit statistics… and then fails miserably when faced with new, unfamiliar data. Generalizability is the key to creating usable long-term ML solutions, and as such, models need to be tested on independent, out-of-sample data before being put into regular use. A solid rule of thumb is to hold back 20-25% of the original data set: this is testing data, and should be kept entirely separate from the 75-80% of data used to build the ML model itself.

3.5 Focusing on the little things: algorithm choice

The range of algorithms available for ML problem solving is astounding. Random forests, support vector machines, neural networks, and Bayesian estimation methods – the list goes on (and on, and on). The question of what algorithm is best for a given ML problem, however, is often less impactful than we might think.

It's true that some approaches, on some questions, will work better than others. In some cases, this difference can even be quite distinct. However, in my experience it's been quite rare that one modeling approach will strictly dominate all other options in answering a given ML question.

A useful middle ground in selecting an algorithm, in my opinion, is to build a 'stable' of robust modeling approaches that can be built quickly and easily for day-to-day use. Running a battery of models on a given data set allows the data scientist to pick whatever approach has the greatest marginal gain on that particular data set.

However, going far afield for exotic new algorithms, or adopting different programming languages, in my opinion, is rarely necessary or even worth the time.

CHAPTER 4

ARCHITECTURE DESIGN OF MACHINE LEARNING

Raw data is rarely of direct benefit. Its true value resides in the amount of information that a model designer can extract from it. Modeling from data is often viewed as an art form, mixing the insight of the expert with the information contained in the observations. Typically, a modeling process is not a sequential process but is better represented as a sort of loop with a lot of feedbacks and a lot of interactions with the designer. Different steps are repeated several times aiming to reach, through continuous refinements, a good model description of the phenomenon underlying the data.

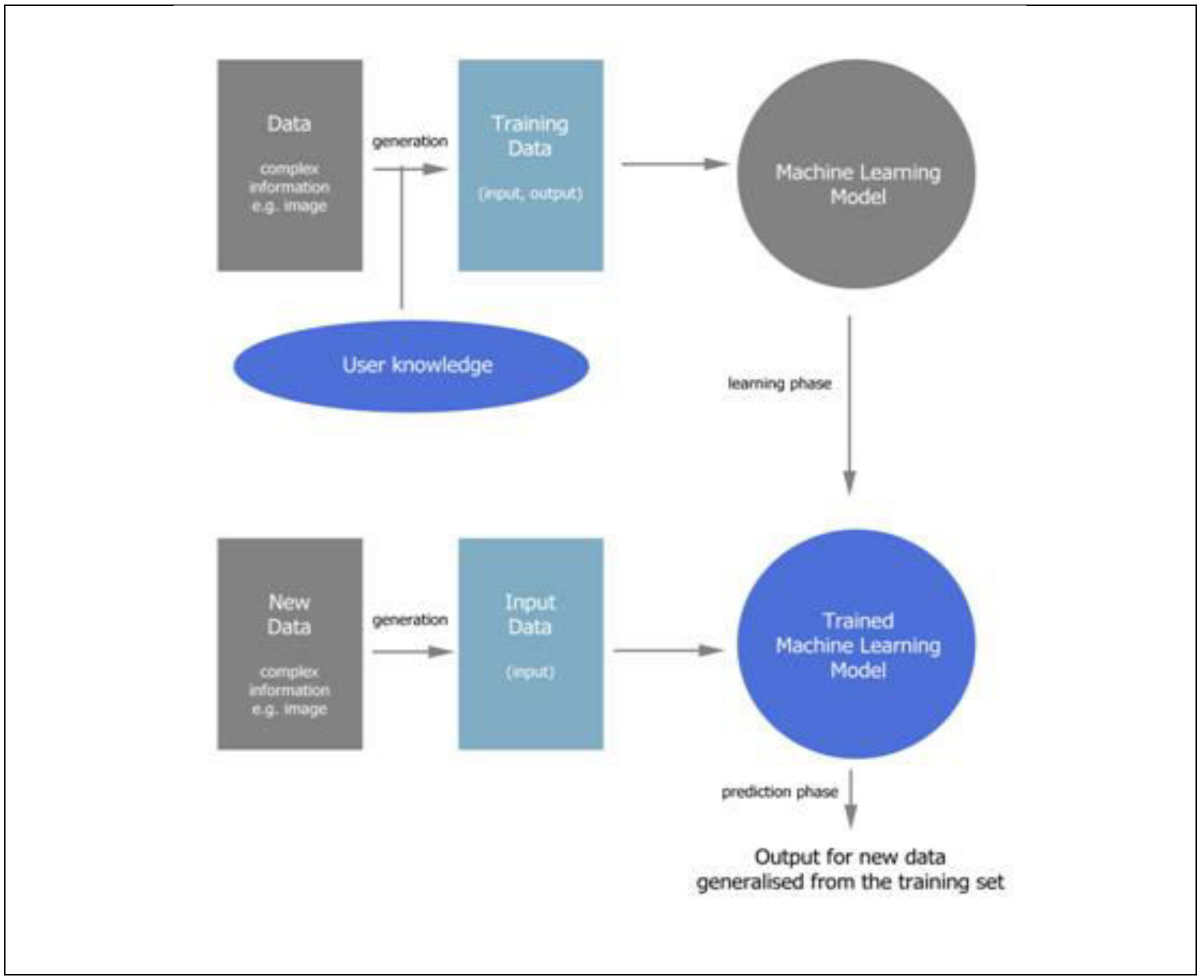

Data
complex information e.g. image
generation
Training Data
(input, output)
Machine Learning Model
User knowledge
learning phase
New Data
complex information e.g. image
generation
Input Data
(input)
Trained Machine Learning Model
prediction phase
Output for new data generalised from the training set

Figure 4.1: Machine Learning System Architecture

The data modeling process is partitioned into 2 phases as shown in Figure 4.1.

- Preliminary phase
- Learning phase

4.1 Preliminary phase

A preliminary phase leads from the raw data to a structured training set. The preliminary phase is made of a problem formulation step where the designer selects the phenomenon of interest and defines the relevant input/output features, an experimental design step where input/output data are collected and a data pre-processing step where a preliminary filtering of data is performed.

4.2 Learning phase

A learning phase leads from the training set to the final model. Once the dataset has been formatted, the learning procedure begins. In qualitative terms, this procedure can be described as follows. First the designer defines a set of parametric models (e.g. polynomial models, neural networks) and a complexity index (or hyper parameter) which controls the approximation power of the model (e.g. degree of polynomial, number of neurons). By means of the complexity index, the set of parametric models is consequently decomposed in a nested sequence of classes of models (e.g. classes of polynomials with increasing degree). Hence, a structural identification procedure loops over the set of classes, first by identifying a parametric model for each class (parametric identification) and then by assessing the prediction error of the identified model on the basis of the finite set of samples (validation). Finally a model selection procedure selects the final model to be used for future predictions.

CHAPTER 5

EXISTING SYSTEM

5.1 Existing System

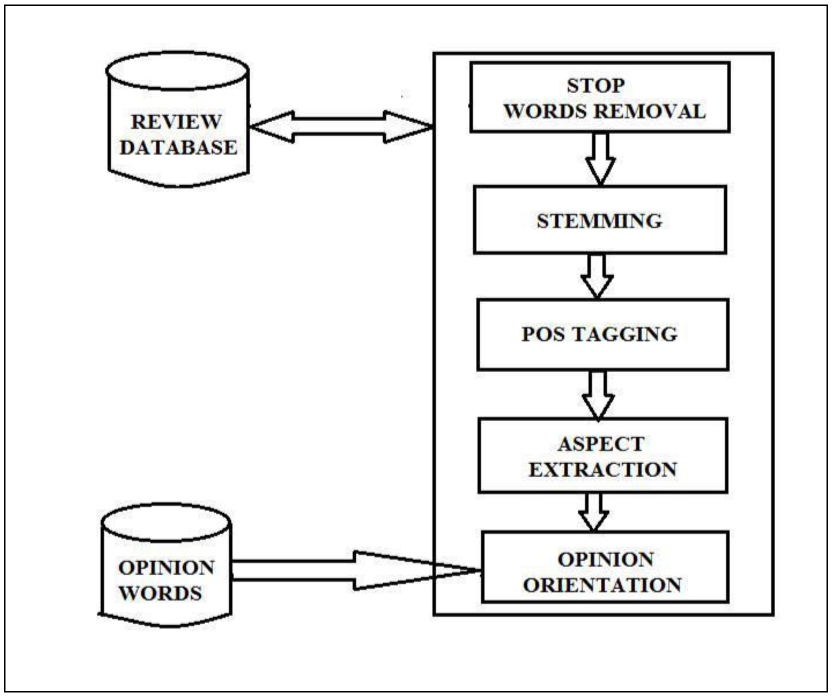


Figure 5.1: Working of Aspects Extraction System Architecture

The existing system uses customer reviews to extract aspect and mine whether given is positive or negative opinion as shown in Figure 5.1. Each review is split into individual sentences. A review sentence is given as input to data preprocessing. Next, it extracts aspect in each review sentence. Stop word removal, stemming and POS tagging are data preprocessing. Sentiment orientation is used to identify whether it is positive or negative opinion sentence. Then it identifies the number of positive and negative opinions of each aspect.

5.1.1 Stop Word Removal

Most frequently used words in English are not useful in text mining. Such words are called stop words. Stop words are language specific functional words which carry no information. It may be of types such as pronouns, prepositions, conjunctions. Stop word removal is used to remove unwanted words in each review sentence. Words like is, are, was etc. Reviews are stored in text file which is given as input to stop word removal. Stop words are collected and stored in a text file. Stop word is removed by checking against stop words list.

5.1.2 Stemming

Stemming is used to form root word of a word. A stemming algorithm reduces the words "longing", "longed", and "longer" to the root word, "long". It consist many algorithms like n-gram analysis, Affix stemmers and Lemmatization algorithms. Porter stemmer algorithm is used to form root word for given input reviews and store it in text file.

5.1.3 POS Tagging

The Part-Of-Speech of a word is a linguistic category that is defined by its syntactic or morphological behavior. Common POS categories in English grammar are: noun, verb, adjective, adverb, pronoun, preposition, conjunction, and interjection. POS tagging is the task of labeling (or tagging) each word in a sentence with its appropriate part of speech. POS tagging is an important phase of opinion mining, it is essential to determine the features and opinion words from the reviews. POS tagging can be done either manually or with the help of POS tagger tool. POS tagging of the reviews by human is time consuming. POS tagger is used to tag all the words of reviews. Stanford tagger is used to tag each word in an online review sentences. Every one sentence in customer reviews are tagged and stored in text file.

5.1.4 Aspect Extraction

Frequent item set mining is used to find all frequent item sets using minimum support count. Here, every sentence is assigned as single transaction. Noun Words in each sentence is assigned as item sets for single transactions.

Aspect extraction is implemented using Figure 5.2. This algorithm first extracts noun and noun phrases in each review sentence and store it in a text file. Minimum support threshold is used to find all frequent aspects for a given review sentences. Aspects like pictures, battery, resolution, memory, lens etc. Then, the frequent aspects are extracted and stored in text file.

5.1.5 Sentence and Aspect Orientation

The existing system first determines the number of positive and negative opinion sentence in reviews using opinion words. The positive and negative labels are collected labels in opinion words. Examples of positive opinion words are long, excellent and good and the negative opinion words are like poor, bad etc. And the next step is to identify the number of positive and negative opinions of each extracted aspect. Both sentence and aspect orientations are implemented using Naïve Bayesian algorithm using supervised term counting based approach. The probabilities of the positive and negative count are found according to the words using Naïve Bayesian classifier.

5.1.6 Opinion words

Opinion word rule in Figure 5.3 gives that, if word is matched with positive opinion words then positive count get increment, or it is negative opinion word then negative count get increment.

In Figure 5.3 *Negation rules* have a negation word or phrase which usually reverses the opinion expressed in a sentence.

Two rules must be applied:

1. Negation Negative->Positive. This will increment positive count.
2. Negation Positive ->Negative. This will increment negative count.

After comparing all the words of the sentence, the found probabilities of the positive and negative counts are compared in the following manners.

a) If the probability of positive count is greater than the negative count, then the sentence or opinion is positive.

b) If the probability of negative count is greater than the positive count, then the sentence or opinion is negative.

c) If the probability of positive count minus probability of negative count is zero, then it is neutral.

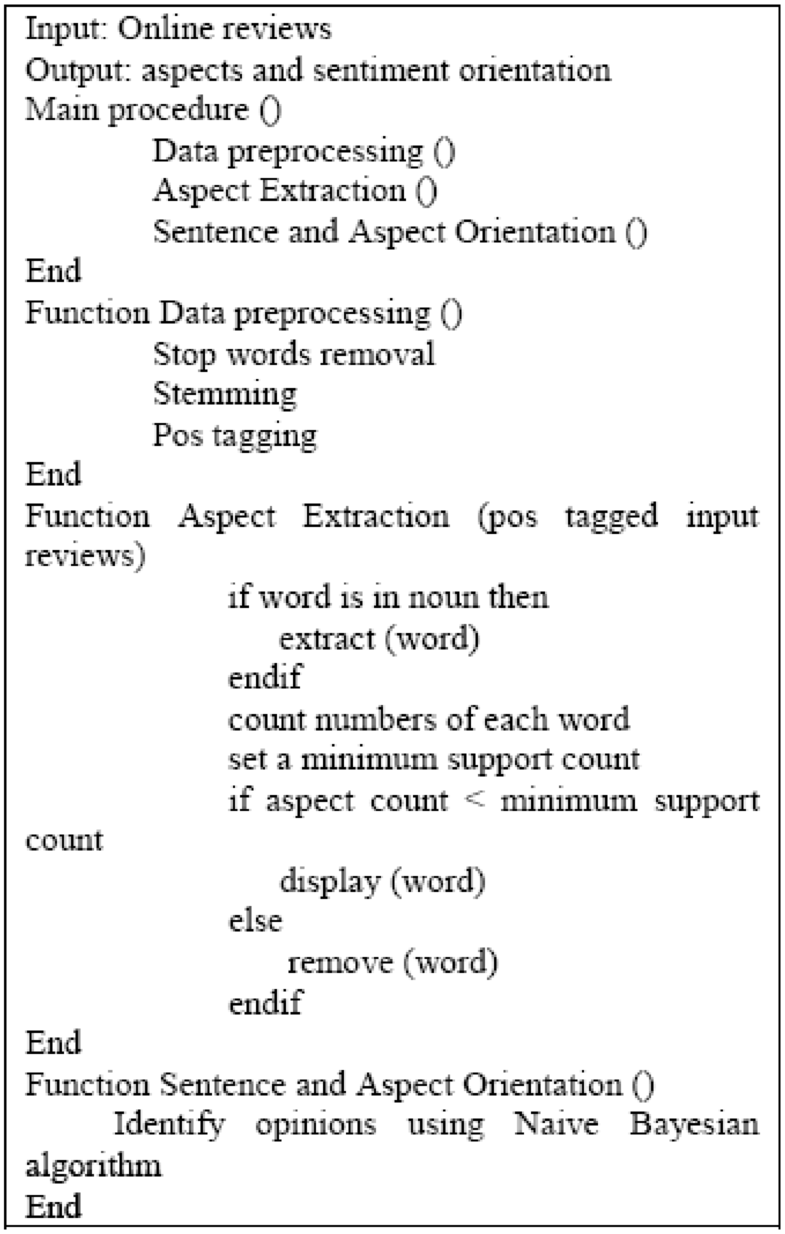
Finally system identifies the number of positive and negative opinion of each extracted aspect in customer reviews.

