1. INTRODUCTION

1.1 MOTIVATION

With the rapid expansion of e-commerce, more and more products are sold on the Web, and more and more people are buying products on the Web. In order to enhance customer satisfaction and their shopping experiences, it has become a common practice for online merchants to enable their customers to review or to express opinions on the products that they buy. With more and more common users becoming comfortable with the Internet, an increasing number of people are writing reviews. As a consequence, the number of reviews that a product receives grows rapidly Some popular products can get hundreds of reviews at some large merchant sites. This makes it very hard for a potential customer to read them to help him order to make a decision on whether to buy the product. In this research, we propose to study the problem of feature-based opinion summarization of customer reviews of products sold online. The task is performed in two steps:

1. Identify the features of the product that customers have expressed opinions on (called opinion features) and rank the features according to their frequencies that they appear in the reviews.

2. For each feature, we identify how many customer reviews have positive or negative opinions. The specific reviews that express these opinions are attached to the feature. This facilitates browsing of the reviews by potential customers.

A question that one may ask is "why not ask the merchant or the manufacturer of the product to provide a list of features?" This is a possible approach. However, it has a number of problems:

1. It is hard for a merchant to provide the features because he/she may sell a large number of products

2. The words used by merchants or the manufacturer may not be the same as those used by common users of the product although they may refer to the same features.

This causes problem in identifying what the customers are interested in. Furthermore, customers may comment on the lack of certain features of the product.

3. Customers may comment on some features that the manufacturer has never thought about, i.e., unexpected features.

4. The manufacturer may not want users of its product to know certain weak features

The large number of online review sites put a lot of useful and relevant information within a consumer’s reach. These reviews can be used to compare offerings by different competitors and consequently to make an informed decision about buying a particular offering. But, for a typical consumer, making this decision would turn out to be difficult for the following reasons:

1. The consumer might not be familiar with the various metrics used to compare

the offerings in that particular domain.

2. The consumer might have to read a lot of reviews to get an overview of the product and its features as reading just a few reviews might not help if they are all biased similarly. Therefore, it would turn out to be helpful if we can somehow

3. Pick out the right metrics that could be useful indicators of the product’s performance, specific to its domain.

4. Summarize the opinions about these important metrics which can be obtained from the large number of reviews into a couple of positive and negative points

These observations in turn led to my decision to develop a software tool that could do precisely what all are mentioned above.

1.2 PROBLEM DEFINITION

The problem is extracting the features form the reviews which are explicit is a difficult task because we need to disambiguate the sense of the reviews and then performing sentiment analysis. The main challenge here is to recognizing the false reviews which doesn’t mean the same as they look when we read. A human being can read and understand weather it is a false review or a genuine review but it is not same with automated machine or tool. More over after finding the state of review then we go for the repetition of the review i.e., how many times it is repeated and finding which product is trending. So this will be the problem we need to look in for.

1.3 OBJECTIVE OF THE PROJECT

At the highest level, the system accomplishes the following tasks:

1. Gather reviews about the product from online websites
2. Select a set of product features to rate on.
3. Determine the ratings for the selected features based on the sentiment of the sentence in which it appears.
4. Summarize the ratings for the features as the total number of positive and negative points for each of the review.
5. Draw a graph based on the summarized ratings and depicting the trend of a particular feature.

1.4 LIMITATIONS OF THE PROJECT

The main limitations of the project are like all opinions, Sentiment is inherently subjective from person to person and can even be outright irrational. It’s critical to mine a large and relevant sample of data when attempting to measure sentiment.no particular data point is necessarily relevant. It’s the aggregate that matters. An individual Sentiment toward a brand or product may be influenced by one or more indirect causes: someone might have a bad day and tweet a negative remark about something they otherwise had a pretty neutral opinion about. with a large enough sample, outliers are diluted in the aggregate. Also since sentiment very likely changes over time according to a person’s mood. So it’s always difficult to calculate sentiment based on these type of conditions.

2. LITERATURE SURVEY

2.1 INTRODUCTION

Reviews A review is an evaluation of a publication, service, or company such as a movie (a movie review), video game, musical composition (music review of a composition or recording), book (book review); a piece of hardware like a car, home appliance, or computer; or an event or performance, such as a live music concert, play, musical theater show, dance show, or art exhibition. In addition to a critical evaluation, the review's author may assign the work a rating to indicate its relative merit. A consumer review refers to a review written by a consumer for a product or a service based on her experience as a user of the reviewed product. A consumer review of a product usually comments on how well the product measures up to expectations based on the specifications provided by the manufacturer or seller.

Frequent Features Generation This step is to find features that people are most interested in. In order to do this, we use association rule mining each transaction consists of a subset of items in I. An association rule is an implication of the form $X \rightarrow Y$, where $X \subset I$, $Y \subset I$, and $X \cap Y = \emptyset$. The rule $X \rightarrow Y$ holds in D with confidence c if c% of transactions in D that support X also support Y. The rule has support s in D if s% of transactions in D contain $X \cup Y$. The problem of mining of transactions in D contain $X \cup Y$. The problem of mining association rules is to generate all association rules in D that have support and confidence greater than the user specified minimum support and minimum confidence. Mining frequent occurring phrases: Each piece of information extracted above is stored in a dataset called a transaction set/file. We then run the association rule mine CBA which is based on the Apriori algorithm. It finds all frequent item sets in the transaction set. Each resulting frequent item set is a possible feature. In our work, we define an item set as frequent if it appears in more than 1%(minimum support) of the review sentences. The A priori algorithm works in two steps. In the first step, it finds all frequent item sets from a set of transactions that satisfy a user-specified minimum support. In the second step, it generates rules from the discovered frequent item sets. For our task, we only need the first step, i.e. finding frequent item sets, which are candidate features. In addition, we only need to find frequent item sets with three words or fewer in this work as we believe that a product feature contains no more than three words (this restriction can be easily relaxed). The generated frequent item sets, which are also called candidate frequent features in this paper, are stored to the feature set for further processing.

Opinion Words Extraction We now identify opinion words. These are words that are primarily used to express subjective opinions. Clearly, this is related to existing work on distinguishing sentences used to express subjective opinions from sentences used to objectively. This opined features are converted into RDF notation so that they will be easily interpreted by the machine.

Orientation Identification for Opinion Words. For each opinion word, we need to identify its semantic orientation, which will be used to predict the semantic orientation of each opinion sentence. The semantic orientation of a word indicates the direction that the word deviates from the norm for its semantic group. Words that encode a desirable state (e.g., beautiful, awesome) have a positive orientation, while words that represent undesirable states have a negative orientation (disappointing). While orientations apply to many adjectives, there are also those adjectives that have no orientation (e.g., external, digital). In this work, we are interested in only positive and negative orientations. Unfortunately, dictionaries and similar sources, i.e., WordNet do not include semantic orientation information for each word. Hatzivassiloglou and McKeown use a supervised learning algorithm to infer the semantic orientation of adjectives from constraints on conjunctions. Although their method achieves high precision, it relies on a large corpus, and needs a large amount of manually tagged training data. In Turney s work, the semantic orientation of a phrase is calculated as the mutual information between the given phrase and the word “excellent” minus the mutual information between the given phrase and the word “poor”. The mutual information is estimated by issuing queries to a search engine and noting the number of hits. The paper, however, does not report the results of semantic orientations of individual words/phrases. Instead it only gives the classification results of reviews. We do not use these techniques in this paper as both works rely on statistical information from a rather big corpus.

Infrequent Feature Identification Frequent features are the "hot" features that people comment most about the given product. However, there are some features that only a small number of people talked about. These features can also be interesting to some potential customers and the manufacturer of the product. The question is how to extract these infrequent features (association mining is unable to identify such features)? Considering the following sentences: "The pictures are absolutely amazing." "The software that comes with it is amazing." Sentences 1 and 2 share the same opinion word amazing yet describing different features: sentence 1 is about the pictures, and sentence 2 is about the software. Since one adjective word can be used to describe different objects, we could use the opinion words to look for features that cannot be found in the frequent feature generation step using association mining. We extract infrequent features using the procedure in Figure 6: for each sentence in the review database if (it contains no frequent feature but one or more opinion words) find the nearest noun/noun phrase around the opinion word. The noun/noun phrase is stored in the feature set as an infrequent feature. In this project, we proposed a set of techniques for mining and summarizing product reviews based on data mining and natural language processing methods. The objective is to provide a language processing methods. The objective is to provide a language processing methods. The objective is to provide a feature-based summary of a large number of customer reviews of a product sold online. Our experimental results indicate that the proposed techniques are very promising in performing their tasks. We believe that this problem will become increasingly important as more people are buying and expressing their opinions on the Web. Summarizing the reviews is not only useful to common shoppers, but also crucial to product manufacturers.

2.2 EXISTING SYSTEM

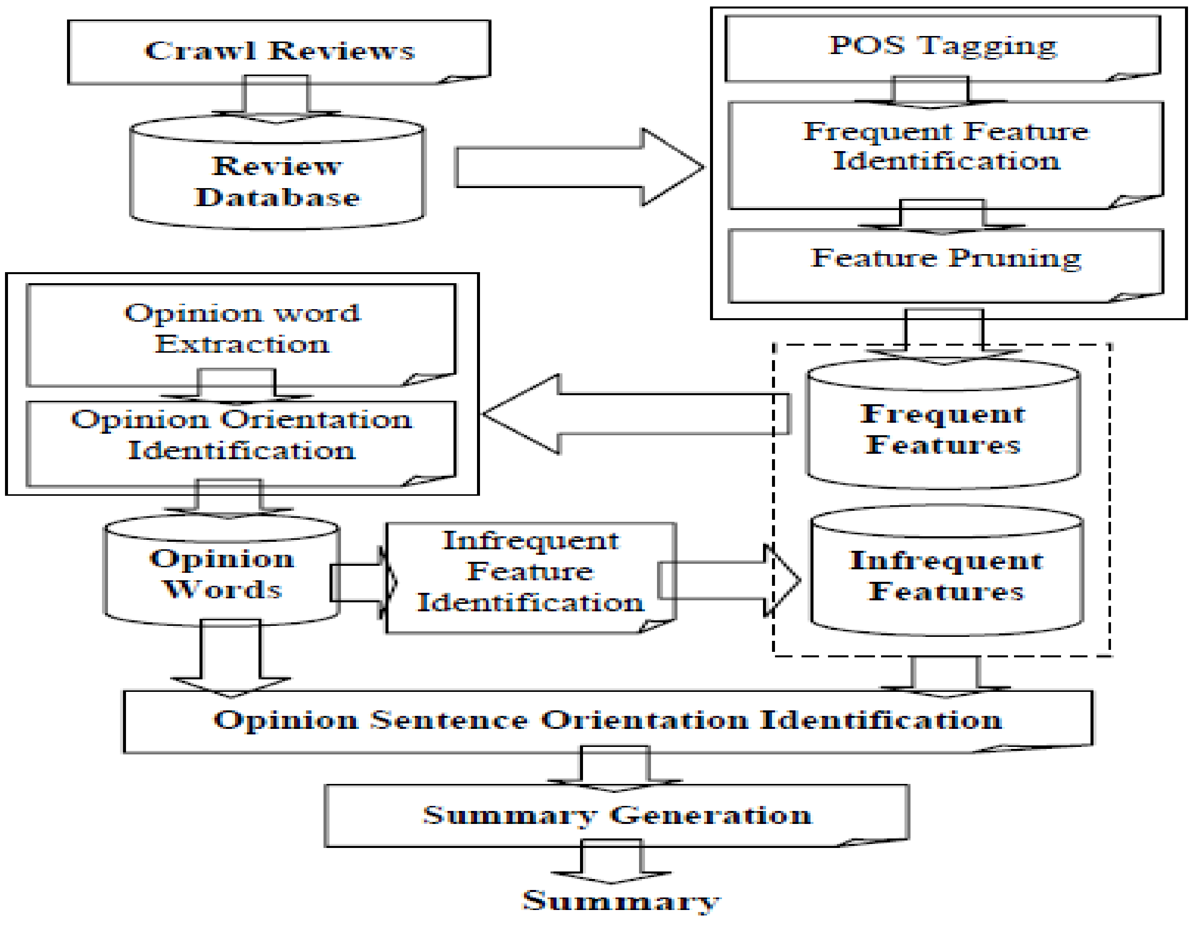


Fig. 2.2.1 Existing System Architecture

2.3 DISADVANTAGES OF EXISTING SYSTEM

1. However, the performance was limited because a sentence contains much less information than a review

2. They propose to use opinion-oriented “scenario templates” to act as summary representations of the opinions expressed in a document, or a set of documents

3. However, their system does not mine product features

4. However, using noun phrases tends to produce too many non-terms, while using reoccurring phrases misses many low frequency terms, terms with variations, and terms with only one work

2.4 PROPOSED SYSTEM

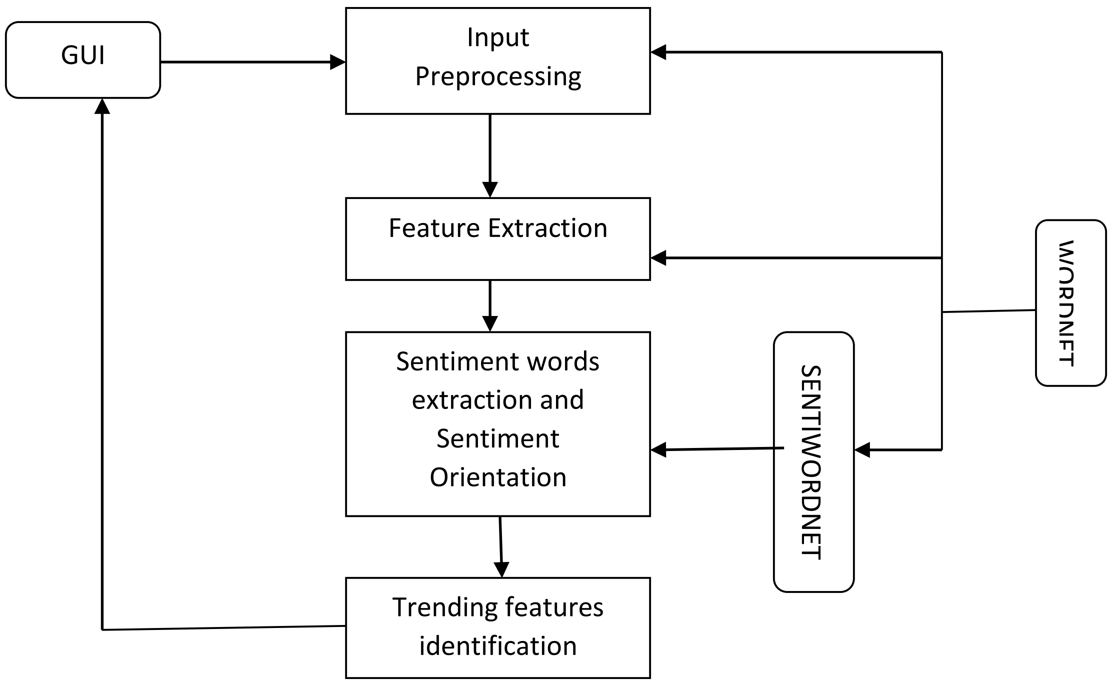

GUI
Input Preprocessing
Feature Extraction
Sentiment words extraction and Sentiment Orientation
Trending features identification
SENTIWORDNET
WORDNET

Fig. 2.4.1 Proposed System Architecture

The above gives an architectural overview for our opinion summarization and sentiment extraction of system along with trend analysis. The system performs the summarization in three main steps: feature extraction and sentiment orientation and finding the Trend. The inputs to the system are a product name and an entry page for all the reviews of the product. The output is the summary of the reviews as the one shown in the introduction section. Given the inputs, the system first downloads (or crawls) all the reviews, and puts them in the review database. The feature extraction function, which is the focus of this project, first extracts "hot" features that a lot of people have expressed their opinions on in their reviews, and then finds those infrequent ones. The opinion direction identification function takes the generated features and summarizes the opinions of the feature into 2 categories: positive and negative. POS tagging is the part-of-speech tagging (Manning and Schütze 1999) from natural tagging (Manning and Schütze 1999) from natural language processing. Then after finding the sentiment of particular reviews we perform to find out the particular trending feature which is the main or additional features to the previous works.

2.5 CONCLUSION

Therefore, we extract the features from the online reviews collected and we will extract sentiments for that extracted features and then we will form a set which will specify us the sentiment count for every feature. Thus we will know the quality of a product or item given in the review. In this paper, we proposed a number of techniques for mining opinion features from product reviews based on data mining and natural language processing methods. The objective is to produce a feature-based summary of a large number of customer reviews of a product sold online. We believe that this problem will become increasingly important as more people are buying and expressing their opinions on the Web. Our experimental results indicate that the proposed techniques are effective in performing their tasks. We also perform to group features according to the strength of the opinions that have been expressed on them, e.g., to determine which features customers strongly like and dislike along with trending of particular object at that particular time.

3. ANALYSIS

3.1 INTRODUCTION

The software requirement specification is produced at the culmination of the analysis task. The function and performance allocated to software as part of system engineering are refined by establishing a complete information description, a representation of system behavior, an indication of performance requirements and design constraints, appropriate validation criteria, and other information pertinent to requirements. Requirements Specification plays an important role to create quality software solution. Requirements are refined and analyzed to assess the clarity.

The introduction of the software requirements specification states the goals and objectives of the software, describing it in the context of the computer-based system. Actually the introduction may be nothing more than the software scope of the planning document.

The information description provides the detailed description of the problem that the software must solve. Information content, flow and structure are documented. Hardware, software and the human interfaces are described for the external system elements and internal software functions.

A description of each function required to solve the problem is presented in the functional description. A processing narrative is provided for each function, design constraints are stated and justified, performance characteristics are stated and one or more diagrams are included to graphically represent the overall structure of the software and interplay among software functions and other system elements.

3.2 SOFTWARE REQUIREMENT SPECIFICATION

There are many good definitions of System and Software Requirements Specifications that will provide us a good basis upon which we can both define a great specification and help us identify deficiencies in our past efforts. There is also a lot of great stuff on the web about writing good specifications. The problem is not lack of knowledge about how to create a correctly formatted specification or even what should go into the specification. The problem is that we don't follow the definitions out there.

We have to keep in mind that the goal is not to create great specifications but to create great products and great software. Can you create a great product without a great specification? Absolutely! You can also make your first million through the lottery – but why take your chances? Systems and software these days are so complex that to embark on the design before knowing what you are going to build is foolish and risky.

The IEEE (www.ieee.org) is an excellent source for definitions of System and Software Specifications. As designers of real-time, embedded system software, we use IEEE STD 830-1998 as the basis for all of our Software Specifications unless specifically requested by our clients. Essential to having a great Software Specification is having a great System Specification. The equivalent IEEE standard for that is IEEE STD 1233-1998. However, for most purposes in smaller systems, the same templates can be used for both.

Benefits of a SRS:

The IEEE 830 standard defines the benefits of a good SRS:

Establish the basis for agreement between the customers and the suppliers on what the software product is to do. The complete description of the functions to be performed by the software specified in the SRS will assist the potential users to determine if the software specified meets their needs or how the software must be modified to meet their needs. [NOTE: We use it as the basis of our contract with our clients all the time].

Reduce the development effort. The preparation of the SRS forces the various concerned groups in the customer’s organization to consider rigorously all of the requirements before design begins and reduces later redesign, recoding, and retesting.

Careful review of the requirements in the SRS can reveal omissions, misunderstandings, and inconsistencies early in the development cycle when these problems are easier to correct.

Provide a basis for estimating costs and schedules. The description of the product to be developed as given in the SRS is a realistic basis for estimating project costs and can be used to obtain approval for bids or price estimates. [NOTE: Again, we use the SRS as the basis for our fixed price estimates].

Provide a baseline for validation and verification. Organizations can develop their validation and Verification plans much more productively from a good SRS. As a part of the development contract, the SRS provides a baseline against which compliance can be measured. [NOTE: We use the SRS to create the Test Plan].

Facilitate transfer. SRS makes it easier to transfer the software product to new users or new machines. Customers thus find it easier to transfer the software to other parts of their organization, and suppliers find it easier to transfer it to new customers.

Serve as a basis for enhancement. Because the SRS discusses the product but not the project that developed it, the SRS serves as a basis for later enhancement of the finished product. The SRS may need to be altered, but it does provide a foundation for continued production evaluation.

Characteristics of a SRS:

a) Correct

b) Unambiguous

c) Complete

d) Consistent

e) Ranked for importance and/or stability

f) Verifiable

g) Modifiable

h) Traceable

a) Correct:

This is like motherhood and apple pie. Of course you want the specification to be correct. No one writes a specification that they know is incorrect. We like "Correct and Ever Correcting." The discipline is keeping the specification up to date when you find things that are not correct.

b) Unambiguous:

An SRS is unambiguous if, and only if, every requirement stated there in has only one interpretation. Again, easier said than done. Spending time on this area prior to releasing the SRS can be a waste of time. But as you find ambiguities - fix them.

c) Complete:

A simple judge of this is that it should be all that is needed by the software designers to create the software.

d) Consistent:

The SRS should be consistent within itself and consistent to its reference documents. If you call an input "Start and Stop" in one place, don't call it "Start/Stop" in another.

e) Ranked for Importance:

Very often a new system has requirements that are really marketing wish lists. Some may not be achievable. It is useful provide this information in the SRS.

f) Verifiable:

Don't put in requirements like - "It should provide the user a fast response". Another of my favorites is - "The system should never crash." Instead, provide a quantitative requirement like: "Every key stroke should provide a user response within 100 milliseconds."

g) Modifiable:

Having the same requirement in more than one place may not be wrong – butt ends to make the document not maintainable.

h) Traceable:

Often, this is not important in a non-politicized environment. However, inmost organizations, it is sometimes useful to connect the requirements in the SRS to a higher level document. Why do we need this requirement?

STAR UML

Star UML was an open source UML tool, licensed under a modified version of GNU GPL. After being abandoned for some time, the project had a last revival to move from Delphi to Java/Eclipse and then stopped again. However, the community is still active.

The stated goal of the project was to replace larger, commercial applications such as Rational Rose and Borland Together.

Star UML supports most of the diagram types specified in UML 2.0. It is currently missing object, package, timing and interaction overview diagrams (though the first two can be adequately modelled through the class diagram editor).

Star UML was written in Delphi, which is one of the reasons. A separate update called White Star UML hopes to address some of the outstanding problems.

Formats:

Star UML uses its own file format, with the. UML extension.

