Learning and Evaluating Quality Features of Web Services through User Review Sentiment Analysis

Ameya Murar
Department of Computer Science
Rochester Institute of Technology
aam9527@rit.edu

ABSTRACT
Web Services can be evaluated in a standard way by analyzing the various features of Quality of Service (QoS). Some of the features include security, invocation fee, reliability, response time etc. It is very important to know what the user feels about these features. User sentiment about a particular service can help the developer of that service to make improvements and also help other users in selecting the correct web service. Traditionally, QoS parameters were based on the information provided by the developers of the service itself or the results of the tests obtained by invoking the service. This is not a reliable way of analyzing the quality features as the authenticity of the provider is largely unknown along with the testing environment. Therefore, a more efficient approach is necessary to evaluate the quality features of Web Services.

The aim of this project was to analyze the sentiments of users from Stack Overflow about the popular 100 Web Services. The list of API's was extracted from Programmable Web. We performed sentiment analysis on the posts extracted from Stack Overflow and accordingly list the various features affecting the user.

1. INTRODUCTION
Identifying and extracting subjective knowledge by using various natural language processing techniques and text mining is known as sentiment analysis. It is a very important tool which helps to understand the opinion of people regarding a product or a movie or a web service. In this project, we performed sentiment analysis on the posts by users who have used certain Web API's and categorizing them as positive and negative. However, this involved a lot of challenges. The questions and comments about a particular API were not grouped under a topic. As a result, we need to perform topic modelling on the posts. We used Latent Aspect Rating Analysis (LARA) for performing topic modelling. Later, we performed the sentiment analysis using the Support Vector Machines (SVM). This paper summarizes the various papers which I have read to understand the problem, develop a solution towards solving the problem, implementation and the results obtained.

When we want to learn a new technology or if we are facing any issue while using an existing software, we generally turn to programming forums for help. These programming forums contain a huge amount of knowledge and information that it becomes tedious to scroll through each post to find if it is relevant to us. One of the disadvantages of using these programming forums is that it contains a lot of unnecessary or redundant posts and comments. If the information is not available, then we can post our question on the forum which may be answered by the experts in that field. We have used one such programming forum for our sentiment mining experiment.

If we perform the following search query ‘YouTube API’ on Stack Overflow, then we will get more than 14,000 results. It is quite time consuming and a tedious thing to scroll through these many results to find the result which we want. Initially, it may seem that a tool can be created which can give us the required information from these forums in a short time. However, creation of such a tool involves a lot of issues. To begin with, the tool will need to collect data from the forum, aggregate it and finally analyze it. This is a non-trivial task as it involves many steps like data cleaning, preparation, analysis etc. Some of the issues are listed below in detail.

2. MOTIVATION
There are various programming forums on the web which may help the users or developers in their development problems. However, the information which the user or the developer wants is not easily available on such forums. They have to literally scroll through many pages to find the solution which they want. In such case, it would be helpful if the user or developer gets the information he/she wants in one click. We decided to build a system which can help the user or the developer in determining the correct API to use based on the previous experiences of the users. Such a system has dual benefits. We believed that such a system will considerably reduce the time for the user who wants to use the API for his development purpose. Also, it can help the developers of the respective API’s to specifically focus on the areas where the users feel the need for improvement.

3. CHALLENGES
Some of the challenges which were discussed briefly in the Introduction are explained in this section.
Data Collection: On Stack Overflow, when we performed a search query, we only got at most 50 questions on each page. Since we had decided to consider 3000 relevant questions from each API for analysis, we needed to consider information from 60 such pages for each API. Also, a lot of unnecessary information was present on each page. Such information included links to careers page, register, sign in, etc. We needed to remove such information as it was not useful for the purpose of analysis. Also, sometimes the questions involved snippets of code written by the user who wants to use it as a reference for asking the question. Such snippets do not give any information and hence were not considered.

Data Integration: The data extracted from Stack Overflow is raw and was converted to a suitable format. The purpose of doing this was that the results of the mining analysis would have been hampered if the data was not in correct format as unnecessary information would have been used in the building of the training model. Another issue was that of storing this huge amount of information. We used a relational database for this purpose. However, if we had performed a join operation, then this operation would have been expensive when using a RDBMS. An alternative would have been to use a NoSQL database such as MongoDB.

Data Analysis: This step was completely dependent on what data was analysed. If the data was irrelevant, then the results of the analysis would have been misleading. As stated earlier, we performed topic modelling on the posts extracted from Stack Overflow. We performed stop word removal on these posts. Stop words included if, or, and etc. If they were not removed, then they might have got considered as key words in process of data analysis which in turn would have distorted the results. If there were sarcastic posts in our corpus, then it would have become difficult to perform sentiment mining. This might have happened because the text of the posts says one thing, which in fact is the opposite of the other thing, which the user wanted to convey. Such posts also could have skewed the results of analysis.

