Aspect-based Opinion Mining from Product Reviews

by

Deepthi Mave (MT2013045)

A thesis submitted

in partial fulfilment of the

requirements for the degree of

Master of Technology

in

Information Technology

International Institute of Information Technology, Bangalore.

June 2015

Thesis Certificate

This is to certify that the thesis titled **Aspect-based Opinion Mining from Product Reviews** submitted to the International Institute of Information Technology, Bangalore, for the award of the degree of **Master of Technology** is a bona fide record of the research work done by **Deepthi Mave (MT2013045)** under my supervision. The contents of this thesis, in full or in parts, have not been submitted to any other Institute or University for the award of any degree or diploma.

_____

Prof.G.Srinivasaraghavan

IIIT-Bangalore,

The 15th of June, 2015.

Abstract

The aim of this thesis work is to develop an automated system that processes a large dataset of product reviews for aspect-based opinion mining and summarization. Natural Language Processing techniques and lexicon-based approaches for opinion mining are used to extract the aspects and customer opinion. The aspect extractor generates the relevant aspect-opinion pairs from a large set of reviews. The information extracted is then classified as positive or negative. By using the labels in the user query, the system aggregates the opinion for each aspect along with the polarity for the products relevant to the user query. The process is directed by the user provided labels to retrieve the relevant product. A set of product reviews is data mined from online sources. The opinion miner-review retriever system developed extracts and evaluates opinion from these reviews.

Acknowledgements

" I would like to express my appreciation to all those who helped me during the course of this thesis work. My deepest thanks to my advisor, ***Prof. G.Srinivasaraghavan*** *for his continuous encouragement and valuable suggestions. It was my pleasure to work under his guidance. I dedicate my work to my family and friends who have always been there to support and cheer me up throughout my ups and downs."*

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Online user reviews are increasingly becoming important for measuring the quality of different products and services. The sheer amount of reviews available in review sites, blogs, news articles discussion forums and social media make it difficult to track and understand customer sentiments. Sentiment analysis or Opinion mining involves studying and building system that collects data from these sources and examines the opinions.

1.1 Background

With the rapid growth in the number of electronic documents and the huge amount of information on the Web, individuals, business entities and other organizations are seeking new ways to extract and utilise public opinion. The sentiments in reviews, online forums etc., about products, services, events are useful in decision making for individual customers, manufacturers or organizations[15]. The customer is exposed to excess information and extracting relevant information has become a challenge. With

the growing importance of sentiment analysis, a considerable amount of research is done in the field.

Sentiment analysis is different from the traditional text mining process. The traditional text mining focuses on analysis of facts from unstructured text, whereas sentiment analysis deals with attitudes of people. The problem is to find the relevant information and to get an overall opinion of what people think about an entity (product, hotel, etc.). But there is no easy method to obtain these. Simply using the ratings given is not enough to get a description of what people think. Manually analysing a large amount of data is a daunting task.

The process of mining opinions would ease this task by providing a summary of customer reviews in terms of aspects of a product or service indicating whether it is positive or negative. In this thesis a system is developed using Natural Language Processing techniques, information extraction and sentiment lexicons generated by the data corpus to extract opinions on product aspects and summarize them. This provides user access to the sentiments expressed in hundreds of reviews in a concise and useful manner.

1.2 Objective

This thesis work deals with aspect-based opinion mining, where the aspects of products are the units under consideration as opposed to sentences and documents. The objective of this thesis is to process a large set of online reviews about products and address the problems in opinion mining outlined below:

- Identifying and extracting features/aspects of the product from the reviews

along with the opinion expressed on each of these aspects. The relevant product features and opinions are extracted from a large set of customer reviews.

- Evaluating the customer opinions on the extracted features/aspects. The opinions for the aspects is identified from reviews and polarity classification of the opinion is performed.

- Summarizing the customer opinions on extracted aspects.

1.3 System Outline

This thesis work focuses on analysing a large corpus of online reviews about products. Since the product search functionality should be faster the opinion mining task itself is not done in real time. NLTK is used to preprocess and POS tag the data. An Aspect-extractor module extracts aspects that the reviewers have commented on. The opinions about the aspects are also identified. The aspect-opinion pairs are then generated.

The opinions are classified as expressing negative or positive sentiment about the aspect. Pruning of the aspects is performed to remove the aspects that are not useful. The opinion about aspects are summarized for each products. The system processes a user query and provides the aspect based summary for products that are relevant to the give query. Review retrieval is done using inverted index method.

Chapter 2

Opinion Mining

This chapter will throw light on area of opinion mining. It presents an overview on the past works, ongoing research, the principal approaches used in opinion mining and aspect-based opinion mining in particular.

Opinion mining or Sentiment analysis is the computational study of peoples opinions, appraisals, attitudes, and emotions toward entities, individuals, issues, events, topics and their attributes [3]. It extracts the subjective information from the sources using Natural Language Processing techniques and text analytics. The process of sentiment analysis can be viewed broadly as in Fig. 2.1

[?]

2.1 Challenges

Opinion mining is a type of Natural Language Processing and has several challenges.

- an opinion word can imply different opinions based on the context.

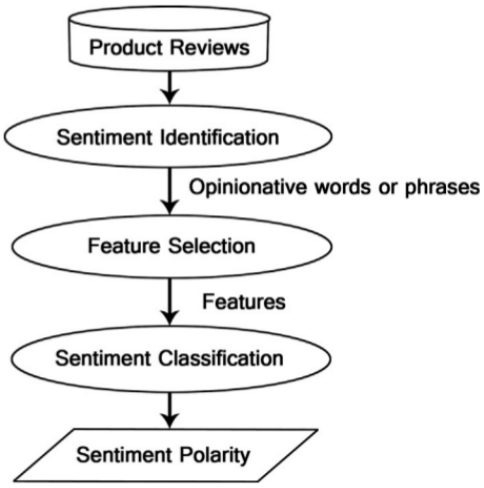


Figure 2.1: Sentiment Analysis process on product reviews.

- people dont always express opinions the same way. The human accuracy for sentiment analysis is 80%, apparently people agree on sentiments only 80% of the time.

- handling implied sentiments, an objective sentence may also imply opinion about the entity. For example, The laptop crashes every time I open a web browser. The sentence expresses a negative sentiment about laptop even though it is not a subjective sentence.

- handling of negation. Negation words reverse the polarity of opinion words. Also, if the sentence has context shifters like but, however etc. or modifiers the implied sentiment changes.

2.2 Levels in sentiment analysis

Opinion mining can be generally classified into 3 levels based on the granularity:

- Document Level
- Sentence Level
- Aspect Level

There exists a good amount of research work and literature for document level opinion mining. In document level sentiment analysis, overall sentiment of the document is determined as either positive or negative. The input can be reviews, blog posts, articles, tweets etc. In sentence level, each sentence of the document is classified subjective or objective. The polarity classification of the sentence is done as expressing positive or negative sentiment.

Even though document level and sentence level opinion mining are useful in many applications, the overall sentiment of a topic or entity may not reflect the sentiment about a sub-topic or aspect/feature of the entity. For example,

I bought a canon camera today. The picture quality is really good. The lenses are not that great in this model. Battery life is manageable. It is worth the price.

In document level opinion mining the above review may be classified as positive. But the reviewer talks about different features of the camera and expresses different opinions about each. The entire information is not captured by document - level or sentence - level opinion mining methods. To a consumer these detailed information may be crucial for decision making. The aspect - level opinion mining tries to address this problem. In this thesis work, a system is developed to extract and evaluate user opinions at the aspects level.

2.2.1 Aspect-based opinion mining

The review can contain opinion about anything such as a product, a service or an organization. The target object of the opinion is denoted with term *entity*. An *entity* consists of subcomponents and set of attributes of the components. These subcomponents and attributes are considered as aspects. Aspect-based opinion mining deals with extracting these aspects and their opinions. The basic steps in aspect-based opinion mining in Fig.2.2. A more precise definition of an opinion is necessary. So using the definitions in [3], an opinion can be defined as the quintuple:

(entity, aspect, opinion orientation, holder, time)

2.2.1.1 Definitions

Entity: An entity is the target of an opinion. ex. camera.

Aspect: An aspect is a specific feature of the entity. For a camera this can be picture quality, lense, video quality etc.

Opinion Orientation: The orientation or polarity of an opinion can be either positive, negative or neutral. The orientation can also be given by sentiment score.

Opinion Holder : The holder of the opinion is the person or organization that expresses the opinion.

Time: Time represent the time when the opinion is expressed by the opinion holder.