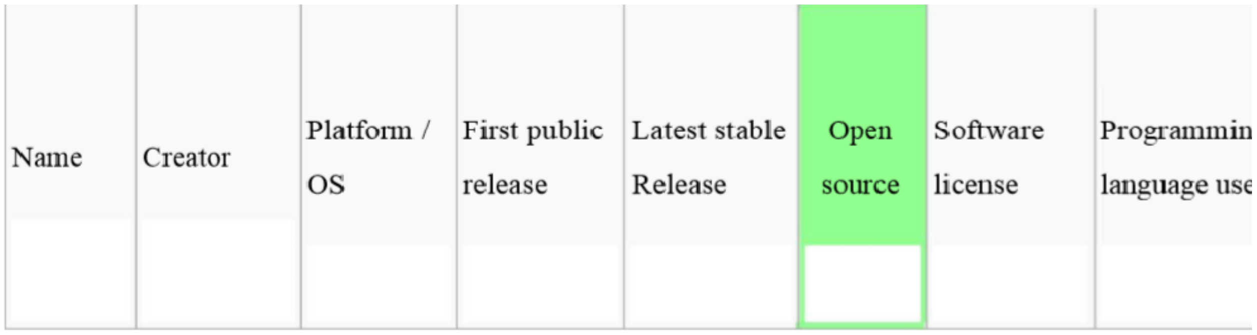
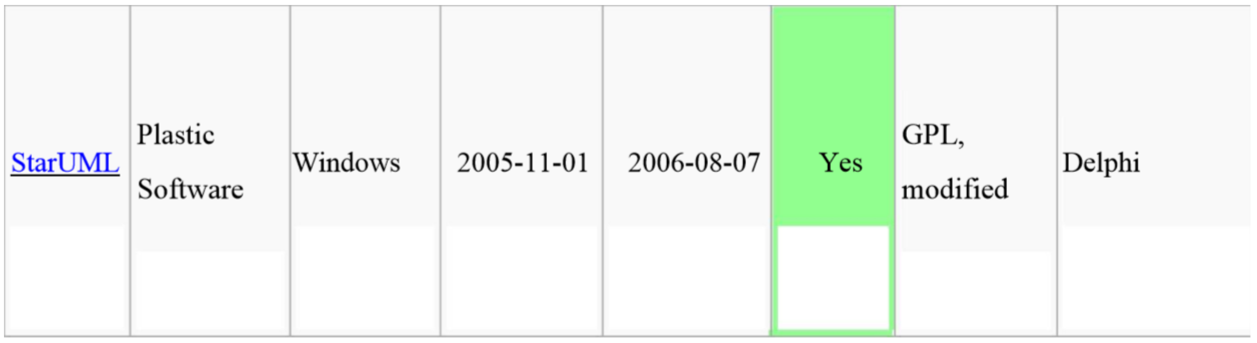
Exports:

• There is an export to raster image format, JPEG, and vector image format Windows Metafile.

• There is no export to SVG and PNG.

Imports:

There is an import from XMI, and

Name	Creator	Platform / OS	First public release	Latest stable Release	Open source	Software license	Programmin language use
StarUML	Plastic Software	Windows	2005-11-01	2006-08-07	Yes	GPL, modified	Delphi

3.2.1 USER REQUIREMENTS

OPERATING SYSTEM: Win XP and above

WEB BROWSER: Mozilla Firefox.

3.2.2 HARDWARE REQUIREMENTS

PROCESSOR: PENTIUM IV 2.6 GHz

RAM: 512 MB DD

RAM MONITOR: 15” COLOR

HARD DISK: 20 GB

CDDRIVE LG 52X

KEYBOARD: STANDARD 102 KEYS

3.2.3 SOFTWARE REQUIREMENTS

TOOLS USED: POS Tagger,SENTIWORD.NET

OPERATING SYSTEM: WINDOWS

LANGUAGES: JAVA

DATABASE: FILE SYSTEMS

3.3 ALGORITHMS

Linear search is the simplest search algorithm; it is a special case of brute-force search. Its worst case cost is proportional to the number of elements in the list. Its expected cost is also proportional to the number of elements if all elements are searched equally. If the list has more than a few elements and is searched often, then more complicated search methods such as binary search or hashing may be appropriate. Linear search is usually very simple to implement, and is practical when the list has only a few elements, or when performing a single search in an unordered list.

1. Get the features from feature set.
2. Get the next review to be mined.
3. Compare each and every word in review with required word.
4. If they are equal, then remove or save it as per corresponding process.
5. Repeat step 2 to 4 until all reviews are processed.

Apriori Algorithm

Apriori is an algorithm for frequent item set mining and association rule learning over transactional databases. It proceeds by identifying the frequent individual items in the database and extending them to larger and larger item sets as long as those item sets appear sufficiently often in the database. The frequent item sets determined by Apriori can be used to determine association rules which highlight general trends in the database: this has applications in domains such as market basket analysis. Apriori algorithm has two steps in which it is used to generate frequent feature item sets.

1. Find minimum support and confidence values and give the values as input to apriori algorithm.

2. Join Step: C k is generated by joining Lk-1with itself.

3. Prune Step: Any (k-1)-item set that is not frequent cannot be a subset of a frequent k item set.

4. Repeat steps 2 and 3 until no item sets are remaining to mine.

Along with these two there are totally six various steps in finding the sentiment of word. These six steps involve six different algorithms which are discussed below.

Step-1:

XML Processing:

There will be an XML file created for storing all the features of any particular product that is available readily on the internet. In this module the XML file that contains all the features of the product are processed to extract the features excluding all the XML tags.

- The XML file containing the features of the product are given as input to the program.
- The program processes the XML file for the value present in the set of tags.
- Then the program excludes the tags and then extracts the values present between the tags which become the actual features of the product.
- Now, the program creates a text file for storing the features.
- The features that are extracted are stored in the text file that is created.

Step-2:

Pre -Processing:

All the stop words present in a given input text file which consists of selected product reviews are removed.

- The text file that contains all the selected product reviews is given as input to the program.
- The program processes the text file and then splits each word of the file.
- Each word of the file is compared with the set of stop words.
- If the word that is present in the file is a stop word, then it is eliminated.
- Since all the stop words are removed the resultant file has no stop words.
- The program creates a file and then writes all the reviews that do not contain the stop words.

Step-3:

Parts of speech tagging:

Each word in the text file is tagged with the respective parts of speech.

- The output text file from the step-2 is given to the step-3 for parts of speech tagging.
- In this step the Stanford-postagger.jar file is used to tag the parts of speech of any given text.
- The .jar file can be accessed by using a command in the command prompt.
- The command is embedded in a java program and use process builder that opens the command prompt and directs to the path that is given where the Stanford-POStagger is present.
- The Stanford-POStagger hence tags the content of the text file containing the reviews.
- The output is then written to a text file.

Step-4:

Discretization:

The adjectives and nouns that are present in the tagged file are separated and stored in two different files.

- The output file from step-3 is dynamically given to step-4 for discretization.
- The Nouns and Adjectives that are present in the tagged file are separated in this step.
- The program checks each element of the file and then retrieves the nouns in the file.
- The Nouns that are retrieved are stored in a text file that is created by the program.

Step-5:

Frequent Noun Generation:

The frequently repeated Nouns are identified and stored in a file.

- The Nouns file from step-3 is dynamically given to step-4 for generating frequent Nouns
- The Nouns that are repeated thrice or more than thrice are identified in this step.
- The nouns are compared with themselves and the number of times the nouns are repeated calculated.
- Depending on the repetitions of the noun, the nouns that are repeated more than thrice are retrieved which are called the frequent nouns.
- The frequent nouns that are retrieved are stored in a text file.

Step-6:

Relevant Noun Generation:

The nouns that are relevant to the product features are extracted.

- The Frequent nouns that are generated in step-5 and the Actual features that are generated in step-1 are dynamically given to step-6 for generating Relevant nouns.
- The actual features are compared with the frequent features and the relevant features are retained eliminating the unwanted and irrelevant features.
- The relevant features thus generated are then written to a text file by the program.

Step-7:

Sentiment word extraction:

In this module, sentiment of the generated nouns and adjectives are retrieved in this module with the help of sentiword.net.

- The relevant features that are retrieved by processing the reviews are then compared with the corresponding adjectives that are generated in the step-3 from the reviews.
- The adjectives are checked for the sentiment using the Sent WordNet tool which consists of the sentiments and meanings of the words.
- The determination of the sentiment of the adjectives is done manually.
- The sentiment values are noted down in an excel sheet for further reference.

Step-8:

Trend Analysis:

In this step the sentiment values which are calculated using Sent WordNet are processed and the trending features are identified.

- The sentiment values that are noted down in the excel sheet are processed.
- The number of positive, negative and neutral values for each feature are then noted down into another excel sheet and their average values are calculated.
- Using appropriate settings, we draw a horizontal bar graph.
- The bar graph contains the feature and the corresponding number of negative and positive scores in graphical format.
- Since the average of the values are known we set the average value as a threshold value.
- The features that have positive count equal or more than the threshold value are considered as trending features.

3.4 OVERVIEW OF TECHNOLOGIES USED

3.4.1 POS TAGGER

A Part-Of-Speech Tagger (POS Tagger) is a piece of software that reads text in some language and assigns parts of speech to each word (and other token), such as noun, verb, adjective, etc., although generally computational applications use more fine-grained POS tags like 'noun-plural'. This software is a Java implementation of the log-linear part-of speech taggers described in these papers (if citing just one paper, cite the 2003 one): Kristina Toutanova and Christopher D. Manning. 2000. Enriching the Knowledge Sources Used in a Maximum Entropy Part-of-Speech Tagger. In Proceedings of the Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora (EMNLP/VLC-2000), pp. 63-70. Kristina Toutanova, Dan Klein, Christopher Manning, and Yoram Singer. 2003. Feature-Rich Part-of-Speech Tagging with a Cyclic Dependency Network. In Proceedings of HLT-NAACL 2003, pp. 252-259. The tagger was originally written by Kristina Toutanova. Since that time, Dan Klein, Christopher Manning, William Morgan, Anna Rafferty, Michel Galley, and John Bauer have improved its speed, performance, usability, and support for other languages.

The system requires Java 1.8+ to be installed. Depending on whether you're running 32 or 64-bit Java and the complexity of the tagger model, you'll need somewhere between 60 and 200 MB of memory to run a trained tagger (i.e., you may need to give java an option like java -mx200m). Plenty of memory is needed to train a tagger. It again depends on the complexity of the model but at least 1GB is usually needed, often more.

Number	**Tag**	**Description**
1.	CC	Coordinating conjunction
2.	CD	Cardinal number
3.	DT	Determiner
4.	EX	Existential there
5.	FW	Foreign word
6.	IN	Preposition or subordinating conjunction
7.	JJ	Adjective
8.	JJR	Adjective, comparative
9.	JJS	Adjective, superlative
10.	LS	List item marker
11.	MD	Modal
12.	NN	Noun, singular or mass
13.	NNS	Noun, plural
14.	NNP	Proper noun, singular
15.	NNPS	Proper noun, plural
16.	PDT	Pre determiner
17.	POS	Possessive ending
18.	PRP	Personal pronoun
19.	PRP$	Possessive pronoun
20.	RB	Adverb
21.	RBR	Adverb, comparative

22.	RBS	Adverb, superlative
23.	RP	Particle
24.	SYM	Symbol
25.	To	to
26.	UH	Interjection
27.	VB	Verb, base form
28.	VBD	Verb, past tense
29.	VBG	Verb, gerund or present participle
30.	VBN	Verb, past participle
31.	VBP	Verb, non-3rd person singular present
32.	VBZ	Verb, 3rd person singular present
33.	WDT	Wh-determiner
34.	WP	Wh-pronoun
35.	WP$	Possessive Wh-pronoun
36.	WRB	Wh-adverb

3.4.2 SENTENCE SEGMENTATION

Sentence segmentation is the process of dividing written text into meaningful units, such as words, sentences, or topics. The term applies both to mental processes used by humans when reading text, and to artificial processes implemented in computers, which are the subject of natural language processing. The problem is non-trivial, because while some written languages have explicit word boundary markers, such as the word spaces of written English and the distinctive initial, medial and final letter shapes of Arabic, such signals are sometimes ambiguous and not present in all written languages.

3.4.3 SENTIWORDNET

SENTIWORDNET 3.0, is an enhanced lexical resource explicitly devised for supporting sentiment classification and opinion mining applications (Pang and Lee, 2008). SENTIWORDNET 3.0 is an improved version of SENTIWORDNET 1.0 (Esuli and Sebastiani, 2006), a lexical resource publicly available for research purposes, now currently licensed to more than 300 research groups and used in a variety of research projects worldwide. SENTIWORDNET is the result of the automatic annotation of all the synsets of WORDNET according to the notions of “positivity”, “negativity”, and “neutrality”. Each synset s is associated to three numerical scores Pos(s), Neg(s), and Obj(s) which indicate how positive, negative, and “objective” (i.e., neutral) the terms contained in the synset are. Different senses of the same term may thus have different opinion-related properties. Each of the three scores ranges in the interval [0.0, 1.0], and their sum is 1.0 for each synset. This means that a synset may have nonzero scores for all the three categories, which would indicate that the corresponding terms have, in the sense indicated by the synset, each of the three opinion related properties to a certain degree.

3.5 Content Diagram of project

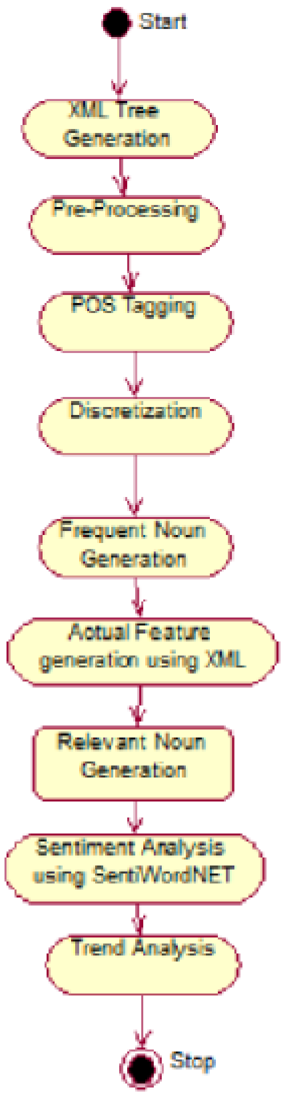


Fig. 3.5.1 Content Diagram

The above diagram represents the content of the project or steps involved to complete the execution. First six steps are already mentioned in the algorithms and the remaining two steps are finding Sentiment Analysis and Trend Analysis. Sentiment Analysis is done by using Sentiword.net and then the trending feature is extracted from the generated graph.

4.DESIGN

4.1 Introduction

UML diagram UML is a method for describing the system architecture in detail using the blueprint. UML represents a collection of best engineering practices that have proven successful in the modeling of large and complex systems. The UML is a very important part of developing objects oriented software and the software development process. The UML uses mostly graphical notations to express the design of software projects.

Using the UML helps project teams to communicate explore potential designs, and validate the architectural design of the software DEFINITION: UML is a general purpose visual modeling language that is used to specify, visualize, construct and document the Arti Crafts of the software system.

UML is a language:

It will provide vocabulary and rules for communication and function on conceptual and physical representation. So it is modeling language.

UML specifying:

Specifying means building models that are precise, unambiguous and complete. In particular, the UML address the specification of all the important analysis, design and implementation decisions that must be made in developing and displaying a software intensive system.

UML visualization:

The UML includes both graphical and textual representation. It makes easy to visualize the system for better understanding.

.

UML constructing:

UML models can be directly connected to a variety of programming languages and it is sufficiently expressive and free from any ambiguity to permit the direct execution of models.

UML documenting:

UML provides variety of documents in addition raw executable codes.

Goals of UML:

The primary goals in the design of the UML were:

1. Provide users with a ready-to-use, expressive visual modeling language so they can develop and exchange meaningful models.
2. Provide extensibility and specialization mechanisms to extend the core concepts.
3. Be independent of particular programming languages and development processes.
4. Provide a formal basis for understanding the modeling language.
5. Encourage the growth of the OO tools market.
6. Support higher-level development concepts such as collaborations, frameworks, patterns and components.
7. Integrate best practices.

Uses of UML:

Information system

Handle large amounts of data with complex relationships, which are stored in relational or object databases

Technical system

Handle control technical equipment such as telecommunications, military systems, or industrial processes. They must handle the special interfaces of the equipment and have less standard software than Information system. There is often real-time system.

Embedded real-time systems

Execute on simple hardware embedded in some other equipment such as a mobile phone, car, household appliance, etc.

Distributed systems

Distributed on a number of machines where data Is transferred easily from one machine to another. They require synchronized communication mechanisms to ensure data integrity and are often built upon object mechanisms such as CORBA, COM/DCOM, or Java Beans/RMI

System software

Define the technical infrastructure that other software uses. Operating system. Databases.

Business system

Describe the goals, the resources (human, computers), the rules (laws, business strategies, policies), and the actual work in the business

Rules of UML:

1. Specify what a well-formed model should look like.

2. The UML has semantic rules for

   (1) Names: It will call things, relationships and diagrams

   (2) Scope: The content that gives specific meaning to the name

   (3) Visibility: How those names can be seen and used by others

   (4) Integrity: How things properly and consistently relate to one another

   (5) Execution: To run or simulate a dynamic model.

BUILDING BLOCKS OF UML:

The building blocks of UML can be defined as:

- Things
- Relationships
- Diagrams

(1) Things:

Things are the most important building blocks of UML. Things can be:

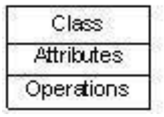
- Structural
- Behavioral
- Grouping
- Annotational

Structural things:

The **Structural things** define the static part of the model. They represent physical and conceptual elements. Following are the brief descriptions of the structural things.

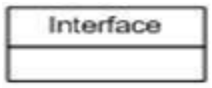
Class:

Class represents set of objects having similar responsibilities.



Interface:

Interface defines a set of operations which specify the responsibility of a class.

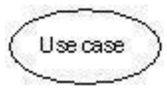


Collaboration:

Collaboration defines interaction between elements.

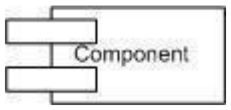
Use case:

Use case represents a set of actions performed by a system for a specific goal.



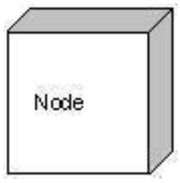
Component:

Component describes physical part of a system.



Node:

A node can be defined as a physical element that exists at run time.

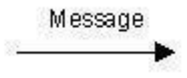


Behavioral things:

A behavioral thing consists of the dynamic parts of UML models. Following are the behavioral things:

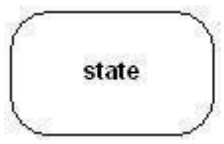
Interaction:

Interaction is defined as a behavior that consists of a group of messages exchanged among elements to accomplish a specific task.



State machine:

State machine is useful when the state of an object in its life cycle is important. It defines the sequence of states an object goes through in response to events. Events are external factors responsible for state change

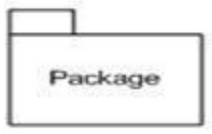


Grouping things:

Grouping things can be defined as a mechanism to group elements of a UML model together. There is only one grouping thing available:

Package:

Package is the only one grouping thing available for gathering structural and behavioral things.

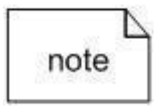


Annotational things:

Annotational things can be defined as a mechanism to capture remarks, descriptions, and comments of UML model elements. **Note** is the only one annotational thing available.

Note:

A note is used to render comments, constraints etc., of an UML element.

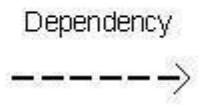


(2) Relationships:

Relationship is another most important building block of UML. It shows how elements are associated with each other and this association describes the functionality of an application. There are four kinds of relationships available.

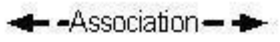
Dependency:

Dependency is a relationship between two things in which change in one element also affects the other one.



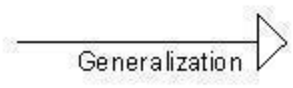
Association:

Association is basically a set of links that connects elements of an UML model. It also describes how many objects are taking part in that relationship.



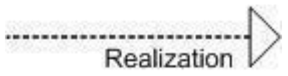
Generalization:

Generalization can be defined as a relationship which connects a specialized element with a generalized element. It basically describes inheritance relationship in the world of objects.



Realization:

Realization can be defined as a relationship in which two elements are connected. One element describes some responsibility which is not implemented and the other one implements them. This relationship exists in case of interfaces.



(3) Diagrams:

UML diagrams are the ultimate output of the entire discussion. All the elements, relationships are used to make a complete UML diagram and the diagram represents a system. The visual effect of the UML diagram is the most important part of the entire process. All the other elements are used to make it a complete one.

UML includes the following nine diagrams and the details are described in the following chapters.

1. Class diagram
2. Object diagram
3. Use case diagram
4. Sequence diagram
5. Collaboration diagram
6. Activity diagram
7. State chart diagram

CLASS DIAGRAM:

Overview:

The class diagram is a static diagram. It represents the static view of an application. Class diagram is not only used for visualizing, describing and documenting different aspects of a system but also for constructing executable code of the software application.

The class diagram describes the attributes and operations of a class and also the constraints imposed on the system. The class diagrams are widely used in the modelling of object oriented systems because they are the only UML diagrams which can be mapped directly with object oriented languages

The class diagram shows a collection of classes, interfaces, associations, collaborations and constraints. It is also known as a *structural diagram*.

Purpose:

The purpose of the class diagram is to model the static view of an application. The class diagrams are the only diagrams which can be directly mapped with object oriented languages and thus widely used at the time of construction.

The UML diagrams like activity diagram, sequence diagram can only give the sequence flow of the application but class diagram is a bit different. So it is the most popular UML diagram in the coder community.

So the purpose of the class diagram can be summarized as:

- Analysis and design of the static view of an application.
- Describe responsibilities of a system.
- Base for component and deployment diagrams.
- Forward and reverse engineering.

Class diagrams are used for:

- Describing the static view of the system.
- Showing the collaboration among the elements of the static view.
- Describing the functionalities performed by the system.
- Construction of software applications using object oriented languages.