5.2 Naive Bayesian algorithm

In machine learning, Naive Bayes classifiers are a family of simple probabilistic classifiers based on applying Bayes' theorem with strong (naive) independence assumptions between the features (shown in Figure 5.2).

Steps are as follows:

1. The positive labels, negative labels and review sentences are stored in separate text file.
2. Split the sentence into the combination of words. It means first combination of two words and then single words.
3. First compare the combination of two words, if it matched then delete that combination from the opinion. Again start comparing of single word.



```
Input: Online reviews
Output: aspects and sentiment orientation
Main procedure ()
        Data preprocessing ()
        Aspect Extraction ()
        Sentence and Aspect Orientation ()
End
Function Data preprocessing ()
        Stop words removal
        Stemming
        Pos tagging
End
Function Aspect Extraction (pos tagged input
reviews)
             if word is in noun then
                 extract (word)
             endif
             count numbers of each word
             set a minimum support count
             if aspect count < minimum support
count
                 display (word)
             else
                 remove (word)
             endif
End
Function Sentence and Aspect Orientation ()
      Identify opinions using Naive Bayesian
algorithm
End
```

Figure 5.2: Naive Bayesian algorithm

```
if word is in opinion_words then mark(word)
orientation
————————————————————$SSO\
————————————————————2SLQLRQ
————————————————————:RUG
————————————————————5XOH
end if
if word is near a negation word then orientation
$SSO\————————————————————1HJDWLRQ
————————————————————5XOHV
end if
return orientation
```

Figure 5.3: Sentiment Orientation Algorithms

Initially, the probabilities of positive and negative count to zero [positive=0, negative=0]. The sentiment orientation algorithm is shown in Figure 5.3.

5.3 Drawbacks of Existing System

- There are no summarized aspects of reviews based on the relative importance of the extracted aspect.
- Unavailability of graphical representation about the overall sentiment of the product.

CHAPTER 6

SYSTEM REQUIREMENT SPECIFICATION

Software Requirement Specification (SRS) is a requirement specification for a software system which is a complete description of the behavior of a system to be developed. In addition to a description of software functions, the SRS also contains non-functional requirements. Software requirements are a sub field of software engineering that deals with the elicitation, analysis, specification and validation of requirements for software.

Detailed software description can serve as a basis for a design or implementation to the developer. The requirement engineering process normally involves in writing a definition and then expanding it into the requirement specification. The different levels of a system specification are useful because it communicates information about the system to different types of developer. The minimum requirements are expected to be described. Transformation on objects/selected area should be possible. Good user interface is to be provided.

6.1 Functional Requirements

In Software engineering and systems engineering, a **functional requirement** defines a function of a system or its component. A function is described as a set of inputs, the behavior, and outputs.

Functional requirements may be calculations, technical details, data manipulation and processing and other specific functionality that define *what* a system is supposed to accomplish.

Functional requirements of the projected system include the following:

- The system shall display all the attribute of the products that can be configured.
- The system shall enable user to add one or more attribute of the product to the configuration.

- The system shall display the sentiments of each product attribute to the user.
- The system should provide the user with overall sentiment of the product.

6.2 Non Functional Requirements

In systems engineering and requirements engineering, a **non-functional requirement** (NFR) is a requirement that specifies criteria that can be used to judge the operation of a system, rather than specific behaviors.

Non Functional requirements of the proposed system:

- The system shall provide a uniform look between all the sentiments.
- The system shall provide storage of all databases on redundant automatic switchover.
- The algorithm should never fail with any of the test cases.

6.3 Software requirements

- Web scrapping - Beautiful Soup.
- Programming Language - Python 3.4.
- Data Mining and Natural Language Processing library – NLTK.
- Machine Learning and Data Analysis Library - Scikit Learn and Pandas.
- Data Visualization –MatPlot.

6.4 Jupyter notebook

Notebook documents contain the inputs and outputs of a interactive session as well as additional text that accompanies the code but is not meant for execution. In this way, notebook files can serve as a complete computational record of a session, interleaving executable code with explanatory text, mathematics, and rich representations of resulting objects. These documents are internally JSON files and are saved with the .ipynb extension. Since JSON is a plain text format, they can be version-controlled and shared with colleagues.

Notebooks may be exported to a range of static formats, including HTML (for example, for blog posts), reStructuredText, LaTeX, PDF, and slide shows, via the new nbconvert command.

Furthermore, any .ipynb notebook document available from a public URL can be shared via the IPython Notebook Viewer (nbviewer). This service loads the notebook document from the URL and renders it as a static web page. The results may thus be shared with a colleague, or as a public blog post, without other users needing to install IPython themselves. In effect, nbviewer is simply nbconvertas a web service, so you can do your own static conversions with nbconvert, without relying on nbviewer.

6.5 Python

Python is a widely used high-level programming language for general-purpose programming, created by Guido van Rossum and first released in 1991. An interpreted language, Python has a design philosophy which emphasizes code readability (notably using whitespace indentation to delimit code blocks rather than curly brackets or keywords), and a syntax which allows programmers to express concepts in fewer lines of code than possible in languages such as C++ or Java. The language provides constructs intended to enable writing clear programs on both a small and large scale.

Python features a dynamic type system and automatic memory management and supports multiple programming paradigms, including object-oriented, imperative, functional programming, and procedural styles. It has a large and comprehensive standard library.

Python interpreters are available for many operating systems, allowing Python code to run on a wide variety of systems. CPython, the reference implementation of Python, is open source software and has a community-based development model, as do nearly all of its variant implementations. CPython is managed by the non-profit Python Software Foundation.

Python includes the following attributes:

- Python is a high-level language, meaning that it abstracts underlying computer-related technical details. For example, Python does not make its users think too much about computer memory management or proper declaration of variables and uses safe assumptions about what the programmer is trying to convey. In addition, a high-level language can be expressed in a manner closer to English prose or mathematical equations. Python is perfect for literate programming because of its lightweight, "low ceremony" nature.
- Python is a general-purpose language meaning that it can be used for all problems that a computer is capable of rather than specializing in a specific area such as statistical analysis. For example, Python can be used for both artificial intelligence and statistical analysis. Python can be used for a variety of heterogeneous tasks within a given work-flow, as the UCAR scientist described earlier.
- Python is an interpreted language meaning that evaluation of code to obtain results can happen immediately rather than having to go through a time-consuming, compile and run cycle, which thereby speeds up the thinking and experimentation processes. IPython is an interactive form of the Python language also invented by Fernando Pérez.
- Python has a standard library, and numerous third-party libraries yielding a vast array of existing codebases and examples for solving problems.
- Python has many, many users which mean that the programmers can quickly find solutions and example code to problems with the help of Google and Stackoverflow.

6.6 Hardware Requirements

- Processor – Intel PC
- Input devices – Mouse and keyboard
- Hard disk – 20GB(approximately)
- Display – VGA color monitor
- RAM – 256MB

CHAPTER 7

PROPOSED SYSTEM

7.1 Problem Statement

The people cannot analyse exact information in the document and sentence level opinion mining on customer reviews.

7.2 Aim of the proposed system

The proposed system aims to simplify and automate the task of identifying the various aspects being talked about a particular product in a review.

7.3 Objective

- The system identifies the customer review aspects.
- Extract these aspects and identify the sentiment associated with each of these aspects.
- It also provides a graph of overall sentiment of the product.

7.4 Use case diagram

A **use case diagram** at its simplest is a representation of a user's interaction with the system that shows the relationship between the user and the different use cases in which the user is involved. A use case diagram can identify the different types of users of a system and the different use cases and will often be accompanied by other types of diagrams as well.

A use case diagram is a coherent piece of functionality that a system can provide by interacting with actors as shown in Figure 7.1. The various interactions of actors with system are quantized into use cases. Each use case involves one or more actors as well as system itself. The set of use cases shows the complete functionality of the system at some level of detail. Similarly, each actor represents one kind of object for which the system can perform behavior. The set of actors represents the complete set of objects that the system can serve. Objects accumulate behavior from all the system with which they interact as actors.

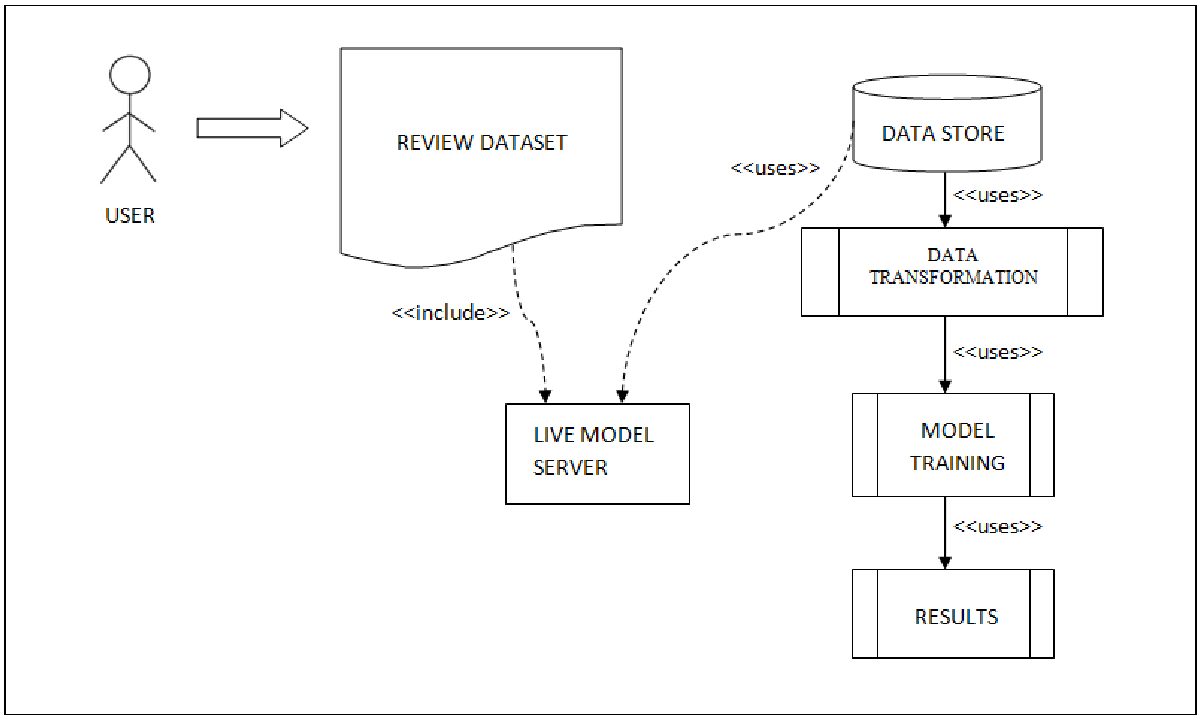


Figure 7.1: Use case diagram for proposed approach

7.5 Proposed System Architecture Design

The architecture design of the proposed system is as shown in Figure 7.2.

The proposed system consists of 3 major components namely,

1) Training Layer
2) Processing Layer
3) Trained Model

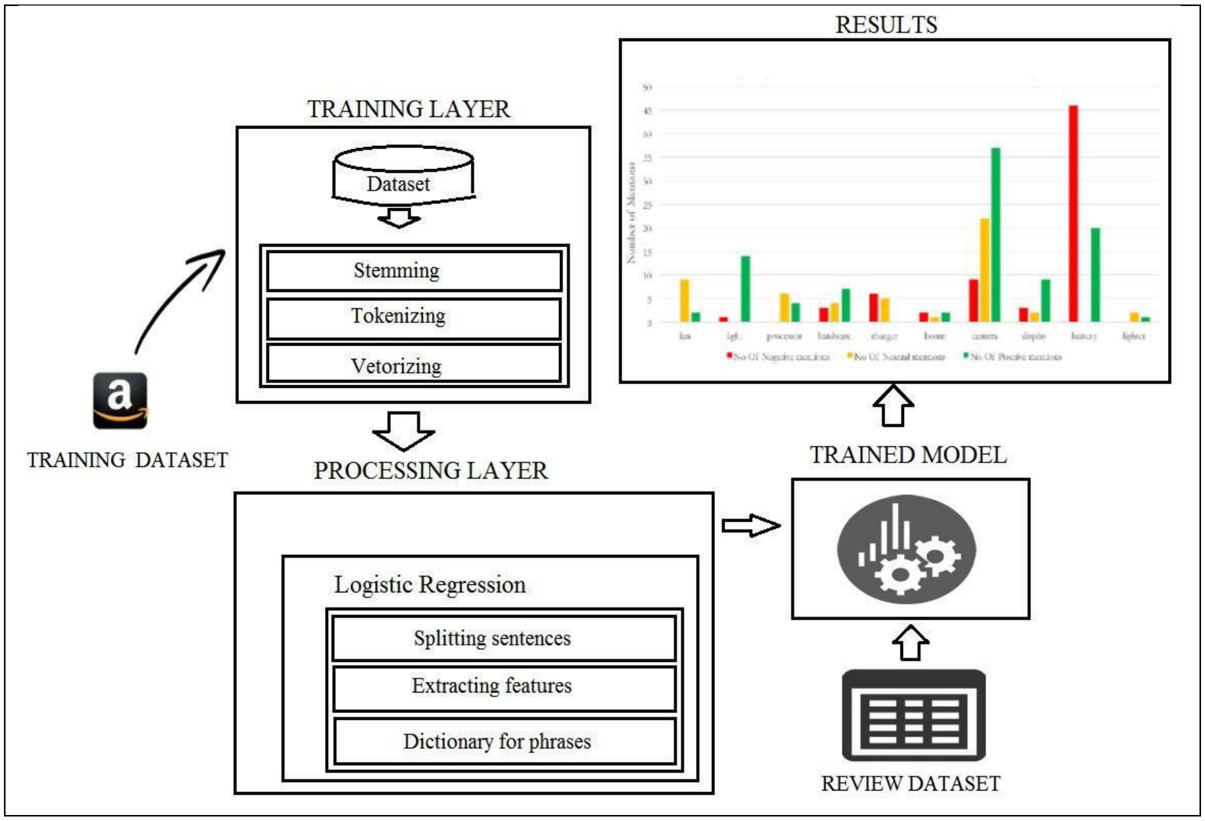


Figure 7.2: The architecture of identifying sentiments of online reviews

7.5.1 Training Layer

The training layer functionality includes stemming, tokenizing and vectorizing.

- **Stemming:** Stemming is used to form root word of a word. A stemming algorithm reduces the words “longing”, “longed” and “longer” to the root word, “long”. It consists of algorithms like Affix stemmers and Lemmatization algorithm.

- **Tokenizing:** To correctly chunk the sentences into single attribute phrases, grammar rules are used to identify the associated descriptors for each product attribute in the sentences, and tokenize them appropriately. It includes algorithms like Context Free Grammar, Using Regex Parsers, Utilizing conjunction and other punctuations in the sentences to perform tokenization. Of all the methods that we tried, the final method i.e.

using conjunctions, and punctuations to tokenize sentences worked the best on the review data.