Now, given an opinionated document, processing it and extracting all the possible quintuples is known as aspect-based opinion mining. In practical applications, there will be a large set of documents or reviews from which the aspect-opinion needs to be

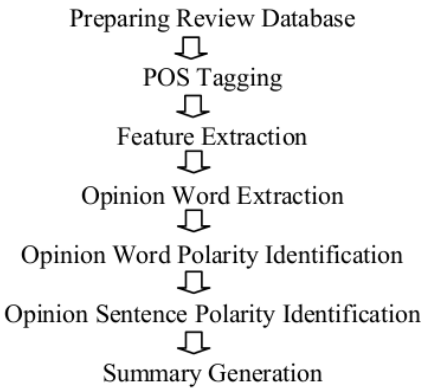


Figure 2.2: Basic steps for feature based opinion mining and summarization

extracted. In this case, summary of the opinions is required. Fig.2.3 shows a common form of opinion summary based on the aspects[1].

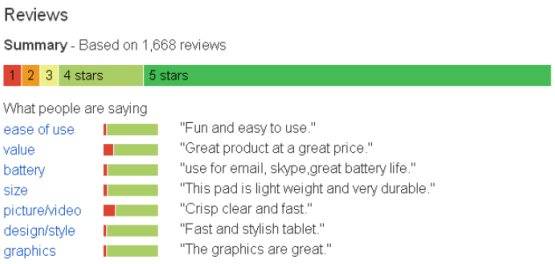


Figure 2.3: opinion summary based on product aspect of iPad (from Google Product)

2.3 Sentiment Classification Techniques

In general, opinion mining task is considered to be an opinion classification problem. The initial step is to extract the features from the corpus. Some of the features are [26]:

[1]www.google.com/shopping

- **Terms – presence and frequency**: These features are individual words or word n-grams and their frequency counts. The word presence can be represented in binary format and frequency by weights.

- **Parts of speech (POS)**: In opinions nouns are important indicators of aspect and adjectives may indicate opinions.

- **Opinion words and phrases**: These are words commonly used to express opinions including good or bad, like or hate.

- **Negations** The appearance of negative words may change the opinion orientation as in not good is equivalent to bad.

Feature selection can be done by either statistical approach or lexicon based approach. In lexicon based approach a seed list of features is manually annotated. Feature selection can treat the words as Bag of Words or n-grams. Feature selection also involves natural language preprocessing techniques like stop words removal, stemming, lemmatization etc. The statistical methods can be tf-idf, pmi etc.

The sentiment classification techniques are generally divided into two broad categories:

- Machine Learning Approach

- Lexicon Based Approach

A third kind of classification is hybrid approach[4], which employs both machine learning techniques and lexicon-based approach. The various important methods and techniques in the above two approaches are illustrated in Fig.2.4.

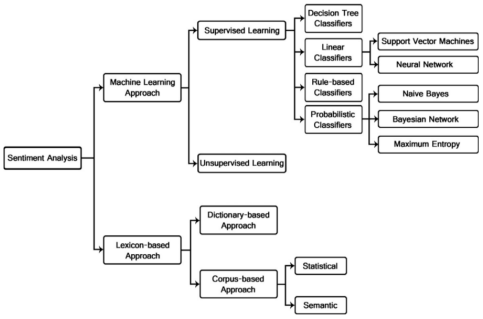


Figure 2.4: Sentiment classification techniques[5]

2.3.1 Lexicon based approach

In this thesis work Lexicon-based approach is used for sentiment polarity detection. This approach is based on the intuition that the polarity of a piece of text can be obtained on the ground of the polarity of the words which compose it. It involves finding the opinion lexicon which is used to analyze the text. So the efficiency of the method depends on the goodness of the lexicon resource. Some of the resources for lexicon based approach are SentiWordNet, Wordnet-Affect, SenticNet etc.[5] SenticNet is used in this thesis work to fine the sentiment score of a opinion word. The two methods are dictionary-based approach and corpus-based approach. In dictionary-based approach a small set of opinion words are collected with semantic orientations and then searches the dictionary of their synonyms and antonyms to grow the list[11,25,13,19,22]. The corpus-based approach begins with a seed list of opinion words, and then finds other opinion words in a large corpus to help in finding opinion words with context specific orientations. This could be done by using statistical or semantic methods[10].

2.4 SenticNet

SenticNet [9] is a lexical resource for concept-level sentiment analysis. It relies on the Sentic Computing [8], a novel multi-disciplinary paradigm for Sentiment Anaylsis. SenticNet is able to associate polarity and affective information to complex concepts such as accomplishing goal, celebrate special occasion and so on. At present, SenticNet provides sentiment scores (in a range between -1 and 1) for 14,000 common sense concepts. The sentiment conveyed by each term is defined on the ground of the intensity of sixteen basic emotions, defined in a model called *Hourglass of Emotions* Fig 2.5. The SenticNet[2] API is used to evaluate the polarity of an opinion word in this thesis work.

2.5 Related Work

This section discusses some useful works relevant to the thesis topic.Many researchers have proposed different methods to solve the problem of mining product features from online product reviews has been addresses by many researchers, proposing different methods. This thesis work is closely related to [11,20,2,6] on mining opinion features in customer reviews. The problem is generally divided into three main subtasks:

- Identifying product features according to the topic
- Identifying opinion expression about the features
- Classifying the sentiment orientation of the opinions [15]

[2]http://sentic.net/api/

Figure 2.5: The Hourglass of Emotions

2.5.1 Exploiting Language Rules

The rules are based on contextual patterns in natural language, which capture various properties of the terms or phrases along with their relation in the text. Hu and Liu[12] first proposed a method to extract aspects from the reviews by finding frequent nouns and noun phrases as frequent aspects and using relations between aspects and opinion words to identify the infrequent aspects. Blair-Goldensohn et al.[4] refined

the approach by considering mainly those noun phrases that are in opinion sentences or in some syntactic patterns which indicate sentiments. Several filters were applied to remove unlikely aspects, for example, dropping aspects which do not have sufficient mentions along-side known sentiment words. The frequency-based idea was also utilized in (Popescu and Etzioni [2]; Ku et al.[14]; Long et al.[18];Moghaddam and Ester[21]; Zhu et al.[29]). This method is simple and has been found to be effective. A modified approach of the frequency-based method is used in this work.

Zhuang et al. employed the dependency relation to extract aspect- opinion pairs from movie reviews. After parsed by a dependency parser (e.g., MINIPAR [3] (Lin, 1998)), words in a sentence are linked to each other by a certain dependency relation. A dependency relation template can be found as the sequence NN - nsubj - VB - dobj - NN. NN and VB are POS tags. nsubj and dobj are dependency tags. Reliable dependency relation templates from training data first obtained from training data, and then used them to identify valid aspect-opinion pairs in test data.

Double Propagation [26] further develops the above discussed idea. The system propagates through a seed opinion list and aspect list, hence the name double propagation. The propagation performs four subtasks:

- Extracting aspects using opinion words
- Extracting aspects using extracted aspects
- Extracting opinion words using the extracted aspects
- Extracting opinion words using both given and extracted opinion words

[3]http://webdocs.cs.ualberta.ca/ lindek/minipar.htm)

The method uses predefined rules to extract the aspects and opinion words. This thesis work uses idea of double propagation algorithm.

2.5.2 Association Rules

Hu and Liu in [20] designed a system to perform review summarization in two main steps: feature extraction and opinion direction identification. Product name and an entry page for all the reviews of the product are given as input. The system generates review summary for the product features. The sentences in review are POS tagged and simple noun phrases(NP) and verb phrases(VP) identified using syntactic chunking.

Association rule mining is used to detect all frequent itemsets, which is a set of words or a phrase that occurs together. The association rule miner CBA [16] based on the Apriori algorithm in [1], finds all frequent itemsets in the transaction set with the user-specified minimum support 1%. In the next step Feature Pruning is performed to remove those incorrect features. The two types of pruning are presented:

(a) Compactness pruning checks features that contain at least two words, which are named feature phrases, and removes those that are likely to be meaningless.

(b) Redundancy pruning removes redundant features that contain single words.

Opinion words extraction with all the remaining frequent features is done after pruning. Infrequent feature identification. Hu and Liu suppose that people like to use the same opinion word to describe different features. So they can use the opinion words to look for features that cannot be found in previous step. This intuition is used in this thesis work to extract the features.

This proposal can produce a number of explicit features however implicit feature cannot be extracted. The irrelevant sentences may be thought of as opinion sentences, and the nouns in irrelevant sentence would be extracted as features.