4.2 UML DIAGRAMS

Fig. 4.2.1. Use Case Diagram

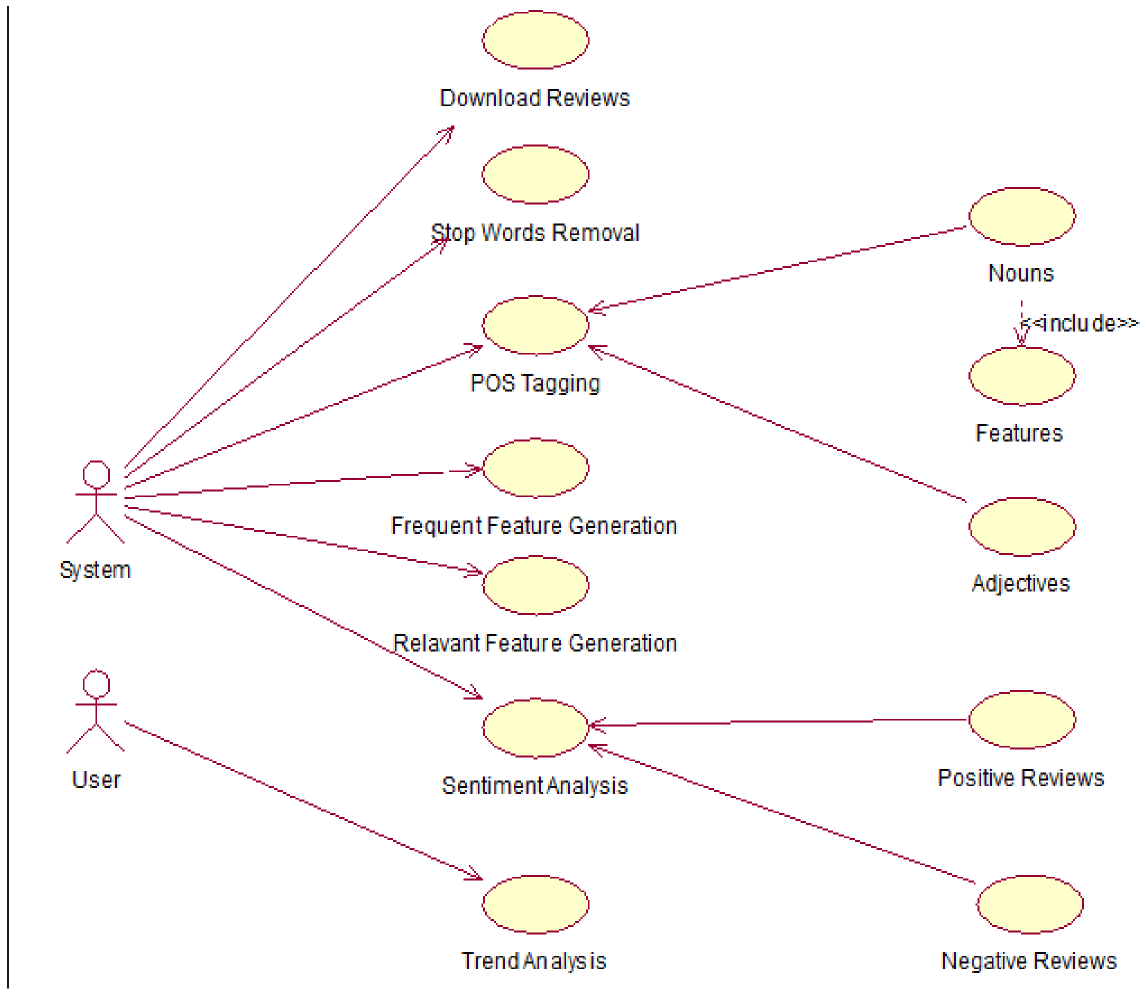


Fig. 4.2.2 Activity Diagram

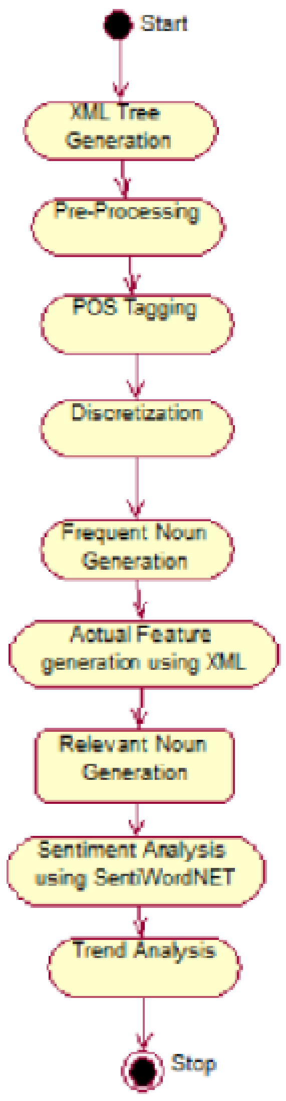


Fig. 4.2.3 Class Diagram

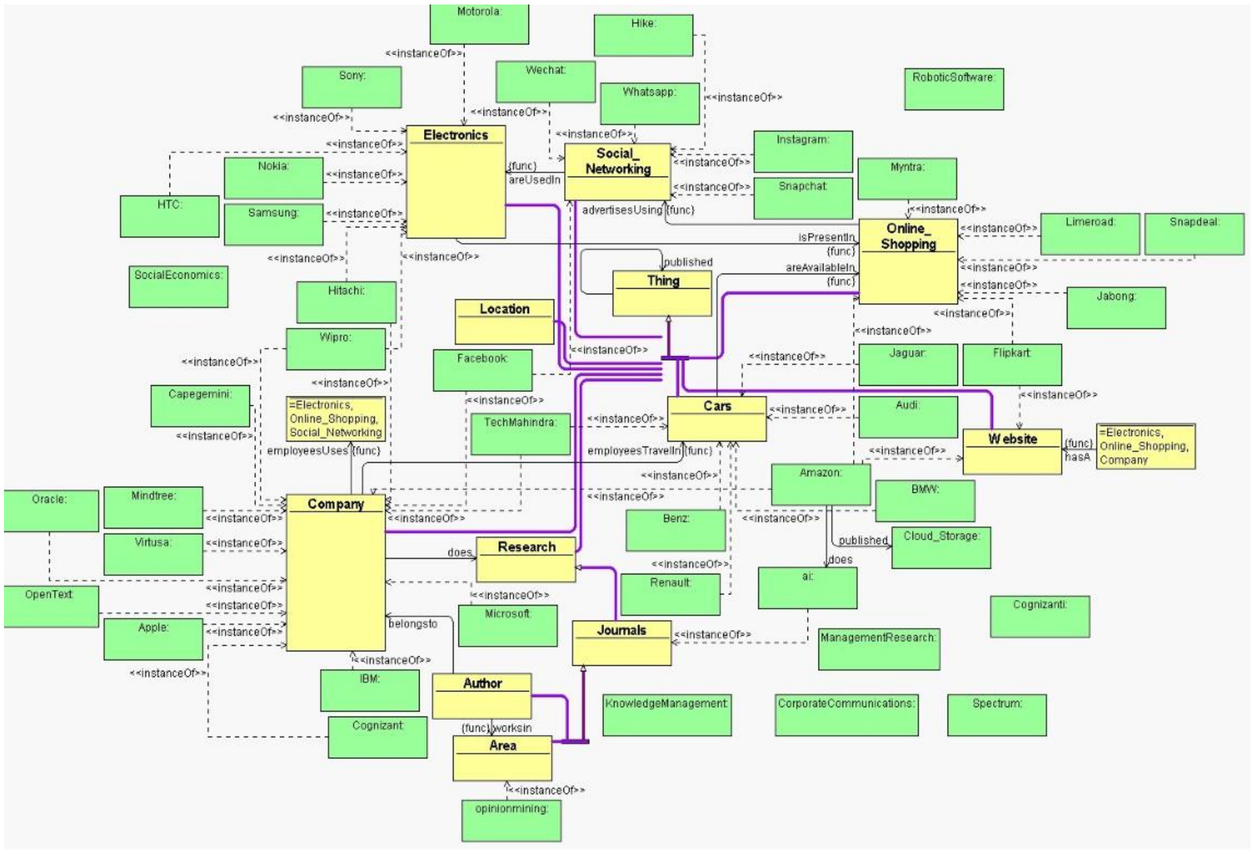


Fig. 4.2.4 Sequence Diagram

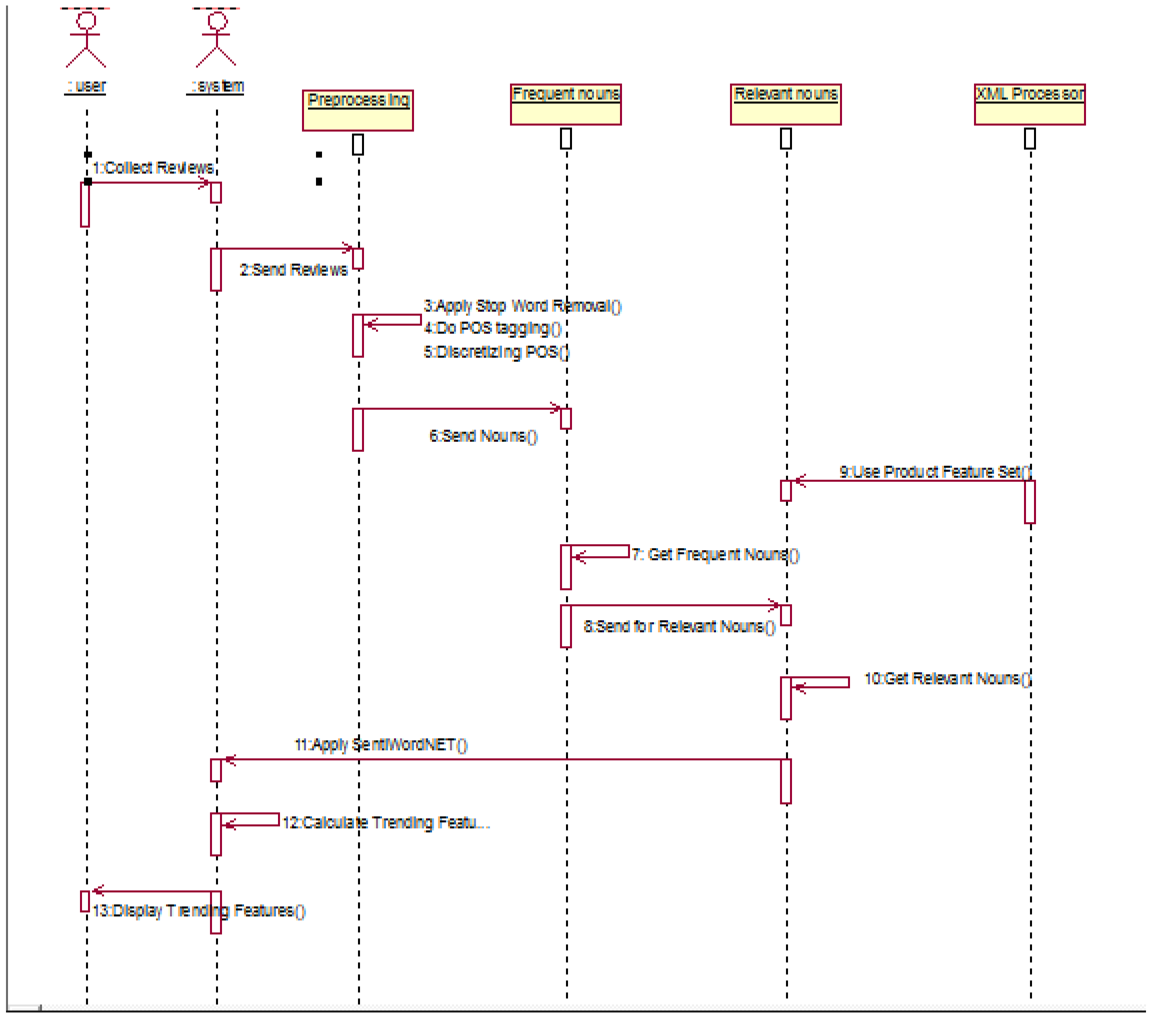


Fig. 4.2.5 Component Diagram

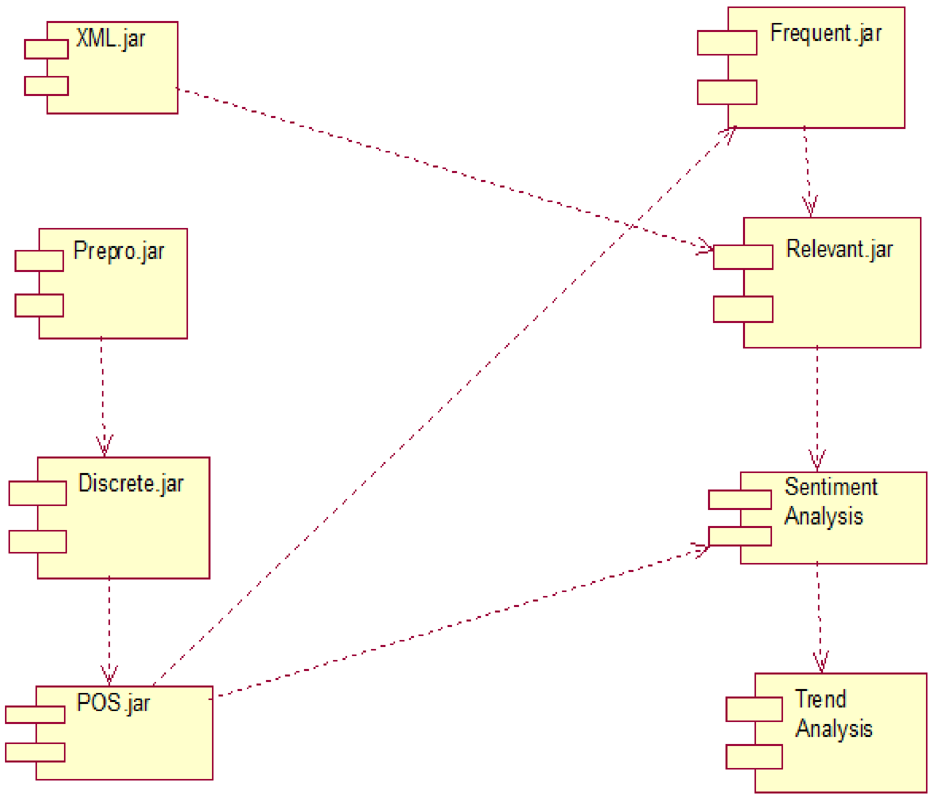
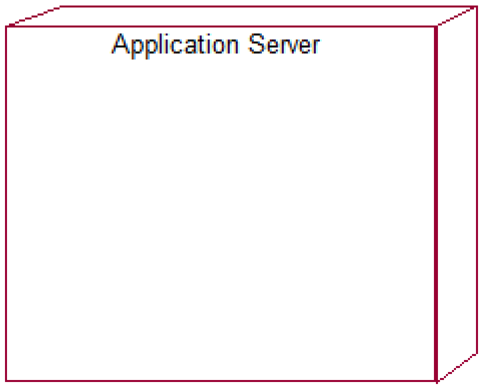


Fig. 4.2.6 Deployment Diagram



4.3 MODULE DESIGN AND ORGANIZATION

Module-1:

XML Processing:

There will be an XML file created for storing all the features of any particular product that is available readily on the internet. In this module the XML file that contains all the features of the product are processed to extract the features excluding all the XML tags.

- The XML file containing the features of the product are given as input to the program.
- The program processes the XML file for the value present in the set of tags.
- Then the program excludes the tags and then extracts the values present between the tags which become the actual features of the product.
- Now, the program creates a text file for storing the features.
- The features that are extracted are stored in the text file that is created.

Module-2:

Pre -Processing:

All the stop words present in a given input text file which consists of selected product reviews are removed.

- The text file that contains all the selected product reviews is given as input to the program.
- The program processes the text file and then splits each word of the file.
- Each word of the file is compared with the set of stop words.
- If the word that is present in the file is a stop word, then it is eliminated.
- Since all the stop words are removed the resultant file has no stop words.
- The program creates a file and then writes all the reviews that do not contain the stop words.

Module-3:

Parts of speech tagging:

Each word in the text file is tagged with the respective parts of speech.

- The output text file from the step-2 is given to the step-3 for parts of speech tagging.
- In this step the Stanford-postagger.jar file is used to tag the parts of speech of any given text.
- The .jar file can be accessed by using a command in the command prompt.
- The command is embedded in a java program and use process builder that opens the command prompt and directs to the path that is given where the Stanford-POStagger is present.
- The Stanford-POStagger hence tags the content of the text file containing the reviews.
- The output is then written to a text file.

Module-4:

Discretization:

The adjectives and nouns that are present in the tagged file are separated and stored in two different files.

- The output file from step-3 is dynamically given to step-4 for discretization.
- The Nouns and Adjectives that are present in the tagged file are separated in this step.
- The program checks each element of the file and then retrieves the nouns in the file.
- The Nouns that are retrieved are stored in a text file that is created by the program.

Module-5:

Frequent Noun Generation:

The frequently repeated Nouns are identified and stored in a file.

- The Nouns file from step-3 is dynamically given to step-4 for generating frequent Nouns
- The Nouns that are repeated thrice or more than thrice are identified in this step.
- The nouns are compared with themselves and the number of times the nouns are repeated calculated.
- Depending on the repetitions of the noun, the nouns that are repeated more than thrice are retrieved which are called the frequent nouns.
- The frequent nouns that are retrieved are stored in a text file.

Module-6:

Relevant Noun Generation:

The nouns that are relevant to the product features are extracted.

- The Frequent nouns that are generated in step-5 and the Actual features that are generated in step-1 are dynamically given to step-6 for generating Relevant nouns.
- The actual features are compared with the frequent features and the relevant features are retained eliminating the unwanted and irrelevant features.
- The relevant features thus generated are then written to a text file by the program.

Module-7:

Sentiment word extraction:

In this module, sentiment of the generated nouns and adjectives are retrieved in this module with the help of sentiword.net.

- The relevant features that are retrieved by processing the reviews are then compared with the corresponding adjectives that are generated in the step-3 from the reviews.
- The adjectives are checked for the sentiment using the Sent WordNet tool which consists of the sentiments and meanings of the words.
- The determination of the sentiment of the adjectives is done manually.
- The sentiment values are noted down in an excel sheet for further reference.

Module-8:

Trend Analysis:

In this step the sentiment values which are calculated using Sent WordNet are processed and the trending features are identified.

- The sentiment values that are noted down in the excel sheet are processed.
- The number of positive, negative and neutral values for each feature are then noted down into another excel sheet and their average values are calculated.
- Using appropriate settings, we draw a horizontal bar graph.
- The bar graph contains the feature and the corresponding number of negative and positive scores in graphical format.
- Since the average of the values are known we set the average value as a threshold value.
- The features that have positive count equal or more than the threshold value are considered as trending features.

5. IMPLEMENTATION AND TESTING

5.1 INTRODUCTION

Implementation literally means to put into effect or to carry out. In the system implementation phase, the software deals with translation of the design specifications into source code. The ultimate goal of the implementation is to write the source code and the internal documentation so that it can be verified easily. The code and documentation should be written in a manner that eases debugging, testing and modification. System flow charts, simple run on packages, sample output etc., is part of the implementation. An effort was made to satisfy the following goals in order:

1. Clarity and simplicity of the code
2. Minimization of Hard Coding
3. Minimization of the amount of memory used
4. Thorough phased implementation has been done so that we can use our proposed system correctly.

5.2 EXPLANATION OF KEY FUNCTIONS

Key functions of the project are:

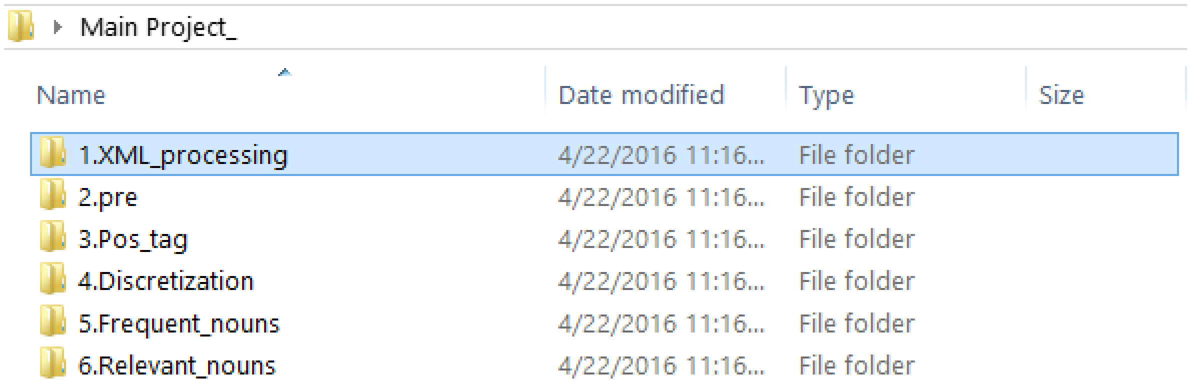
5.2.1 XML Processing
5.2.2 PRE Processing
5.2.3 POS-Tagging
5.2.4 Discretization
5.2.5 Frequent Noun Generation
5.2.6 Relevant Noun Generation
5.2.7 Sentiment Word Extraction
5.2.8 Trend Analysis

INPUT REVIEWS

1 audio , bluetooth and price are awesome .
2 it has good audio output with cheap price and fast bluetooth .
3 good bluetooth and good music at this good price .
4 bluetooth and music are not worth buying with this reasonless price .
5 rich sound quality with a reasonable price and blue tooth .
6 blue tooth is slow , music is not convenient ,price is not worth buying .
7 price is not high but this output sound is extremely bad .
8 affordable phone but it is has poor sound output .
9 this is very comparable and cheaper budget alternative to the s4 the music features were defective .
10 music is very slow and get stuck sometimes and the size isn't good enough .
11 audio is very low but has a good user interface for beginners .
12 galaxy grand has good audio with a powerful bluetooth .
13 very cool audio features and bluetooth is awsome .
14 extremely low sound and touch response is very good .
15 audio response is not good .
16 music features are magnificant .
17 excellent audio .
18 audio is acceptable .
19 i am very impressed with loud sound feature .
20 camera and display are functioning awesome with great clarity , the phone has got good size also .
21 size is not satisfactory , camera and display are lively .
22 camera takes good pictures and pictures are display is good .
23 picture clarity is not good and display works bad sometimes .
24 the camera is acceptable with pretty good flash .
25 perfect camera with brilliant flash light .
26 camera not agreeable but flash is working good .
27 the camera isn't the best at 5 mp and it has bad audio .
28 camera takes good pictures .
29 the camera takes very colorful photos .
30 camera worked great and pictures were wonderful .
31 the battery only lasts half day and its disappointing , memory is not good enough , keypad is a little tight .

32 battery has been disappointing memory is erratic , great screen .
33 battery has bad issues capability to hold two sim cards is superb installation of applications is inconsistent .
34 low batter life but good dual sim feature and clear call clarity .
35 worst memory and battery because they always low .
36 poor battery life and memory .
37 display is nice but erratic battery .
38 disadvantage is its poor battery life
has the plus: double sim card feature is great .
39 dual sim card is great feature in this but battery is weak .
40 battery and dual sim card are very bad .
41 call clarity is great due to strong dual sim card operations .
42 the dual sim comfortable ! enjoys the benefits , it has great apps and easy to handle .
43 battery is acceptable .
44 battery is low .
45 display and dual sim are pretty good .
46 problematic calls and display .
47 troublesome calls and display is not good enough .
48 only bad thing is that you can not use the two sim cards together .
49 great phone for a reasonable price .
50 budget of phone is impressive .
51 the price was great for the features .
52 upgrade is good .
53 the size is comfortable .
54 great size for travel .
55 the dual sim slots is super .
56 dual sim option which is very lovable .
57 the screen size is perfect for my needs .
58 not able to increase brightness in this poor display .
59 size is perfect .
60 comfortable size .