To begin with, a list of API’s was needed which can be used to query Stack Overflow. Such a list was obtained from programmableweb.com. Programmable Web is a repository containing all the public API’s. After such a list was obtained, we got a list of questions for each API from Stack Overflow. After we got the posts, we performed topic modelling. Topic Modelling was performed using Latent Dirichlet Allocation. Then, we manually labelled the posts as positive or negative. Finally, we built our training model based on the sentiments labelled to the posts.

4. OVERVIEW

In the next section, a comprehensive summary of all the papers read is given. In order to gain a complete understanding of the research work already done in this area, the papers were read and analyzed. After reading the papers, a 6 step approach to address the problem was devised.

5. RELATED WORK

To get a good understanding of the problem and to devise an efficient strategy for implementation, ten papers were read. This section summarizes all the papers.

5.1 Sentiment Analysis on Web Services

The paper by Liu et al.[5] proposes a novel approach for extracting, ranking and evaluating quality features of Web Services. Their approach involves classification of reviews into positive and negative. For performing user classification, they have used augmented logistic regression also known as l1 regularized logistic regression. Initially, their approach involves learning QoS features from User Reviews. The learning process is divided into two phases: data pre-processing and sentiment analysis. The data pre-processing phase involves four steps: Review cleaning, Word tokenizing and stemming, Stop word removing and POS tagging. Review cleaning involves evaluation of each review and filtering out the invalid reviews. Word tokenizing and stemming involves tokenization of each review into a bag of words followed by stemming of words to their root forms, such as from production to produce. Stop word removing involves removal of words which are useless for feature selection and sentiment analysis. POS tagging involves tagging each word of a review into different parts of speech. After the data pre-processing phase, a frequency matrix is generated, which is then given as input to the next phase: sentiment analysis. Sentiment analysis gives twofold result. A model which confirms whether a reviews is positive or negative is obtained. The feature list where each feature is a parameter that a user takes into consideration when the QoS of a service is evaluated is also obtained. The use of logistic regression for feature selection is an interesting approach and could be used in our project. It can help in improving the accuracy of the results obtained by sentiment analysis and also quality features can be extracted more elegantly.

The paper by Rajat et al.[1] proposes a novel method of ranking users by their reputation for each web service. Their approach involves a step-wise methodology which culminates in visualization of sentiment distribution. The first phase involves extracting the most popular API’s. JSoup is used for getting the HTML code from programmable web. The list of API includes the 100 most popular API’s. The second phase involves performing query on stack overflow. For each of the 100 API’s, a query is performed on stack overflow to get a list of questions in the form of URL’s. Similar to the first phase, JSoup is used to extract the HTML code from stack overflow. The third phase involves retrieving details from posts. Relevant information like the title of the post, date posted, text of the post, comments, user who answered the question, reputation of the user and the number of upvotes are extracted and stored in a MongoDB database. The fourth phase involves topic modelling using latent dirichlet allocation. This step is necessary to extract the quality features which affect the users. The University of Massachusetts has created the LDA implementation known as MALLETT. The input to the MALLET model is the text of the post extracted in the previous phase. The output from MALLET will be a list of words describing each document. The fifth phase involves sentiment analysis. Stanford sentiment analyzer is used for the same. However, the results obtained from the analyzer do not reflect the correct sentiment. To verify these results, Support Vector Machines (SVM) are used. The sixth phase involves user rankings. In this phase, extraction of the user information is done. User information consists of current reputation, number of upvotes received etc. A new reputation is computed based on a certain formula for each user. Finally, the users are ranked in decreasing order of reputation for each web service. The
In the ranking phase, a ranking model is developed based on the weight vectors. In the Visualization phase, the results generated by the USTM are visualized. The authors also talk about the potential limitations of the framework. They have mentioned two drawbacks: 1) The category rules defined by the authors are not always applicable to all the other app developers. 2) The framework is not generalized.

5.2 User-Sentiment Topic Model

The paper by Zong et al.[10] proposes an interesting model known as user-sentiment topic model (USTM). Social network analysis along with the user’s interests and sentiments can be captured efficiently by this model. The features of the model are that: 1) For detecting the sentiments of the authors, USTM is used in an unsupervised way. 2) Sentiment analysis is incorporated into topic modelling to understand the user’s sentiments about the topics specifically by USTM.