2.5.3 Unsupervised Information Extraction System

The process of aspect extraction can be modelled as information extraction problem. Popescu and Etzioni[2] introduce an unsupervised formation-extraction system (OPINE) to extract aspects from reviews. OPINE performs four main tasks:

(1) Identifying product features

(2) Identifying opinions of product features

(3) Determining the polarity of opinions

(4) Ranking opinions based on their strength.

It is built on top of KnowItAll, a domain-independent information extraction system based on Web [14], which instantiates relation-specific generic extraction patterns into extraction rules to find candidate facts. A form of Point-wise Mutual Information (PMI) is assigned by KnowItAlls Assessor between phrases that is estimated from Web search engine hit counts [24]. The PMI is computed between each fact and automatically generated discriminator phrases. Given fact f and discriminator d, the PMI score is computed as (1):

$$PMI(f,d) = \frac{Hits(d+f)}{hits(d) + hits(f)} \tag{2.1}$$

In order to output a probability associated with each fact, the PMI scores are converted to binary features for a Naive Bayes Classifier.

Given a product class, OPINE extracts explicit feature from parsed review data.The system recursively identifies both the parts and the properties of the given product class and their parts and properties, in turn, continuing until no candidates are found. The system finds related concepts and extracts their parts and properties. Opine achieves 22% higher precision than Hus, but has 3% lower recall.

2.5.4 Probability-based Algorithm

Scaffidi presents a probability-based algorithm and compares it to an existing support based approach in [6]. The system uses two algorithms (probability-based and support-based) to extract features from Amazon.com product categories to ask customers to rate the features in terms of helpfulness for choosing products. The features identified by the probability-based algorithm are preferred by the customers. The probability-based algorithm can identify features that comprise a single noun or two successive nouns The system has 6 sub parts:

1. Configuring with baseline frequency data
2. Calculating single-lemma statistics
3. Calculating the probability of observed lemma occurrence counts
4. Calculating the probability of observed bigram occurrence counts
5. Filtering words inherent to the category
6. Selecting the final features.

The limitations in this system are: it needs some supervised machine learning to offer quality feature-extraction, and cannot extract implicit features from reviews.

A lot of research has been carried out on product aspect extraction, opinion identification, sentiment classification and opinion summarization. However there is little work done on integrating all these parts together. Therefore, this thesis work focuses on integrating all these parts to mine product aspects by using a lexicon-based approach. This thesis differs from many past works by integrating four parts and thus developing a system which gives a practically useful result. The advantage of mining product aspects from the data using the lexicon-based approach as employed in this thesis work is that, training data is not required for opinion mining. So this approach can be employed when training data is not available or when data is from multiple domains. In this thesis lexicon based method is used for opinion mining. The seed lexicon sources used for mining are generated from the data corpus itself.

Chapter 3

Linguistic Resources Development

This chapter describes how the dataset is obtained from online review site, processed, generating the opinion seed list and aspect seed list. For a lexicon-based approach two NLP resources are required:

- A corpus of reviews
- Seed lexicon set to bootstrap the aspect and opinion words extraction

3.1 Data Corpus

The online reviews can be mainly of two formats[3].

Format 1 - Pros, Cons and the detailed review: The reviewer is asked to write pros and cons of the product separately and also write a full review.

Format 2 - Free Format: No guidelines are specified, the reviewer can write freely.

In this thesis work the customer reviews on Camera is used as the data for opinion

mining task. The reviews on Camera are obtained from Amazon[1]. The data miner retrieves the html pages containing the reviews. The reviews and related information are then extracted from these html pages to a csv file.

The reviews used in this thesis work are of Free Format. The review does not have separate pros and cons information. The data corpus statistics are:

- **Number of Products**: 65
- **Number of reviews**: 12770
- **Number of sentences**: 65289

The product name and product code are mined from site into a csv file. For each product the reviews are mined and written into a csv file. Product code and product name are mapped to the review ,each review is given a unique id and written into a xml file.

3.1.1 Data Format

Each review in the xml file has following information :

- **review Id**:unique id that represents the review. ex: RC100001.
- **Product Name**: name of the product as mined from the site. ex : nikon d7100 24.1 mp dx-format cmos digital slr (body only)
- **Product Type**: the category of the product. In this thesis the category is Camera

[1]http://www.amazon.com/

- **Product Id**: unique id of the product mined from the site. ex: B00BI9X7UC.
- **Review Title**: title of the review.
- **Review Text**: the customer review about the product.
- **Rating**: rating of the review. range is from 1-5 stars. 5 is the maximum rating and 1 is the minimum
- **Date** :date when the review is posted
- **yesHelpfulness**: Count of "yes" helpfulness votes. the reviewer tells whther the review has been helpful for him or not
- **totalHelpfulness**:Count of "yes" + no helpfulness votes.

3.2 Generating Lexicon Resources

In this thesis work, a lexicon based approach is used for opinion mining which requires two seed lexicon files. The first lexicon set is seed opinion words and the second is seed aspects. Both the lists are generated using the data corpus and manually filtered to remove the inconsistencies.

3.2.1 Aspect lexicon

The seed aspect lexicon file is generated using the product names. Each product name is tokenized and POS tagged using NLTK. The nouns added to the aspect lexicon. The intuition here is that, it is in human nature to expect more in less time. So it is natural that a customer browses through product name more frequently than go

to product details. So, to attract customer interest the manufacturer/seller mentions important features in the product name. The inconsistencies in this lexicon is checked manually as the number of aspects obtained is small.

3.2.2 Opinion lexicon

The seed opinion lexicon file is generated using the review title. Title of a review is in a way a concise summary of the user opinion. It can indicate overall sentiment of the reviewer. Similarly, star rating also reflects the overall opinion of the reviewer. With this insight, the title of the reviews are considered as candidates for the seed opinion lexicon generation and the ratings are used to decide polarity. The title is POS Tagged and adjectives are considered as it is an important indicator of opinion. The polarity of each opinion word is assigned using the rating of the review. Opinion word is classified positive if the rating is 4 or 5, negative if the rating is 1,2. Polarity of words from review title with rating 3 is assigned manually according to semantics.

Chapter 4

System Architecture

This chapter gives a detailed description of system architecture, the algorithms for aspect-opinion extraction, SenticNet-based polarity detection and aspect-opinion summarization.

4.1 Design

The system developed in this thesis work integrates the tasks of (1)Aspect extraction; (2) Aspect-Opinion Identification (3) Aspect-Opinion polarity classification and (4)Aspect-based product review summary generation using user provided labels.It can be broadly divided into two parts:

- Opinion Mining Subsystem
- Review Retrieval Subsystem

The system is illustrated in Fig.4.1. The opinion mining subsystem consists of the modules: (1) Preprocessing the review corpus and POS Tagging; (2) Noun Phrase

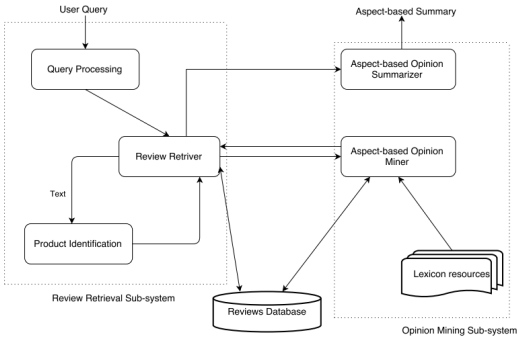
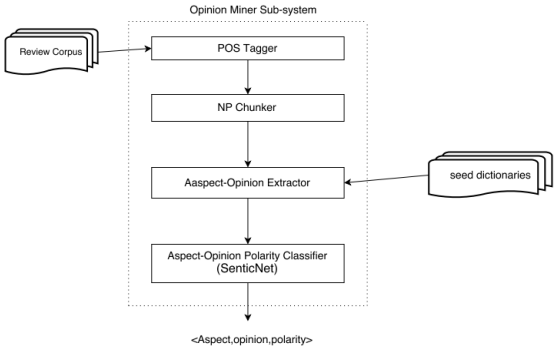


Figure 4.1: System architecture for aspect-based opinion mining

Chunking of tagged reviews; (3) Aspect-Opinion Extraction from the noun phrases; (4) Aspect-Opinion polarity classification using SenticNet. Fig.6 shows the architecture of the Opinion Miner.



miner.png

Figure 4.2: Opinion Miner architecture

The information retrieval subsystem has components that process the user input, Builds an inverted index on the aspects and aspect-opinion pairs, retrieves the camera reviews based on the user specified labels. Each of these modules are detailed below.