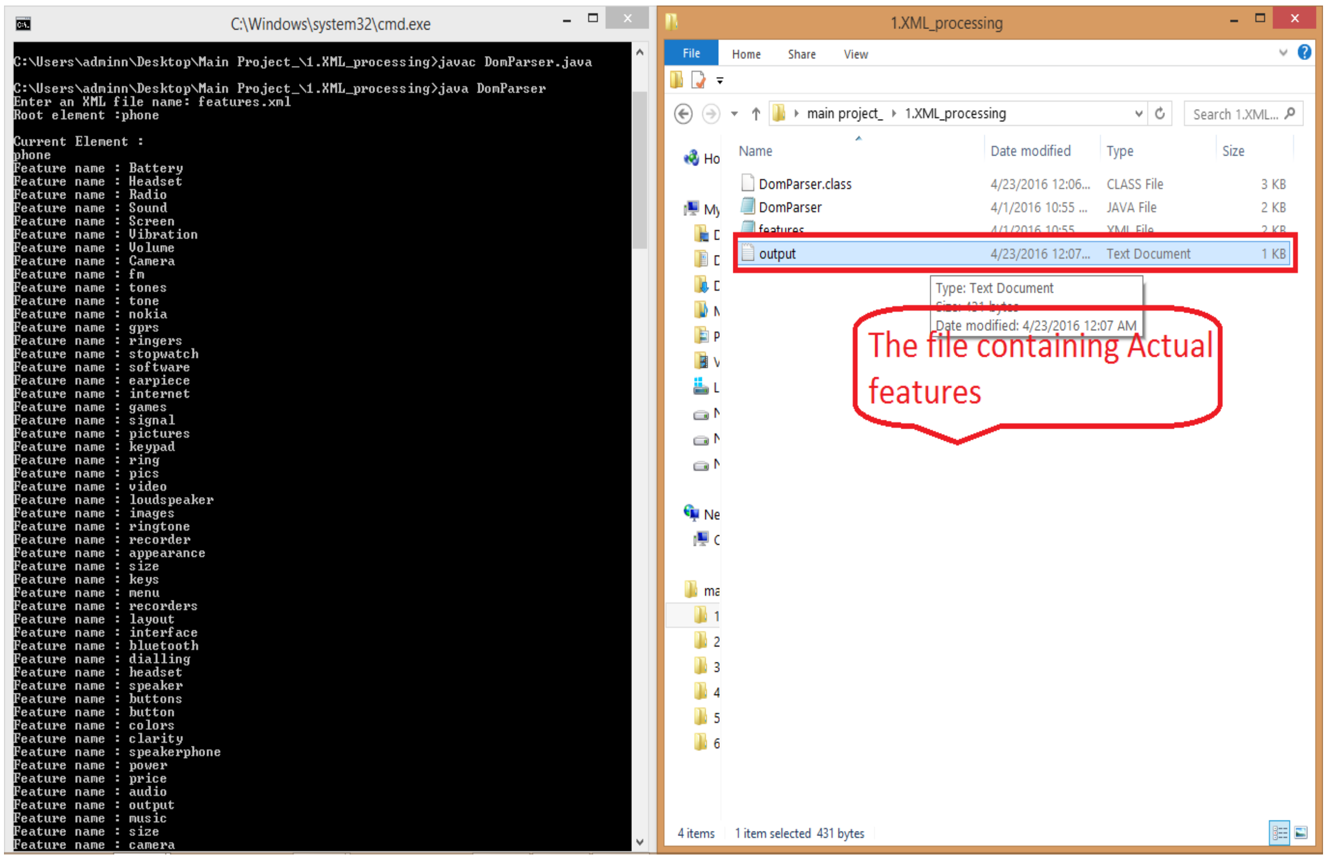
FOLDER HIERARCHY



Main Project_

Name	Date modified	Type	Size
1.XML_processing	4/22/2016 11:16...	File folder	
2.pre	4/22/2016 11:16...	File folder	
3.Pos_tag	4/22/2016 11:16...	File folder	
4.Discretization	4/22/2016 11:16...	File folder	
5.Frequent_nouns	4/22/2016 11:16...	File folder	
6.Relevant_nouns	4/22/2016 11:16...	File folder	

Fig. 5.2.1 Xml Processing



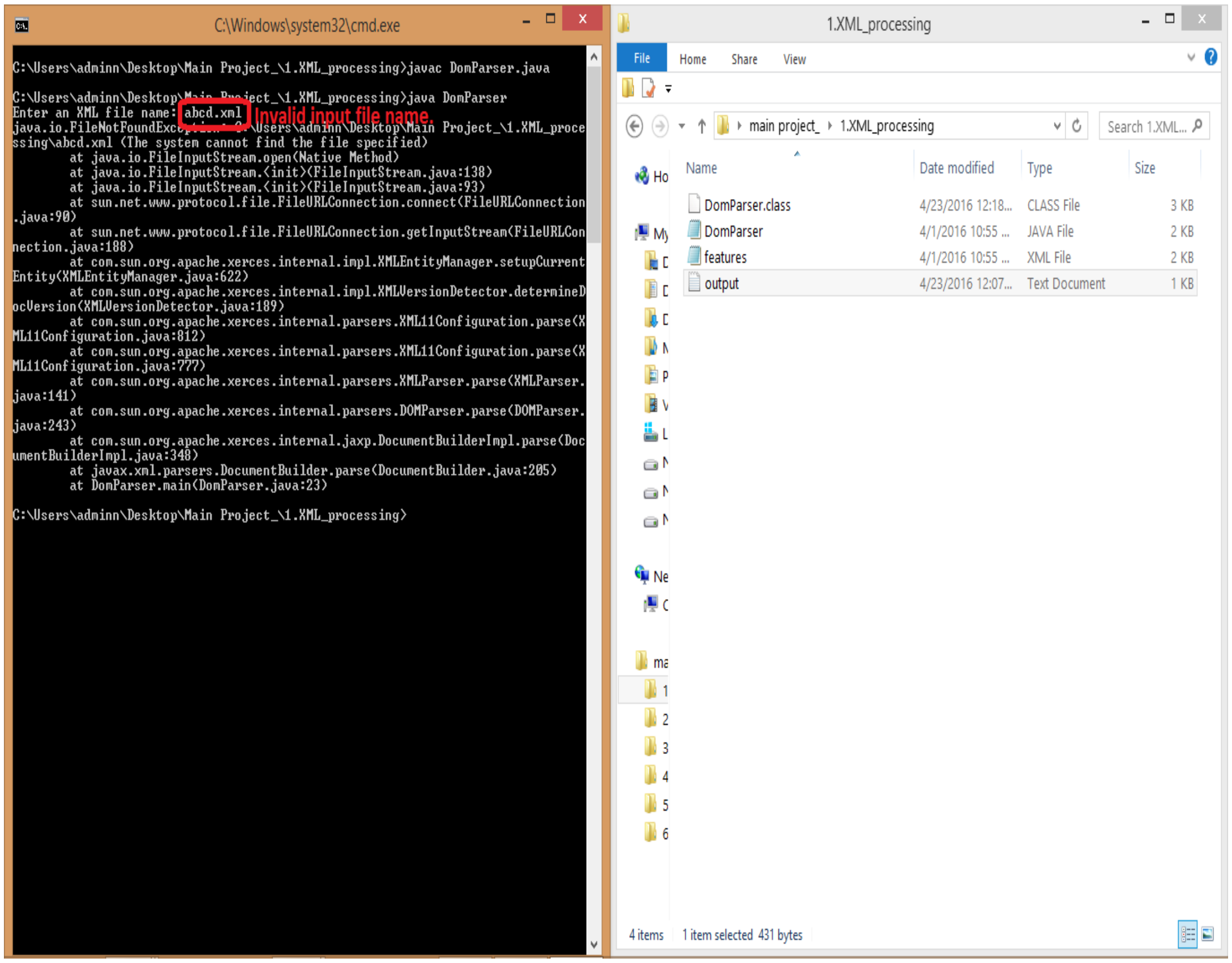
```
C:\Users\adminn\Desktop\Main Project_\1.XML_processing>javac DomParser.java

C:\Users\adminn\Desktop\Main Project_\1.XML_processing>java DomParser
Enter an XML file name: features.xml
Root element :phone

Current Element :
phone
Feature name : Battery
Feature name : Headset
Feature name : Radio
Feature name : Sound
Feature name : Screen
Feature name : Vibration
Feature name : Volume
Feature name : Camera
Feature name : fm
Feature name : tones
Feature name : tone
Feature name : nokia
Feature name : gprs
Feature name : ringers
Feature name : stopwatch
Feature name : software
Feature name : earpiece
Feature name : internet
Feature name : games
Feature name : signal
Feature name : pictures
Feature name : keypad
Feature name : ring
Feature name : pics
Feature name : video
Feature name : loudspeaker
Feature name : images
Feature name : ringtone
Feature name : recorder
Feature name : appearance
Feature name : size
Feature name : keys
Feature name : menu
Feature name : recorders
Feature name : layout
Feature name : interface
Feature name : bluetooth
Feature name : dialling
Feature name : headset
Feature name : speaker
Feature name : buttons
Feature name : button
Feature name : colors
Feature name : clarity
Feature name : speakerphone
Feature name : power
Feature name : price
Feature name : audio
Feature name : output
Feature name : music
Feature name : size
Feature name : camera
```

Name	Date modified	Type	Size
DomParser.class	4/23/2016 12:06...	CLASS File	3 KB
DomParser	4/1/2016 10:55 ...	JAVA File	2 KB
features	4/1/2016 10:55 ...	XML File	2 KB
output	4/23/2016 12:07...	Text Document	1 KB

Type: Text Document
Size: 431 bytes
Date modified: 4/23/2016 12:07 AM

C:\Windows\system32\cmd.exe
1.XML_processing
File
Home
Share
View
main project_
1.XML_processing
Search 1.XML...
4 items
1 item selected 431 bytes
The file containing Actual features

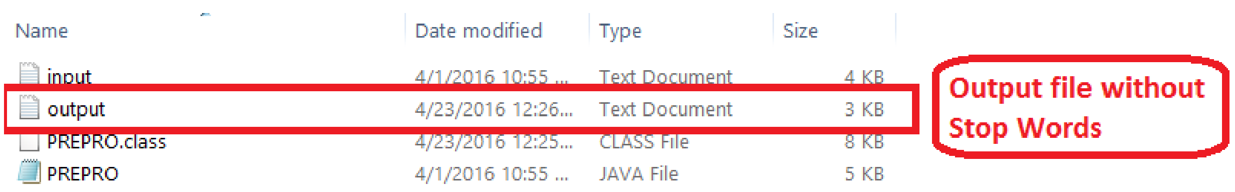
Fig. 5.2.1.1 Test Case



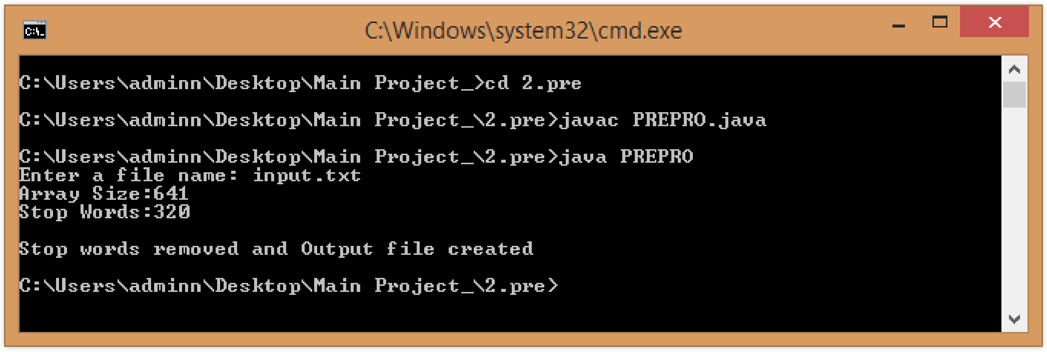
```
C:\Users\adminn\Desktop\Main Project_\1.XML_processing>javac DomParser.java

C:\Users\adminn\Desktop\Main Project_\1.XML_processing>java DomParser
Enter an XML file name: abcd.xml
java.io.FileNotFoundException: C:\Users\adminn\Desktop\Main Project_\1.XML_proce
ssing\abcd.xml (The system cannot find the file specified)
        at java.io.FileInputStream.open(Native Method)
        at java.io.FileInputStream.<init>(FileInputStream.java:138)
        at java.io.FileInputStream.<init>(FileInputStream.java:93)
        at sun.net.www.protocol.file.FileURLConnection.connect(FileURLConnection
.java:90)
        at sun.net.www.protocol.file.FileURLConnection.getInputStream(FileURLCon
nection.java:188)
        at com.sun.org.apache.xerces.internal.impl.XMLEntityManager.setupCurrent
Entity(XMLEntityManager.java:622)
        at com.sun.org.apache.xerces.internal.impl.XMLVersionDetector.determineD
ocVersion(XMLVersionDetector.java:189)
        at com.sun.org.apache.xerces.internal.parsers.XML11Configuration.parse(X
ML11Configuration.java:812)
        at com.sun.org.apache.xerces.internal.parsers.XML11Configuration.parse(X
ML11Configuration.java:777)
        at com.sun.org.apache.xerces.internal.parsers.XMLParser.parse(XMLParser.
java:141)
        at com.sun.org.apache.xerces.internal.parsers.DOMParser.parse(DOMParser.
java:243)
        at com.sun.org.apache.xerces.internal.jaxp.DocumentBuilderImpl.parse(Doc
umentBuilderImpl.java:348)
        at javax.xml.parsers.DocumentBuilder.parse(DocumentBuilder.java:205)
        at DomParser.main(DomParser.java:23)

C:\Users\adminn\Desktop\Main Project_\1.XML_processing>
```

Name	Date modified	Type	Size
DomParser.class	4/23/2016 12:18...	CLASS File	3 KB
DomParser	4/1/2016 10:55 ...	JAVA File	2 KB
features	4/1/2016 10:55 ...	XML File	2 KB
output	4/23/2016 12:07...	Text Document	1 KB

Fig. 5.2.2 Pre-Processing

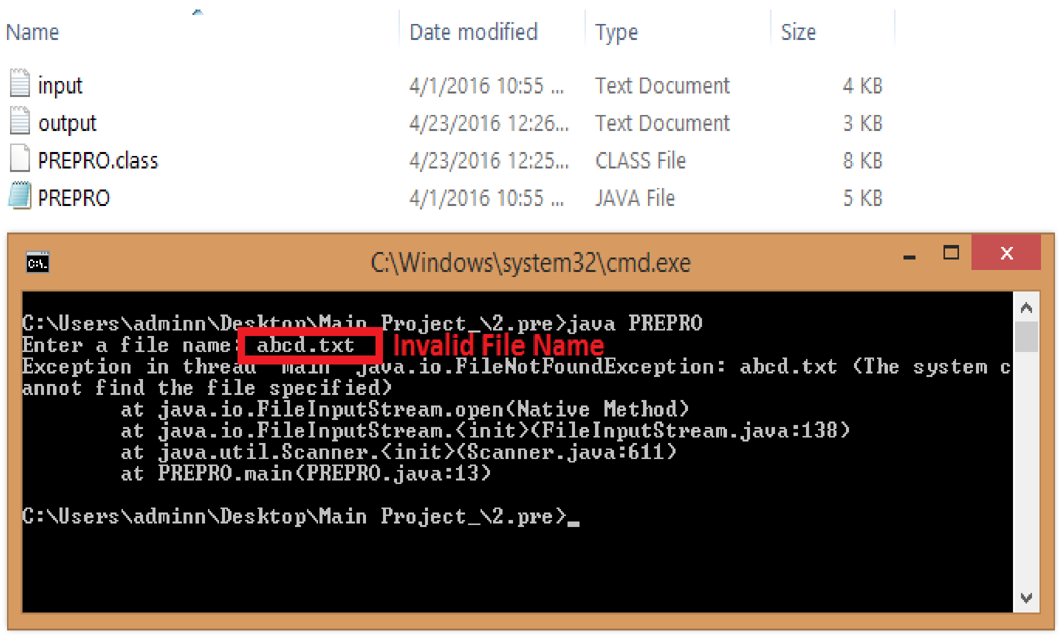


Name	Date modified	Type	Size
input	4/1/2016 10:55 ...	Text Document	4 KB
output	4/23/2016 12:26...	Text Document	3 KB
PREPRO.class	4/23/2016 12:25...	CLASS File	8 KB
PREPRO	4/1/2016 10:55 ...	JAVA File	5 KB



```
C:\Users\adminn\Desktop\Main Project_>cd 2.pre

C:\Users\adminn\Desktop\Main Project_\2.pre>javac PREPRO.java

C:\Users\adminn\Desktop\Main Project_\2.pre>java PREPRO
Enter a file name: input.txt
Array Size:641
Stop Words:320

Stop words removed and Output file created

C:\Users\adminn\Desktop\Main Project_\2.pre>
```

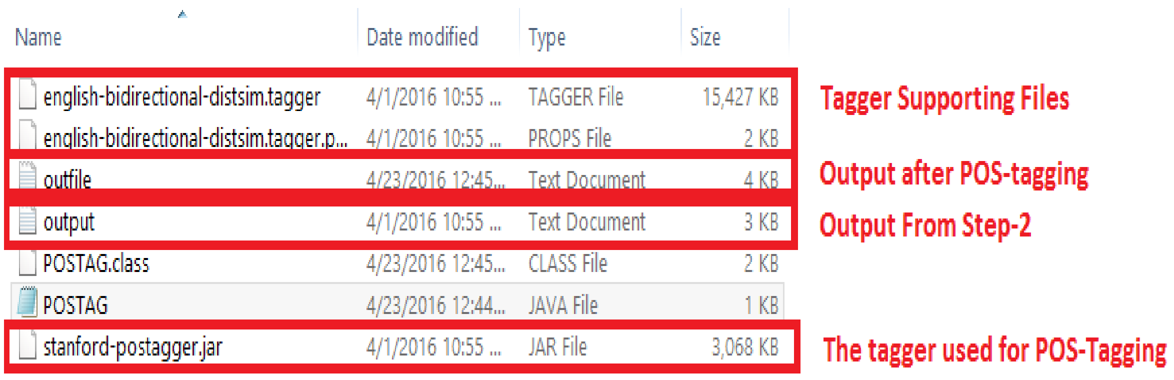
Fig. 5.2.2.1 Test Case



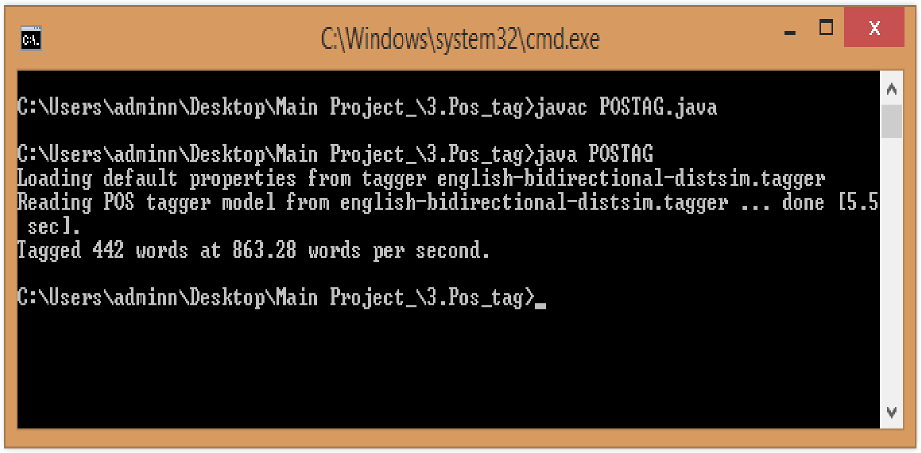
Name	Date modified	Type	Size
input	4/1/2016 10:55 ...	Text Document	4 KB
output	4/23/2016 12:26...	Text Document	3 KB
PREPRO.class	4/23/2016 12:25...	CLASS File	8 KB
PREPRO	4/1/2016 10:55 ...	JAVA File	5 KB

```
C:\Users\adminn\Desktop\Main Project_\2.pre>java PREPRO
Enter a file name: abcd.txt
Exception in thread "main" java.io.FileNotFoundException: abcd.txt (The system c
annot find the file specified)
        at java.io.FileInputStream.open(Native Method)
        at java.io.FileInputStream.<init>(FileInputStream.java:138)
        at java.util.Scanner.<init>(Scanner.java:611)
        at PREPRO.main(PREPRO.java:13)

C:\Users\adminn\Desktop\Main Project_\2.pre>_
```

Fig. 5.2.3 POS-Tagging

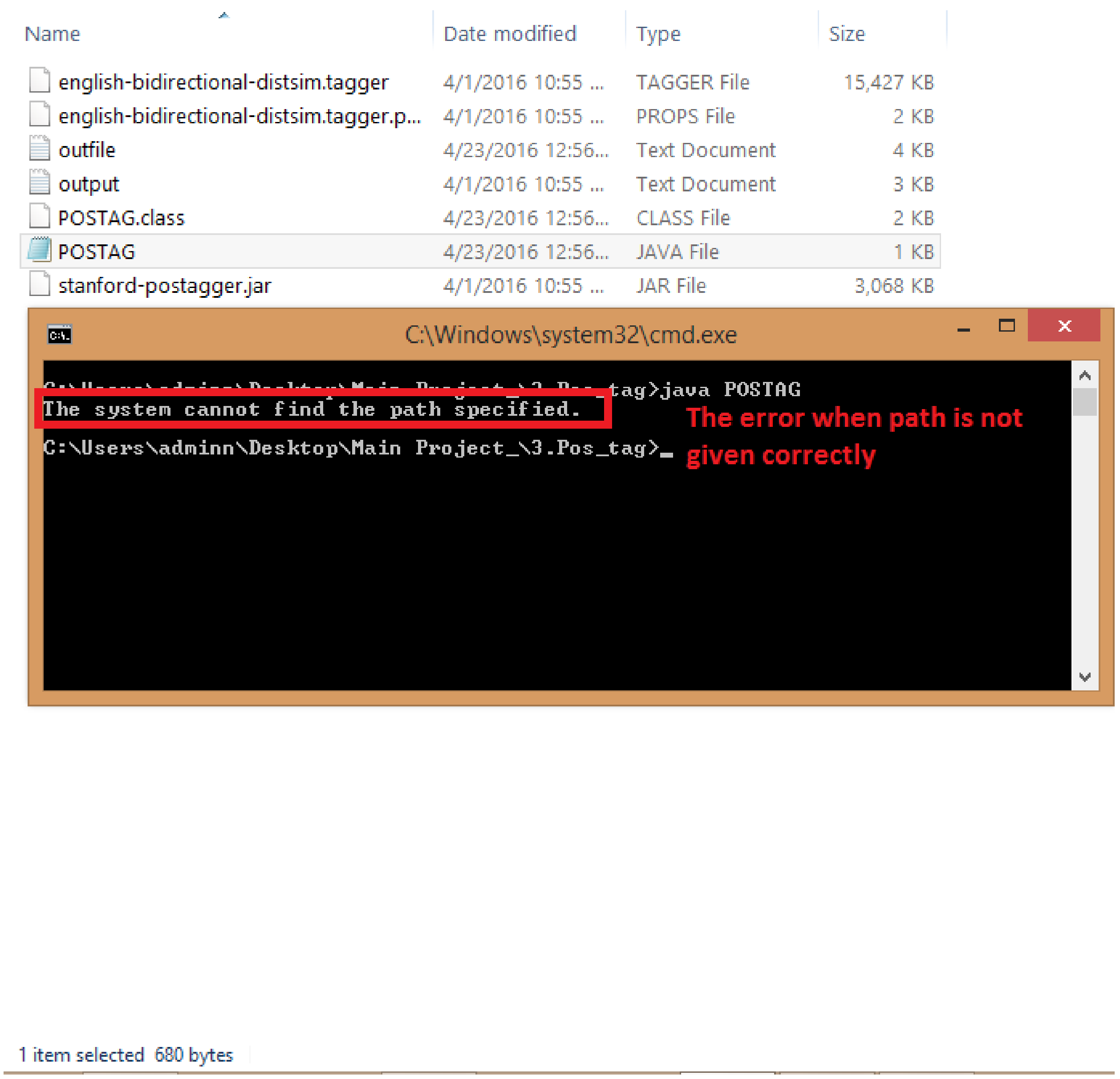


Name	Date modified	Type	Size
english-bidirectional-distsim.tagger	4/1/2016 10:55 ...	TAGGER File	15,427 KB
english-bidirectional-distsim.tagger.p...	4/1/2016 10:55 ...	PROPS File	2 KB
outfile	4/23/2016 12:45...	Text Document	4 KB
output	4/1/2016 10:55 ...	Text Document	3 KB
POSTAG.class	4/23/2016 12:45...	CLASS File	2 KB
POSTAG	4/23/2016 12:44...	JAVA File	1 KB
stanford-postagger.jar	4/1/2016 10:55 ...	JAR File	3,068 KB



```
C:\Users\adminn\Desktop\Main Project_\3.Pos_tag>javac POSTAG.java

C:\Users\adminn\Desktop\Main Project_\3.Pos_tag>java POSTAG
Loading default properties from tagger english-bidirectional-distsim.tagger
Reading POS tagger model from english-bidirectional-distsim.tagger ... done [5.5
 sec].
Tagged 442 words at 863.28 words per second.

C:\Users\adminn\Desktop\Main Project_\3.Pos_tag>_
```

Fig. 5.2.3.1 Test Case



Name	Date modified	Type	Size
english-bidirectional-distsim.tagger	4/1/2016 10:55 ...	TAGGER File	15,427 KB
english-bidirectional-distsim.tagger.p...	4/1/2016 10:55 ...	PROPS File	2 KB
outfile	4/23/2016 12:56...	Text Document	4 KB
output	4/1/2016 10:55 ...	Text Document	3 KB
POSTAG.class	4/23/2016 12:56...	CLASS File	2 KB
POSTAG	4/23/2016 12:56...	JAVA File	1 KB
stanford-postagger.jar	4/1/2016 10:55 ...	JAR File	3,068 KB