The model considers the sentiment and the topic simultaneously which helps in getting a more accurate result. The USTM is a four-layered model which is devised from the Latent Dirichlet Allocation (LDA). An assumption is made that the sentiment information is related to the user, topics are related to both the user and the sentiment and words are related to both the sentiment label and the topics. In the experiments which they performed, they have used one Chinese dataset and two English datasets. The Chinese dataset consisted of 7192 documents and it was named ‘Douban’. One of the English dataset consisted of 2000 documents and it was obtained from the Cornell University. It was named ‘Movie review’. Another one consisted of 39861 documents and it was obtained from CMU. It was named ‘Enron’. This was followed by preprocessing, which consisted of two steps: 1) Chinese Word Segmentation and 2) Filtering. ICTCLAS (Chinese Lexical Analysis System) was used to differentiate each word in a Chinese sentence. ICTCLAS was also used for POS tagging. This process was not needed for the English datasets. In filtering, stop words were removed to reduce the size of the vocabulary along with the punctuations and non-English characters from the English datasets.

Next, they defined their model priors. They felt that detecting sentiments is more complex than extracting features or topics. Prior information such as sentiment lexicon can be incorporated to improve the sentiment detection. However, it was observed that as more and more sentiment lexicon was incorporated, the performance did not show any improvement. As a result, they chose the sentiment paradigm word list which consisted of 21 positive and negative words respectively. Finally, in the model which they setup, sentiment labels for each word were only initialized by the model priors. The sentiment labels were set as positive, negative and neutral. Three performance metrics: Convergence Analysis, Accuracy Analysis and Scalability Analysis were used to evaluate the model qualitatively and quantitatively.

The paper by Pletea et al.[7] proposes a discussion on security-related issues on GitHub. For the purpose of analysis, they have used 60,658 commits and 54,892 pull requests from the projects available in MSR 2014 Mining Challenge Dataset. They have used a keywords-based approach for the identification of cluster with application security. Common words related to application security are manually selected. Only the top 25 co-occurring tags are selected to ensure that the tags are relevant. Also, the tags of programming languages are excluded. After this step, porter’s stemming algorithm is applied on the keyword set. For performing
sentiement analysis, the Natural Language Text Processing Tool (NLTK) is used. The output we obtain is a neutral, positive or negative text. We also obtain an aggregate label which summarizes the three scores. After performing statistical analysis along with a case study, it was found that security-related discussions comprise 10 percent of all the discussions on GitHub.

The paper by Murgia et al.[6] studies developer emotions in a relative context of software maintenance and evolution. 800 comments were studied for this purpose. The study also analyzes issue reports and checks if any emotional information about development of software is carried by them. This is the initial step in the development of a tool for mining emotions in software development reports. The reports are taken from open source systems. The dataset used for the purpose consists of the repository belonging to the Apache Software Foundation. They also parsed Apache’s Jira-based repository and extracted all the issue reports since last 16 years. The developers’ comments were also extracted. Their focus was only on publicly available issue reports. To avoid coarse granularity, sampling was performed on the comments. The next step was to perform emotion mining. The six primary emotions of Parrott i.e., love, joy, fear, surprise, sadness and anger were used to label the emotions. Cohen’s value or Fleiss’ value is used to calculate the degree of inter-rater agreement.

5.3 Posts Recommendation

The paper by Wang et al.[9] proposes a methodology that recommends posts from stack overflow which may concern API design-related issues using social network analysis. They have proposed two recommendation approaches: reputation-based recommender approach and a random recommender approach for comparison. The first step in their approach involves experts identification. Only the questions from experienced users are kept and the questions from novice users are removed. A bow-tie approach is used for this purpose. The second step involves dimension reduction. The topics are organized according to the interests or concerns of the large developer community. Latent Dirichlet Allocation (LDA) was used for this purpose. The third step involves Post Selection. Questions which have been answered after a considerable amount of time after they have been posted indicates that the developer is still facing some issues. If they have been answered quickly, it means that it is a known issue which might have been solved before. The fourth and the final step involves ranking of questions and posts according to their difficulty. Difficulty here stands for the time taken to answer the question. A parameter known as Questions Score (QS) is used for ranking the questions. Finally, they have performed a case study using posts related to iOS and Android to evaluate their methodology. A ground truth is established for evaluation.

5.4 Aspect Model for Online Reviews

The paper by Jo et al.[4] proposes a probabilistic model known as Sentence-LDA (SLDA) which focus on aspect oriented sentence generation. Then, the SLDA is extended to Aspect and Sentiment Unification Model (ASUM). The authors have introduced the concept of Senti-aspects in which the pairs of aspect, sentiment are discovered by ASUM. SLDA and ASUM were applied to the reviews about electronic devices and restaurants. The advantage of ASUM is that no pre-labeling of sentiments to reviews is necessary. For the purpose of experimentation, 22000 reviews about Electronics and 30000 reviews about Restaurants were selected. Preprocessing was done to remove web URL’s and punctuations. Then, stemming was performed using the Porter’s stemming algorithm. Regular expressions were used to handle negation in reviews. They performed four experiments to test the two models: ASUM and SLDA. The first experiment involved testing the results evaluated by SLDA. The second experiment involved testing the sentiment aspects evaluated by ASUM. The third experiment involved testing the sentiment words found by ASUM. The fourth experiment involved testing the performance of sentiment classification of ASUM.