4.1.1 Database

The database *productsentisum* has two tables reviews and products which has review and product data respectively.. The data corpus is read and the information relevant to the system are stored into the database tables. These data are used in opinion mining and also in review retrieval to generate the aspect-opinion summary. The database schema for *productsentisum*:

Reviews

review_Id	review_productName	review_productId	review_productType	review_title	revi

Products

product_Id	product_Name	product_Type

4.2 Modules and Algorithms

In this section the components of proposed system for aspect-based opinion mining is discussed in detail.

- **Data preprocessing and POS Tagger**

  The reviews are preprocessed and POS Tagging is performed. Each sentence in the review is tokenized, the stop-words and punctuations are removed a list is obtained. These sentences are POS Tagged. POS tagging is also performed on product names and review titles to generate the seed lexicon for aspect-opinion extraction.

- **Noun Phrase Chunker**

  Chunking is a method of extracting keywords or important phrases from the text. The chunker takes a POS tagged sentence as input. It searches for Noun Phrases and Verb groups. For aspect extraction the first step is to extract the noun phrases from the corpus. Chunking can be done in two ways, using regular expression and by training a chunk parser. Chunking with regular expressions are used in this thesis work to obtain noun phrases as outlined in Algorithm 1. The output of the algorithm is dumped into a file.

  Ex: sentence : The little yellow dog barked at the cat Reg-ex pattern : "NP: $\{< DT >? < JJ > * < NN >\}$"

  The above regular expression chunks two Noun Phrases in the given sentence,

  (S

  (NP the/DT little/JJ yellow/JJ dog/NN)

  barked/VBD

  at/IN

  (NP the/DT cat/NN))

  Algorithm 1: NP Chunking

```
Input : All the reviews from the corpus
Output : list of  Noun Phrases for each review
### Specify the pattern to be chunked
```

```
grammar = "
    NBAR:
        {<RB>?<JJ.*>*<NN.*>}  # Nouns and Adjectives, terminated with Nouns
        {<NN.*>*<RB>?<JJ.*>}
    NP:
        {<NBAR>}
        {<NBAR><IN><NBAR>}  # Above, connected with in/of/etc.
For each review in reviews do
    sent_tok = Tokenize(sentences)
    sent_tag = POSTagger(sent_tok)
    #Chunking tree
    tree = NPChunker(sent_tag)
    terms = get_terms_parse_tree(tree)
    #variable list and dictionary to hold the NP
    nounPhrasesList = [];
    nounPhrases = {}
    For each term in terms do
        nounPhrasesList.append(term)
    end
    nounPhrases[reviewId] = nounPhrasesList;
end
```

- **Aspect - Opinion Extractor**

  The aspect extraction is the main task in the opinion mining system developed in

this thesis. A lexicon based propagation algorithm (Algorithm 2)is proposed in this work, the algorithm takes as input seed aspect lexicon, seed opinion lexicon and the Noun phrases for each reviews generated in the NP Chunking step. Noun Phrases are scanned two times in the algorithm. In the first propagation, the aspect seed list is used to find the match the noun in the Noun Phrase, the corresponding adjective is extracted and the seed adjective list is expanded.

Hu and Liu [11] suppose that people like to use the same opinion word to describe different features. The opinion words can be used to look for features that cannot be found in previous step. This intuition is used in this thesis work to extract more the features. In the second propagation over the Noun Phrases, the adjectives in the expanded seed adjective list is used to find more aspects. These aspects are added to the aspect list. Also, the identified aspects and the corresponding adjective pairs are listed for every review.

This process generates a considerable amount of aspects and aspect-opinion pairs. Due to the inherent ambiguity of natural languages, all these may not be useful for the task at hand. So, the aspects are pruned based on aspect frequency. This approach is simple and has been found to be effective. The aspects above the threshold are retained. The threshold value for frequency is arrived experimentally. The process is outlined in **Algoritm 2**.

Algorithm 2: Aspect-Opinion Extractor

```
Input: noun-phrases from NP-chunking, seed opinion-polarity, seed opinions
Output: new adjectives and aspect-opinion pairs
```

```
initialize newAspects=[]


For revId in nounPhrases do
    For npPair in nounPhrases[revId] do
        taggedPair=pos_tag(npPair)
        initialize nouns=[]
        initialize adjs=[]
        For taggedWord in taggedPair do
            if JJ in taggedWord(pos_tag) then
                adjs.append(taggedWord)
            end
            if NN in taggedWord(pos_tag) then
                nouns.append(taggedWord)
            end
        end
        For noun in nouns do
            if noun in seedAspects then
                For adj in adjs do
                    if adj not in seedAdjs then
                        seedAdjs.append(adj)
                    end
                end
            end
        end
```

```
    end
end
Result: new opinion list


initialize aspectOpinionPairDict={}
For revId in nounPhrases do
    For npPair in nounPhrases[revId] do
        taggedPair=pos_tag(npPair)
        initialize nouns=[]
        initialize adjs=[]
        For taggedWord in taggedPair do
            if JJ in taggedWord(pos_tag) then
                    adjs.append(taggedWord)
            end
            if NN in taggedWord(pos_tag) then
                    nouns.append(taggedWord)
            end
        end
        initialize appendList=[]
        For adj in adjs do
            For noun in nouns do
                appendList.append([adj,noun])
            end
            if adj in seedAdjs then
```

```
                for noun in nouns do
                    if noun not in seedAspects then
                        newAspects.append(noun)
                    end
                end
            end
        end
        aspectOpinionPairDict[revId]=appendList
    end
end
```

- **Aspect - Opinion Polarity Classification**

  The opinion miner uses the seed opinion list generated initially and SenticNet API. The seed opinion list has polarity labels. To assign the polarity for aspect-opinion pair, the polarity of unlabeled opinion is determined by calculating its sentiment score using the SenticNet API. The opinions with a sentiment score less than zero are classified as negative and the rest as positive. The words for which the score was not evaluated by the SenticNet API is labeled as unknown. This list of labeled opinions id used to assign the polarity for the aspect- opinion pair. This process is outlined in **Algorithm 3**.

  Algorithm 3

```
Input: Set of opinion words where initial seed opinions are labeled
Output: Aspect-opinion pair polarity classification as positive or negative
newPolarityDict = []
```

```
for word in opinionlist do
    if word not in newPolarityDict then
            url='http://sentic.net/api/en/concept/'+str(wordj)+'/polarity/'
    else if  word newPolarityDict  and newPolarityDict[word] == obj then
            url='http://sentic.net/api/en/concept/'+str(adj)+'/polarity/
    else
            continue
    end
    search  polaritytag
    if present then
        polarity = polarityinTag;
        if polarity > 0 then
                polarity_label = pos;
        else
                polarity_label = neg
        end
        for opinion in newopinionlist do
            if opinion_in_pair == opinion then
                aspect_opinion_polarity = polarity
            end
        end
    end
end
```

- **Inverted Index**

  The inverted index module builds an inverted index for aspects in reviews and aspect-opinion pairs in reviews. These inverted indexes are used by the review retriever module to get the reviews that match the aspects or opinion present in the user query. Inverted indexes allow fast full text searches, the inverted indexes built here are record level inverted index. The process is outlined in **Algorithm 4**.

- **Review Retriever**

  The review retriever is another sub system, it uses the inverted index, product lists, product and review details in database and results of the Opinion Miner to retrieve the products that best matches the user query along with aspect-opinion summarization for that product.

  Algorithm 4

```
Input: Full aspects list, all reviews with review ID, aspect ID
Output: aspect inverted index file invertedIndex.p

initialize: invertedIndex={}

for aspect in combinedAspects do
    initalize aspectPresenceList=[]
    For revID in allReviewsDict do
initalize revText=
```

```
For rev in allReviewsDict[revID] do
            revText=revText+' '+str(rev)
end
        revText = word_tokenize(revText)
        if aspect in revText then
            aspectPresenceList.append(revID)
        end
    end
    invertedIndex[aspectID[aspect]]=aspectPresenceList
end
```

The implementation details of all the modules in the aspect-based opinion mining system mentioned above will be presented in next chapter.

Chapter 5

Implementation

This chapter deals with the implementation of all the modules of aspect-based opinion mining system described in chapter 5. It also includes the prerequisites for running the implementation in a computer. The programming language chosen for implementation is Python. The following sections will contain the code for different parts of the opinion miner. The NLTK library is used for the text processing.