C:\Windows\system32\cmd.exe

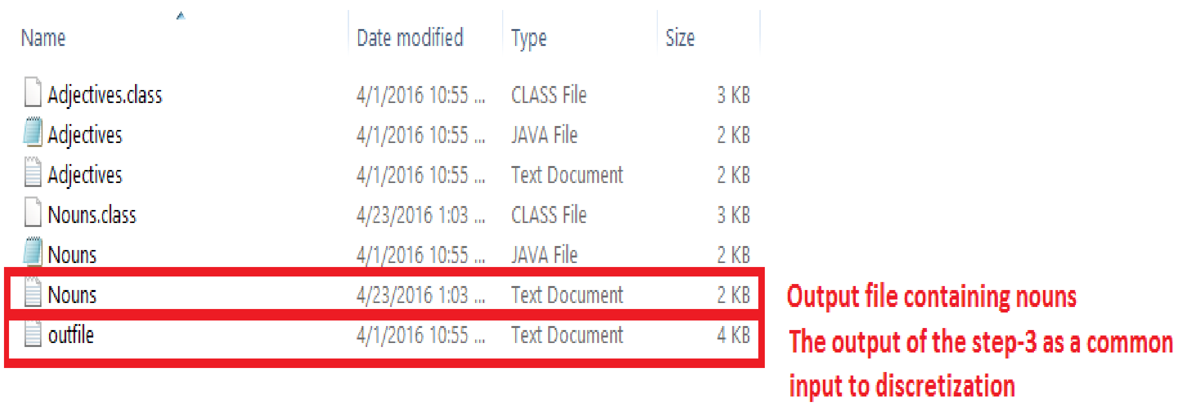
```
C:\Users\adminn\Desktop\Main Project_\3.Pos_tag>java POSTAG
The system cannot find the path specified.

C:\Users\adminn\Desktop\Main Project_\3.Pos_tag>_
```

The error when path is not given correctly

1 item selected 680 bytes

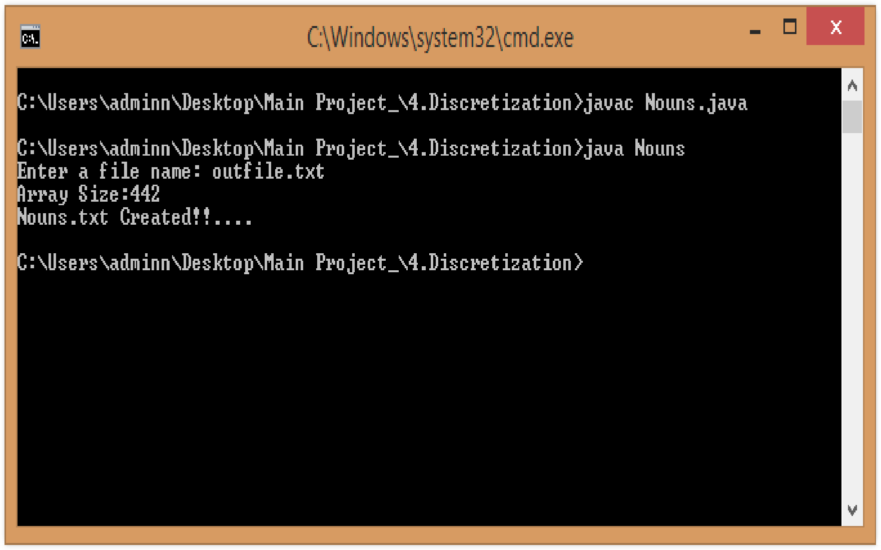
Fig. 5.2.4. Discretization

Fig. 5.2.4.1 Nouns



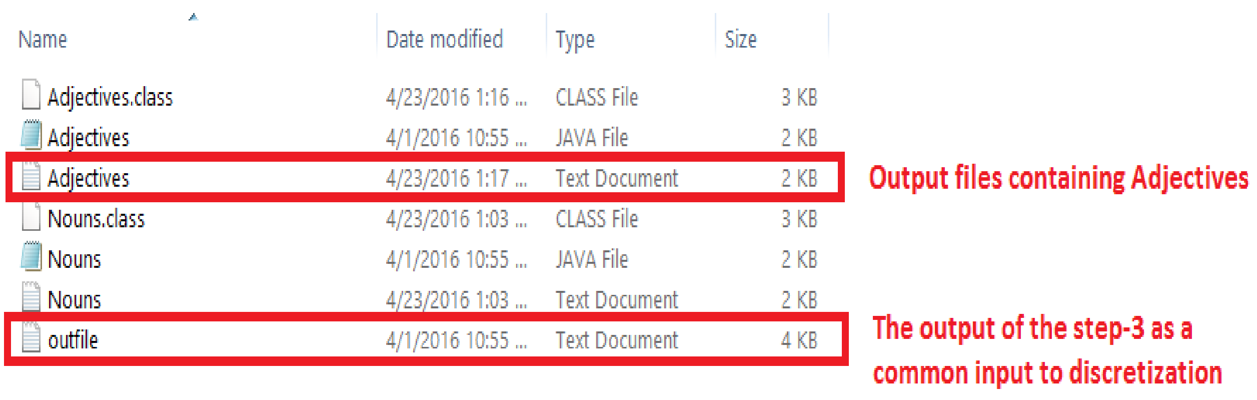
Name	Date modified	Type	Size
Adjectives.class	4/1/2016 10:55 ...	CLASS File	3 KB
Adjectives	4/1/2016 10:55 ...	JAVA File	2 KB
Adjectives	4/1/2016 10:55 ...	Text Document	2 KB
Nouns.class	4/23/2016 1:03 ...	CLASS File	3 KB
Nouns	4/1/2016 10:55 ...	JAVA File	2 KB
Nouns	4/23/2016 1:03 ...	Text Document	2 KB
outfile	4/1/2016 10:55 ...	Text Document	4 KB

Output file containing nouns
The output of the step-3 as a common input to discretization



```
C:\Users\adminn\Desktop\Main Project_\4.Discretization>javac Nouns.java

C:\Users\adminn\Desktop\Main Project_\4.Discretization>java Nouns
Enter a file name: outfile.txt
Array Size:442
Nouns.txt Created!!....

C:\Users\adminn\Desktop\Main Project_\4.Discretization>
```

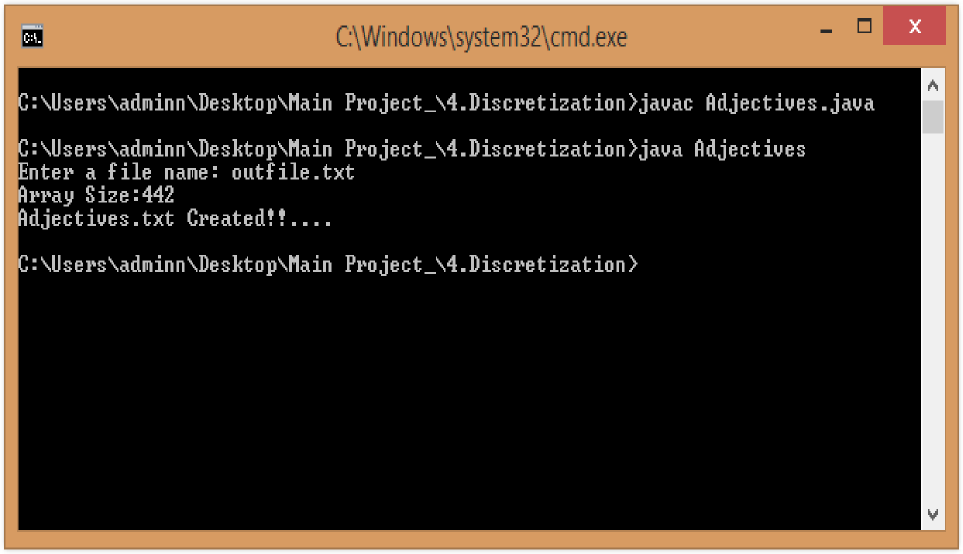
Fig. 5.2.4.2 Adjectives



Name	Date modified	Type	Size
Adjectives.class	4/23/2016 1:16 ...	CLASS File	3 KB
Adjectives	4/1/2016 10:55 ...	JAVA File	2 KB
Adjectives	4/23/2016 1:17 ...	Text Document	2 KB
Nouns.class	4/23/2016 1:03 ...	CLASS File	3 KB
Nouns	4/1/2016 10:55 ...	JAVA File	2 KB
Nouns	4/23/2016 1:03 ...	Text Document	2 KB
outfile	4/1/2016 10:55 ...	Text Document	4 KB

Output files containing Adjectives

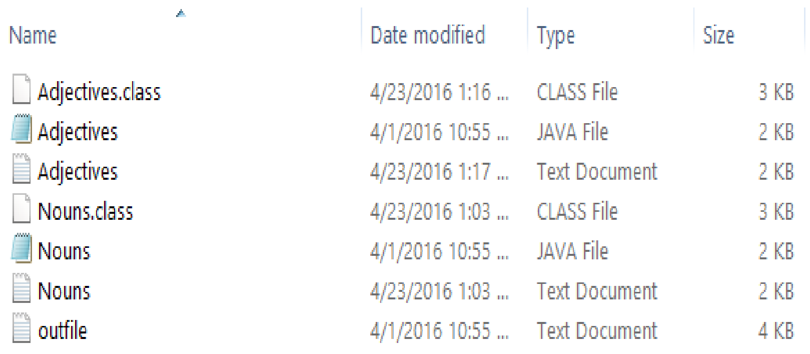
The output of the step-3 as a common input to discretization



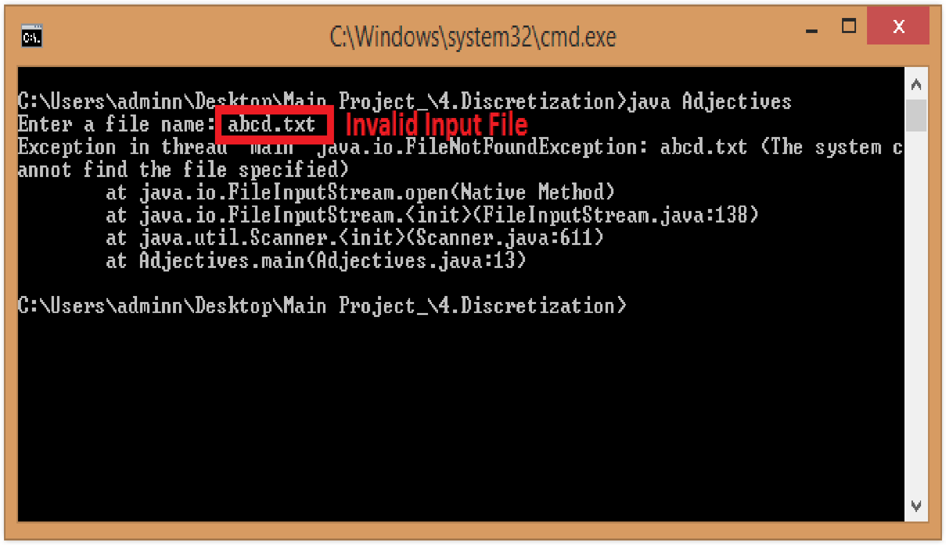
```
C:\Users\adminn\Desktop\Main Project_\4.Discretization>javac Adjectives.java

C:\Users\adminn\Desktop\Main Project_\4.Discretization>java Adjectives
Enter a file name: outfile.txt
Array Size:442
Adjectives.txt Created!!....

C:\Users\adminn\Desktop\Main Project_\4.Discretization>
```

Fig. 5.2.4.3 Test Cases

Fig. 5.2.4.3.1 Adjectives

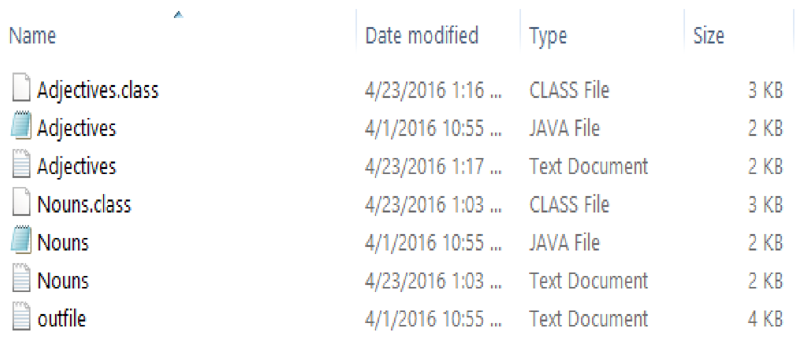


Name	Date modified	Type	Size
Adjectives.class	4/23/2016 1:16 ...	CLASS File	3 KB
Adjectives	4/1/2016 10:55 ...	JAVA File	2 KB
Adjectives	4/23/2016 1:17 ...	Text Document	2 KB
Nouns.class	4/23/2016 1:03 ...	CLASS File	3 KB
Nouns	4/1/2016 10:55 ...	JAVA File	2 KB
Nouns	4/23/2016 1:03 ...	Text Document	2 KB
outfile	4/1/2016 10:55 ...	Text Document	4 KB

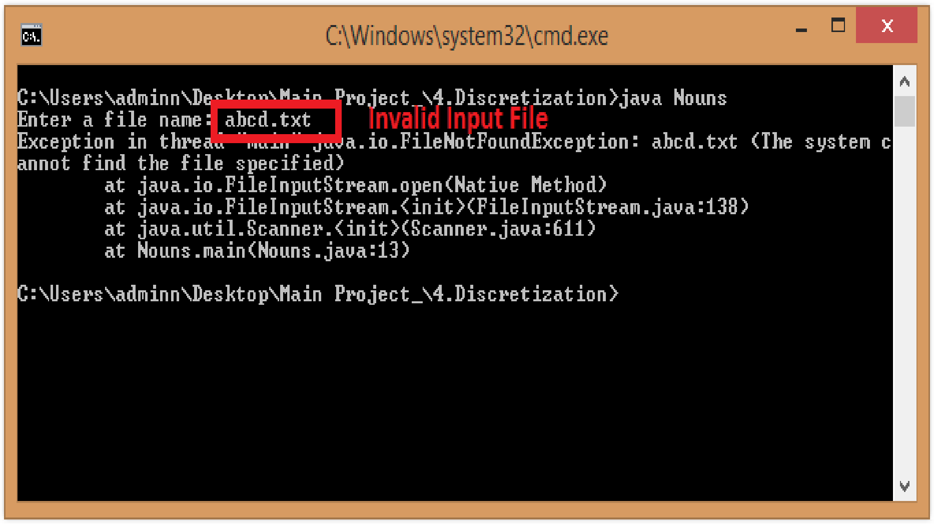


```
C:\Users\adminn\Desktop\Main Project_\4.Discretization>java Adjectives
Enter a file name: abcd.txt
Exception in thread "main" java.io.FileNotFoundException: abcd.txt (The system c
annot find the file specified)
        at java.io.FileInputStream.open(Native Method)
        at java.io.FileInputStream.<init>(FileInputStream.java:138)
        at java.util.Scanner.<init>(Scanner.java:611)
        at Adjectives.main(Adjectives.java:13)

C:\Users\adminn\Desktop\Main Project_\4.Discretization>
```

Fig. 5.2.4.3.2 Nouns

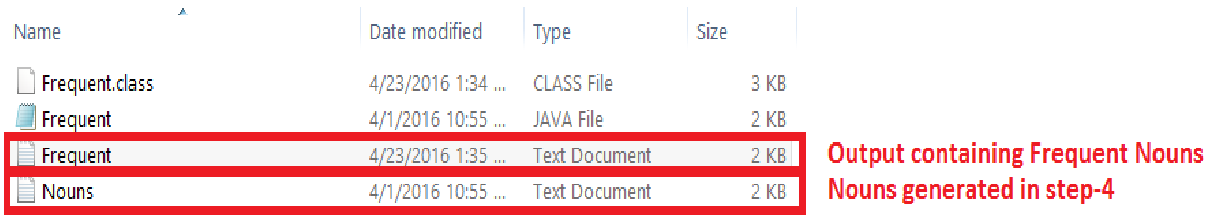


Name	Date modified	Type	Size
Adjectives.class	4/23/2016 1:16 ...	CLASS File	3 KB
Adjectives	4/1/2016 10:55 ...	JAVA File	2 KB
Adjectives	4/23/2016 1:17 ...	Text Document	2 KB
Nouns.class	4/23/2016 1:03 ...	CLASS File	3 KB
Nouns	4/1/2016 10:55 ...	JAVA File	2 KB
Nouns	4/23/2016 1:03 ...	Text Document	2 KB
outfile	4/1/2016 10:55 ...	Text Document	4 KB

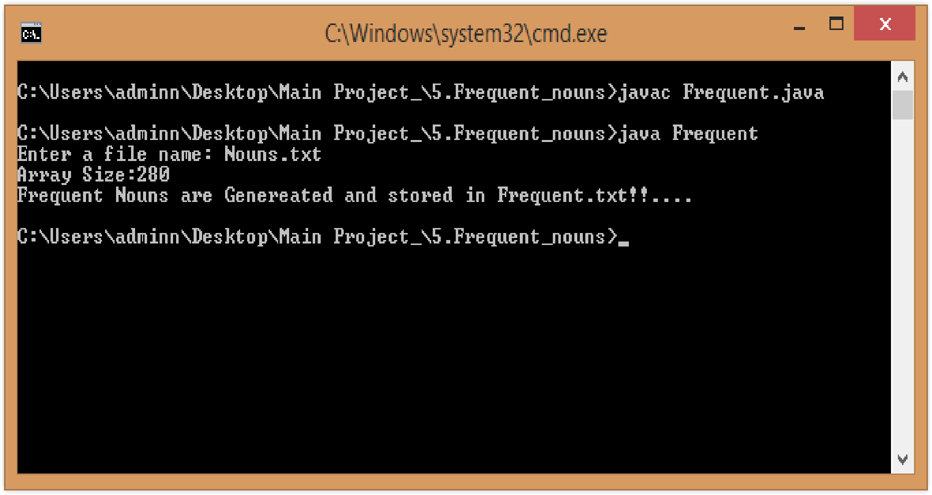


```
C:\Users\adminn\Desktop\Main Project_\4.Discretization>java Nouns
Enter a file name: abcd.txt
Exception in thread "main" java.io.FileNotFoundException: abcd.txt (The system c
annot find the file specified)
        at java.io.FileInputStream.open(Native Method)
        at java.io.FileInputStream.<init>(FileInputStream.java:138)
        at java.util.Scanner.<init>(Scanner.java:611)
        at Nouns.main(Nouns.java:13)

C:\Users\adminn\Desktop\Main Project_\4.Discretization>
```

Fig. 5.2.5 Frequent Noun Generation

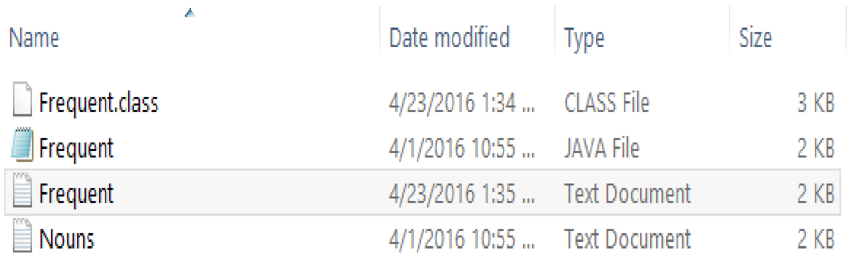


Name	Date modified	Type	Size
Frequent.class	4/23/2016 1:34 ...	CLASS File	3 KB
Frequent	4/1/2016 10:55 ...	JAVA File	2 KB
Frequent	4/23/2016 1:35 ...	Text Document	2 KB
Nouns	4/1/2016 10:55 ...	Text Document	2 KB

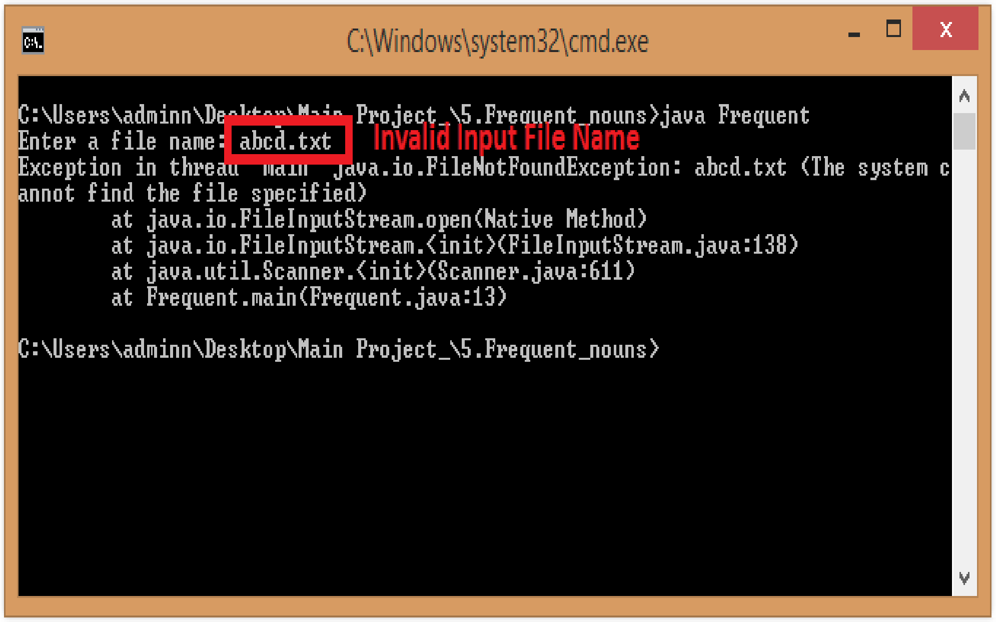


```
C:\Users\adminn\Desktop\Main Project_\5.Frequent_nouns>javac Frequent.java

C:\Users\adminn\Desktop\Main Project_\5.Frequent_nouns>java Frequent
Enter a file name: Nouns.txt
Array Size:280
Frequent Nouns are Genereated and stored in Frequent.txt!!....

C:\Users\adminn\Desktop\Main Project_\5.Frequent_nouns>_
```

Fig. 5.2.5.1 Test Case

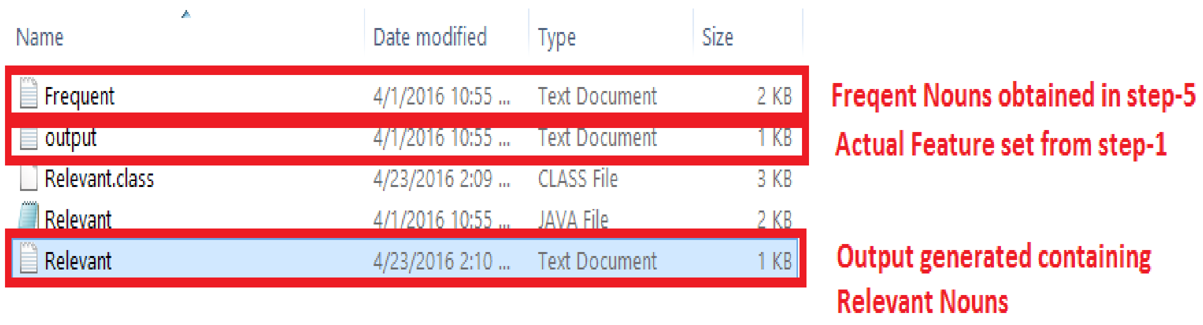


Name	Date modified	Type	Size
Frequent.class	4/23/2016 1:34 ...	CLASS File	3 KB
Frequent	4/1/2016 10:55 ...	JAVA File	2 KB
Frequent	4/23/2016 1:35 ...	Text Document	2 KB
Nouns	4/1/2016 10:55 ...	Text Document	2 KB



```
C:\Users\adminn\Desktop\Main Project_\5.Frequent_nouns>java Frequent
Enter a file name: abcd.txt
Exception in thread "main" java.io.FileNotFoundException: abcd.txt (The system c
annot find the file specified)
        at java.io.FileInputStream.open(Native Method)
        at java.io.FileInputStream.<init>(FileInputStream.java:138)
        at java.util.Scanner.<init>(Scanner.java:611)
        at Frequent.main(Frequent.java:13)

C:\Users\adminn\Desktop\Main Project_\5.Frequent_nouns>
```