- **Vectorizing:** Vectorization is the term for converting a scalar program to a vector program. Vectorized program can run multiple operations from a single instruction. The different vectors used here are:
    - **Count Vectorizer:** This vectorizer converts text 'documents' into a matrix of token counts, specifically, a sparse matrix representation of the counts using the *scipy.sparse.coo_matrix*. We specified the analyzer to 'words', allowed lowercasing of tokens, and also asked the vectorizer to consider Unigrams, Bigrams, and Trigrams as features.
    - **Tf-Idf Vectorizer:** This vectorizer converts the collection of raw 'documents' into a matrix of TF-IDF features. It is equivalent to applying CountVectorizer on the set of 'documents' and then applying Tf-Idf transformation on it. "Tf–Idf, short for term frequency–inverse document frequency, is a numerical statistic that is intended to reflect how important a word is to a document in a collection or corpus. "In this vectorizer as well, we tried different combinations of various parameters, to output the best results.

7.5.2 Processing Layer

Logistic Regression is a discriminative model, which can be used to directly estimate the probability of occurrence of y given occurrence of x (p(y|x)). For this model to work correctly there is no restriction on the features to be co-related. It can be used to provide multi-category classification in cases where the categories are exhaustive, and mutually exclusive, i.e. every instance belongs to one, and only one category.

A sample code for the LR Classifier:

log_model=**LogisticRegression5**()

log_model=log_model.**fit5**(X=X_train,y=y_train)

y_log_pred=log_model.**predict5**(X_dev)

The functionalities of Logistic regression include the following:

- Splitting sentences
- Extracting features
- Dictionary for phrases

A Logistic model that predicts probabilities between 0 and 1, that is S-shaped is built as shown in Figure 7.3.

Probability=$1/[1+\exp(\beta_0+\beta_1 X)]$ or $\log_e[P/(1-P)] = \beta_0+\beta_1 X$

The function $\log_e[P/(1-P)]$ is called the logistic function.

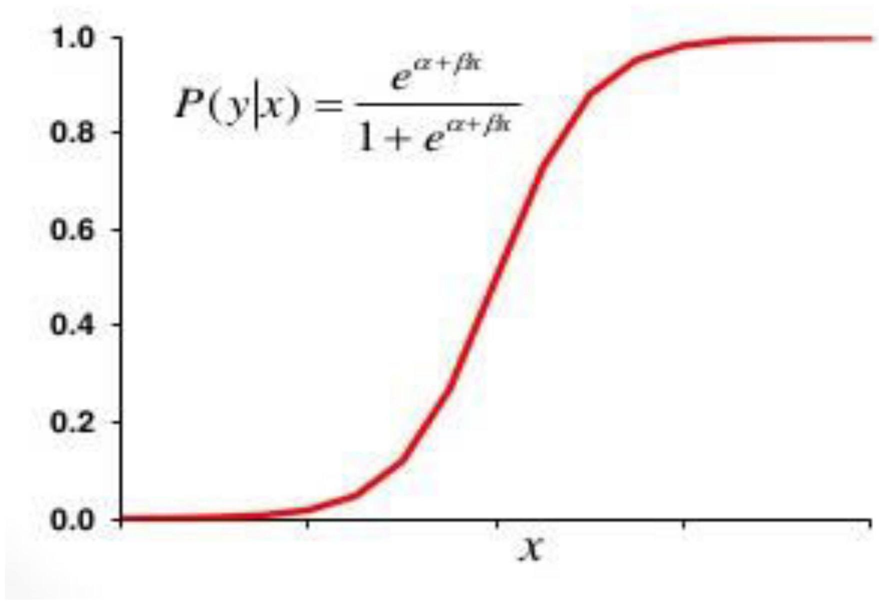


Figure 7.3: Logistic Curve

7.5.3 Trained Model

You should always *evaluate a model* to determine if it will do a good job of predicting the target on new and future data. Because future instances have unknown target values, you need to check the accuracy metric of the ML model on data for which you already know the target answer, and use this assessment as a proxy for predictive accuracy on future data.

To properly evaluate a model, you hold out a sample of data that has been labelled with the target (ground truth) from the training data source. Evaluating the predictive accuracy of an ML model with the same data that was used for training is not useful, because it rewards models that can "remember" the training data, as opposed to generalizing from it. Once you have finished training the ML model, you send the model the held-out observations for which you know the target values. You then compare the predictions returned by the ML model against the known target value. Finally, you compute a summary metric that tells you how well the predicted and true values match.

7.6 Workflow of the proposed system

The Figure 7.4 shows the workflow of the proposed system. The major 3 components included are Product Attribute Extraction, Data Processing and Sentiment Prediction.

7.6.1 Product Attribute Extraction

One of the most popular applications for named entity detection is product attribute extraction. Automatically extracting product attributes from text helps E-commerce company correctly process user query and match it the right products. Marketers want to know what people are talking about related to their products. Corporate strategists' wants to find out what products are trendy based on user discussions.

Product attributes are specific properties related to a product. For example, Apple iPhone 5 with black color has 4 major attributes: Company name as "Apple", brand name as "iPhone", generation as "5", and color as "black". A complete set of attributes define a product.

What challenge do we face in extracting product attributes? We can create a dictionary of products and their corresponding attributes, but simply relying on a dictionary has its limitations:

- A dictionary is not complete as new products are created.
- There is ambiguity in matching the attributes.

Misspelling, abbreviations, and acronyms are used often, particularly in social media such as Twitter.

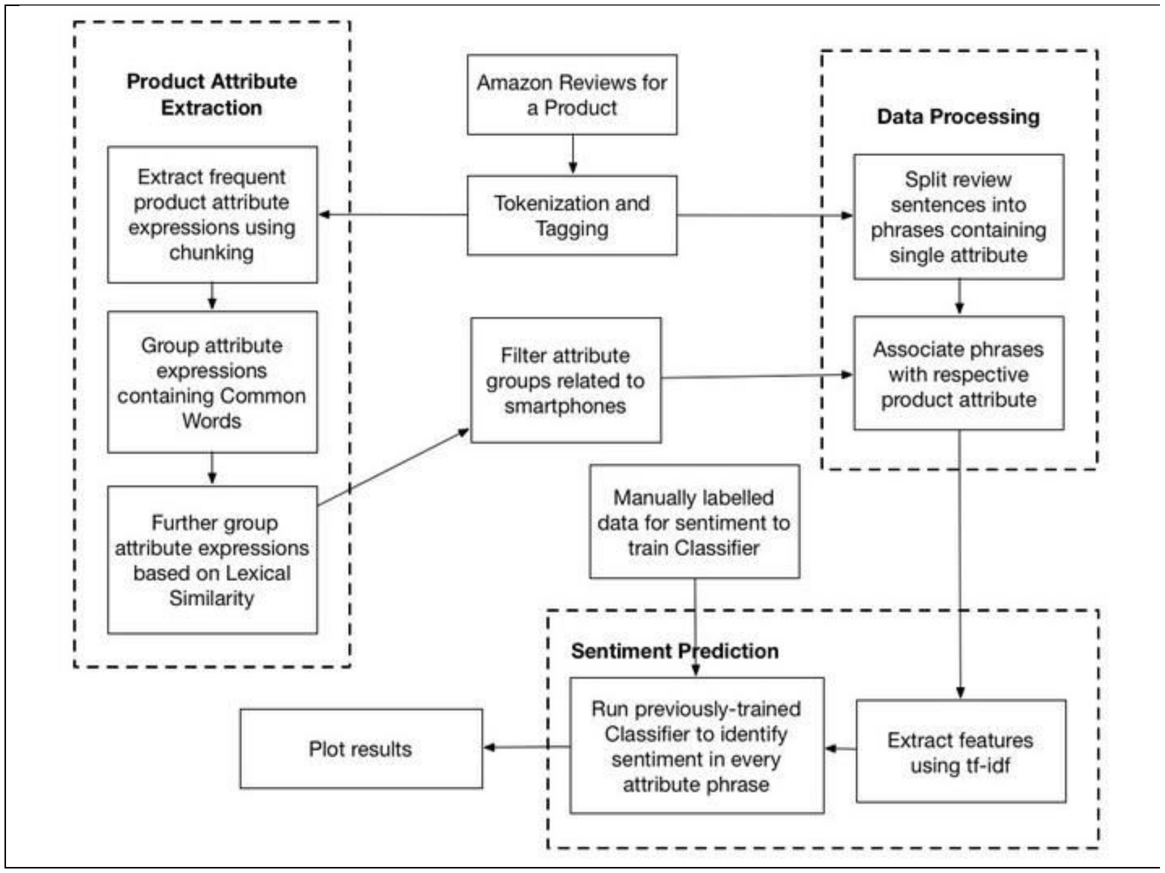

Product Attribute Extraction
Extract frequent product attribute expressions using chunking
Group attribute expressions containing Common Words
Further group attribute expressions based on Lexical Similarity
Amazon Reviews for a Product
Tokenization and Tagging
Filter attribute groups related to smartphones
Data Processing
Split review sentences into phrases containing single attribute
Associate phrases with respective product attribute
Manually labelled data for sentiment to train Classifier
Sentiment Prediction
Run previously-trained Classifier to identify sentiment in every attribute phrase
Extract features using tf-idf
Plot results

Figure 7.4: Workflow of the system

7.6.2 Data Processing

Data processing is, generally, "the collection and manipulation of items of data to produce meaningful information. In this sense it can be considered a subset of *information processing*, "the change (processing) of information in any manner detectable by an observer. "The term Data Processing (DP) has also been used previously to refer to a department within an organization responsible for the operation of data processing applications.

Data processing may involve various processes, including:

- Validation – Ensuring that supplied data is correct and relevant.
- Sorting – "arranging items in some sequence and/or in different sets."
- Summarization – reducing detail data to its main points.
- Aggregation – combining multiple pieces of data.
- Analysis – the "collection, organization, analysis, interpretation and presentation of data."
- Reporting – list detail or summary data or computed information.
- Classification – separates data into various categories.

7.6.3 Sentiment Prediction

Sentiment analysis (sometimes known as opinion mining or emotion AI) refers to the use of natural language processing, text analysis, computational linguistics, and biometrics to systematically identify, extract, quantify, and study affective states and subjective information. Sentiment analysis is widely applied to voice of the customer materials such as reviews and survey responses, online and social media, and healthcare materials for applications that range from marketing to customer service to clinical medicine.

Generally speaking, sentiment analysis aims to determine the attitude of a speaker, writer, or other subject with respect to some topic or the overall contextual polarity or emotional reaction to a document, interaction, or event. The attitude may be a judgment or evaluation (see appraisal theory), affective state (that is to say, the emotional state of the author or speaker), or the intended emotional communication (that is to say, the emotional effect intended by the author or interlocutor).

In order to extract sentiments from the phrases derived in the Product Phrase Extraction and Association step, we decided to use various machine learning techniques. We used vectorizers to extract and learn syntactic patterns accompanying different sentiments in the reviews. We then trained classifiers on manually labelled training dataset to predict the sentiment contained in each of the phrases.

7.7 Data Flow Diagram

A **data flow diagram** (**DFD**) is a graphical representation of the "flow" of data through an information system, modeling its *process* aspects. A DFD is often used as a preliminary step to create an overview of the system without going into great detail, which can later be elaborated. DFDs can also be used for the visualization of data processing (structured design).

LEVEL 0:

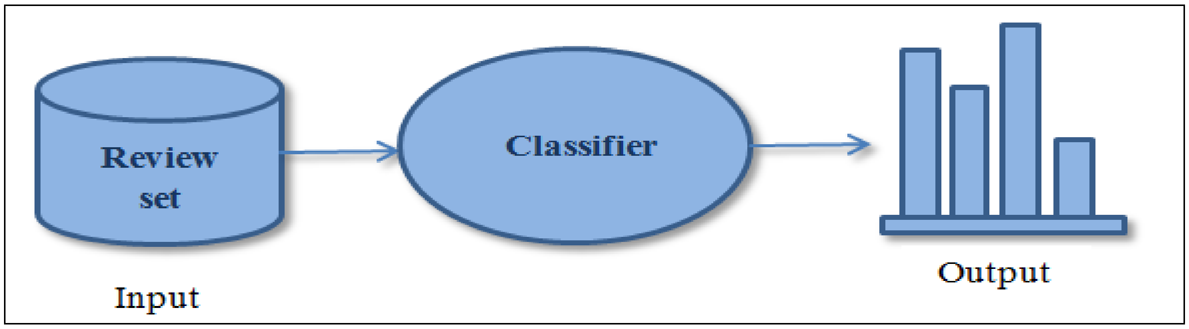


Figure 7.5: Level 0 DFD

Data level 0 is the basic level of DFD. It indicates the input and output of the system. The input to the Classifier is the review dataset and output is the graphical representation of the opinions as show in Figure 7.5.

LEVEL 1:

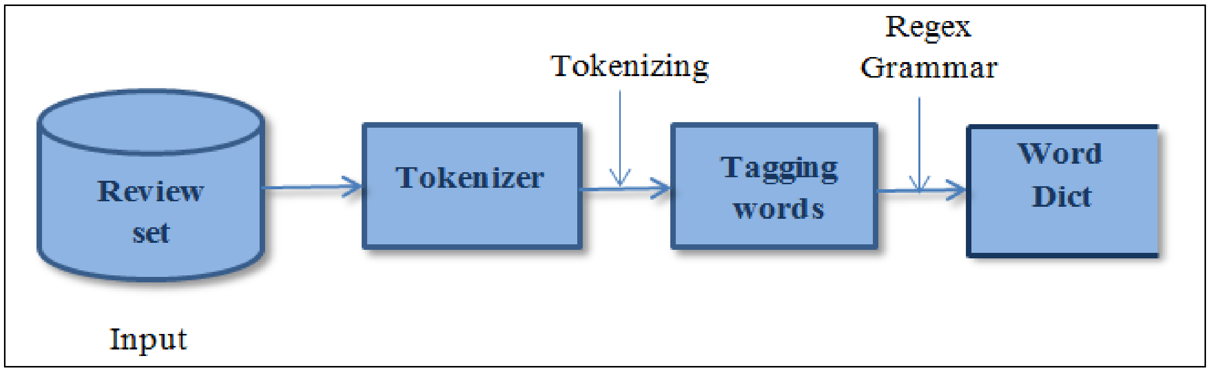


Figure 7.6: Level 1 DFD

Level 1 DFD is shown in Figure 7.6. It describes the first processing stage, Product Attribute Extraction. Tokenizing and Regex grammars are the algorithm used in this process.