The paper by Wang et al.[8] proposes a unique approach towards the problem of data analysis on opinionated text. They define this problem as ‘Latent Aspect Rating Analysis’ (LARA). While analyzing the sentiment of the review, the latent opinion hidden in each of the reviews and the emphasis on the various reviews are considered. Their approach involves designing a model which can perform probabilistic rating regression. The name of this model is ‘Latent Rating Regression’ (LRR). This model helps in solving the above mentioned problem generally. This model also assists in unearthing a lot of things. First, it helps in discovering the latent ratings of every reviewer on the topics given along with the different weights which the reviewer has placed on the various topics. It takes as input a collection of text reviews along with the total ratings and the topic specifications. The authors used a data set consisting of hotel reviews to prove their hypothesis about the ‘Latent Rating Regression’ model. They found out some dissimilarities in the topic ratings even when there were similarities in the total topic ratings. They have based their approach on two stages. The first stage is that of Aspect Segmentation. In this stage, subsets implying to each topic are created. Then, they are mapped to the text sentences present in a review. This is the motivation behind this step. The output of the first stage is used as the input for the second stage. In the second stage, the ‘Latent Rating Regression’ is applied to the results obtained in the first stage. In this stage, the ratings and the weights of the topics are analyzed by the model. First, an assumption of the behavior of the user’s review is done. Then, this assumption is passed to the ‘Latent Rating Regression’ model. For the purpose of analysis, approximately 240,000 reviews from the website ‘TripAdvisor’ were crawled and stored in a database. The reviews were taken for the time duration of 14th February 2009 to 15th March 2009. Basic pre-processing steps were performed on the review texts. These steps include removal of punctuation symbols and stop words, conversion into lower case of every word. Also, the words which did not occur more than 5 times in a document were removed. They applied the ‘Porter Stemmer’ algorithm to each word to perform stemming. They performed Qualitative and Quantitative Evaluation of the reviews.

6. APPROACH

I decided to follow a 6 step approach towards solving the problem of evaluating the quality features of web services through sentiment analysis. The six steps are listed in figure 1.
6.1 Extraction of popular API’s

This was the first step in the project. The list of 100 popular API’s was obtained from www.programmableweb.com. Programmable Web is a popular repository which contains a list of more than 14,000 API’s. These API’s are further classified into various categories like popular, most recent, featured, offering support for REST and/or JSON. Further, the API’s are also classified into several different topics like Weather, Mapping, Mobile, Transportation, Social etc.

JSoup was used to extract the list of API’s. For the purpose of crawling HTML pages, JSoup is popularly used. JSoup is a Java library and integrates easily with any Java framework. It can conveniently parse web pages. JSoup library provides a connect method which can be used to extract the HTML source code of the web page. It can be done by using the URL of the web page as the parameter to be passed. The required tags are selected by the resultant document element.

Some of the popular API’s include: Google Maps API, Twitter API, YouTube API, Flickr API, Facebook API, Amazon Product Advertising API, Twilio API, Microsoft Bing Maps API etc. Figure 2 gives the most popular API’s which we got after selecting the most popular API’s option.

6.2 Performing query on Stack Overflow

This was the second step in the project. In this step, a search query was performed on Stack Overflow to extract questions related to each of the 100 API’s. These queries had to be run one at a time. This is because Stack Overflow asks for human verification when too many URL requests are made. We can immediately navigate to the API page by modifying the URL of the webpage of the Stack Overflow.

For example, the URL can be modified as follows if we want to generate a query on Twitter API.

http://stackoverflow.com/search?q=Twitter+API

Like we extracted the list of API’s in the previous step, JSoup was used to extract the HTML source code of the above page. When the query is extracted, the result is as shown in figure 3.

When we get the HTML source code of the web page after querying Stack Overflow, we got the links to the questions for further processing. The code for extracting the links to the questions are given in figure 4. The links to the questions (posts) on the web page were extracted using a regular expression. A pattern was observed in the link to each question. It was exploited using the regular expression in figure 5.