Fig. 5.2.6 Relevant Noun Generation

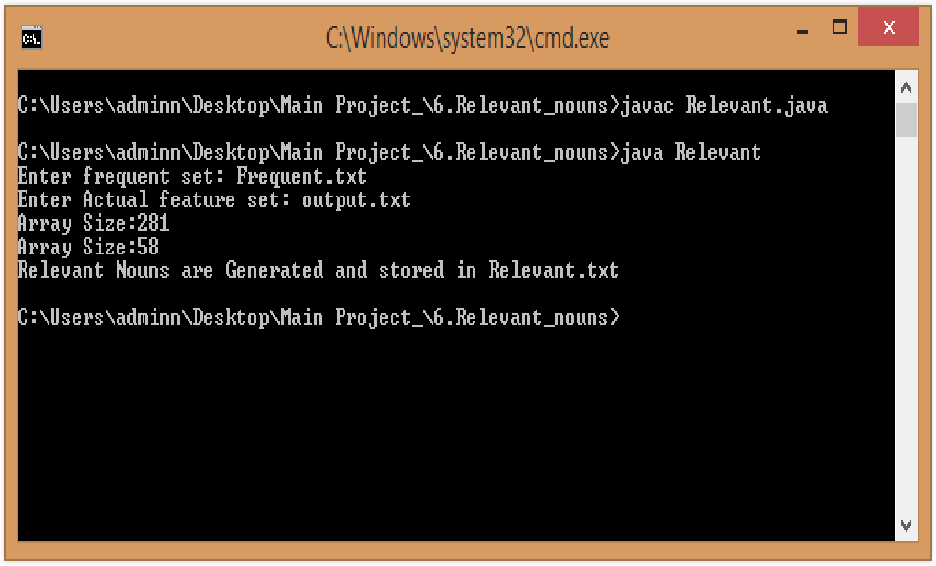


Name	Date modified	Type	Size
Frequent	4/1/2016 10:55 ...	Text Document	2 KB
output	4/1/2016 10:55 ...	Text Document	1 KB
Relevant.class	4/23/2016 2:09 ...	CLASS File	3 KB
Relevant	4/1/2016 10:55 ...	JAVA File	2 KB
Relevant	4/23/2016 2:10 ...	Text Document	1 KB

Freqent Nouns obtained in step-5

Actual Feature set from step-1

Output generated containing Relevant Nouns

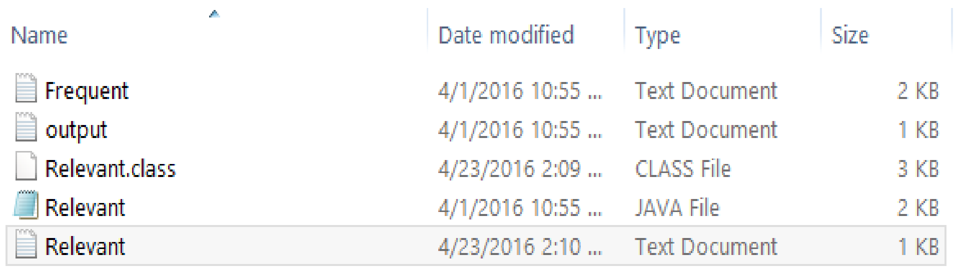


C:\Windows\system32\cmd.exe

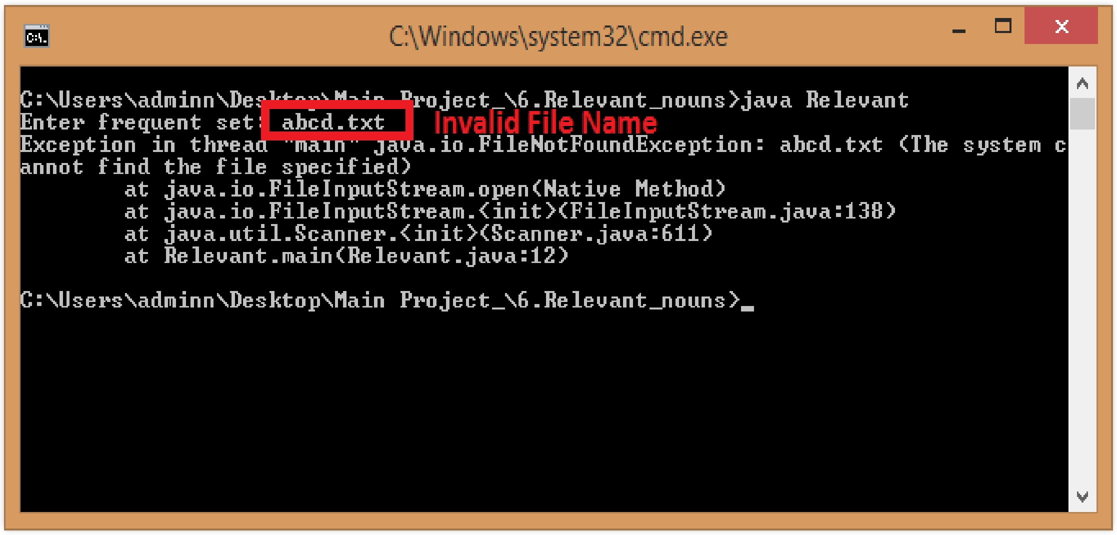
```
C:\Users\adminn\Desktop\Main Project_\6.Relevant_nouns>javac Relevant.java

C:\Users\adminn\Desktop\Main Project_\6.Relevant_nouns>java Relevant
Enter frequent set: Frequent.txt
Enter Actual feature set: output.txt
Array Size:281
Array Size:58
Relevant Nouns are Generated and stored in Relevant.txt

C:\Users\adminn\Desktop\Main Project_\6.Relevant_nouns>
```

Fig. 5.2.6.1 Test Cases

Fig. 5.2.6.1.1 Frequent Set

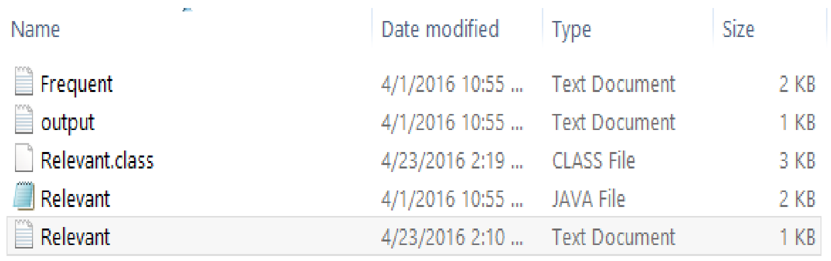


Name	Date modified	Type	Size
Frequent	4/1/2016 10:55 ...	Text Document	2 KB
output	4/1/2016 10:55 ...	Text Document	1 KB
Relevant.class	4/23/2016 2:09 ...	CLASS File	3 KB
Relevant	4/1/2016 10:55 ...	JAVA File	2 KB
Relevant	4/23/2016 2:10 ...	Text Document	1 KB

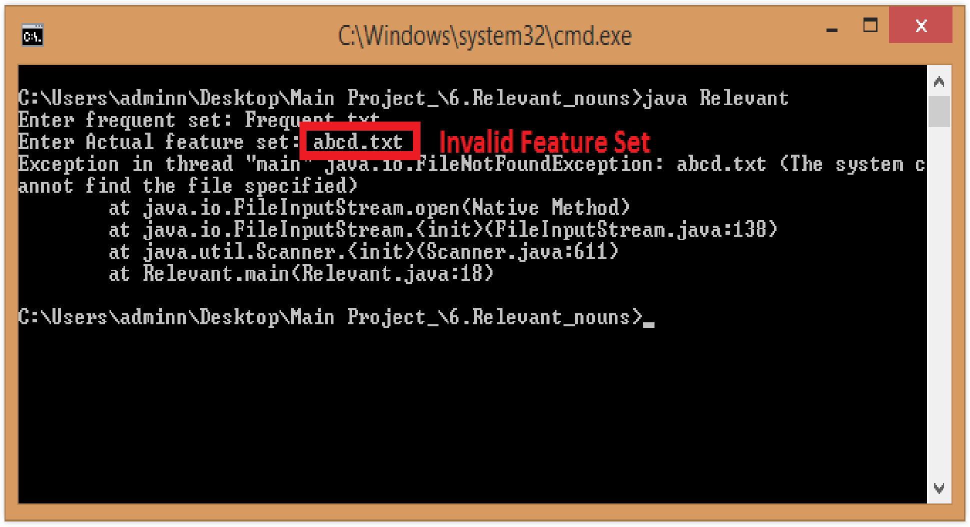


```
C:\Users\adminn\Desktop\Main Project_\6.Relevant_nouns>java Relevant
Enter frequent set: abcd.txt
Exception in thread "main" java.io.FileNotFoundException: abcd.txt (The system c
annot find the file specified)
        at java.io.FileInputStream.open(Native Method)
        at java.io.FileInputStream.<init>(FileInputStream.java:138)
        at java.util.Scanner.<init>(Scanner.java:611)
        at Relevant.main(Relevant.java:12)

C:\Users\adminn\Desktop\Main Project_\6.Relevant_nouns>_
```

Fig. 5.2.6.1.2 Actual Feature Set

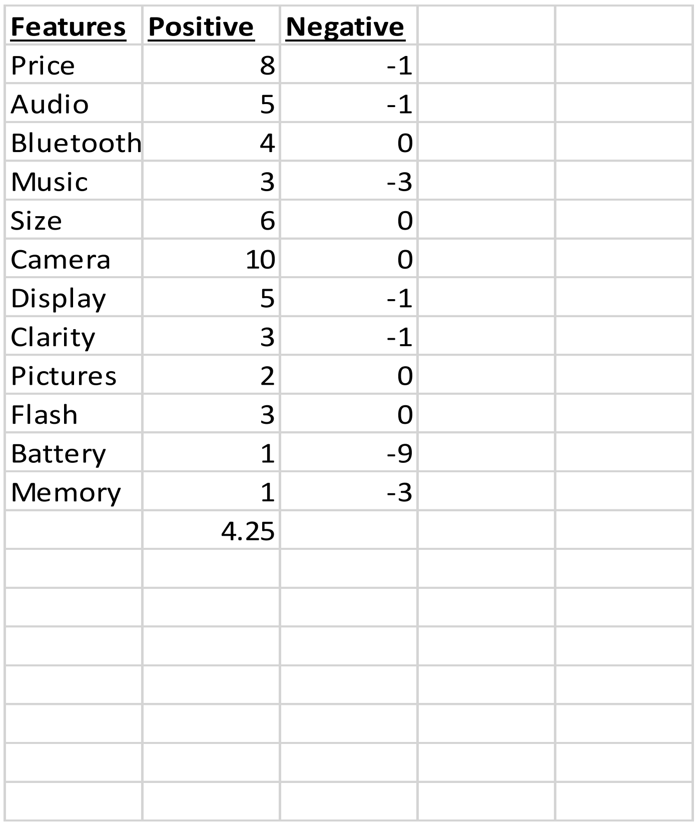


Name	Date modified	Type	Size
Frequent	4/1/2016 10:55 ...	Text Document	2 KB
output	4/1/2016 10:55 ...	Text Document	1 KB
Relevant.class	4/23/2016 2:19 ...	CLASS File	3 KB
Relevant	4/1/2016 10:55 ...	JAVA File	2 KB
Relevant	4/23/2016 2:10 ...	Text Document	1 KB



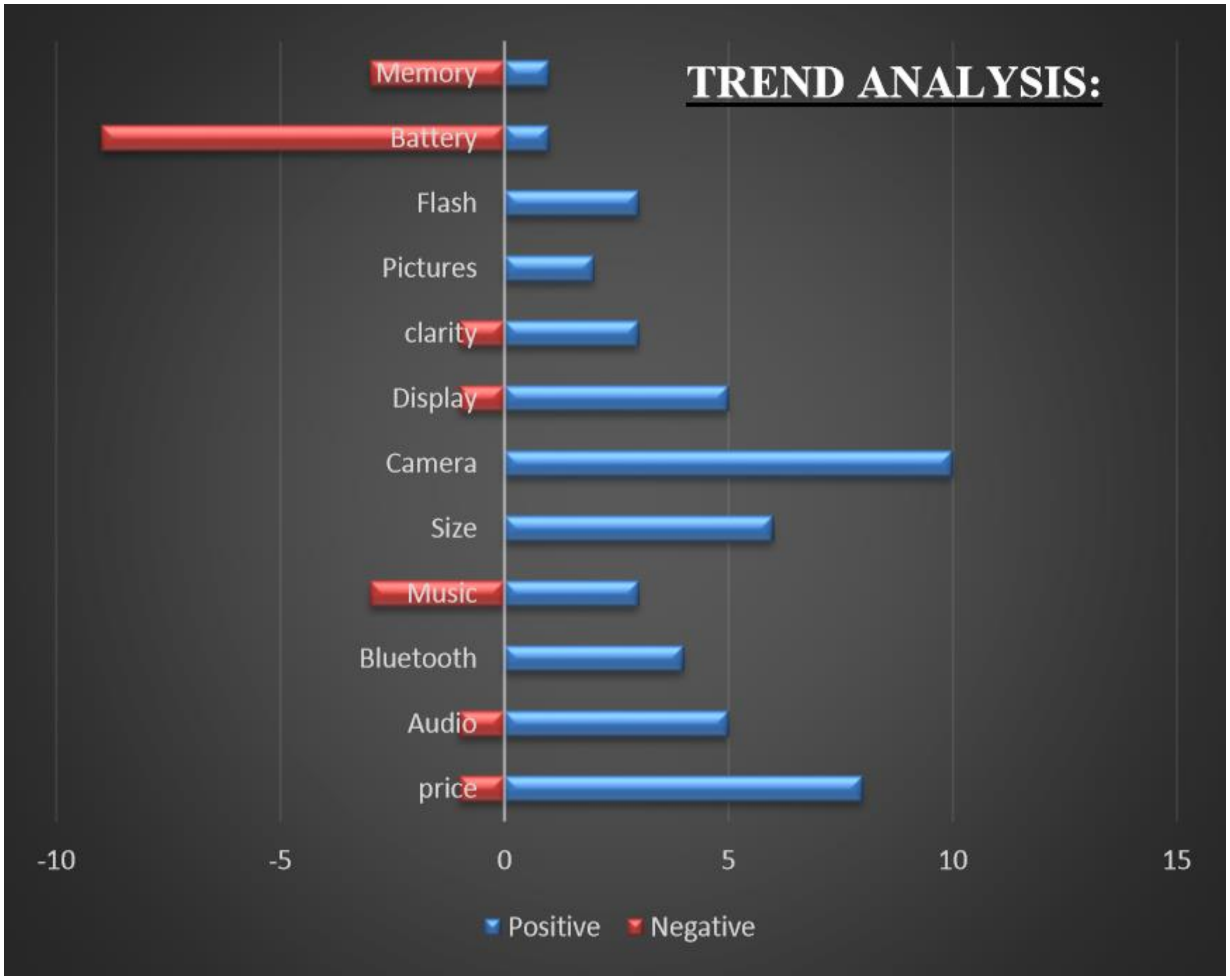
```
C:\Users\adminn\Desktop\Main Project_\6.Relevant_nouns>java Relevant
Enter frequent set: Frequent.txt
Enter Actual feature set: abcd.txt
Exception in thread "main" java.io.FileNotFoundException: abcd.txt (The system c
annot find the file specified)
        at java.io.FileInputStream.open(Native Method)
        at java.io.FileInputStream.<init>(FileInputStream.java:138)
        at java.util.Scanner.<init>(Scanner.java:611)
        at Relevant.main(Relevant.java:18)

C:\Users\adminn\Desktop\Main Project_\6.Relevant_nouns>_
```

Fig. 5.2.7 Sentiment Word Extraction



Features	**Positive**	**Negative**
Price	8	-1
Audio	5	-1
Bluetooth	4	0
Music	3	-3
Size	6	0
Camera	10	0
Display	5	-1
Clarity	3	-1
Pictures	2	0
Flash	3	0
Battery	1	-9
Memory	1	-3
	4.25	

Fig. 5.2.8 Trend Analysis



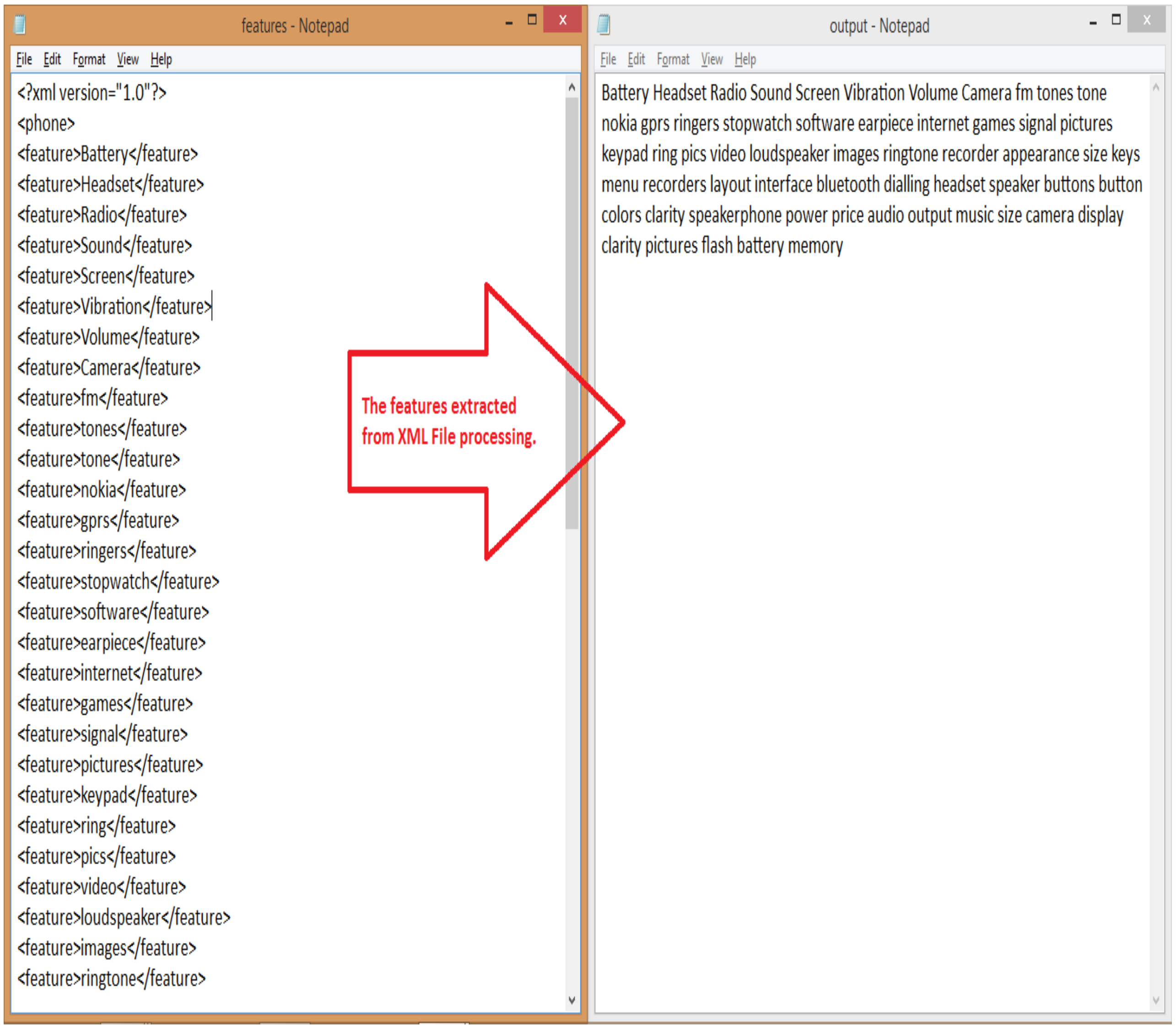
5.3 RESULT ANALYSIS

The Results obtained in the above all phases are formed to analysis the project i.e. results helps in finding the accuracy of result.