LEVEL 2:

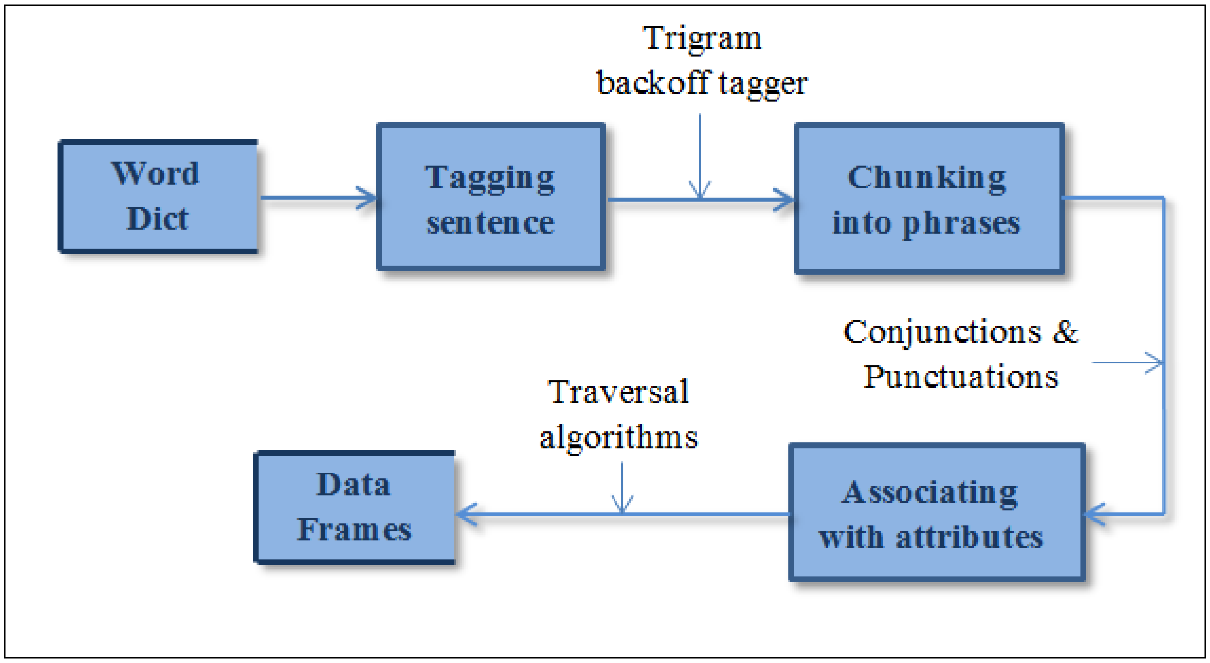


Figure 7.7: Level 2 DFD

The next processing stage of Sentiment Analysis system is the Attribute Phrases Extraction shown in the Figure 7.7. It Associates every phrases in Word Dict with its respective product attributes.

LEVEL 3:

The final processing level is the Opinion Sentiment Prediction as shown in Figure 7.8.
This includes vectorizing and classifying algorithms for the prediction of sentiments associated with product and its attributes.

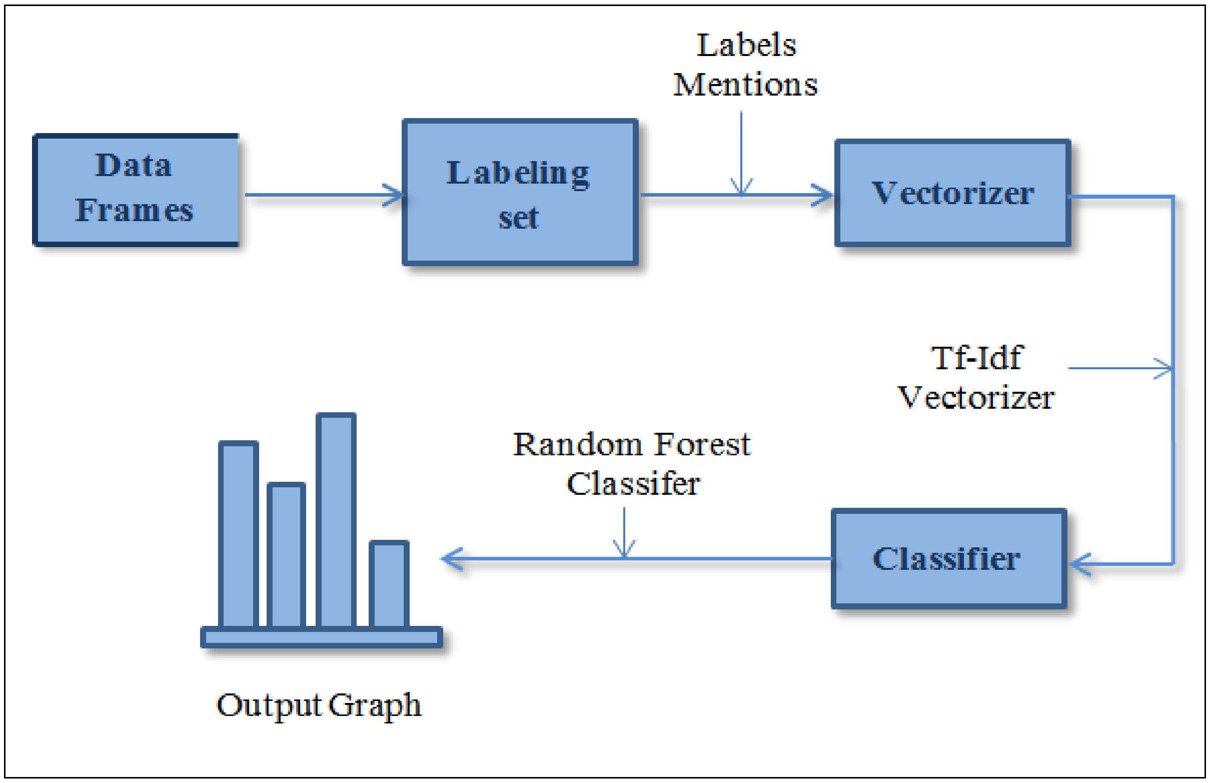


Figure 7.8: Level 3 DFD

7.8 Proposed System Advantages

- The system automatically extracts product attributes from a given set of reviews.
- The user will have a summary of all the reviews listing the pros and cons of the specified product.
- The system provides a graph of overall sentiment of the product.
- The user will be provided with a graph showing the sentiments for each product attribute.
- The system shows the general opinion around a product attribute – positive or negative or neutral also the number of people who feel that way.

CHAPTER 8

IMPLEMENTATION

Given the nature and challenges of our data, the goal of this project was threefold:

1. To automatically extract product attributes from a given set of reviews.
2. To extract the phrases which are associated with each of the product attributes.
3. To perform sentiment analysis on these extracted attributes.

The projected system consists of three modules:

1. Product attributes extraction.
2. Attribute phrase extraction.
3. Opinion sentiment prediction.

8.1 Flowchart

Flowchart is a type of diagram that represents an algorithm, workflow/process. The flowchart of the proposed system is as shown in Figure 8.1. The process starts with the training data set as the input to the classifier. The trained model uses these ground truth values for the sentiment prediction of review data set.

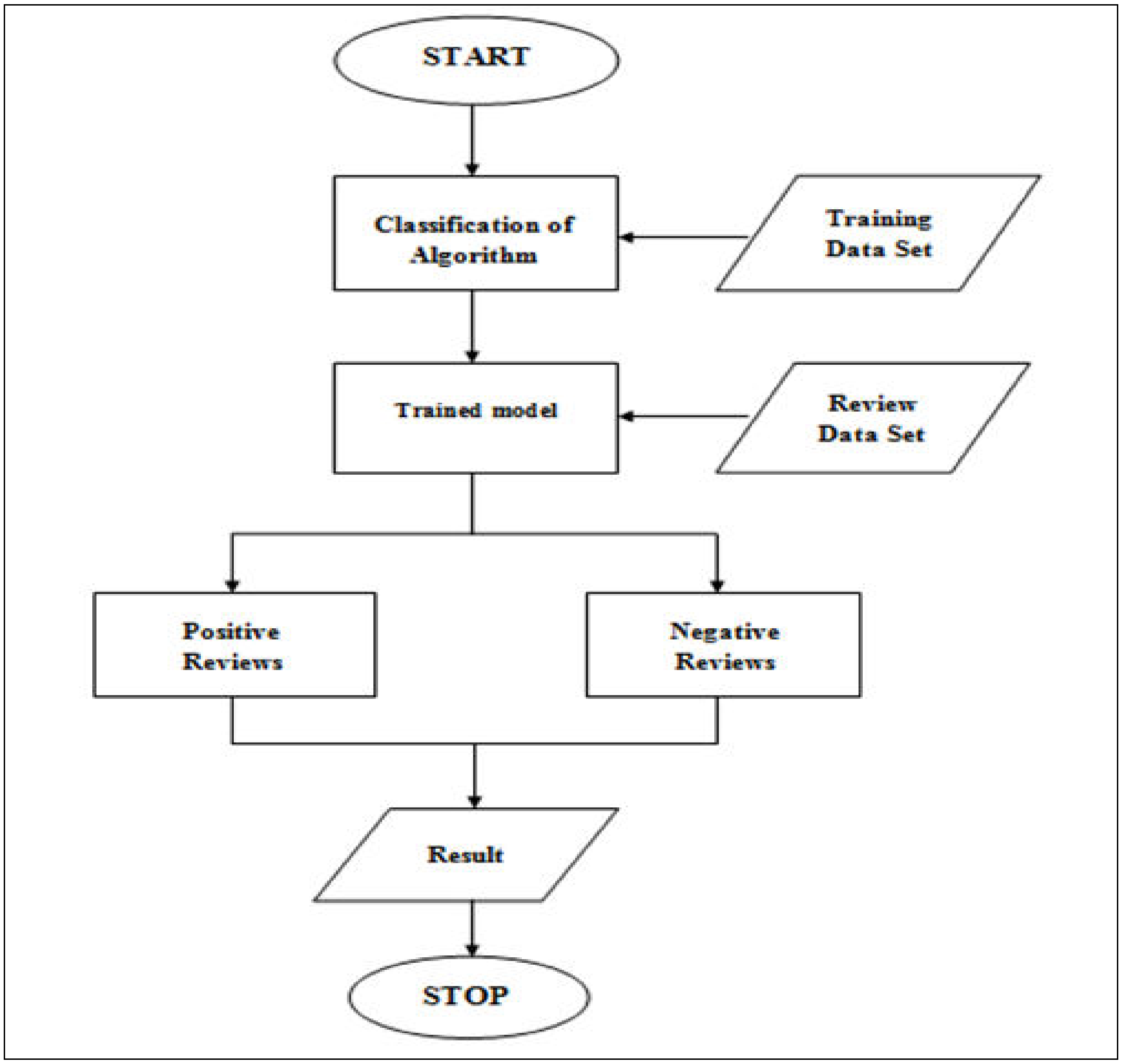


Figure 8.1: Flow chart of the proposed system

8.2 Product Attributes Extraction

The reviews are tokenized into different sentences and tagging individual words init. A manual inspection of different review sentences showed that most of the product attributes were either nouns, adjectives, adverbs or a combination of them. We then extracted only those word phrases which used the patterns mentioned in the following pseudo code and occurred a certain number of time. Regex grammar is used for tokenizing to extract noun phrases.

P1:{<JJ.*|NN.*><NN.*>+} # battery charging system
P2:{<RB|RBR|RBS><JJ.*|NN.*>+} # Shows noun doing action
P3:{<NN.*>+} # battery life, lcd screen

Many of these extracted phrases were synonyms of each other.
In order to club them into similar groups, we used the following 3 properties:

- **Common Words**

  Attribute expressions sharing some common words are likely to belong to the same group. For example, "battery life", "battery", "battery charger" etc.
- **Lexical Similarity**

  Attribute expressions whose words are synonymous in WordNet [11,16,23,33,36] are likely to belong in the same to the same group. For example, "battery" and "charger"
- **Domain Filtering**

  Attribute expressions whose words are related with the application domain or product are likely to be most relevant product attributes. For example, "screen", "internet", "camera" are essential features of a cellphone and hence relevant.

Keeping these constraints in mind, we proceeded with the following:

i. We formed a dictionary (called "word_dict") with words as keys and a list of all extracted attribute phrases containing that word as values.
ii. From these dictionary, we related every word key with every other word key if their corresponding phrase list had noun-based lexical similarity in WordNet above a particular threshold (0.8 in this case).

Following a psuedocode to achieve this:

```
//Calculating pairwise similarity
for word_key1 in word_dict:
for every word_key2 in word_dict:
```

```
sim(word_key1,word_key2)=
Avg(PhraseSim(Phrase1word_dict[word_key1],Phrase2word_dict[word_key2]))
ifsim(word_key1,word_key2)>!0.8:
Club(word_key1,word_key2)
//Sub function for calculating similarity betweenphrases
PhraseSim(Phrase1,Phrase2):
return Max(word1Phrase1,word2Phrase2)
```

iii. We now had a list of tuples of words which were similar to each other due to the phrases in which they were contained. From this list, we wanted to group together all the word tuples which had a common word. Thus, we formed another dictionary (called "topics")with key as the common word and value as set of all the words in the tuples containing that common word.

iv. The dictionary "topics" contained many words unrelated to phone attributes such as "hour", "work", "good", "simple", "expectation", "fine" etc. Thus, we filtered out those words keys which did not have a noun-based lexical similarity with phones in WordNet.

Following is a psuedocode to achieve this:

```
//Calculating similarity with phone
ProductType=['cell','phone','telephone','mobile']
forword_key in topic:
sim(word_key,ProductType)=Avg(WordSim(word_key,ProductProductType))
ifsim(word_key1,word_key2)>!0.7:
Club(word_key1,word_key2)
SubQfunctionfor calculating similarity with phone
WordSim(word1,Product):
returnMax(word1,!Product)
```

8.3 Attribute Phrase Extraction

Initially, we tried to associate the entire sentences to the product attributes they contained, under the assumption that usually customer reviews are written by regular people and are hence, simple in structure - reviews describe one product attribute per sentence. But on analysing the training /development data set, we found that this was not the case. Each sentence in the review were mostly complex, with multiple product attributes and multiple set of descriptors.

We classified the complex sentences into the following five categories:

1. Sentences with a single product attribute, and a single set of descriptors.
   E.g. 'The battery life is very bad."
2. Sentences with multiple product attributes, and a single set of descriptors.
   E.g. "The camera, as well as the processor is very good."
3. Sentences with a single product attribute, and multiple sets of descriptors.
   E.g. The screen is not so good, but it is enough for general usage.
4. Sentences with multiple product attributes, and multiple set of descriptors.
   E.g. "I hate the camera, the screen is also bad, however the battery is good."
5. Sentences with pronouns, and associated set of descriptors.
   E.g. "It has a 13 Megapixel camera. It takes really good pictures."

In order to overcome these challenges, and correctly extract single attribute phrases from the above 5 type of sentences, we tried using different algorithms to check which one provided the best results.

The implementation can be divided into three stages:

1. Tagging the sentences
2. Chunking / Tokenizing the sentences into phrases
3. Associating phrases with the respective product attribute

8.3.1 Tagging the Sentences

We implemented a trigram backoff tagger, which was trained on brown corpus categories of ‘news’, ‘editorial’, and ‘reviews’ since they were similar to our product reviews.

In order to align the tags of some phrases / terms (like "not") to our liking, we added some additional training data sets.By doing so, we were able to appropriately tag the set of adverbs and conjunctions present in the text, which was not being done by either the “nltk.pos_tag” or by the trigram back-off tagger with brown corpus training data set.

8.3.2 Tokenizing the sentences into phrases

To correctly chunk the sentences into single attribute phrases, we use grammar rules to identify the associated descriptors for each product attribute in the sentences, and tokenize them appropriately.