6.3 Retrieving user information from posts

This was the third step in the project. In the previous step, we extracted the links of the URL’s of the questions posted on Stack Overflow. These links were then matched with the regular expression given in figure 4. We were able to generate a list of URL’s which would then be used to extract information. An example of such post is given in figure 5. The data about which we were concerned includes:

1. Name of the user
2. Date
3. Text of the post
4. Reputation of the user

The data extracted from the posts was stored in a CSV file. After such information is gathered and stored, a sentiment is assigned to the post. As the data is unlabelled, it was necessary to label it for the purpose of training the classification model. We manually labeled around 500 posts from each API. An example for the ‘Amazon EC2 API’ can be seen in figure 6. After labels were assigned to many posts, a training model was built using Support Vector Machines.
Figure 3: Screenshot of Stack Overflow after querying for Amazon EC2 API

```java
for(int i=1;i<11;i++) {
    String src=Integer.toString(i);
    Elements links=doc.select("a[href=");
    String pattern="http://stackoverflow.com/questions/[0-9]+/";
    Pattern regex=Pattern.compile(pattern);
    for(Element src:links) {
        Matcher m=regex.matcher(src.attr("abs:href"));
        if(m.find()) {
            System.out.println(src.attr("abs:href"));
        }
    }
}
```

Figure 4: Code for retrieving the URL's of questions

Figure 5: Regex for retrieving the URL's of questions

Figure 6: Screenshot of a question (post) on Amazon EC2 API in Stack Overflow

Figure 7: Screenshot of the CSV file containing posts related to all the API's
6.4 Topic Modelling

This was the fourth step in the project. The specifics of the various posts were determined in this step. There were some trends or patterns hidden inside documents, in our case, posts. The process of discovering these trends or patterns is known as Topic Modelling. It is an important constituent of text mining. We performed a query with "Twitter API" as the parameter. If the results are observed closely, the first question/post is about posting. The user is asking for a method which can change the name of the application from which the question is posted. We observed that the user is talking about ‘posting’. Thus, such hidden features which are important in text mining were unearthed by performing topic modelling.

To perform topic modelling, we implemented the Latent Aspect Rating Analysis (LARA) methodology. This methodology extracts the latent aspects from the reviews by placing different weights on different aspects before finally converging into a set of aspects based on the overall entity judgment. Topic modeling is also known as Aspect Segmentation. In Aspect Segmentation, various parts of aspects are created. Subsets which correspond to every topic are mapped to from the review sentences. The authors of the Latent Aspect Rating Analysis (LARA) devised a boot-strapping algorithm which helped in finding the latent ratings. The input to the algorithm is a set of reviews along with a selection threshold and a step limit for iteration. Initially, all the reviews were split into sentences. Then, keywords for each aspect in the sentences were matched and each aspect’s hits which were matching were stored in a variable. An aspect label is assigned to each sentence by taking into consideration the hit value stored in a variable before. If multiple aspects are assigned to a sentence, it means that the hits matched which resulted in a tie. Then the square of matching hits is calculated, which is named as ‘sq’. Following this step, according to the value of ‘sq’, the words under each topic are ranked. Considering the selection threshold for each aspect, the topmost words occurring in the sentences are joined. This joining is based on a list containing keywords for each aspect. After each iteration, if there is no change in this list containing keywords for each aspect or if the iteration step limit is breached, then the aspects are presented in a segmented manner. Otherwise, the process is repeated until the condition explained previously matches.

6.5 Sentiment Analysis

6.5.1 Using WEKA

This was the fifth step in the project. This step was to be performed after the posts are combined on the basis of their topics, in the previous step. But, since the posts obtained from Stack Overflow do not have a sentiment value, it was essential to assign a sentiment to them. While there are several sentiment engines available online, I decided to manually assign a sentiment to approximately 1000 posts for each API. This is because these online sentiment engines generally tend to assign a negative sentiment to most of the posts. The technical aspects of the posts cause the sentiment engines to fail on most of the posts and a negative sentiment is generated.

After manually labeling the posts, we planned to build a training model using the Support Vector Machine classification technique. The data used for classification and regression analysis was analyzed by learning supervised models and algorithms known as Support Vector Machines. They are a key component in Machine Learning. When a training dataset is provided, a model is built by the Support Vector Machine which classifies the data into two categories. This makes the Support Vector Machine a non-probabilistic binary linear classifier. A clear gap must exist between the data belonging to the two different categories. This gap must be as wide as possible, as the data is mapped to points in space. When unlabeled data is processed by the Support Vector Machine, it is also mapped as points in the same space and then classified into either of the two categories, so as to determine which side of the gap, defined earlier, they fall on. This is the concept of linear classification. Support Vector Machines also perform non-linear classification. In non-linear classification, input data is mapped to higher-dimensional feature spaces. If the data is unlabeled, then it is not possible to employ a supervised learning algorithm. In such a case, an unsupervised learning algorithm must be implemented. Support Vectors can even handle such cases, where ‘Support Vector Clustering’ must be used. Support Vector Clustering is an unsupervised version of the Support Vector Machines. Industrial applications normally use the Support Vector Clustering algorithm, because in these applications, generally, the data is unlabeled or partially labeled.