5.1 NP Chunker

The following piece of code 5.1.1 takes all the reviews as input and generates the Noun Phrases. The algorithm is explained in chapter 4, Algorithm 1

5.1.1 NPChunking.py

```
'''
NP chunker to extract Noun Phrase from the reviews and generate a list of phrase
```

```
'''
import nltk, csv, codecs, pickle
from nltk.corpus import stopwords

stopwords = stopwords.words('english')



allreviewList = []
# Used for word tokenization
sentence_re = r'''(?x)
      ([A-Z])(\.[A-Z])+\.?
    | \w+(-\w+)*
    | \$?\d+(\.\d+)?%?
    | \.\.\.
    | [][.,;"'?():-_`]
'''
# The regex patter for chunking
grammar = r"""
    NP1:
        {<RB>?<JJ.*>*<NN.*>}
        {<NN.*>*<RB>?<JJ.*>}
    NP:
        {<NP1>}
        {<NP1><IN><NP1>}
```

```
"""
chunker = nltk.RegexpParser(grammar)

# Open the reviews input file
with open('reviewinput.csv', 'r') as incsvfile:
    csv_r = csv.reader(incsvfile, delimiter=',')
    for row in csv_r:
        allreviewList.append(row)
rows=len(allreviewList)

#Dictionary to hold the list of all the NPs for each review
main_dict={}

for i in range(rows):
    print(i)
    revId=allreviewList[i][0]
    text=allreviewList[i][1]

    tokens = nltk.regexp_tokenize(text, sentence_re)
    postokens = nltk.tag.pos_tag(tokens)
    tree = chunker.parse(postokes)

    # Finds NP leaf nodes of a chunk tree
    def leaves(tree):
```

```
    for subtree in tree.subtrees(filter = lambda t: t.label()=='NP'):
        yield subtree.leaves()

# converts all the words to lower case
def caseChange(word):
    word = word.lower()
    return word

# Removing stopwords and meaningless words
def acceptable_word(word):
    accepted = bool(2 <= len(word) <= 40
        and word.lower() not in stopwords)
    return accepted

# get the phrases fom chunk trees
def get_terms(tree):
    for leaf in leaves(tree):
        term = [ caseChange(w) for w,t in leaf if acceptable_word(w) ]
        yield term

terms = get_terms(tree)
aspectList=[]

for term in terms:
```

```
        if len(term)>1:
            aspectList.append(term)

    main_dict[revId]=aspectList

# Dump the Nps to a file
pickle.dump(main_dict,open('main_dict.p','wb'))
```

5.2 Aspect-Opinion Extractor

The aspect-opinion extractor takes the Noun Phrases dump file from the Np Chunker, and seed aspects seed opinions. The process is detailed in **Algorithm 2** of previous chapter.

```
'''
Aspect- opinion extraction using the seed lexicons - aspect and opinions to gene
'''

import pickle, csv, nltk
import numpy as np
from nltk.probability import FreqDist

# Lexicon based propagation
print('Lexicon-based propagation algorithm to generate aspects')
```

```
#input files
# load the Noun Phrases
nounPhrases=pickle.load(open('nounPhrases.p','rb'))

# load seed lexicon files
opinionPolarityDict=pickle.load(open('opinionPolarityDict.p','rb'))
seedAdjs=pickle.load(open('seedAdjs.p','rb'))
seedAspects=pickle.load(open('seedAspects.p','rb'))
# load all the reviews
allReviewsDict=pickle.load(open('allReviewsDict.p','rb'))

print(len(seedAspects))
# list to store the expanded aspect list
newAspects=[]

# Propagation using aspect lexicon to expand opinion words
i=0
print('Number of seed opinions::',str((len(seedAdjs))))
for revId in nounPhrases:
    i+=1
    if i in np.linspace(0,2000,5) or i in 6*np.linspace(0,2000,5):
        print(i)

    for npPair in nounPhrases[revId]:
```

```
        taggedPair=nltk.pos_tag(npPair)


        nouns=[]
        adjs=[]
        for taggedWord in taggedPair:
            if 'JJ' in taggedWord[1] and taggedWord[0]:
                adjs.append(taggedWord[0])
            if 'NN' in taggedWord[1] and taggedWord[0]:
                nouns.append(taggedWord[0])
        for noun in nouns:
            if noun in seedAspects:
                for adj in adjs:
                    if adj not in seedAdjs:
                        seedAdjs.append(adj)
seedAdjs=list(set(seedAdjs))
# dump the new opinion list
pickle.dump(seedAdjs,open('seedAdjsnew.p','wb'))


# Propagation using expanded opinion lexicon to find aspects
i=0
print('Number of seed aspects::',str((len(seedAspects))))
aspectOpinionPairDict={}
for revId in nounPhrases:
```

```
i+=1
if i in np.linspace(0,2000,5) or i in 6*np.linspace(0,2000,5):
    print(i)

for npPair in nounPhrases[revId]:

    taggedPair=nltk.pos_tag(npPair)

    nouns=[]
    adjs=[]
    for taggedWord in taggedPair:
        if 'JJ' in taggedWord[1] and taggedWord[0]:
            adjs.append(taggedWord[0])
        if 'NN' in taggedWord[1] and taggedWord[0]:
            nouns.append(taggedWord[0])
    appendList=[]
    for adj in adjs:
        for noun in nouns:
            appendList.append([adj,noun])
        if adj in seedAdjs:
            for noun in nouns:
                if noun not in seedAspects:
                    newAspects.append(noun)
    aspectOpinionPairDict[revId]=appendList
```

```
# dump new aspect list and aspect opinion pair for each review.
pickle.dump(newAspects,open('newSeedAspects.p','wb'))
pickle.dump(aspectOpinionPairDict,open('aspectOpinionPairDict.p','wb'))

print('Number of seed adjectives::',str((len(seedAdjs))))
print('Number of new aspects::',str((len(newAspects))))


# Aspect Pruning-Populate most frequent aspects from the new aspects list
print('\n\nselecting only most frequent of the aspects')
thresholdval = 98
newSeedAspects=pickle.load(open('newSeedAspects.p','rb'))
allNounsList=pickle.load(open('allNounsList.p','rb'))
fd=FreqDist()
for noun in allNounsList:
    fd[noun]+=1
i=0
topAspects=[]
for w in sorted(fd, key=fd.get, reverse=True):
    i+=1
    #print(w,fd[w])
    topAspects.append(w)
    if i==thresholdval:
        break
```

```
with open('topAspects.csv','w') as file:
    for item in topAspects:
        file.write(item+',')
pickle.dump(topAspects,open('topAspects.p','wb'))

# Merge the two aspect lists and remove duplicates
print('\n\nMerge the two aspect lists and remove duplicates')
aspectsToRemove=open('AspectsToRemove.txt','r').read()
aspectsToRemove=aspectsToRemove.split(',')
combinedAspects=list(set(topAspects+seedAspects))

for item in aspectsToRemove:
    try:
        combinedAspects.remove(item)
    except:
        continue

print('Total aspects after combining and pruning::',str(len(combinedAspects)))
pickle.dump(combinedAspects,open('combinedAspects.p','wb'))

# prune aspect opinon pair - use pruned aspect list
print("\n\nPrune aspect opinion pair to remove non-usable aspects")
newAspectOpinionPairDict={}
for revID in aspectOpinionPairDict:
```

```
    listOfPairs=aspectOpinionPairDict[revID]
    appendPair=[]
    for pair in listOfPairs:
        noun=pair[1]
        if noun in combinedAspects:
            appendPair.append(pair)
    newAspectOpinionPairDict[revID]=appendPair
# dump the pruned result
pickle.dump(newAspectOpinionPairDict,open('newAspectOpinionPairDict.p','wb'))
```

5.3 Aspect-Opinion Polarity Classifier

The aspect-pinion classifier presented in **Algorithm 3** of chapter 4 is implemented as 5.3.1 and 5.3.2. The opinion words are classified in 5.3.1. The aspect- opinion pair is assigned polarity, by mapping the polarity label to the aspect- opinion pair ID in 5.3.2