Following are the algorithms which we used:

- **Context Free Grammar Algorithm:**

  This algorithm utilizes "grammar", which specifies which trees can represent thestructure of a given text. These can then be used to find possible syntactic structures forthe sentences.

  An example code for this algorithm is :

```
grammar1 = nltk.CFG.fromstring("""
S -> NP VP
PP -> P NP
V -> "saw" | "ate" | "walked"
NP -> "John" | "Mary" | "Bob" | Det N | Det N PP
N -> "man" | "dog" | "cat" | "telescope" | "park"
P -> "in" | "on" | "by" | "with"
""")
sentence = "Mary saw Bob".split()
rd_parser = nltk.RecursiveDescentParser(grammar1)
for tree in rd_parser.parse(sent):
print(tree)
Output:(S (NP Mary) (VP (V saw) (NP Bob)))
```

- **Using Regex Parsers to tokenize sentences:**

  In this we tried to find patterns in the tags of the phrases, and then define regex patterns to try and chunk them.

  A sample code for this algorithm is

```
P1:{<JJ.*|NN.*><NN.*>+} # battery charging system
P2:{<RB|RBR|RBS><JJ.*|NN.*>+} # Shows noun doing action
P3:{<NN.*>+} # battery life, lcd screen
```

- **Utilizing conjunctions, and other punctuations in the sentences to perform tokenization:**

  In this, we tried using the conjunctions and the punctuations present in the sentences to determine distinct phrases.

  A pseudo-code for this is shown below:

```
def5chunk_Data_On_CC(tagged_review_Data):
for each sentence in tagged sentences:
Split the sentence at each Conjunction
Create a list of sentences
for each item in the list of sentences:
```

```
Split them on punctuations like commas, semiQcolons, and colons
Store the list of phrases in the list
return5Dictionary_of_list_of_phrases_for_all_reviews
```

Of all the methods that we tried, the final method i.e. using conjunctions, and punctuations to
tokenize sentences worked the best on the review data.

8.3.3 Associating phrases with product attributes

The extracted “most talked about" product attributes and split the review sentences in phrases containing a single product attribute, we are left we the job of associating each phrase with the corresponding product attribute it contains. We need to build this association so that the sentiments predicted per phrase can be summed-up for the corresponding product attribute. Phrases not containing the extracted product attributes will be discarded.

To do this, we first create a dataframe for every extracted product attribute. We then use a simple algorithm to traverse through each phrase and find whether it contains a product attribute or its synonyms. If yes, then the phrase is added to corresponding attribute's dataframe. If a phrase contains multiple product attributes then the phrase is added to the dataframes of all the contained attributes.

Following is a psuedocode showing the algorithm:

```
for each phrase in a review sentence:
set1= all the words in that Phrase
for every Product Attribute in Topics: #Topics is a dictionary containing all the product
set2=synonyms for that product attribute #attributes and its synonyms
If there are any common terms in set1 and set2 then:
Associate that Phrase to that Product Attribute
```

8.4 Sentiment Prediction

In order to extract sentiments from the phrases derived in the Product Phrase Extraction and Association step, we decided to use various machine learning techniques. We used vectorizers to extract and learn syntactic patterns accompanying different sentiments in the reviews. We then trained classifiers on manually labelled training dataset to predict the sentiment contained in each of the phrases.

Thus, this step can be separated into three segments:

1. Manually labelling the training set into positive, negative and neutral sentiments
2. Identifying the best vectorizer to extract characteristic features differentiating one sentiment from another
3. Identifying the best classifier which will used these characteristic features to predict sentiments

This involved finding which combination of vectorizer and classifiers works the best for the dataset, without over-fitting to the training dataset.

8.4.1 Labelling Training Set

Using the procedures explained in the Data pre-processing section, we split the Amazon review sentences into single attribute phrases. Then we manually labelled the data, classifying them into ‘Negative mentions’ (represented as -1), Neutral Mentions (represented as 0), and Positive mentions (represented as 1).

8.4.2 Vectorizer

We tried various vectorizers found in Scikit Learn python machine-learning library with different parameters values before finalizing which vectorizer will be the best fit for our dataset.

Following are the vectorizers we considered, along with the variations in parameters we did.

- ***Count Vectorizer***

  This vectorizer converts text 'documents' into a matrix of token counts, specifically, a sparse matrix representation of the counts using the *scipy.sparse.coo_matrix*. We specified the analyzer to 'words', allowed lowercasing of tokens, and also asked the vectorizer to consider Unigrams, Bigrams, and Trigrams as features. We limited the number of features to 1000.

  A sample code of the vectorizer is given below:

```
vectorizer=CountVectorizer5(
analyzer='word',#Assigning analyzer as word
tokenizer=tokenize,
lowercase=True,#Lower casing the words
stop_words='english',#For removing stop words from being considered as features
ngram_range=(1,3)
Allowing unigrams, bigrams, and trigrams to be considered as features
max_features=1000 # Using the top 1000 features)
```

- ***Tf-Idf Vectorizer***

  This vectorizer converts the collection of raw 'documents' into a matrix of TF-IDF features. It is equivalent to applying CountVectorizer on the set of 'documents' and then applying Tf-Idf transformation on it. "Tf–Idf, short for term frequency–inverse document frequency, is a numerical statistic that is intended to reflect how important a word is to a document in a collection or corpus". In this vectorizer as well, we tried different combinations of various parameters, to output the best results.

A sample code for this vectorizer:

```
vectorizer=TfidfVectorizer5(
analyzer='word',
min_df=1,
tokenizer=tokenize,
lowercase=True,
stop_words='english',
ngram_range=(1,3),
max_features=1000)
```

8.4.3 Classifier

To perform sentiment analysis on the phrases associated with different product attributes, in conjunction with the vectorizers, we tried three different classifier algorithms to analyze which combination provided the best precision, and re-call, without over-fitting to the training data set.

Following are the classifiers which we considered.

- ***Logistic Regression Classifier***

  This model is a discriminative model, which can be used to directly estimate the probability of occurrence of y given occurrence of x (p(y|x)). For this model to work correctly there is no restriction on the features to be co-related. It can be used to provide multi-category classification in cases where the categories are exhaustive, and mutually exclusive, i.e. every instance belongs to one, and only one category.

  A sample code for the LR Classifier:

```
log_model=LogisticRegression5()
log_model=log_model.fit5(X=X_train,y=y_train)
y_log_pred=log_model.predict5(X_dev)
```

- ***SVM Classifier***

This is also a discriminative classifier, which is formally defined by a hyper-plane, i.e. provided labelled training data set, this classifier outputs an optimal hyper-plane, which can then we utilized to classify new examples.

A sample code for SVM classifier:

svm_model=**nltk.svm.SVC**!(kernel='linear',C=1.0)
svm_model=svm_model.**fit5**(X=X_train,y=y_train)
y_svm_pred=svm_model.**predict5**(X_dev)

- ***Random Forest Classifier***

A random forest is a meta estimator that fits a number of decision tree classifiers on various sub-samples of the dataset and use averaging to improve the predictive accuracy and control over-fitting. The sub-sample size is always the same as the original input sample size but the samples are drawn with replacement if bootstrap=True (default).

A sample code for RF classifier:

rf_model=**RandomForestClassifier**(n_estimators=1000)
rf_model=rf_model.**fit5**(X=X_train,y=y_train)
y_rf_pred=rf_model.**predict5**(X_dev)

CHAPTER 9

PSEUDOCODE

9.1 Introduction

Pseudocode is an informal high-level description of the operating principle of a computer program or other algorithm.

It uses the structural conventions of a normal programming language, but is intended for human reading rather than machine reading. Pseudocode typically omits details that are essential for machine understanding of the algorithm, such as variable declarations, system-specific code and some subroutines. The programming language is augmented with natural language description details, where convenient, or with compact mathematical notation. The purpose of using psuedocode is that it is easier for people to understand than conventional programming language code, and that it is an efficient and environment-independent description of the key principles of an algorithm. It is commonly used in textbooks and scientific publications that are documenting various algorithms, and also in planning of computer program development, for sketching out the structure of the program before the actual coding takes place.

No standard for pseudocode syntax exists, as a program in pseudocode is not an executable program. Pseudocode resembles, but should not be confused with skeleton programs which can be compiled without errors. Flowcharts, drakon-charts and Unified Modeling Language (UML) charts can be thought of as a graphical alternative to pseudocode, but are more spacious on paper.

9.2 Stemming, tokenizing and vectorizing the features

```
stemmer = pstemmer
defstem_tokens(tokens, stemmer):
stemmed = []
```

```
for item in tokens:
stemmed.append(stemmer.stem(item))
return stemmed

def tokenize(text):
    # remove non letters
text = re.sub("[^a-zA-Z]", " ", text)
    # tokenize
tokens = nltk.word_tokenize(text)
    # stem
stems = stem_tokens(tokens, stemmer)
return stems
```

9.3 Training the classifiers on the training set

```
log_model = LogisticRegression()
rf_model = RandomForestClassifier(n_estimators=1000)
svm_model = SVC(kernel='linear', C = 1.0)
log_model = log_model.fit(X=X_train, y=y_train)
rf_model = rf_model.fit(X=X_train, y=y_train)
svm_model = svm_model.fit(X=X_train, y=y_train)

# Performing predictions on the dev set
y_log_pred = log_model.predict(X_dev)
y_rf_pred = rf_model.predict(X_dev)
y_svm_pred = svm_model.predict(X_dev)
```

- **Classifier report for the Logistic Regression model**

```
fromsklearn.metrics import classification_report

print(classification_report(y_dev, y_log_pred))
```

- **Classifier report for the Random Forest Model**

```
print(classification_report(y_dev, y_rf_pred))
```

- **Classifier report for the SVM Model**

```
print(classification_report(y_dev, y_svm_pred))
```

9.4 Logistic Regression Model

- **Training the classifier on the whole training data available**

```
log_model = LogisticRegression()
log_model = log_model.fit(X=corpus_data_features_nd, y=train_data_df.Sentiment)
```

- **Creating method to get predictions for the test data**

```
defget_Predictions_LR(df_dict, vectorizer):
predictions_dict = { }
for feature in df_dict:
test_data_df = df_dict[feature]
corpus_data_features_test_vector = vectorizer.transform(test_data_df.Text.tolist())
corpus_data_features_test = corpus_data_features_test_vector.toarray()
print (feature)
test_log_pred = log_model.predict(corpus_data_features_test)
predictions_dict[feature] = test_log_pred
returnpredictions_dict
```

9.5 Random Forest Model

- **Training the classifier on the whole training data available**

```
rf_model = RandomForestClassifier(n_estimators=1000)
rf_model = rf_model.fit(X=corpus_data_features_nd, y=train_data_df.Sentiment)
```

- **Creating method to get predictions for the test data**

```
defget_Predictions_RF(df_dict, vectorizer):
predictions_dict = { }
for feature in df_dict:
test_data_df = df_dict[feature]
corpus_data_features_test_vector = vectorizer.transform(test_data_df.Text.tolist())
corpus_data_features_test = corpus_data_features_test_vector.toarray()
test_log_pred = rf_model.predict(corpus_data_features_test)
predictions_dict[feature] = test_log_pred
returnpredictions_dict
```

CHAPTER 10

TESTING

The purpose of testing is to discover errors, every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub-assemblies and/or finished products. It is the process of executing software with intent of ensuring that the software system meets its requirements and user exceptions and does not fail in an unacceptable manner.

10.1 Attribute Extraction

In order to test our algorithm's precision, we manually inspected all the reviews in test data and drew out the most common product attributes being talked about. We considered these as baseline attributes as they were common attributes across reviews of other cellphones as well. We then drew out a list of attributes given by our proposed algorithm and compared them as shown in the Table 10.1.

Manual Inspection	Iphone 5s (test data)	Galaxy S5	Nexus 5
battery	battery	battery	battery
screen	display	monitor, display	display
charger	charger	charger	
speakers	headphone		
camera	camera	camera	camera
keyboard	button, home	button, home	home
software	system	system	
size	light ,lighter		light ,lighter
connection	connection	wireless	
hardware	hardware	resistance, sensor, port, scanner	processor, hardware
body	back	back	
price			
experience			
speed			
	fan		fan

Table 10.1: Expected Features v/s Actual Features obtained for various products

Our algorithm does a pretty good job of identifying the various product attributes with no manual intervention. The precision is about 78.57% on test data set and 64.28% on Galaxy S5 data set. It has surprisingly good precision on Nexus data set which has very few reviews (226 in total).

This is comparable to the baseline precision range of 69-75% given by "Hu and Liu's" work2. The comparatively low precision in Galaxy S5 and Nexus 5 can be due to the fact that our algorithm was unable to capture indirect/derived product attributes such as "price", "speed", "user experience" etc. This is because in our algorithm, the product attributes are purely derived form the words present in the reviews and any other word is not used. So in the case of "speed", reviews will contain words like "fast", "faster" or "slow" and in the case of "price", reviews will contain words like "cheap", "expensive", "rip off" etc. Because the algorithm is not able to club them under one group, the frequency of such words (like "cheap", "expensive", "rip off") is not high enough to be captured.

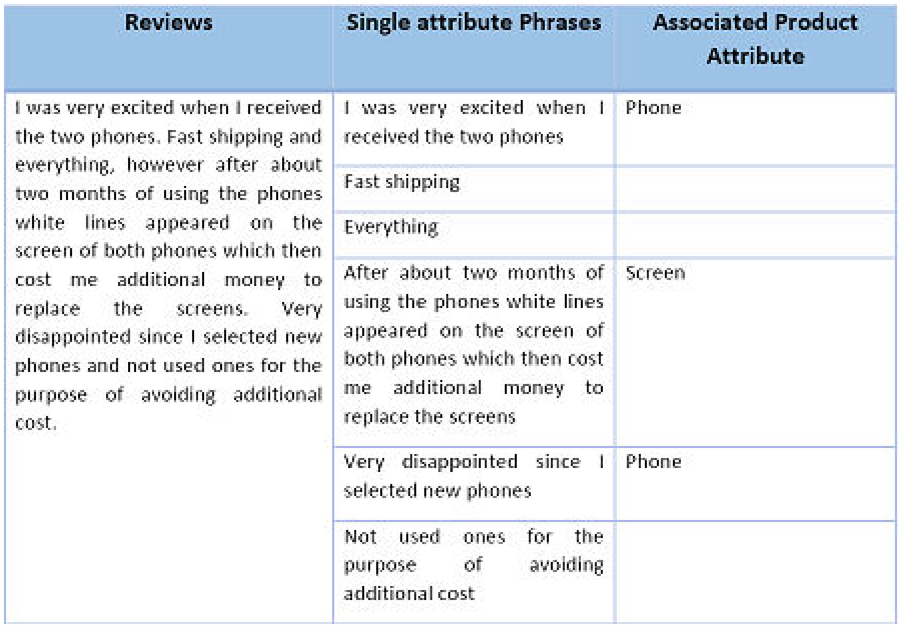
10.2 Attribute Phrase Extraction

The method to split sentences on conjunctions and punctuations, though simple, gave surprisingly good results on the product reviews. Most of the phrases contain a single attribute with a single set of descriptors. In other words, the resulting phrases are complete in themselves and sufficient to extract sentiments for each of the product attribute they contain as shown in Table 10.2.