We decided to use WEKA and python to perform the sentiment analysis by Support Vector Machines. WEKA provides a library ‘LIBSVM’ which implements the ‘Sequential Minimal Optimization’ algorithm. This algorithm is the basis for kernelized Support Vector Machines. The algorithm handles both classification as well as regression. Going further, it also supports multi-class classification.

Some of the posts to which sentiments were assigned manually:
Positive Posts for Amazon EC2 API
1. The documentation for Amazon EC2 API clearly demonstrates the various functions which can be used when needed.
2. Amazon EC2 API can be invoked at a very low cost as compared to other APIs which offer the same functionality.
3. In the right EC2RDS configuration instance sizes for your traffic levels Magento performs reasonably well on RDS.

Negative Posts for Amazon EC2 API
1. The Spanish version of the documentation is not updated yet.
2. But I have got some new error can u just help with what the error is about.
3. It took a lot of time for me to understand its implementation.

Positive Posts for Google Maps API
1. That part can be loaded with the help of a simple script.
2. It was very easy and it worked fine with my application.
3. I created a mashup with this API and weather API and it worked wonders.

Negative Posts for Google Maps API
1. Tried activating but it didn’t work.
2. I guess everyone who invokes it gets that error.
3. The place and the latitudes don’t match.

Positive Posts for Facebook API
1. My initial thoughts on reading the docs was that it was correct.
2. Awesome, that worked for me.
3. Thanks a bunch. This insight into relative protocol is really helpful.

Negative Posts for Facebook API
1. Unfortunately it doesn’t work that way
2. The comments took a lot of time to load
3. The video was not in sync with the audio

Figure 9: Initial WEKA window on startup

Figure 10: WEKA window after importing the data file

The step-by-step process from loading the file into WEKA to generating the classifier output is given below.

- Loading the CSV data file into WEKA.
  This is the first step in the classification process. We directly loaded the data file into WEKA. The other option is to convert the CSV file into ARFF. ARFF stands for Attribute Relation File Format. Each line in the input data file contains the name of the API, the user who posted the question, the year in which it was posted, the post, the aspect which it talks about and the sentiment. All these fields are comma separated. Figure 9 visualizes the initial WEKA window on startup.

  The figure 10 depicts the WEKA window after importing the data file.

  We can see the 6 attributes on the left and the different aspects of the posts displayed on the right. The
count of each aspect for the ‘Amazon EC2 API’ is also displayed. The blue color indicates that a particular aspect is used in a negative context while the red color indicates a particular aspect in used in a positive context.

- Applying filters on the imported data file.

WEKA provides various pre-processing mechanisms including filters which can be applied to text or numerical data. We can apply various filters as needed to improve the classification accuracy of our dataset. Here, we applied ‘StringToWordVector’ filter. The string attributes are converted by the ‘StringToWordVector’ filter into a collection of attributes. These attributes represent the occurrence of word information. This information is obtained from the text which is present in these strings. The function parameters are modified such that they include the detection of stop words along with stemming. Stemming is a unique information retrieval technique in which a word is traced back to its root. For example, production is traced back to produce. An advantage of using this technique is that many similar words can be tracked down, which helps in optimizing the accuracy. Also, Term Frequency Inverse Document Frequency (TFIDF) can be invoked through this filter. The words which are more important for classification are thus selected. When the ‘StringToWordVector’ filter is applied, the WEKA window looks like the one shown in figure 11.

- Applying the Support Vector Machine algorithm.

After importing the data file and applying the filters to it, the next step is to apply the Support Vector Machine algorithm to it. WEKA provides a function called as ‘LIBSVM’ which is an implementation of the SVM. This library is not pre-installed inside WEKA. It needs to be downloaded from the web and installed on the CLASSPATH. After this installation is complete, we can run the ‘LIBSVM’ library on the imported data file. We can visualize in figure 12 the classifier output which is obtained after applying the ‘LIBSVM’ function on the data.

As we can see in figure 12, we got an accuracy of 86.5%. Out of 1386 instances, 1199 were correctly classified and 187 were incorrectly classified. For the purpose of testing, we used the 10 folds cross validation methodology. Cross validation is also known as rotation estimation. Random partition of the original dataset into 10 same sized subsets of the dataset takes place in 10 fold cross validation. We also increased the folds to 20 to see whether the accuracy of the model can be increased. However, we got the same accuracy of 86.5% which we had obtained when the number of folds were 10. This can be visualized in figure 13.

The reason for the skewed confusion matrix in both the cases (10 folds and 20 folds cross validation) is because the number of negative and positive instances are not equal. This indicates that over the years, people are unsatisfied while using the ‘Amazon EC2 API’.