5.3.1 Polarity Classifier.py

```
'''
Find sentiment score and classify unlabelled opinion
'''

import urllib.request, re, pickle
```

```
adj1=pickle.load(open('seedAdjs.p','rb'))
opinionPolarityDict=pickle.load(open('opinionPolarityDict.p','rb'))

newPolarityDict = opinionPolarityDict
errAdj=''
success=0
i=0
for adj in adj1:
    if adj not in newPolarityDict:
        url='http://sentic.net/api/en/concept/'+str(adj)+'/polarity/'
    elif adj in newPolarityDict and newPolarityDict[adj]=='obj':
        url='http://sentic.net/api/en/concept/'+str(adj)+'/polarity/'
    else:
        continue
    i+=1
    print(str(i),url)
    try:
        html=str(urllib.request.urlopen(url).read())
    except:
        errAdj=errAdj+','+adj
        continue
    pattern=r'float">(-{0,1}\d+\.{0,1}\d*)</polarity>'
    obj=re.search(pattern,html)
    if obj:
```

```
        polarity=float(obj.group(1))
        if polarity>=0:
            polarity='pos'
        else:
            polarity='neg'
        newPolarityDict[adj]=polarity
        success+=1
    else:
        polarity="NA"
        errAdj=errAdj+','+adj

        print(adj, str(polarity))
# dump the new opinion-polarity list
pickle.dump(newPolarityDict,open('newPolarityDict.p','wb'))
file=open('failedOpinionWords.txt','w')
file.write(errAdj)
file.close()
print(str(success)+'/'+str(i)+' requests successful')
```

5.3.2 AOIDPolarity.py

```
'''
Assign polarity to asect-opinion pairs using new labelled opinion words
'''
```

```
import pickle


AOID=pickle.load(open('AspectOpinionID.p','rb'))
newPolarityDict=pickle.load(open('newPolarityDict.p','rb'))


AOIDPolarityDict={}


for AOPair in AOID:
    polarity='unk'
    adj=AOPair.split(' ')[0]
    if adj in list(newPolarityDict.keys()):
        polarity=newPolarityDict[adj]
    AOIDPolarityDict[AOID[AOPair]]=polarity


pickle.dump(AOIDPolarityDict,open('AOPolarityDict.p','wb'))
```

5.4 Build Inverted Index

The inverted index is built for aspect list and aspect-opinion pair. **Algorithm 4** in chapter 4 explains the process. Inverted index built are used in Review retrieval based on the user query.

```
'''
Buil inverted index for aspects
```

```
'''


from nltk.corpus import stopwords
from nltk.tokenize import sent_tokenize, word_tokenize
import re, os, pickle, sys

stop_words = set(stopwords.words("english")) #set of stopwords

combinedAspects=pickle.load(open('combinedAspects.p','rb'))
allReviewsDict=pickle.load(open('allReviewsDict.p','rb'))
aspectID=pickle.load(open('aspectID.p','rb'))

print(len(combinedAspects))

invertedIndex={}
for aspect in combinedAspects:
    print(combinedAspects.index(aspect))
    aspectPresenceList=[]
    for revID in allReviewsDict:
        revText=''
        for rev in allReviewsDict[revID]:
            revText=revText+' '+str(rev)
        revText=revText.replace('|','')
```

```
        revText=word_tokenize(revText)


        if aspect in revText:
            aspectPresenceList.append(revID)
    invertedIndex[aspectID[aspect]]=aspectPresenceList


# dump the inverted index
pickle.dump(invertedIndex,open('invertedIndex.p','wb'))
```

5.5 Review Retriever

Review retriever uses the product and review information from database, and ¡aspect,opinion,polarity¿ tuple generated by the opinion-miner. For a user query,the query is parsed. Initially it searches for a direct match in the product name. If found the product review summary is generated and displayed. If the user query is not a product name, then the input is POS Tagged and noun list and aspect-opinion pair list is generated for the query. The review Ids are matched using inverted index and product details are fetched from the database. A summary for each aspect is generated with positive review count and negative review count, which is displayed with the product name that best matches the query.

5.5.1 ProductAspectOpinionSum.py

```
'''
Review retrieval directed by the user input and the aspect - opinion summary is
```

```
'''

from nltk.tokenize import word_tokenize
import nltk, csv,pickle, mysql.connector
import os

# Summarizes the aspect - opinion polarity from the reviews for each product
def prodSummarize(prodId):
    import pickle, mysql.connector
    AOInvertedIndex=pickle.load(open('AspectOpinionInvertedIndex.p','rb'))
    AOPolarityDict=pickle.load(open('AOPolarityDict.p','rb'))
    AOID=pickle.load(open('AspectOpinionID.p','rb'))

    # Open database connection
    cnx = mysql.connector.connect(user='root', password='gdp123', host='localhos

    # prepare a cursor object using cursor() method
    cursor = cnx.cursor()

    query='SELECT review_Id from reviews where review_productId = "%s"'%(prodId)
    cursor.execute(query)
    dbrows = cursor.fetchall()

    rlist=[]
```

```
for row in dbrows:
    rlist.append(row[0])
cnx.commit()
cnx.close()


reviewsDict={}
for aoPairID in AOInvertedIndex:
    pairReviewList=[]
    reviewList=AOInvertedIndex[aoPairID]
    for review in reviewList:
        if review in rlist:
            pairReviewList.append(review)
    if pairReviewList:
        reviewsDict[aoPairID]=pairReviewList



polarityCountDict={}
for aoPairID in reviewsDict:
    for key,val in AOID.items():
        if val==aoPairID:
            aoPair=key


    aspect=aoPair.split(' ')[1]
    if aspect not in list(polarityCountDict.keys()):
```

```
            polarityCountDict[aspect]=[0,0]
        currentCount=polarityCountDict[aspect]
        posCount=currentCount[0]
        negCount=currentCount[1]

        polarity=AOPolarityDict[aoPairID]
        if polarity=='pos':
            posCount+=len(reviewsDict[aoPairID])
        if polarity=='neg':
            negCount+=len(reviewsDict[aoPairID])
        updatedCount=[posCount,negCount]
        polarityCountDict[aspect]=updatedCount

    return polarityCountDict
# The products list which has the aspect-opinion pair
def GetAOProdList(AOList):
    aspectInvertedIndex=pickle.load(open('invertedIndex.p','rb'))
    aoInvertedIndex=pickle.load(open('AspectOpinionInvertedIndex.p','rb'))
    AOID=pickle.load(open('AspectOpinionID.p','rb'))

    # Open database connection
    cnx = mysql.connector.connect(user='root', password='gdp123', host='localhos
    # prepare a cursor object using cursor() method
    cursor = cnx.cursor()
```

```
AOProdList=[]
AOCount=0
for item in AOList:
    ao=''
    for i in item:
        ao=ao+str(i)+' '
    ao=ao[:-1]
    if ao not in list(AOID.keys()):
        continue
    else:
        AOCount+=1
    aoID=AOID[ao]
    revIdList=aoInvertedIndex[aoID]
    for revId in revIdList:
        query='SELECT review_productId from reviews where review_Id = "%s"'%
        cursor.execute(query)
        dbrows = cursor.fetchall()
        for row in dbrows:
            AOProdList.append(row[0])
cnx.commit()
cnx.close()

AOProdList=list(set(AOProdList))
```

```
    return AOProdList


# Parse query to get the aspect-opinion pair list and nounlist
def getNounlistAOList(queryInput):
    tokenizedQuery=word_tokenize(queryInput)
    taggedQuery=nltk.pos_tag(tokenizedQuery)
    nounList=[]
    AOList=[]
    for tokenWord in taggedQuery:
        if 'NN' in tokenWord[1]:
            index=taggedQuery.index(tokenWord)
            nounList.append(str(tokenWord[0]))
            try:
                if 'JJ' in taggedQuery[index-1][1]:
                    appendList=[taggedQuery[index-1][0],tokenWord[0]]
                    AOList.append(appendList)
            except:
                continue
    obj=[nounList,AOList]
    return obj


# Get product Id for the list of review Ids
def GetProdID(RevIDList):
    prodIDList=[]
```

```
    # Open database connection
    cnx = mysql.connector.connect(user='root', password='gdp123', host='localhos
    # prepare a cursor object using cursor() method
    cursor = cnx.cursor()
    for revID in revIDList:
        query='SELECT review_productId from reviews where review_Id = "%s"'%(rev
        #print(query)
        cursor.execute(query)
        dbrows = cursor.fetchall()
        for row in dbrows:
            if row not in prodIDList:
                prodIDList.append(row[0])
        cnx.commit()
        cnx.close()
    return prodIDList
# Display the result
def dispResults(resultsDictofDict):
    import csv
    print('\t\t\t\t\t'.join(['pName','Aspect','+ve reviews','-ve reviews']))
    prodNames={}
    with open('cameradata.csv','r') as file:
        csv_r=csv.reader(file,delimiter=',')
        for row in csv_r:
            prodNames[row[0]]=row[1]
```

```
    for item in resultsDictofDict:
        pName=prodNames[item]
        print(pName)
        dispAspect=resultsDictofDict[item]
        for aspect in dispAspect:
            posRev=dispAspect[aspect][0]
            negRev=dispAspect[aspect][1]
            if posRev==0 and negRev == 0:
                continue
            print('\t\t\t\t\t','\t\t\t\t\t\t'.join([str(aspect),str(posRev),str(ne



#####Input necessary variables
prodNames={}
with open('cameradata.csv','r') as file:
    csv_r=csv.reader(file,delimiter=',')
    for row in csv_r:
        prodNames[row[0]]=row[1]



aspectInvertedIndex=pickle.load(open('invertedIndex.p','rb'))
aoInvertedIndex=pickle.load(open('AspectOpinionInvertedIndex.p','rb'))
AOID=pickle.load(open('AspectOpinionID.p','rb'))
```

```
queryInput=input('Enter your requirements in your own words (upto 140 characters

queryInput=queryInput.lower()

if type(queryInput)!=str:
    print('Your query was not valid. Please try again.')
    os._exit(1)
if len(queryInput)>140:
    print("Query too long. Try again..")
    os._exit(1)

direct_list=[]
for val in list(prodNames.values()):
    if queryInput in val.lower():
        for key in prodNames.keys():
            if queryInput in prodNames[key].lower():
                direct_list.append(key)
        break
    break

if direct_list:
    resultsDict={}
```

```
    for ID in direct_list:
        resultsDict[ID]=prodSummarize(ID)
    dispResults(resultsDict)
    os._exit(1)

[nounList,AOList]=getNounlistAOList(queryInput)

if AOList:
    try:
        AOProdList=GetAOProdList(AOList)

    except:
        print('AO not found')
        pass
    if AOProdList:
        resultsDict={}
        for ID in AOProdList:
            resultsDict[ID]=prodSummarize(ID)
        dispResults(resultsDict)
        print('Matched using Aspect-Opinion pairs')
        os._exit(1)

elif len(nounList)==1:
    if nounList[0] not in list(aspectInvertedIndex.keys()):
```

```
            print('Unable to populate any results. Try again with a different query
            os._exit(1)
        else:
            revIDList=aspectInvertedIndex(nounList[0])
            prodIDList=GetProdID(RevIDList)
            if len(prodIDList)>25:
                print('too many results. Try a better search.')
                os._exit(1)
            else:
                resultsDict={}
                for ID in prodIDList:
                    resultsDict[ID]=prodSummarize(ID)

                print('Matched using single Aspect')
                os._exit(1)

    elif len(nounList)>1:
        itemDict={}
        itemList=[]
        for item in nounList:
            try:
                revIDList=aspectInvertedIndex[item]
            except:
                continue
```

```
        prodIDList=GetProdID(RevIDList)
        itemDict[item]=prodIDList
        itemList.append(prodIDList)
    print(len(itemList))

else:
    print('Unable to populate any results. Try again with a different query.')
    os._exit(1)
```