10.3 Opinion Sentiment Predictions

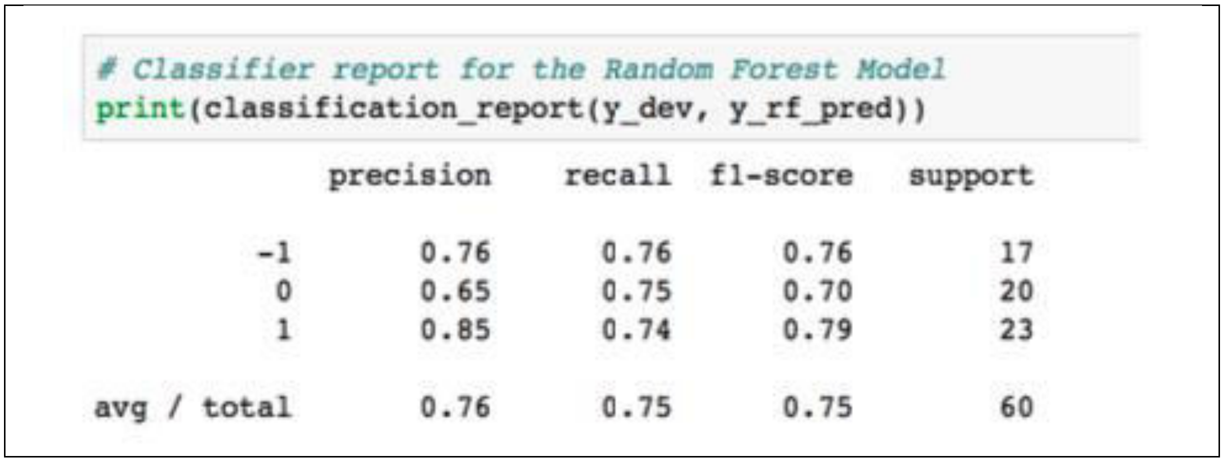
On the training dataset, the combination of Count Vectorizer and Logistic Regression Classifier gave a precision of 81%. However, running this combination on the test dataset made us realize that Count Vectorizer and Logistic Regression Classifier were over-fitting on the training data set and not producing as good results as expected.

On the other hand, though the combination of Tf-Idf Vectorizer and Random Forest Classifier was giving only 76% precision on the training data set, it was giving much better results on the test dataset.



Reviews	Single attribute Phrases	Associated Product Attribute
I was very excited when I received the two phones. Fast shipping and everything, however after about two months of using the phones white lines appeared on the screen of both phones which then cost me additional money to replace the screens. Very disappointed since I selected new phones and not used ones for the purpose of avoiding additional cost.	I was very excited when I received the two phones	Phone
	Fast shipping	
	Everything	
	After about two months of using the phones white lines appeared on the screen of both phones which then cost me additional money to replace the screens	Screen
	Very disappointed since I selected new phones	Phone
	Not used ones for the purpose of avoiding additional cost	

Table 10.2: Original reviews and the phrases extracted for product attributes



```
# Classifier report for the Random Forest Model
print(classification_report(y_dev, y_rf_pred))
```

```
             precision    recall  f1-score   support

         -1       0.76      0.76      0.76        17
          0       0.65      0.75      0.70        20
          1       0.85      0.74      0.79        23

avg / total       0.76      0.75      0.75        60
```

Table 10.3: Precision and recall for the final combination of Tf-Idf Vectorizer and Random Forest Classifier

Hence, we decided to trade-off on lower accuracy on training set for better results on the test data set and finalized on using Tf-Idf Vectorizer in combination with Random Forest Classifier to predict sentiments. The precision and recall for Tf-Idf Vectorizer and Random Forest Classifier combination is shown in Table 10.3.

CHAPTER 11

RESULTS ANALYSIS

We ran our algorithm on three datasets: Training/Dev dataset of I-Phone 5s (1141 reviews), large test dataset of Samsung Galaxy 5s (2880 reviews) and a smaller dataset of LG Nexus 5 (226 reviews).

Our Training/Dev dataset was manually tagged for sentiment analysis. Hence, we were able to measure the precision and recall for this dataset. On an average, the precision obtained was 0.76, with precision being as high as 0.85 for positive sentiments and as low as 0.65 for neutral sentiments. The recall obtained was on an average 0.75, with little variance across all the sentiments. The recall obtained was very consistent. The overall f-1 score was 0.75.

In terms of product attribute extraction, while tagging the training data manually, we made a list of attributes we were expecting from the algorithm. The extracted attributes for all three data sets are as shown below in the Table 11.1.

Manual Inspection	Iphone 5s (test data)	Galaxy S5	Nexus 5
battery	battery	battery	battery
screen	display	monitor, display	display
charger	charger	charger	
speakers	headphone		
camera	camera	camera	camera
keyboard	button, home	button, home	home
software	system	system	
size	light ,lighter		light ,lighter
connection	connection	wireless	
hardware	hardware	resistance, sensor, port, scanner	processor, hardware
body	back	back	
price			
experience			
speed			
	fan		fan

Table 11.1: The extracted attributes of the data sets

Some of the expected attributes were not obtained in the test data. But on closer inspection we found that this was because these attributes were not frequently talked about in the reviews for the test data.

For I-Phone 5s (training data) the overall results were as shown below in the Figure 11.1. As seen in the graphs, neutral mentions are included while talking about individual product attributes but not when talking about the product overall. This is because, most of the neutral mentions about the overall product were not sentiments, but general statements that do not convey sentiments like "I gifted this phone to my wife", "I travel a lot with this phone" etc. Hence we removed the neutral sentiments from the final results for the phone as a whole. Overall, 441 users were happy with the device, while 335 were not as happy, conveying that overall users are not as happy as one would expect. Battery and Charger were the biggest pain points for I-Phone 5s. The camera, the weight and the system was a plus for it.

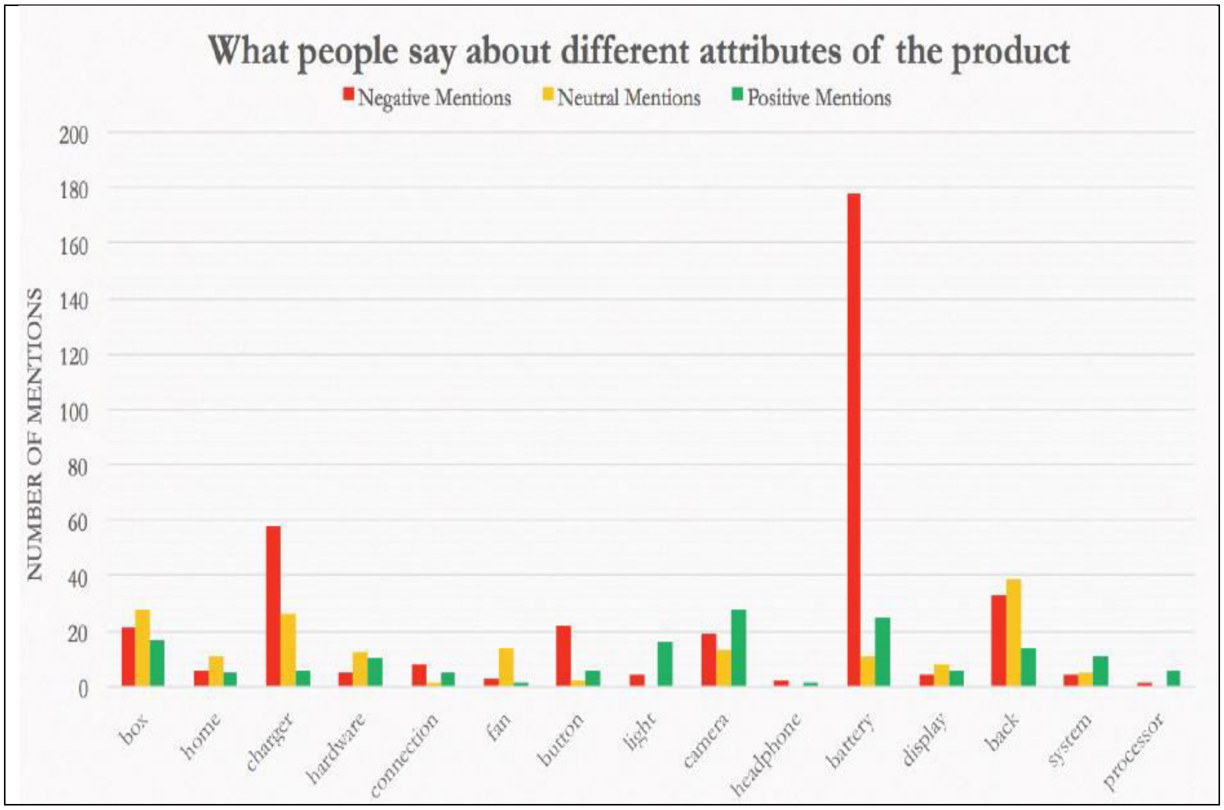


Figure 11.1: Sentiments for each product attribute for IPhone 5s

The results were pretty good in both the test data sets. Samsung Galaxy S5 was evaluated with 2880 reviews and the graph is shown in Figure 11.2. Overall 1937 positive mentions were obtained with 1005 negative mentions, pointing to overall a better sentiment from the users towards the phone. Charger and Battery were again big pain points along with one of its ports getting negative mentions. The camera and the display are being the positive attributes of the phone.

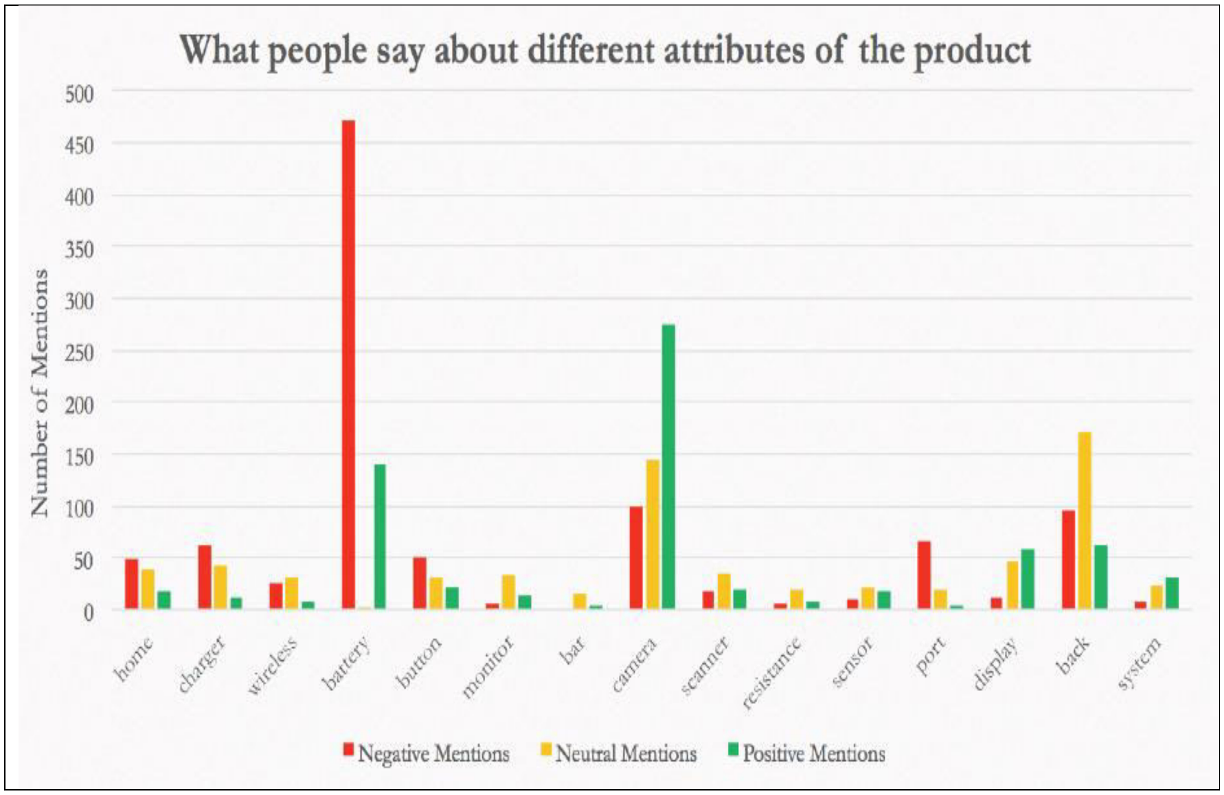


Figure 11.2: Sentiments for each product attribute for Samsung Galaxy S5

The results on a test dataset of LG Nexus 5 with relatively less reviews (226 reviews) were also very good. Overall, there were 189 positive mentions along with 99 negative mentions, indicating an overall positive sentiment. Battery and Charger again are getting negative reviews (Smartphones). However, LG Nexus got positive reviews in many categories like camera, weight, display, processor and hardware as shown in Figure 11.3.