The following are the sets of input and output files that are formed as input and output for each phase:

5.3.1 XML Processing
5.3.2 PRE Processing
5.3.3 POS-Tagging
5.3.4 Discretization
5.3.5 Frequent Noun Generation
5.3.6 Relevant Noun Generation
5.3.7 Sentiment Word Extraction
5.3.8 Trend Analysis

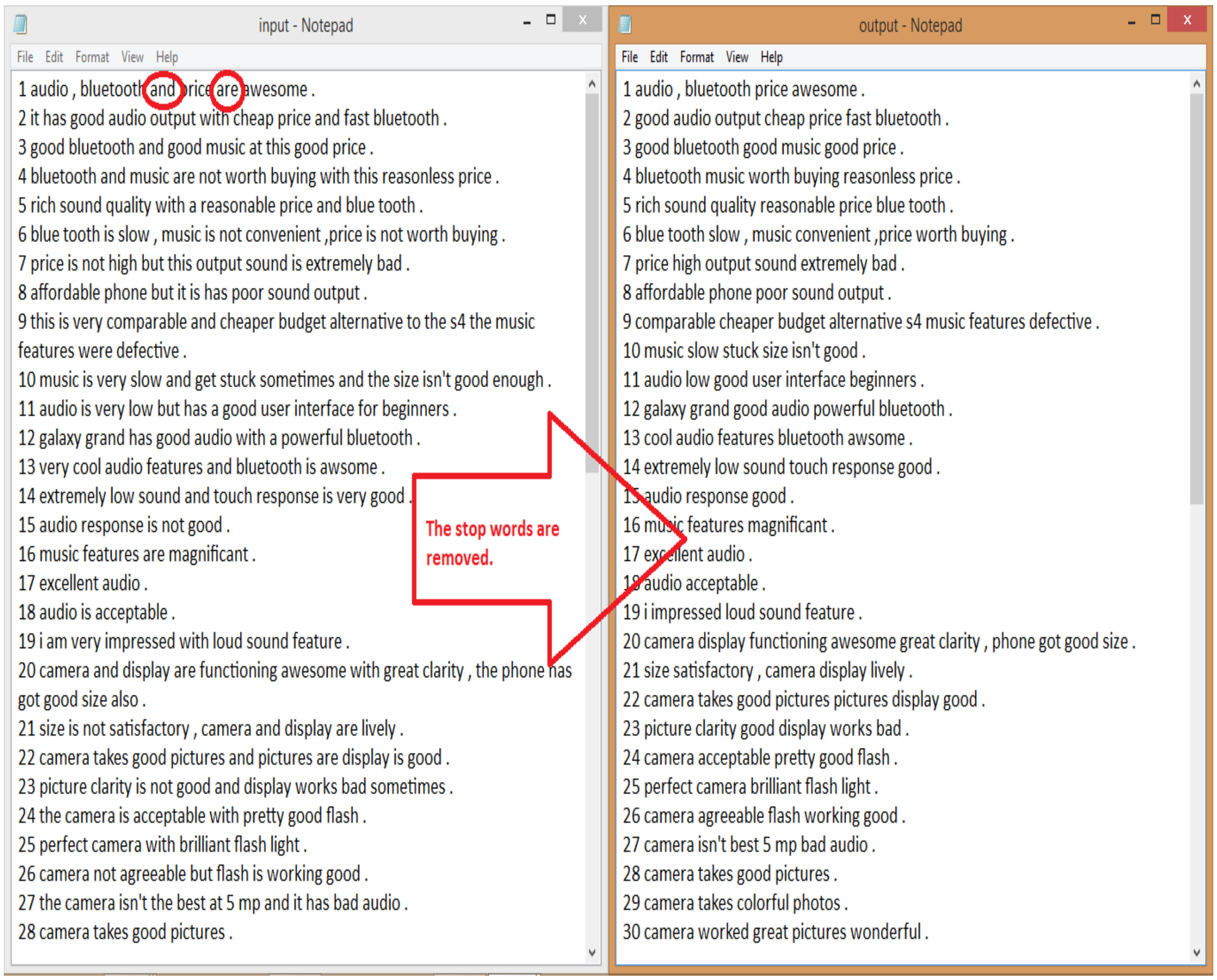
Fig. 5.3.1 XML Processing


features - Notepad
File Edit Format View Help
output - Notepad
File Edit Format View Help
The features extracted from XML File processing.

```
<?xml version="1.0"?>
<phone>
<feature>Battery</feature>
<feature>Headset</feature>
<feature>Radio</feature>
<feature>Sound</feature>
<feature>Screen</feature>
<feature>Vibration</feature>
<feature>Volume</feature>
<feature>Camera</feature>
<feature>fm</feature>
<feature>tones</feature>
<feature>tone</feature>
<feature>nokia</feature>
<feature>gprs</feature>
<feature>ringers</feature>
<feature>stopwatch</feature>
<feature>software</feature>
<feature>earpiece</feature>
<feature>internet</feature>
<feature>games</feature>
<feature>signal</feature>
<feature>pictures</feature>
<feature>keypad</feature>
<feature>ring</feature>
<feature>pics</feature>
<feature>video</feature>
<feature>loudspeaker</feature>
<feature>images</feature>
<feature>ringtone</feature>
```

```
Battery Headset Radio Sound Screen Vibration Volume Camera fm tones tone
nokia gprs ringers stopwatch software earpiece internet games signal pictures
keypad ring pics video loudspeaker images ringtone recorder appearance size keys
menu recorders layout interface bluetooth dialling headset speaker buttons button
colors clarity speakerphone power price audio output music size camera display
clarity pictures flash battery memory
```

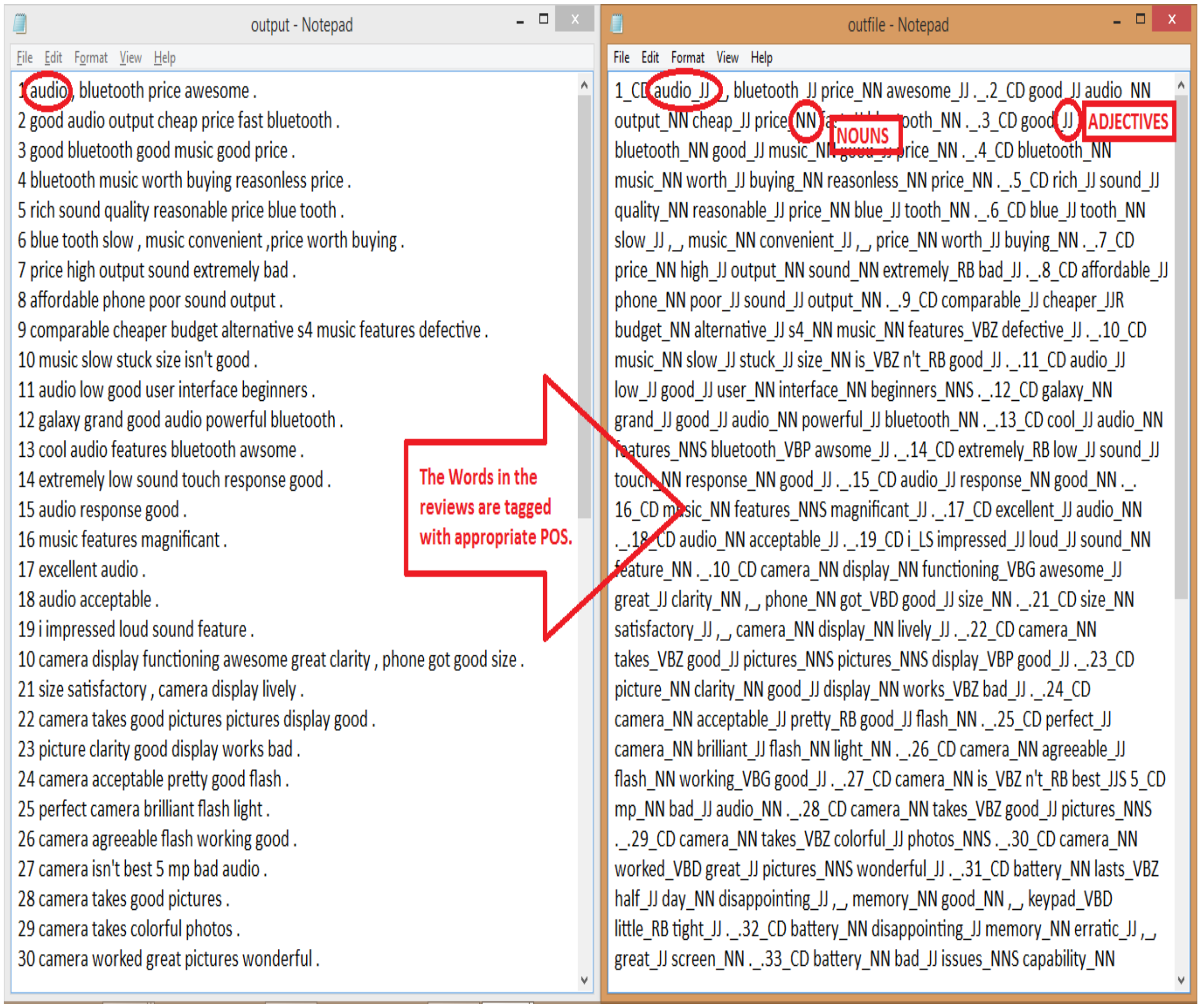
Fig. 5.3.2 Pre-Processing


input - Notepad
File Edit Format View Help
output - Notepad
File Edit Format View Help
The stop words are removed.

```
1 audio , bluetooth and price are awesome .
2 it has good audio output with cheap price and fast bluetooth .
3 good bluetooth and good music at this good price .
4 bluetooth and music are not worth buying with this reasonless price .
5 rich sound quality with a reasonable price and blue tooth .
6 blue tooth is slow , music is not convenient ,price is not worth buying .
7 price is not high but this output sound is extremely bad .
8 affordable phone but it is has poor sound output .
9 this is very comparable and cheaper budget alternative to the s4 the music
features were defective .
10 music is very slow and get stuck sometimes and the size isn't good enough .
11 audio is very low but has a good user interface for beginners .
12 galaxy grand has good audio with a powerful bluetooth .
13 very cool audio features and bluetooth is awsome .
14 extremely low sound and touch response is very good .
15 audio response is not good .
16 music features are magnificant .
17 excellent audio .
18 audio is acceptable .
19 i am very impressed with loud sound feature .
20 camera and display are functioning awesome with great clarity , the phone has
got good size also .
21 size is not satisfactory , camera and display are lively .
22 camera takes good pictures and pictures are display is good .
23 picture clarity is not good and display works bad sometimes .
24 the camera is acceptable with pretty good flash .
25 perfect camera with brilliant flash light .
26 camera not agreeable but flash is working good .
27 the camera isn't the best at 5 mp and it has bad audio .
28 camera takes good pictures .
```

```
1 audio , bluetooth price awesome .
2 good audio output cheap price fast bluetooth .
3 good bluetooth good music good price .
4 bluetooth music worth buying reasonless price .
5 rich sound quality reasonable price blue tooth .
6 blue tooth slow , music convenient ,price worth buying .
7 price high output sound extremely bad .
8 affordable phone poor sound output .
9 comparable cheaper budget alternative s4 music features defective .
10 music slow stuck size isn't good .
11 audio low good user interface beginners .
12 galaxy grand good audio powerful bluetooth .
13 cool audio features bluetooth awsome .
14 extremely low sound touch response good .
15 audio response good .
16 music features magnificant .
17 excellent audio .
18 audio acceptable .
19 i impressed loud sound feature .
20 camera display functioning awesome great clarity , phone got good size .
21 size satisfactory , camera display lively .
22 camera takes good pictures pictures display good .
23 picture clarity good display works bad .
24 camera acceptable pretty good flash .
25 perfect camera brilliant flash light .
26 camera agreeable flash working good .
27 camera isn't best 5 mp bad audio .
28 camera takes good pictures .
29 camera takes colorful photos .
30 camera worked great pictures wonderful .
```

Fig. 5.3.3 POS-Tagging

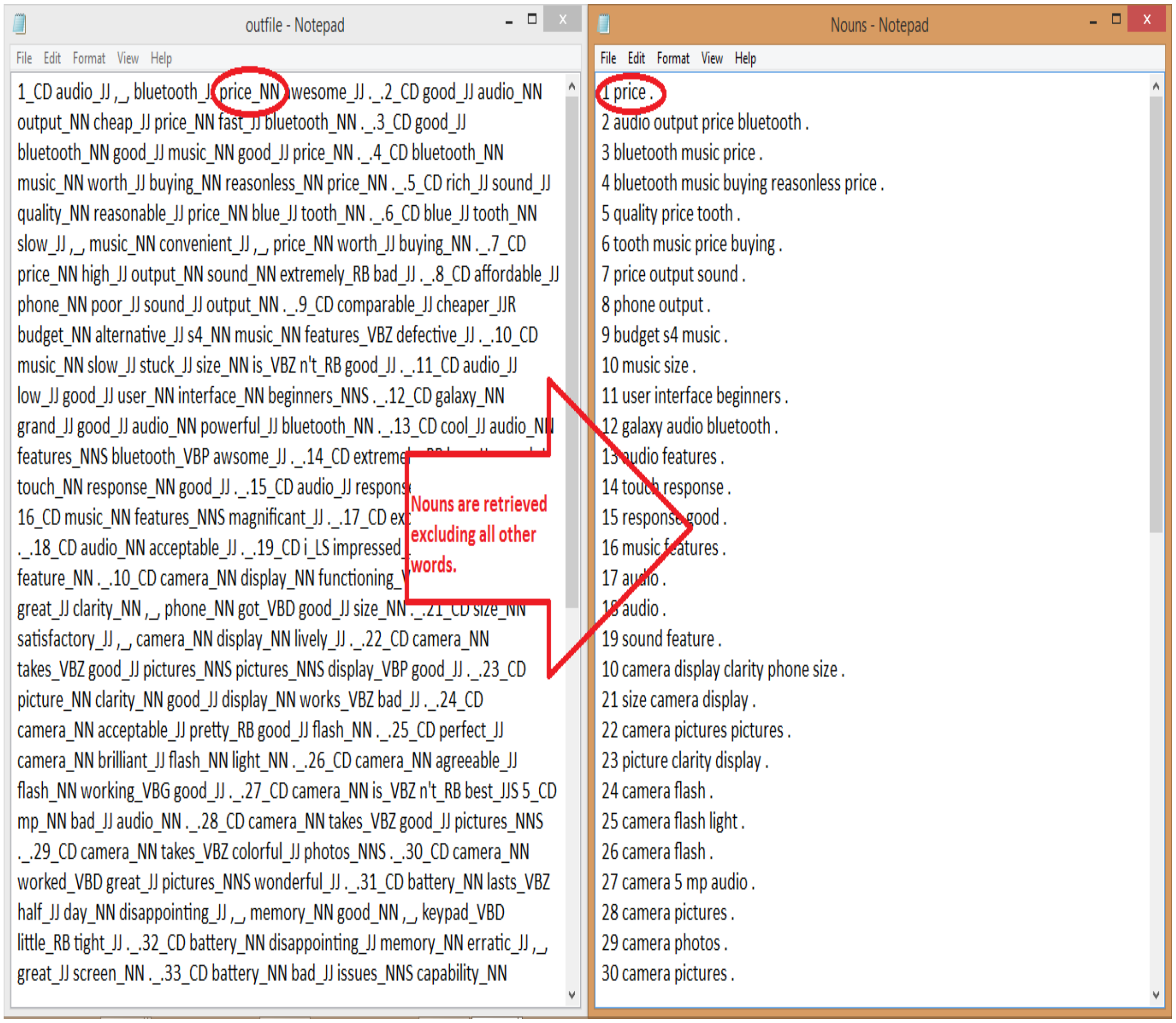

output - Notepad
File Edit Format View Help
outfile - Notepad
File Edit Format View Help
NOUNS
ADJECTIVES
The Words in the reviews are tagged with appropriate POS.

```
1 audio , bluetooth price awesome .
2 good audio output cheap price fast bluetooth .
3 good bluetooth good music good price .
4 bluetooth music worth buying reasonless price .
5 rich sound quality reasonable price blue tooth .
6 blue tooth slow , music convenient ,price worth buying .
7 price high output sound extremely bad .
8 affordable phone poor sound output .
9 comparable cheaper budget alternative s4 music features defective .
10 music slow stuck size isn't good .
11 audio low good user interface beginners .
12 galaxy grand good audio powerful bluetooth .
13 cool audio features bluetooth awsome .
14 extremely low sound touch response good .
15 audio response good .
16 music features magnificant .
17 excellent audio .
18 audio acceptable .
19 i impressed loud sound feature .
10 camera display functioning awesome great clarity , phone got good size .
21 size satisfactory , camera display lively .
22 camera takes good pictures pictures display good .
23 picture clarity good display works bad .
24 camera acceptable pretty good flash .
25 perfect camera brilliant flash light .
26 camera agreeable flash working good .
27 camera isn't best 5 mp bad audio .
28 camera takes good pictures .
29 camera takes colorful photos .
30 camera worked great pictures wonderful .
```

```
1_CD audio_JJ ,_, bluetooth_JJ price_NN awesome_JJ ._.2_CD good_JJ audio_NN
output_NN cheap_JJ price_NN fast_JJ bluetooth_NN ._.3_CD good_JJ
bluetooth_NN good_JJ music_NN good_JJ price_NN ._.4_CD bluetooth_NN
music_NN worth_JJ buying_NN reasonless_NN price_NN ._.5_CD rich_JJ sound_JJ
quality_NN reasonable_JJ price_NN blue_JJ tooth_NN ._.6_CD blue_JJ tooth_NN
slow_JJ ,_, music_NN convenient_JJ ,_, price_NN worth_JJ buying_NN ._.7_CD
price_NN high_JJ output_NN sound_NN extremely_RB bad_JJ ._.8_CD affordable_JJ
phone_NN poor_JJ sound_JJ output_NN ._.9_CD comparable_JJ cheaper_JJR
budget_NN alternative_JJ s4_NN music_NN features_VBZ defective_JJ ._.10_CD
music_NN slow_JJ stuck_JJ size_NN is_VBZ n't_RB good_JJ ._.11_CD audio_JJ
low_JJ good_JJ user_NN interface_NN beginners_NNS ._.12_CD galaxy_NN
grand_JJ good_JJ audio_NN powerful_JJ bluetooth_NN ._.13_CD cool_JJ audio_NN
features_NNS bluetooth_VBP awsome_JJ ._.14_CD extremely_RB low_JJ sound_JJ
touch_NN response_NN good_JJ ._.15_CD audio_JJ response_NN good_NN ._.
16_CD music_NN features_NNS magnificant_JJ ._.17_CD excellent_JJ audio_NN
._.18_CD audio_NN acceptable_JJ ._.19_CD i_LS impressed_JJ loud_JJ sound_NN
feature_NN ._.10_CD camera_NN display_NN functioning_VBG awesome_JJ
great_JJ clarity_NN ,_, phone_NN got_VBD good_JJ size_NN ._.21_CD size_NN
satisfactory_JJ ,_, camera_NN display_NN lively_JJ ._.22_CD camera_NN
takes_VBZ good_JJ pictures_NNS pictures_NNS display_VBP good_JJ ._.23_CD
picture_NN clarity_NN good_JJ display_NN works_VBZ bad_JJ ._.24_CD
camera_NN acceptable_JJ pretty_RB good_JJ flash_NN ._.25_CD perfect_JJ
camera_NN brilliant_JJ flash_NN light_NN ._.26_CD camera_NN agreeable_JJ
flash_NN working_VBG good_JJ ._.27_CD camera_NN is_VBZ n't_RB best_JJS 5_CD
mp_NN bad_JJ audio_NN ._.28_CD camera_NN takes_VBZ good_JJ pictures_NNS
._.29_CD camera_NN takes_VBZ colorful_JJ photos_NNS ._.30_CD camera_NN
worked_VBD great_JJ pictures_NNS wonderful_JJ ._.31_CD battery_NN lasts_VBZ
half_JJ day_NN disappointing_JJ ,_, memory_NN good_NN ,_, keypad_VBD
little_RB tight_JJ ._.32_CD battery_NN disappointing_JJ memory_NN erratic_JJ ,_,
great_JJ screen_NN ._.33_CD battery_NN bad_JJ issues_NNS capability_NN
```

Fig. 5.3.4 Discretization

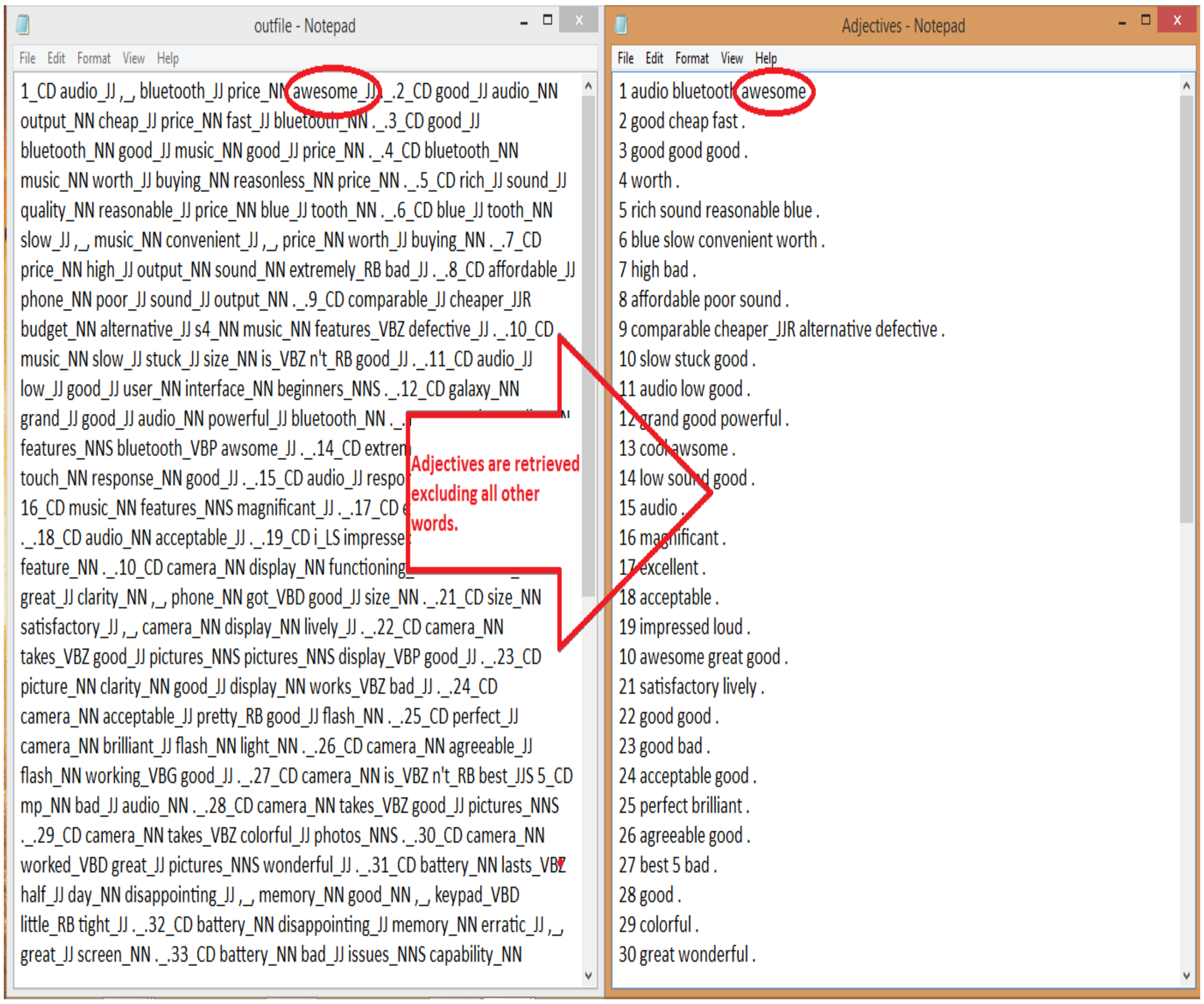
Fig. 5.3.4.1 Nouns


outfile - Notepad
File Edit Format View Help
Nouns - Notepad
File Edit Format View Help
Nouns are retrieved excluding all other words.

```
1_CD audio_JJ ,_, bluetooth_JJ price_NN awesome_JJ ._.2_CD good_JJ audio_NN
output_NN cheap_JJ price_NN fast_JJ bluetooth_NN ._.3_CD good_JJ
bluetooth_NN good_JJ music_NN good_JJ price_NN ._.4_CD bluetooth_NN
music_NN worth_JJ buying_NN reasonless_NN price_NN ._.5_CD rich_JJ sound_JJ
quality_NN reasonable_JJ price_NN blue_JJ tooth_NN ._.6_CD blue_JJ tooth_NN
slow_JJ ,_, music_NN convenient_JJ ,_, price_NN worth_JJ buying_NN ._.7_CD
price_NN high_JJ output_NN sound_NN extremely_RB bad_JJ ._.8_CD affordable_JJ
phone_NN poor_JJ sound_JJ output_NN ._.9_CD comparable_JJ cheaper_JJR
budget_NN alternative_JJ s4_NN music_NN features_VBZ defective_JJ ._.10_CD
music_NN slow_JJ stuck_JJ size_NN is_VBZ n't_RB good_JJ ._.11_CD audio_JJ
low_JJ good_JJ user_NN interface_NN beginners_NNS ._.12_CD galaxy_NN
grand_JJ good_JJ audio_NN powerful_JJ bluetooth_NN ._.13_CD cool_JJ audio_NN
features_NNS bluetooth_VBP awsome_JJ ._.14_CD extreme
touch_NN response_NN good_JJ ._.15_CD audio_JJ respons
16_CD music_NN features_NNS magnificant_JJ ._.17_CD ex
._.18_CD audio_NN acceptable_JJ ._.19_CD i_LS impressed_
feature_NN ._.10_CD camera_NN display_NN functioning_V
great_JJ clarity_NN ,_, phone_NN got_VBD good_JJ size_NN ._.21_CD size_NN
satisfactory_JJ ,_, camera_NN display_NN lively_JJ ._.22_CD camera_NN
takes_VBZ good_JJ pictures_NNS pictures_NNS display_VBP good_JJ ._.23_CD
picture_NN clarity_NN good_JJ display_NN works_VBZ bad_JJ ._.24_CD
camera_NN acceptable_JJ pretty_RB good_JJ flash_NN ._.25_CD perfect_JJ
camera_NN brilliant_JJ flash_NN light_NN ._.26_CD camera_NN agreeable_JJ
flash_NN working_VBG good_JJ ._.27_CD camera_NN is_VBZ n't_RB best_JJS 5_CD
mp_NN bad_JJ audio_NN ._.28_CD camera_NN takes_VBZ good_JJ pictures_NNS
._.29_CD camera_NN takes_VBZ colorful_JJ photos_NNS ._.30_CD camera_NN
worked_VBD great_JJ pictures_NNS wonderful_JJ ._.31_CD battery_NN lasts_VBZ
half_JJ day_NN disappointing_JJ ,_, memory_NN good_NN ,_, keypad_VBD
little_RB tight_JJ ._.32_CD battery_NN disappointing_JJ memory_NN erratic_JJ ,_,
great_JJ screen_NN ._.33_CD battery_NN bad_JJ issues_NNS capability_NN
```

```
1 price .
2 audio output price bluetooth .
3 bluetooth music price .
4 bluetooth music buying reasonless price .
5 quality price tooth .
6 tooth music price buying .
7 price output sound .
8 phone output .
9 budget s4 music .
10 music size .
11 user interface beginners .
12 galaxy audio bluetooth .
13 audio features .
14 touch response .
15 response good .
16 music features .
17 audio .
18 audio .
19 sound feature .
10 camera display clarity phone size .
21 size camera display .
22 camera pictures pictures .
23 picture clarity display .
24 camera flash .
25 camera flash light .
26 camera flash .
27 camera 5 mp audio .
28 camera pictures .
29 camera photos .
30 camera pictures .
```