All the above steps were followed in the same way for the ‘Google Maps API’. As we can see in figure 14, the accuracy obtained was 86.09%. Out of 482 instances considered for testing, 415 instances were correctly classified and 67 instances were incorrectly classified. For the purpose of testing, we used the 66% split method. In this method, 66% of the data was considered for training the dataset while 33% of the data was considered for testing the dataset.

6.5.2 Using scikit-learn

In order to validate the results obtained from WEKA, we decided to use another classification technique. In python, there is a machine learning library known as ‘scikit-learn’ which can be used to perform all the data mining tasks such as classification, clustering, regression etc. Algorithms such as k-means, support vector machines and DBSCAN are included in the ‘scikit-learn’ library. It is a open-source li-
As can be seen in figure 15, we obtained an accuracy of approximately 0.82 which corresponds to 82%. In this case, we worked with the ‘Google Maps API’ dataset. The first two lines in figure 15 imply that ‘pandas’ and ‘numpy’ are imported. Pandas is a library in python which gives several implementations of data structures and data analysis tools. Numpy is also a library in python which can handle scientific computing. The third line indicates how to read a CSV file. ‘X’ and ‘Y’ store the review and the sentiment respectively. The next two lines indicate the training and testing procedure for the given dataset. ‘CountVectorizer’ is used for the conversion to a token count matrix from a set of text documents. ‘TfidfTransformer’ is used for the transformation to a tf-idf representation which is normalized from a token count matrix. ‘SGDClassifier’ provides the implementation for linear classifiers like Support Vector Machines, Logistic Regression etc. It provides training with SGD which is the abbreviation for Stochastic Gradient Descent. The ‘metrics’ library is used to determine the classification accuracy and also other information such as precision, recall and f1-score. The last line gives the accuracy of the model, in this case, it was approximately 0.82 which is nothing but 82%.

As can be seen in figure 16, the metrics for the classification are obtained by building a text report.

### 6.5.3 Using Latent Aspect Rating Analysis method (LARA)

We also decided to use the Latent Aspect Rating Analysis (LARA) method to perform the sentiment analysis. From the topic modeling step which we performed earlier, we had the matrix consisting of the word frequency. For every topic from the matrix, we had the frequency of the words which...
was normalized. The overall sentiment of a review is considered as a response variable. This response variable is nothing but the prediction variable. The frequency of each word in the ratings does not determine the overall sentiment of the review. It is determined by the different weights given to the latent topics which are in each review text.

Initially, we generated a topic rating for each topic and for each review for each of the three API's namely ‘Amazon EC2 API’, ‘Google Maps API’ and ‘Facebook API’. Two vectors namely n-dimensional topic weight vector and topic rating vector were defined. Based on the overall sum of the weights of each review, a rating was generated and assigned to each review. Gaussian distribution was the basis of this final rating for each review. The main motivation behind this technique was to narrow the distance between the total rating and the review descriptions for each API's. On the basis of the ratings of individual topics, the total rating was modelled on the latent topic weights and the sentiment weights of the terms.

Then, we further worked upon the individual behaviors of the different ratings assigned to each review. A multivariate Gaussian distribution was employed for the topic weights prior by considering the parameters of mean and variance. There were various corpus level model parameters which we taken into consideration while generating the overall rating for each review.

Going further, we obtained results in the form of numbers in the range of 0.0045 to 0.67 for the ‘Google Maps API’. For the ‘Amazon EC2 API’, we obtained results in the range of 0.002 to 0.43 and for the ‘Facebook API’, the results were obtained in the range of 0.019 to 0.29. Considering the range of the numbers and going by the way suggested in the paper[8], we decided on setting a threshold value of 0.1. All the ratings which have a rating less than 0.1 were assigned a sentiment tag of ‘Negative’ while all the ratings which have a rating more than 0.1 were assigned a sentiment tag of ‘Positive’. This step was justified by the results which we got below.

For the ‘Google Maps API’, we obtained an accuracy of 0.7773 which is nothing but 78%. Out of 1417 reviews, 1101 reviews were correctly classified whereas 316 reviews were incorrectly classified. For the ‘Amazon EC2 API’, we obtained an accuracy of 0.7569 which is equivalent to 76%. For this API, out of 1387 reviews, 1050 reviews were correctly classified whereas 337 reviews were incorrectly classified. For the ‘Facebook API’, we got an accuracy of 0.7328 which corresponds to 73%. For this particular API, out of 1439 reviews, 1054 were correctly classified and 385 were incorrectly classified.