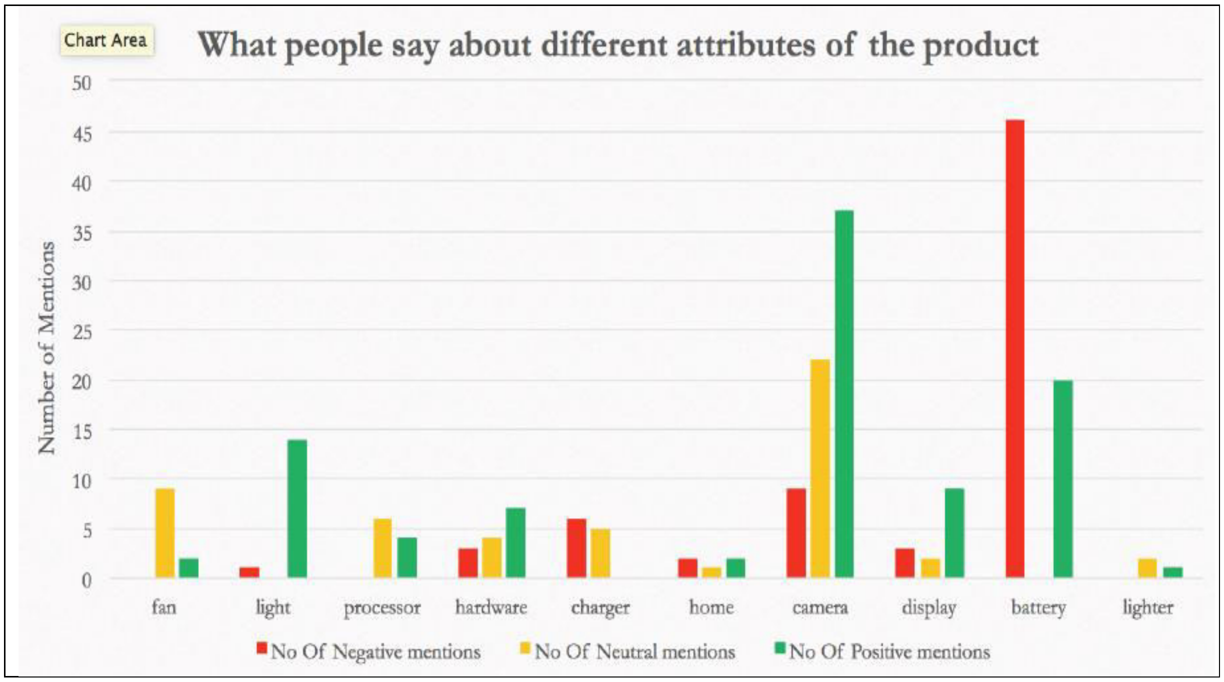
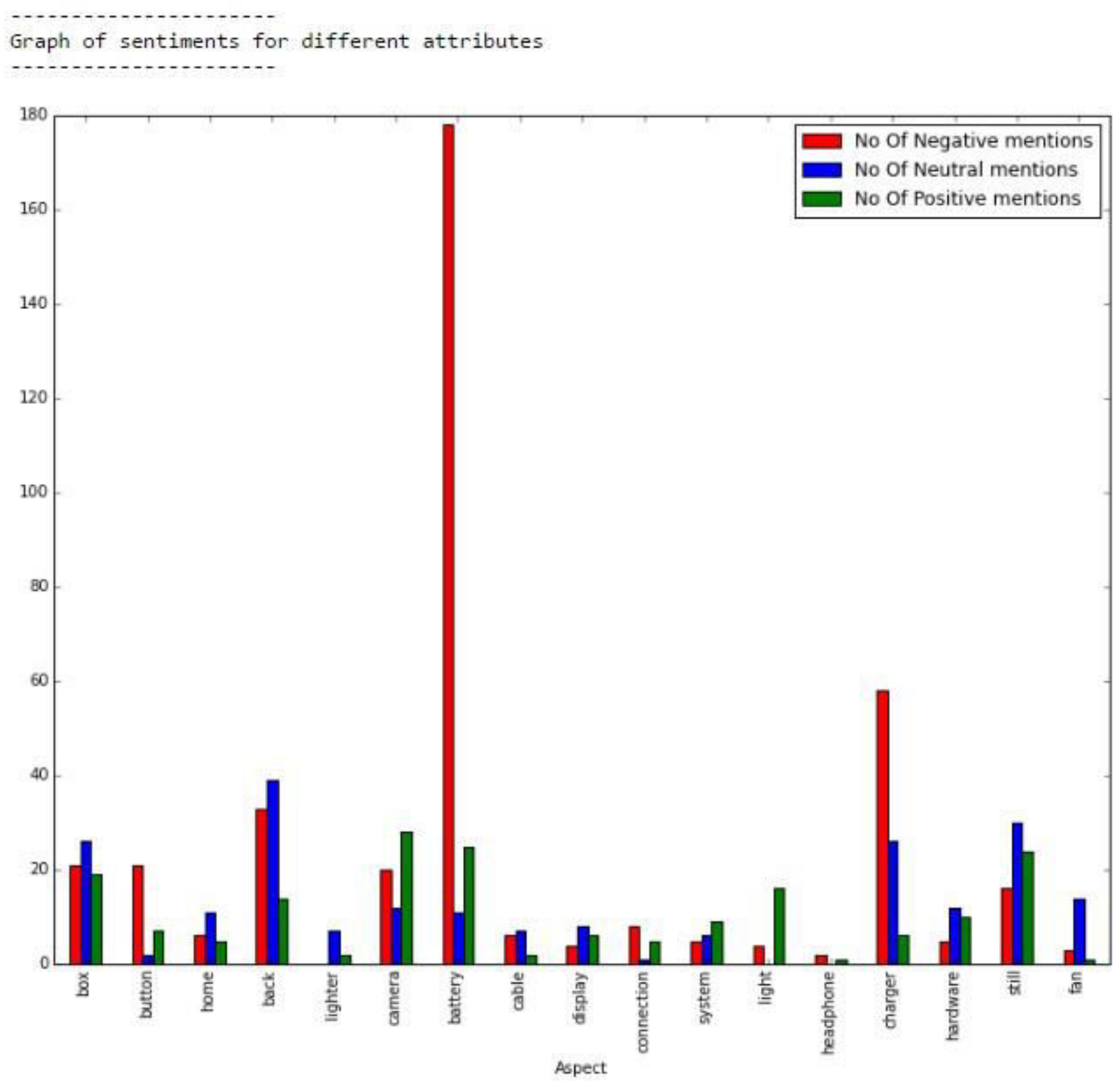


Figure 11.3: Sentiments for each product attribute for Nexus 5

CHAPTER 12

SCREENSHOTS

12.1 Screenshot 1:

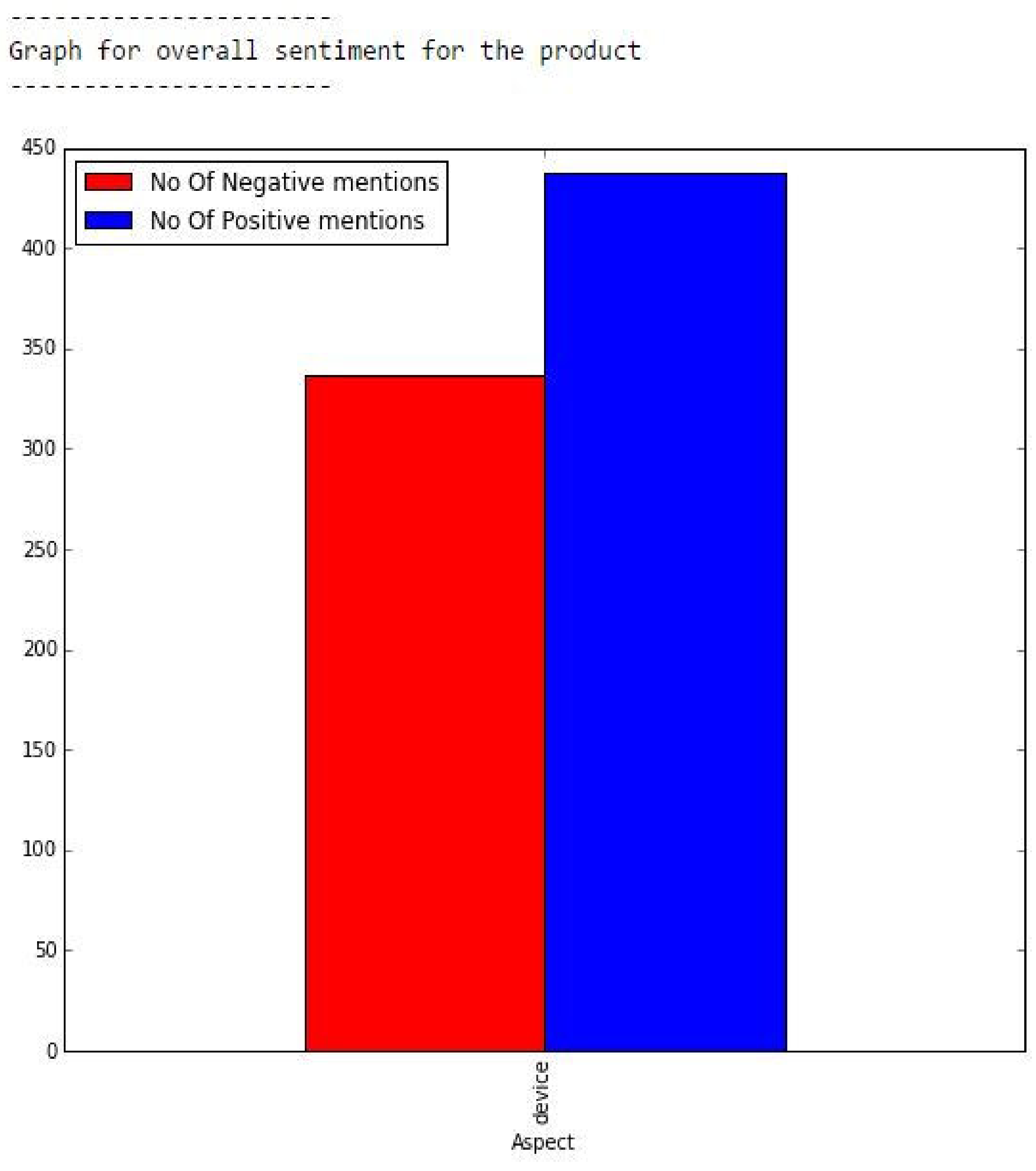


```
----------------------
Graph of sentiments for different attributes
----------------------
```

Screenshot 12.1: Sentiments for each product attribute for IPhone 5s

Screenshot 12.1 shows the results on a dataset of IPhone 5s. The different bars in the graph represent the sentiment of each aspects of the product.

12.2 Screenshot 2:

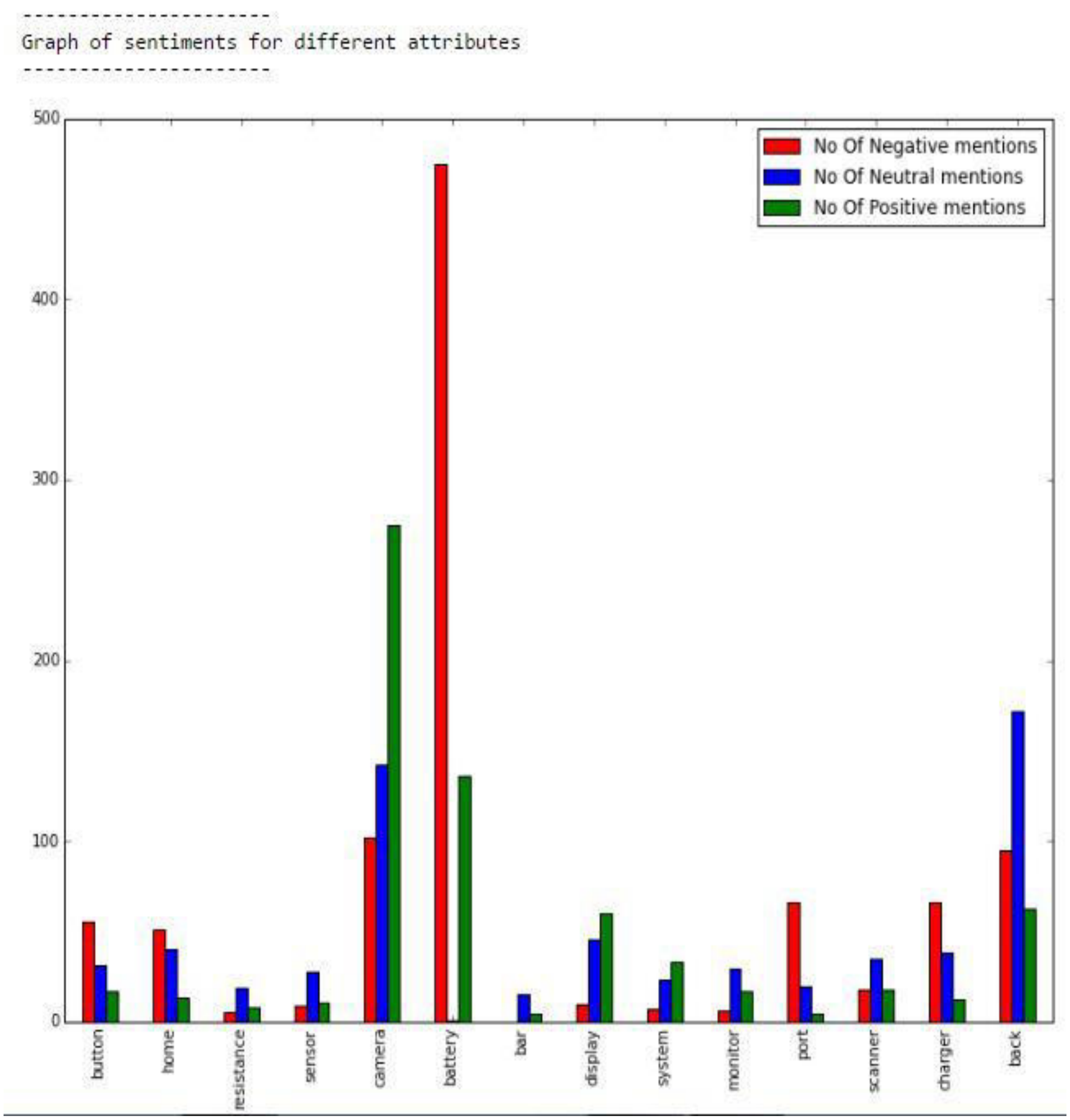


```
----------------------
Graph for overall sentiment for the product
----------------------
```

Screenshot 12.2: Overall sentiments for IPhone 5S

The overall sentiment of IPhone 5s is shown in screenshot 12.2. The graph includes the positive and negative mentions of the IPhone 5s.

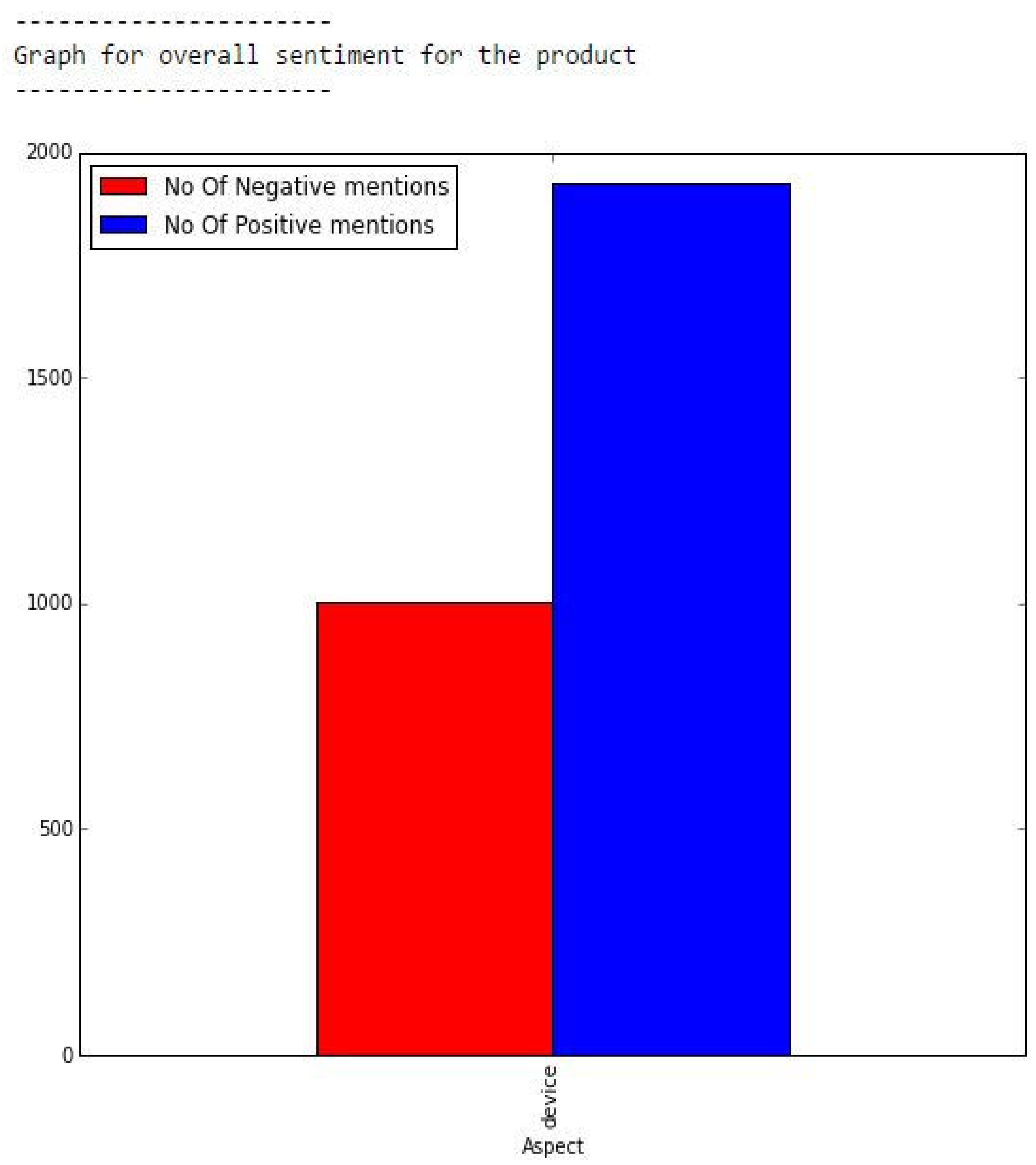
12.3 Screenshot 3:


```
----------------------
Graph of sentiments for different attributes
----------------------
```

Screenshot 12.3: Sentiments for each product attribute for Samsung Galaxy S5

Screenshot 12.3 shows the results on a training dataset of Samsung Galaxy S5 The different bars in the graph represent the sentiment of each aspects of the product.