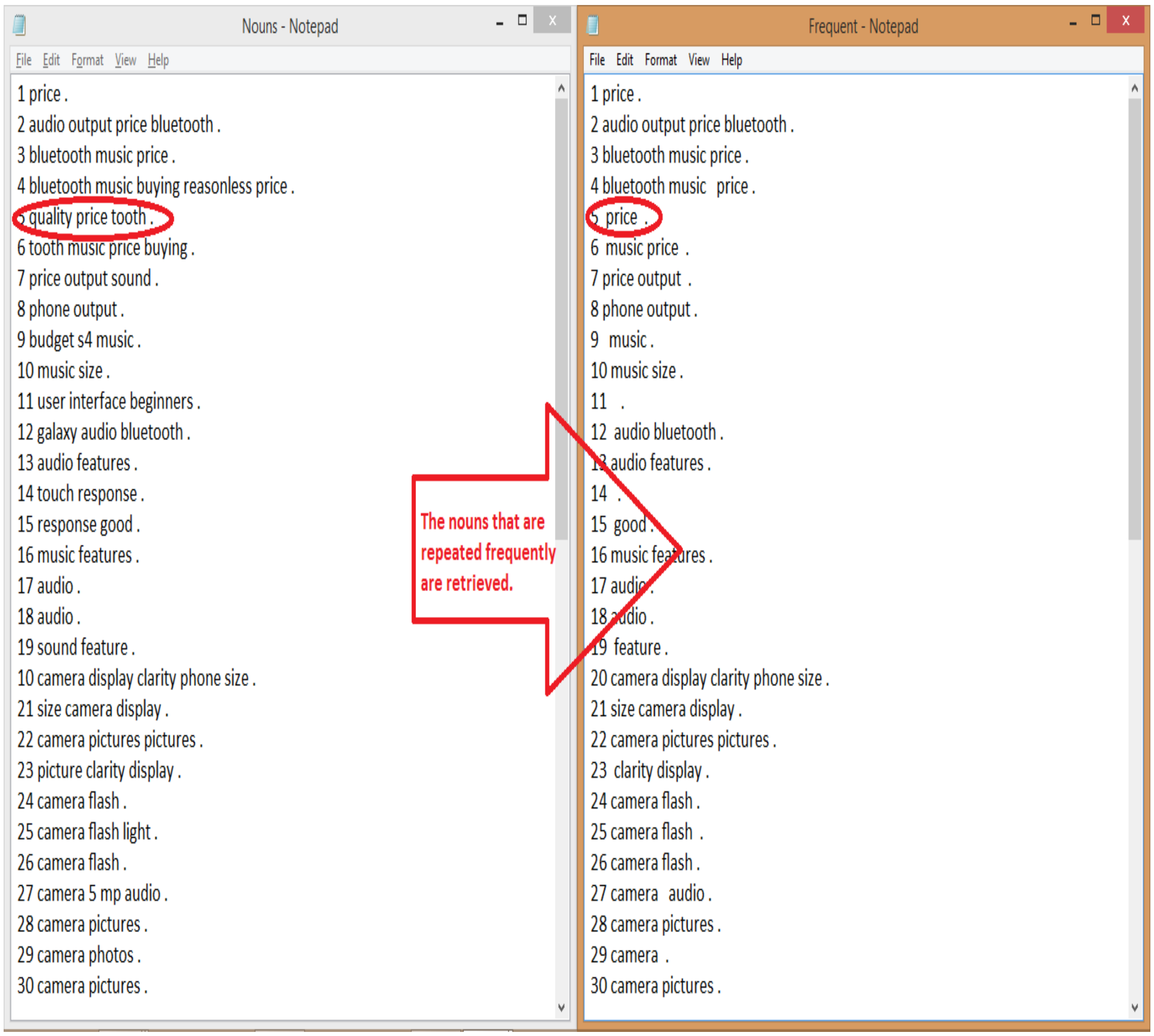
Fig. 5.3.4.2 Adjectives


outfile - Notepad
File Edit Format View Help
Adjectives - Notepad
File Edit Format View Help
Adjectives are retrieved excluding all other words.

```
1_CD audio_JJ ,_, bluetooth_JJ price_NN awesome_JJ ._.2_CD good_JJ audio_NN
output_NN cheap_JJ price_NN fast_JJ bluetooth_NN ._.3_CD good_JJ
bluetooth_NN good_JJ music_NN good_JJ price_NN ._.4_CD bluetooth_NN
music_NN worth_JJ buying_NN reasonless_NN price_NN ._.5_CD rich_JJ sound_JJ
quality_NN reasonable_JJ price_NN blue_JJ tooth_NN ._.6_CD blue_JJ tooth_NN
slow_JJ ,_, music_NN convenient_JJ ,_, price_NN worth_JJ buying_NN ._.7_CD
price_NN high_JJ output_NN sound_NN extremely_RB bad_JJ ._.8_CD affordable_JJ
phone_NN poor_JJ sound_JJ output_NN ._.9_CD comparable_JJ cheaper_JJR
budget_NN alternative_JJ s4_NN music_NN features_VBZ defective_JJ ._.10_CD
music_NN slow_JJ stuck_JJ size_NN is_VBZ n't_RB good_JJ ._.11_CD audio_JJ
low_JJ good_JJ user_NN interface_NN beginners_NNS ._.12_CD galaxy_NN
grand_JJ good_JJ audio_NN powerful_JJ bluetooth_NN ._.
features_NNS bluetooth_VBP awsome_JJ ._.14_CD extren
touch_NN response_NN good_JJ ._.15_CD audio_JJ respo
16_CD music_NN features_NNS magnificant_JJ ._.17_CD
._.18_CD audio_NN acceptable_JJ ._.19_CD i_LS impresse
feature_NN ._.10_CD camera_NN display_NN functioning_
great_JJ clarity_NN ,_, phone_NN got_VBD good_JJ size_NN ._.21_CD size_NN
satisfactory_JJ ,_, camera_NN display_NN lively_JJ ._.22_CD camera_NN
takes_VBZ good_JJ pictures_NNS pictures_NNS display_VBP good_JJ ._.23_CD
picture_NN clarity_NN good_JJ display_NN works_VBZ bad_JJ ._.24_CD
camera_NN acceptable_JJ pretty_RB good_JJ flash_NN ._.25_CD perfect_JJ
camera_NN brilliant_JJ flash_NN light_NN ._.26_CD camera_NN agreeable_JJ
flash_NN working_VBG good_JJ ._.27_CD camera_NN is_VBZ n't_RB best_JJS 5_CD
mp_NN bad_JJ audio_NN ._.28_CD camera_NN takes_VBZ good_JJ pictures_NNS
._.29_CD camera_NN takes_VBZ colorful_JJ photos_NNS ._.30_CD camera_NN
worked_VBD great_JJ pictures_NNS wonderful_JJ ._.31_CD battery_NN lasts_VBZ
half_JJ day_NN disappointing_JJ ,_, memory_NN good_NN ,_, keypad_VBD
little_RB tight_JJ ._.32_CD battery_NN disappointing_JJ memory_NN erratic_JJ ,_,
great_JJ screen_NN ._.33_CD battery_NN bad_JJ issues_NNS capability_NN
```

```
1 audio bluetooth awesome
2 good cheap fast .
3 good good good .
4 worth .
5 rich sound reasonable blue .
6 blue slow convenient worth .
7 high bad .
8 affordable poor sound .
9 comparable cheaper_JJR alternative defective .
10 slow stuck good .
11 audio low good .
12 grand good powerful .
13 cool awsome .
14 low sound good .
15 audio .
16 magnificant .
17 excellent .
18 acceptable .
19 impressed loud .
10 awesome great good .
21 satisfactory lively .
22 good good .
23 good bad .
24 acceptable good .
25 perfect brilliant .
26 agreeable good .
27 best 5 bad .
28 good .
29 colorful .
30 great wonderful .
```

Fig. 5.3.5 Frequent Noun Generation


Nouns - Notepad
File Edit Format View Help
Frequent - Notepad
File Edit Format View Help
The nouns that are repeated frequently are retrieved.

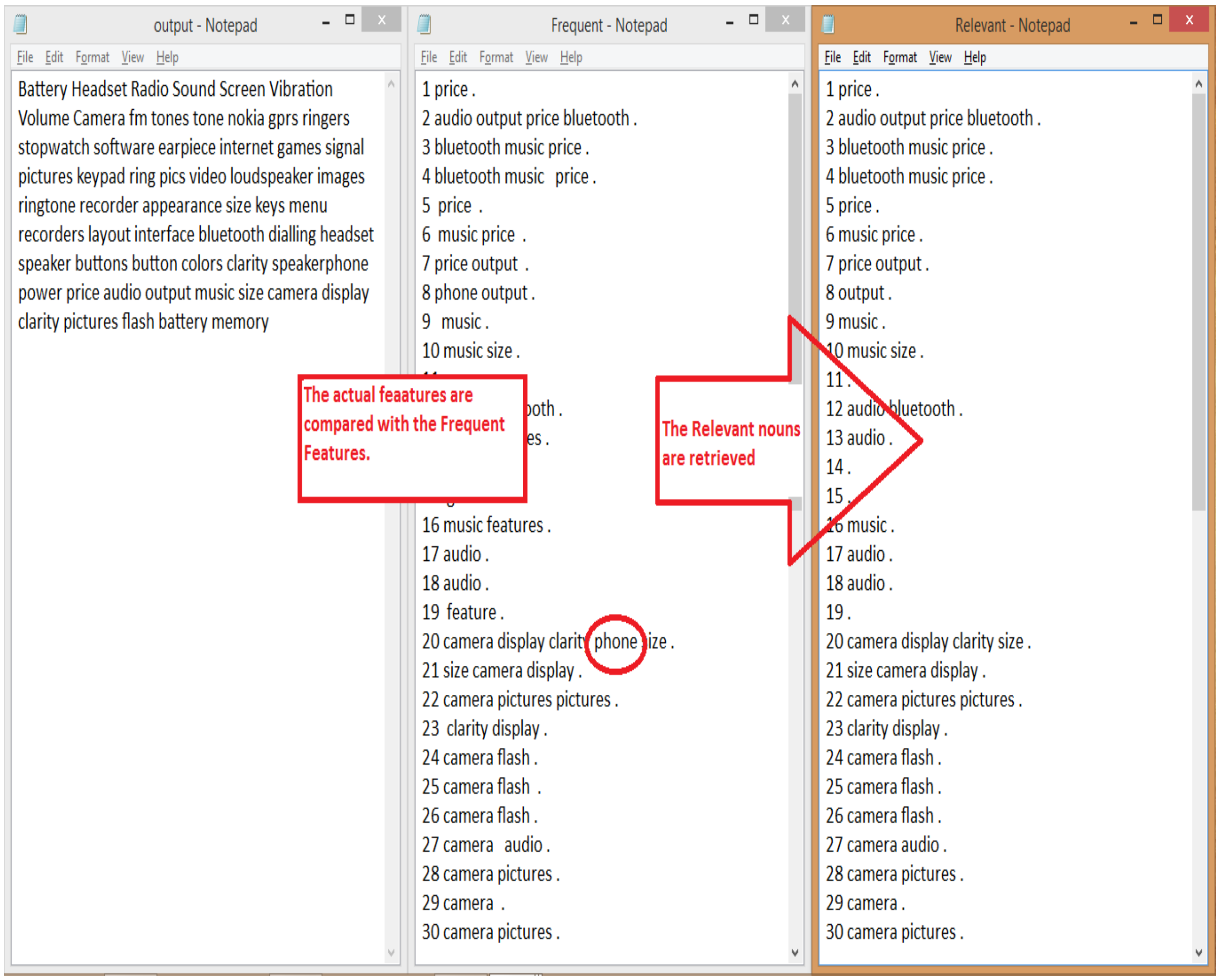
Nouns - Notepad:

```
1 price .
2 audio output price bluetooth .
3 bluetooth music price .
4 bluetooth music buying reasonless price .
5 quality price tooth .
6 tooth music price buying .
7 price output sound .
8 phone output .
9 budget s4 music .
10 music size .
11 user interface beginners .
12 galaxy audio bluetooth .
13 audio features .
14 touch response .
15 response good .
16 music features .
17 audio .
18 audio .
19 sound feature .
10 camera display clarity phone size .
21 size camera display .
22 camera pictures pictures .
23 picture clarity display .
24 camera flash .
25 camera flash light .
26 camera flash .
27 camera 5 mp audio .
28 camera pictures .
29 camera photos .
30 camera pictures .
```

Frequent - Notepad:

```
1 price .
2 audio output price bluetooth .
3 bluetooth music price .
4 bluetooth music  price .
5 price .
6 music price .
7 price output .
8 phone output .
9  music .
10 music size .
11  .
12 audio bluetooth .
13 audio features .
14 .
15 good .
16 music features .
17 audio .
18 audio .
19 feature .
20 camera display clarity phone size .
21 size camera display .
22 camera pictures pictures .
23 clarity display .
24 camera flash .
25 camera flash .
26 camera flash .
27 camera  audio .
28 camera pictures .
29 camera .
30 camera pictures .
```

Fig. 5.3.6 Relevant Noun Generation

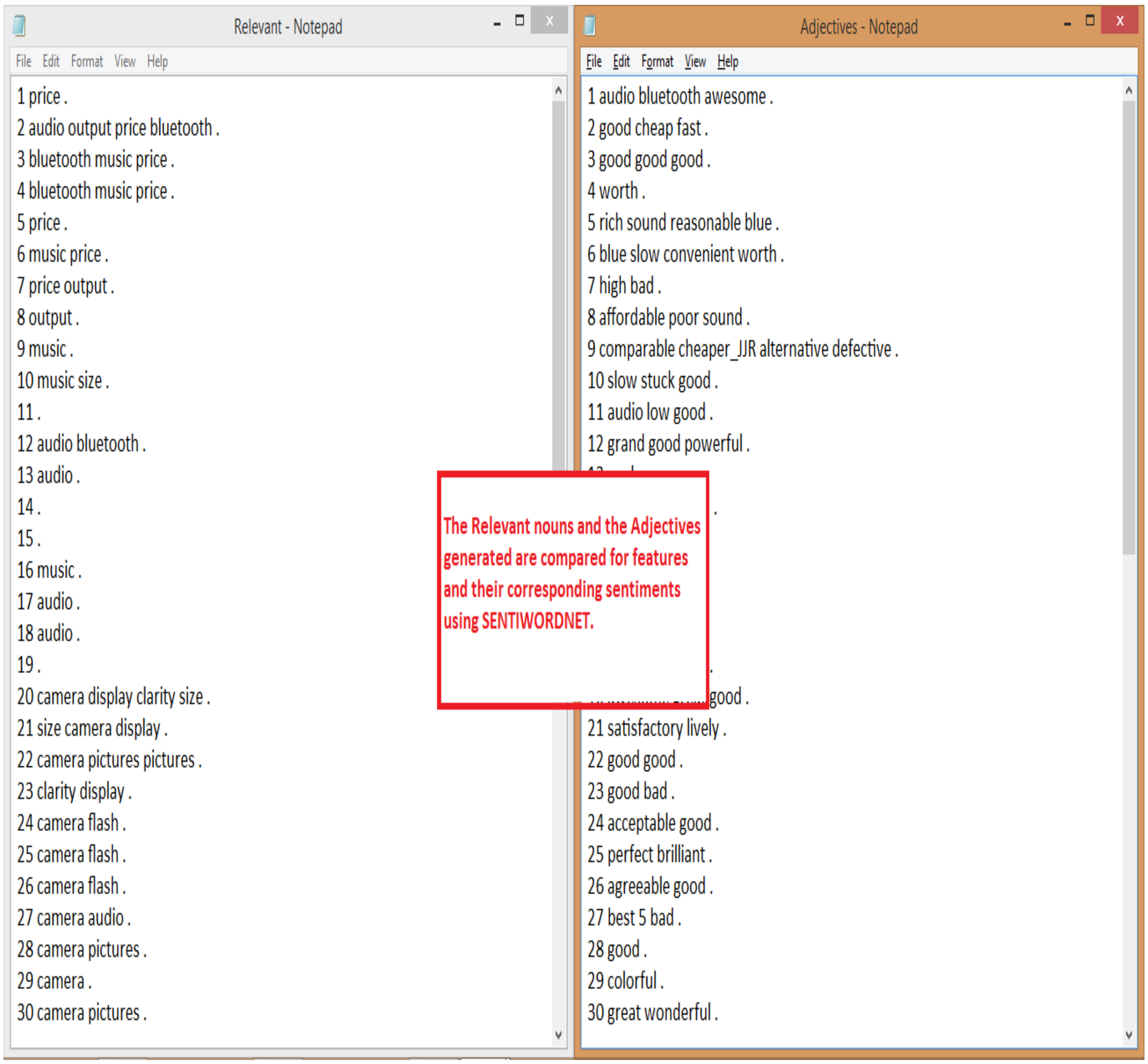

output - Notepad
File Edit Format View Help
Frequent - Notepad
File Edit Format View Help
Relevant - Notepad
File Edit Format View Help
The actual feaatures are compared with the Frequent Features.
The Relevant nouns are retrieved

```
Battery Headset Radio Sound Screen Vibration
Volume Camera fm tones tone nokia gprs ringers
stopwatch software earpiece internet games signal
pictures keypad ring pics video loudspeaker images
ringtone recorder appearance size keys menu
recorders layout interface bluetooth dialling headset
speaker buttons button colors clarity speakerphone
power price audio output music size camera display
clarity pictures flash battery memory
```

```
1 price .
2 audio output price bluetooth .
3 bluetooth music price .
4 bluetooth music   price .
5  price  .
6  music price  .
7 price output  .
8 phone output .
9   music .
10 music size .
11
              ooth .
              es .


16 music features .
17 audio .
18 audio .
19  feature .
20 camera display clarity phone size .
21 size camera display .
22 camera pictures pictures .
23  clarity display .
24 camera flash .
25 camera flash  .
26 camera flash .
27 camera   audio .
28 camera pictures .
29 camera  .
30 camera pictures .
```

```
1 price .
2 audio output price bluetooth .
3 bluetooth music price .
4 bluetooth music price .
5 price .
6 music price .
7 price output .
8 output .
9 music .
10 music size .
11 .
12 audio bluetooth .
13 audio .
14 .
15 .
16 music .
17 audio .
18 audio .
19 .
20 camera display clarity size .
21 size camera display .
22 camera pictures pictures .
23 clarity display .
24 camera flash .
25 camera flash .
26 camera flash .
27 camera audio .
28 camera pictures .
29 camera .
30 camera pictures .
```

Fig. 5.3.7 Sentiment Word Extraction



```
1 price .
2 audio output price bluetooth .
3 bluetooth music price .
4 bluetooth music price .
5 price .
6 music price .
7 price output .
8 output .
9 music .
10 music size .
11 .
12 audio bluetooth .
13 audio .
14 .
15 .
16 music .
17 audio .
18 audio .
19 .
20 camera display clarity size .
21 size camera display .
22 camera pictures pictures .
23 clarity display .
24 camera flash .
25 camera flash .
26 camera flash .
27 camera audio .
28 camera pictures .
29 camera .
30 camera pictures .
```

```
1 audio bluetooth awesome .
2 good cheap fast .
3 good good good .
4 worth .
5 rich sound reasonable blue .
6 blue slow convenient worth .
7 high bad .
8 affordable poor sound .
9 comparable cheaper_JJR alternative defective .
10 slow stuck good .
11 audio low good .
12 grand good powerful .
[illegible]
[illegible] .
[illegible]
[illegible]
[illegible]
[illegible]
[illegible] .
[illegible] good .
21 satisfactory lively .
22 good good .
23 good bad .
24 acceptable good .
25 perfect brilliant .
26 agreeable good .
27 best 5 bad .
28 good .
29 colorful .
30 great wonderful .
```

The Relevant nouns and the Adjectives generated are compared for features and their corresponding sentiments using SENTIWORDNET.

6. CONCLUSION

Explicit online customer reviews on a product are processed in such a way that they are repaired, tokenized, preprocessed, their parts of speech are known and only frequently discussed relevant features in the review set are mined using different algorithms. The reviews which have relevant frequent features are considered for sentiment analysis. sentiments are extracted from the reviews using sentiword.net. And then Trend analysis of the product is extracted.

In future, more work is needed on further refining techniques mentioned in this project and to deal with the outstanding problems like subjectivity classification, word sentiment classification based on machine learning techniques and opinion extraction problem. A variety of steps can be taken to expand this work further by creation of categories for sentiment word senses, where the SentiWordNET is mined for categories of word senses which are particular to products and make the sentiment analysis much easier and reduce the manual work done.

7.REFERENCES

1. Aspect and Entity Extraction for Opinion Mining-Lei Zhang and Bing Liu.
2. A survey on sentiment detection of reviews-Huifeng Tang, Songbo Tan, Xueqi cheng.
3. Stop Word List: -- http://xpo6.com/list-of-english-stop-words/.
4. http://www.tutorialspoint.com/java/java_files_io.htm
5. http://www.tutorialspoint.com/java_xml/java_dom_parse_document.htm
6. http://www.tutorialspoint.com/java_xml/java_dom_query_document.htm
7. http://www.tutorialspoint.com/java/java_string_split.htm
8. http://www.tutorialspoint.com/java/java_string_trim.htm
9. http://blog.ostermiller.org/opening-jar-files
10. https://docs.oracle.com/javase/tutorial/deployment/jar/basicsindex.html
11. https://docs.oracle.com/javase/tutorial/essential/io/file.html
12. https://www.caveofprogramming.com/java/java-file-reading-and-writing-files-in-java.html
13. SentiWordNET source: -- http://sentiwordnet.isti.cnr.it/
14. Obtaining feature and sentiment based linked instance RDF data from unstructured reviews using ontology-based machine learning --D. Teja Santosh, B. Vishnu Vardhan.
15. http://nlp.stanford.edu/software/tagger.shtml
16. http://nlp.stanford.edu/software/pos-tagger-faq.shtml#b