The accuracy obtained from Latent Aspect Rating Analysis is less as compared to the accuracy which we obtained from Support Vector Machines. This might have happened because we used the string to word vectorizers in WEKA and ‘scikit-learn’. These vectorizers do not work on all the cases and may fail on certain cases where the text is too technical. Similarly, ‘scikit-learn’ provides ‘countvectorizer’ which is known to perform better than the one provided by WEKA. However, Latent Aspect Rating Analysis (LARA) works in an entirely different way. It works on the principal of multivariate Gaussian distribution which is known to work better than Support Vector Machines according to the paper[8]. The less accuracy which we got from Latent Aspect Rating Analysis suggests that the training model built by WEKA and ‘scikit-learn’ did not accurately train all the words, most of which were technical. We think that Latent Aspect Rating Analysis (LARA) technique would have given better results had the reviews been generalized and not too technical. However, since our domain was Stack Overflow, we had to work with more technical reviews rather than generalized reviews.

### 6.6 Results and Analysis

This was the sixth step in the project. We planned to perform two types of visualization. They are: Complete topic wise sentiment analysis and Web Service sentiment analysis. We used Microsoft Excel to perform this visualization. Such visualizations could be made for each of the 100 API's. Here, the results for ‘Amazon EC2 API’ and ‘Google Maps API’ have been presented.

- **Web Service Sentiment Analysis**

  In this category, the first diagram which we constructed was a pie-chart to visualize the total number of positive as well as negative sentiments. The pie-chart is given in figure 17. Figure 17 illustrates the distribution of sentiments for ‘Amazon EC2 API’. Similarly, figure 18 illustrates the distribution of sentiments for ‘Google Maps API’.

  As can be seen in figure 17, the negative sentiments are around 6 times the number of positive sentiments. This indicates that ‘Amazon EC2 API’ has more users.
or developers who are unsatisfied. The above result implies to the developers of the Amazon EC2 API that they need to take corrective measures in order to make their API more popular.

Similarly, as can be seen in figure 18, the positive sentiments are more than negative sentiments. This indicates that 'Google Maps API' has more users or developers who are satisfied. The above result implies to the developers of the Google Maps API that they are doing good work and they only need to focus on specific areas within the API.

The second diagram in this category visualizes the number of posts about the 'Amazon EC2 API' over the years starting from 2009 to 2016. The visualization can be seen in figure 19.

Similarly, as seen in figure 20, the number of posts about the 'Google Maps API' over the years starting from 2008 to 2016 are visualized.

As seen in figure 19, the number of posts showed a gradual increase from 2009 to 2012. In 2012, the highest number of posts were recorded for the 'Amazon EC2 API'. From 2012 to 2016, the number of posts seem to decline gradually.

- Complete Topic wise Sentiment Analysis

The diagram which we generated in this category was a chart visualizing all the topics for the 'Google Maps API'. The distribution of all the topics can be seen in figure 21. The topics have been categorized into positive and negative. Thus, we can visualize the topics which has the most positive or negative posts. Likewise, we can also visualize the topics which has the least positive or negative posts. We can see from figure 21 that the most discussed topic in 'Google Maps API' is 'maps'. This is expected since the primary service which 'Google Maps API' provides is maps and navigation. Other topics include 'url', 'video' etc.
Figure 21: Chart showing the distribution of posts for ‘Google Maps API’ across the years

Figure 22: Chart showing the distribution of posts for ‘Facebook API’ across the years

Figure 23: Chart showing the distribution of topics and their classification into positive or negative for ‘Google Maps API’

Figure 24: Chart showing the distribution of topics and their classification into positive or negative for ‘Facebook API’
7. FUTURE WORK

The future work in this research area includes generating the various aspects hidden in the posts by using a different approach than the current one. The approach that was used in this paper involves generating a latent aspect rating model on the basis of which aspects can be found. Another approach would be to use hierarchical topic modeling where it is not required to mention the number of aspects before the text is passed to the model. In this approach, after the text is passed, the number of aspects are generated randomly.

8. CONCLUSION

We would be concluding the paper by stating that the implied hypothesis holds true. The implied hypothesis was that since the data which we worked with came from Stack Overflow and since Stack Overflow is a website for solving grievances and problems, most of posts tend to be negative.

We proposed a new method for classifying the text about the API’s on Stack Overflow by using Support Vector Machines. We used Support Vector Machines as for text classification, they tend to give most accurate results.

We obtained an average accuracy of 84% which proves that the training model was built correctly and that the test data also got classified accurately for most of its instances.

9. ACKNOWLEDGMENTS

I would like to thank Dr. Xumin Liu for her continuous guidance and suggesting unique ways in implementing the various aspects of the project. She also encouraged my ideas and suggestions for the project. I would also like to thank Prof. Leonid Reznik for his constant suggestions on my progress throughout the project while also providing invaluable help in the preparation of the poster.

10. REFERENCES