The above section described the implementation of all the modules of the opinion miner-review retrieval system. The results are discussed in the following chapter.

Chapter 6

Results

The results obtained from the opinion miner - review retrieval system is presented in this chapter. For the given user query, the reviews from the products which has matching features retrieved. For each of the products the count of positive and negative reviews for the features mentioned in those reviews are displayed. The following are the sample results for different queries:

6.1 Sample Results

6.1.1 Query-1 : Camera with good lens

The result is shown in Fig 6.1.

6.1.2 Query-2 : Nikon with good lens

The result is shown in Fig 6.2.

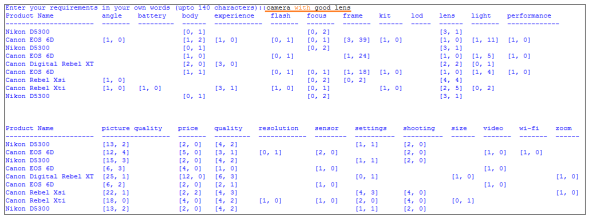


Figure 6.1: Query-1 : Camera with good lens

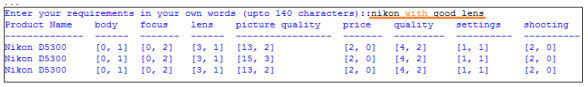


Figure 6.2: Query-2 : Nikon with good lens

6.1.3 Query-3 : Canon eos

The results are shown in Fig 6.3 and 6.4.

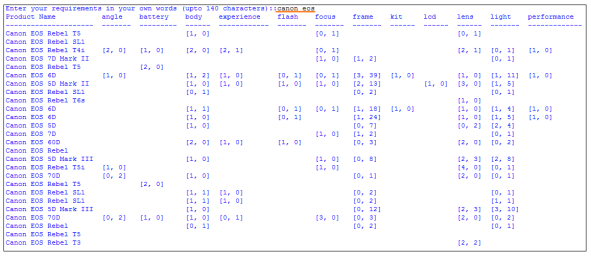


Figure 6.3: Query-3 : Canon eos(1)

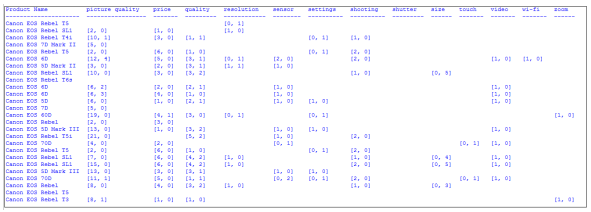
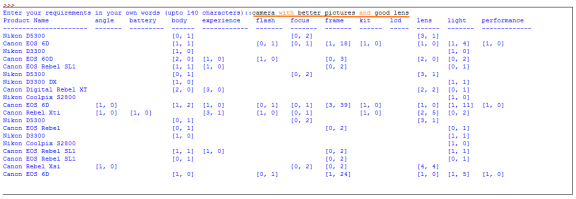


Figure 6.4: Query-3 : Canon eos(2)

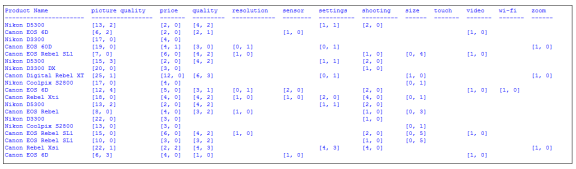
6.1.4 Query-4 : Camera with better pictures and good lens

The results are shown in Fig 6.5 and 6.6.



```
>>>
Enter your requirements in your own words (upto 140 characters):>>camera with better pictures and good lens
Product Name                angle   battery   body   experience   flash   focus   frame   kit    lcd   lens   light   performance
--------------------------  ------  --------  -----  ----------  ------  ------  ------  -----  ----  ------  ------  -----------
Nikon D5300                                   [0, 1]                      [0, 2]                       [3, 1]
Canon EOS 6D                                  [1, 1]               [0, 1] [0, 1]  [1, 18]  [1, 0]      [1, 0]  [1, 4]  [1, 0]
Nikon D3300                                   [1, 0]                                                    [1, 0]
Canon EOS 60D                                 [2, 0]  [1, 0]       [1, 0]          [0, 3]               [2, 0]  [0, 2]
Canon EOS Rebel SL1                           [1, 1]  [1, 0]                       [0, 2]                       [0, 1]
Nikon D5300                                   [0, 1]                      [0, 2]                       [3, 1]
Nikon D3300 DX                                [1, 0]                                                            [1, 1]
Canon Digital Rebel XT                        [2, 0]  [3, 0]                                               [2, 2]  [0, 1]
Nikon Coolpix S2600                                                                                            [1, 0]
Canon EOS 6D                [1, 0]            [1, 2]  [1, 0]       [0, 1] [0, 1]  [3, 39]  [1, 0]      [1, 0]  [1, 11] [1, 0]
Canon Rebel Xti             [1, 0]  [1, 0]            [0, 1]       [1, 0] [0, 1]          [1, 0]      [2, 5]  [0, 2]
Nikon D5300                                   [0, 1]                      [0, 2]                       [3, 1]
Canon EOS Rebel                               [0, 1]                                   [0, 2]                       [0, 1]
Nikon D3300                                   [1, 0]                                                            [1, 1]
Nikon Coolpix S2600                                                                                            [1, 0]
Canon EOS Rebel SL1                           [1, 1]  [1, 0]                           [0, 2]               [1, 1]
Canon EOS Rebel SL1                           [0, 1]                                   [0, 2]               [0, 1]
Canon Rebel Xti             [1, 0]                                          [0, 2]   [0, 2]               [4, 4]
Canon EOS 6D                                  [1, 0]               [0, 1]          [1, 24]              [1, 0]  [1, 5]  [1, 0]
```