12.4 Screenshot 4:

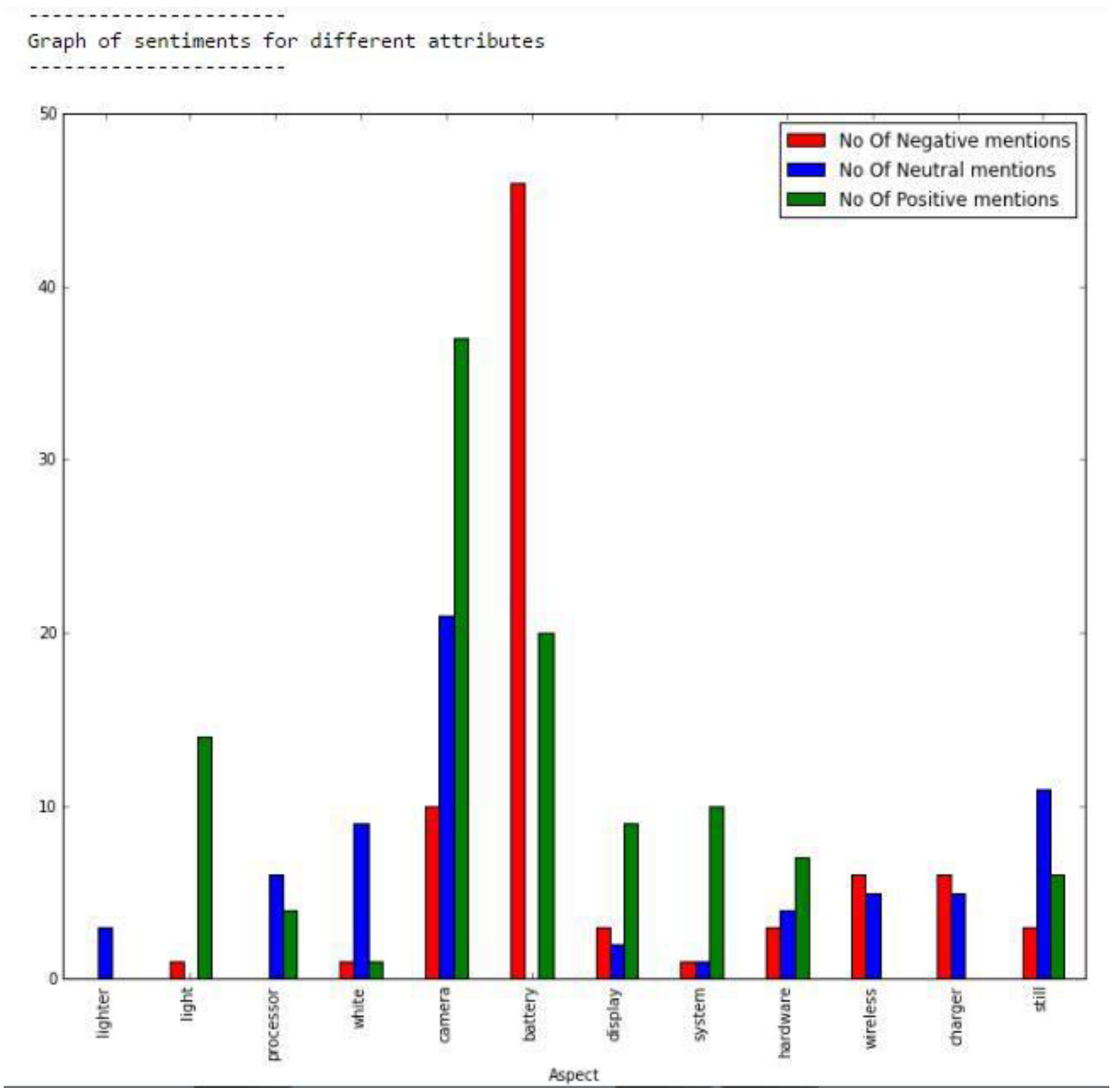


```
----------------------
Graph for overall sentiment for the product
----------------------
```

Screenshot 12.4: Overall sentiments for Samsung Galaxy S5

The overall sentiment of Samsung Galaxy S5 is shown in screenshot 12.4. The graph includes the positive and negative mentions of the Samsung Galaxy S5.

12.5 Screenshot 5:

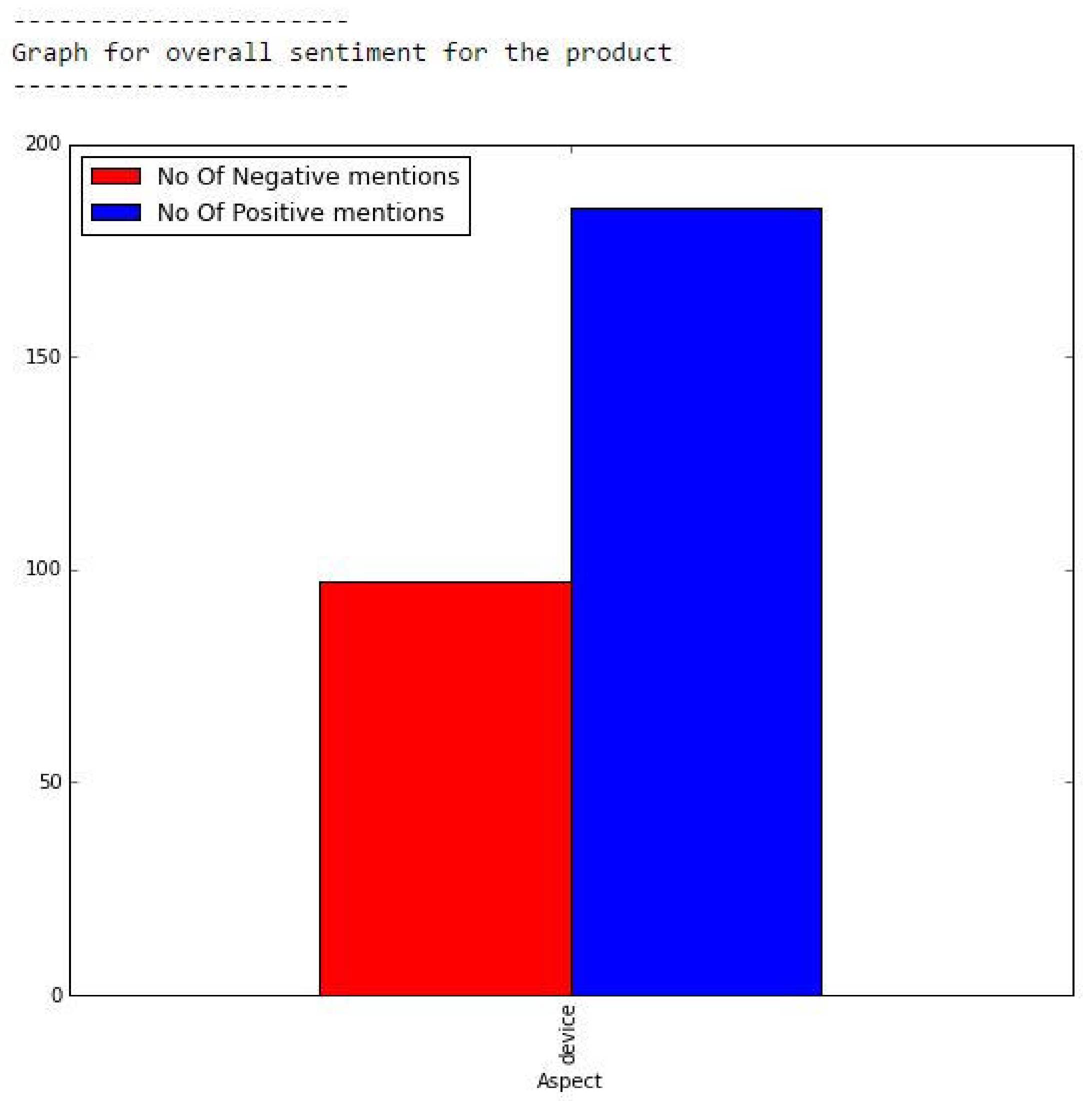


```
----------------------
Graph of sentiments for different attributes
----------------------
```

Screenshot 12.5: Sentiments for each product attribute for LG Nexus 5

Screenshot 12.5 shows the results on a training dataset of LG Nexus 5
The different bars in the graph represent the sentiment of each aspects of the product.

12.6 Screenshot 6:



```
----------------------
Graph for overall sentiment for the product
----------------------
```

Screenshot 12.6: Overall sentiments for LG Nexus 5

The overall sentiment of LG Nexus 5 is shown in screenshot 12.6. The graph includes the positive and negative mentions of the Samsung LG Nexus 5.

CONCLUSION

The algorithm worked well within the product category of Cell phones. The overall precision and accuracy obtained was good, both in terms of feature extraction as well as sentiment analysis.

Every part of the algorithm was automated, and hence the results were obtained with minimum human intervention. The algorithm worked well even when the numbers of reviews were less. The entire algorithm was written in existing libraries and packages within NLTK and hence is affected by the performance of these libraries and packages.

Applications

- For a recommender system, sentiment analysis has been proven to be a valuable technique. A recommender system aims to predict the preference to an item of a target user. Mainstream recommender systems work on explicit data set. For example, collaborative filtering works on the rating matrix, and content-based filtering works on the meta-data of the items.
- In many social networking services or E-commerce websites, users can provide text review, comment or feedback to the items. These user-generated text provide a rich source of user's sentiment opinions about numerous products and items. Potentially, for an item, such text can reveal both the related feature/aspects of the item and the users' sentiments on each feature. The item's feature/aspects described in the text play the same role with the meta-data in content-based filtering, but the former are more valuable for the recommender system. Since these features are broadly mentioned by users in their reviews, they can be seen as the most crucial features that can significantly influence the user's experience on the item, while the meta-data of the item (usually provided by the producers instead of consumers) may ignore features that are concerned by the users. For different items with common features, a user may give different sentiments. Also, a feature of the same item may receive different sentiments from different users. Users' sentiments on the features can be regarded as a multi-dimensional rating score, reflecting their preference on the items.

Future work

The overall results for the project were very good, but there is always room for improvement.

We were able to extract very good features by selecting features that were lexically more similar to the product category ("cellphones") with the help of WordNet. WordNet however, sometimes does not give an accurate measure of how close words are to each other. Substituting it with another resource that can establish a more accurate lexical similarity can maybe provide better results in terms of feature extraction. Creating ontology for products and its attributes is another way of achieving a higher accuracy.

The words are grouped together based on their similarity to each other and clubbed together as one product attribute, and then the attributes are filtered based on their similarity to the actual product category. This process is performance intensive, and hence there is a scope for improvement in terms of performance. Substituting WordNet with some other resource, creating a manual ontology, manually intervening to club potential product attributes into final product attributes etc. are some of the ways in which this can be achieved.

BIBLIOGRAPHY

[1] Abinaya.R, Aishwaryaa.P, Baavana.S, ThamaraiSelvi.N.D, Automatic Sentiment Analysis of User Reviews, 2016.

[2] Erik Cambria, SoujanyaPoria, Rajiv Bajpai, Bj¨ornSchuller, SenticNet 4: A Semantic Resource for Sentiment Analysis Based on Conceptual Primitives, 2016.

[3] ApoorvAgarwal, Vivek Sharma, GeethaSikka, RenuDhir, Opinion Mining of News Headlines using SentiWordNet, 2016.

[4] Xing Fang, Justin Zhan, Sentiment Analysis using Product review data, 2015.

[5] A.Jeyapriya and C.S.KanimozhiSelvi, Extracting Aspects and Mining Opinions in Product Reviews using Supervised Learning Algorithm, 2015.

[6] Alexandra Cornian, ValentinSgarcian, Bogdan Martin, Sentiment analysis from product reviews using SentiWordNet as lexical resource, 2015.

[7] Kang Liu, LihengXu, and Jun Zhao, Co-Extracting Opinion Targets and Opinion Words from Online Reviews Based on the Word Alignment Model, 2015.

[8] AnkitaSrivastava, Dr. M.P.Singh, Prabhat Kumar, Supervised Sentiment Analysis of Product Reviews using K-NN Classifier, 2014.

[9] MalharAnjaria, Ram Mohan Reddy Guddeti, Influence factor based opinion mining of Twitter data using supervised learning, 2014.

[10] S.J.Veeraselvi, C.Saranya, Semantic orientation approach for sentiment classification,

2014.

[11] Zheng-Jun Zha, Jianxing Yu, Meng Wang and Tat-Seng Chua, Product Aspect Ranking and Its Applications, 2013.

[12] BasantAgarwal, VijayKumar Sharma, Namita Mittal, Sentiment classification of review documents using phrase patterns, 2013.

[13] Bing Liu, Sentiment Analysis and Opinion mining, 2012.

[14] Sadhasivam, Kanimozhi SC, TamilarasiAngamuthu, Mining Rare Itemset with automated support threshold, 2011.

[15] David Garcia and Frank Schweitzer, Emotions in product reviews – Empirics and models, 2011.

[16] Alexander Hogenbom, Paul Iterson, Bas Herschoop, FlaviousFrasincas, UzayKaymak, Determining negation scope and strength in sentiment analysis, 2011.

[17] JantimaPolpinij and Aditya K. Ghose,An Ontology-based Sentiment Classification Methodology for Online Consumer Reviews, 2008.

[18] S. B. Kotsiantis, Supervised Machine Learning: A Review of Classification Techniques,2007.

[19] Kanayama, Hiroshi, Tetsuya Nasukawa, Fully automatic lexicon expansion for domain-oriented sentiment analysis, 2006.

APPENDIX

- M Sowmya, Manasa B A, Puneeth, P Sathish Kumar and Prof. Mangala C N exhibited this project with the title **“Sentiment Analysis on Customer Reviews using Machine Learning” in** the Project Exhibition-2017 organized by East West Institute of Technology, Bangalore, held on 27th April 2017.