Figure 6.5: Query-4 : Camera with better pictures and good lens(1)



```
Product Name                picture quality   price   quality   resolution   sensor   settings   shooting   size   touch   video   wi-fi   zoom
--------------------------  ---------------  ------  --------  -----------  -------  ---------  ---------  -----  ------  ------  ------  ------
Nikon D5300                 [13, 2]          [2, 0]  [6, 2]                          [1, 1]     [2, 0]
Canon EOS 6D                [6, 2]           [2, 0]  [2, 1]                 [1, 0]                                 [1, 0]
Nikon D3300                 [17, 0]          [4, 0]
Canon EOS 60D               [19, 0]          [4, 1]  [3, 0]    [0, 1]                   [0, 1]                                                  [1, 0]
Canon EOS Rebel SL1         [7, 0]           [6, 0]  [6, 2]    [1, 0]                                      [1, 0]     [0, 6]          [1, 0]
Nikon D5300                 [15, 3]          [2, 0]  [4, 2]                           [1, 1]     [2, 0]
Nikon D3300 DX              [20, 0]          [3, 0]                                              [1, 0]
Canon Digital Rebel XT      [26, 1]          [12, 0] [4, 9]                           [0, 1]                [1, 0]                            [1, 0]
Nikon Coolpix S2600         [17, 0]          [4, 0]                                                         [0, 1]
Canon EOS 6D                [12, 4]          [5, 0]  [3, 1]    [0, 1]       [2, 0]                      [2, 0]                 [1, 0]  [1, 0]
Canon Rebel Xti             [18, 0]          [4, 0]  [4, 2]    [1, 0]       [1, 0]     [2, 0]           [4, 0]     [0, 1]
Nikon D5300                 [13, 2]          [2, 0]  [4, 2]                           [1, 1]     [2, 0]
Canon EOS Rebel             [8, 0]           [4, 0]  [2, 2]    [1, 0]                            [1, 0]     [0, 8]
Nikon D3300                 [22, 0]          [3, 0]                                              [1, 0]
Nikon Coolpix S2600         [13, 0]          [3, 0]                                                         [0, 1]
Canon EOS Rebel SL1         [15, 0]          [6, 0]  [4, 2]    [1, 0]                            [2, 0]     [0, 5]          [1, 0]
Canon EOS Rebel SL1         [10, 0]          [3, 0]  [3, 2]                                      [1, 0]     [0, 5]
Canon Rebel Xsi             [22, 1]          [2, 2]  [5, 3]                           [4, 3]     [4, 0]                            [1, 0]
Canon EOS 6D                [6, 3]           [4, 0]  [1, 0]                 [1, 0]                                 [1, 0]
```

Figure 6.6: Query-4 : Camera with better pictures and good lens(2)

6.2 Future work

In this thesis work, the effect of negation in the review text is not considered. Considering the negation will vary the result as negation reverses the opinion. The accuracy of the result will increase. For generation of noun phrases, simple relation pattern is used. Use of more patterns can ensure capturing of more relations, resulting in good number of Noun Phrases. This will improve the result of opinion mining and summarization. Integrating these to the system developed can be the future work.

Bibliography

[1] Agrawal R. and Srikant R., Fast algorithm for mining association rules, VLDB94, 1994.

[2] Ana-Maria Popesu and Oren Etzioni, Extracting Product Features and Opinions from Reviews, Proceeding of Human Language Technology Conference and Conference on Empirical Methods in Natural Language, ACL, Vancouver, October 2005, pp. 339-346.

[3] Bing Liu,Lei Zhang A Survey of Opinion Mining and Sentiment Analysis

[4] Blair-Goldensohn, S., K. Hannan, R. McDonald, T. Neylon, G. A. Reis, and J. Reyna. Building a sentiment summarizer for local service reviews. In Proceedings of International Conference on World Wide Web Workshop of NLPIX (WWW-NLPIX-2008), 2008.

[5] Cataldo Musto, Giovanni Semeraro, Marco Polignano, A comparison of Lexicon-based approaches for Sentiment Analysis of microblog posts, Department of Computer Science University of Bari Aldo Moro, Italy

[6] Christopher Scaffidi, Application of a Probability-based Algorithm to Extraction of Product Features from Online Reviews, USA, 2006.

[7] Diana Maynard, Adam Funk. Automatic detection of political opinions in tweets. In: Proceedings of the 8th international conference on the semantic web, ESWC11; 2011. p. 8899.

[8] Erik Cambria and Amir Hussain. Sentic computing. Springer, 2012.

[9] Erik Cambria, Daniel Olsher, and Dheeraj Rajagopal. Senticnet 3: a common and common-sense knowledge base for cognition-driven sentiment analysis. AAAI,

Quebec City, pages 15151521, 2014.

[10] Hatzivassiloglou V, McKeown K. Predicting the semantic orientation of adjectives. In: Proceedings of annual meeting of the Association for Computational Linguistics (ACL97); 1997.

[11] Hu Minging, Liu Bing. Mining and summarizing customer reviews. In: Proceedings of ACM SIGKDD international conference on Knowledge Discovery and Data Mining (KDD04);2004.

[12] Hu, M. and B. Liu. Mining opinion features in customer reviews. In Proceedings of National Conference on Artificial Intelligence (AAAI-2004), 2004a.

[13] Kim S, Hovy E. Determining the sentiment of opinions. In: Proceedings of international conference on Computational Linguistics (COLING04); 2004.

[14] Ku, L., Y. Liang, and H. Chen. Opinion extraction, summarization and tracking in news and blog corpora. In Proceedings of AAAI-CAAW-2006, 2006.

[15] Liliana Ferreira, Niklas Jakob, Iryna Gurevych, "A Comparative Study of Feature Extraction Algorithms in Customer Reviews" icsc, pp.144-151, 2008 IEEE International Conference on Semantic Computing, 2008

[16] Liu B., Hsu W., Ma Y., Integrating Classification and Association Rule Mining, KDD98, 1998.

[17] Liu, B., 2010. Sentiment Analysis and Subjectivity. In: To Appear in Handbook of Natural Language Processing, Indurkhya, N. and F.J. Damerau (Eds.). 2nd Edn., University of Illinois, Chicago, USA., pp: 1-38.

[18] Long, C., J. Zhang, and X. Zhu. A review selection approach for accurate feature rating estimation. In Proceedings of International Conference on Computational Linguistics (COLING-2010), 2010.

[19] Miller G, Beckwith R, Fellbaum C, Gross D, Miller K. WordNet: an on-line lexical database. Oxford Univ. Press; 1990.

[20] Mingqing Hu and Bing Liu, Mining Opinion Features in Customer Reviews, American Association for Artificial Intelligence, 2004.

[21] Moghaddam, S. and M. Ester. Opinion digger: an unsupervised opinion miner from unstructured product reviews. In Proceedings of ACM International conference on Information and Knowledge Management (CIKM-2010), 2010.

[22] Mohammad S, Dunne C, Dorr B. Generating high-coverage semantic orientation lexicons from overly marked words and a thesaurus. In: Proceedings of the conference on Empirical Methods in Natural Language Processing (EMNLP09); 2009.

[23] O. Etzioni, M. Cafarella et al., Unsupervised named-entity extraction from the web: An experimental study, Artificial Intelligence, 2005, pp. 91-134.

[24] P. D. Turney, Mining the Web for Synonyms: PMI-IR versus LSA on ToEFL, In Procs. Of the 20th European Conference on Machine Learning (ECML2001),Freiburg Germany, pp. 491-502.

[25] Qiu Guang, He Xiaofei, Zhang Feng, Shi Yuan, Bu Jiajun, Chen Chun. DASA: dissatisfaction-oriented advertising based on sentiment analysis. Expert Syst Appl 2010;37:618291.

[26] Qiu, G., B. Liu, J. Bu, and C. Chen. Opinion word expansion and target extraction through double propagation. Computational Linguistics, 2011.

[27] Walaa Medhat ,, Ahmed Hassan , Hoda Korashy, Sentiment analysis algorithms and applications: A survey, Ain Shams Engineering Journal (2014)

[28] Yelena Mejova, Padmini Srinivasan. Exploring feature definition and selection for sentiment classifiers. In: Proceedings of the fifth international AAAI conference

on weblogs and social media; 2011.

[29]Zhu, J., H. Wang, B. K. Tsou, and M. Zhu. Multi-aspect opinion polling from textual reviews. In Proceedings of ACM International conference on Information and Knowledge Management (CIKM-2009), 2009.