# Integration of Let's Chat and Sentiment Analysis on Messages in FoodStrap

Mayura Dhivya Nehruji (mayuradhivya.nehruji@sjsu.edu)
Parvathy Kannankumarath Madom Krishnan (parvathy.kannankumarathmadomkrishnan@sjsu.edu)

San Jose State University

## Abstract

An enormous amount of food is wasted or thrown out at restaurants across the country every single day. Over preparation of food and large portion sizes are two major contributing factors. FoodStrap, a web application, aims to bridge the gap between hunger and food waste. It makes it easy to donate leftover and surplus food. The application provides a platform where all the stakeholders for food donation can participate and make the process easy. Through the application, restaurants can donate leftover food at the end of each day and connect with homeless shelters and volunteers in its community. Homeless shelters can sign up in the application and claim food donated by restaurants. Philanthropic people can register themselves as volunteers for homeless shelters and can drive for them. They can pick up the food from restaurants and deliver them to the shelters.

The FoodStrap application has several actors and communication between them is required. In this paper, an open source chatting application, Let's Chat, is integrated with the FoodStrap application to enable users to communicate with each other. To assess customer satisfaction, that is a crucial factor for any application, and to obtain an overview of the emotions exhibited by the users, sentiment analysis on the chat messages is done.

## 1. Introduction

America wastes roughly 40 percent of its food. Of the estimated 125 to 160 billion pounds of food that goes to waste every year, much of it is perfectly edible and nutritious [1]. At a time when 12 percent of American households are food insecure, reducing food waste by just 15 percent could provide enough sustenance to feed more than 25 million people, annually [1]. Currently, only 10 percent of edible wasted food is recovered each year, in the US. FoodStrap provides a medium that helps to reduce food waste and to divert the leftover from restaurants to homeless shelters.

Communication between various stakeholders is desirable for any transaction or activity to be successful and engaging. Currently, a restaurant manager with leftover food, submits a form for donation. And shelters must proactively check the Claims tab to see the list of all available donations. Also, there is no way for the shelters to express their gratitude to restaurants and volunteers other than to individually thank them. Integration with a chatting application, provides a common channel for all users, which can make the donation process smooth and engaging. Restaurant managers need a forum where they can let all homeless shelters know of left-over food available for donation. A restaurant manager can post in the channel whenever leftover food is available for donation and shelters will be immediately notified. They can then claim and pick up the donation. Shelters can use the same forum to compliment and thank restaurants for donating food and volunteers for picking up donations.

Identification of user sentiment and emotion towards the application in general and fellow users is required to assess user experience and

to discover whether users are happy or not. Sentiment analysis is the interpretation and classification of emotions as either positive, negative, or neutral within a given text using various analysis techniques [2]. The chat messages posted by users provides a dataset that can be used for analyzing the sentiment of users. For instance, a shelter manager expresses his appreciation for a restaurant that donated a hundred portions of food by posting a message in the chat. Analyzing this message would show that the shelter manager was happy and thankful. With hundreds of messages posted by different users, manual examination is difficult, and a sentiment analysis tool is required. Assessing the sentiment of users provides an insight into the user experience and user satisfaction.

In this paper, Let's Chat, a chatting application, is integrated with FoodStrap. The users can use the channel for their community to chat and post donation information. Sentiment analysis of chat messages is done to get the sentiment score and emotion for each message and the overall average for the application. Analysis for different users, that is done by examining all the messages by a user, is not in scope.

The next section outlines current approaches for these topics. Further sections delineate the method for chat application integration, sentiment analysis, its result evaluation, and weaknesses. Lastly the conclusion summarizes the solution and mentions future research work and testing.

## 2. Related Works

Sentiment analysis is garnering more attention, because of a myriad of use cases. The article [2], describes the main types of algorithms used for sentiment analysis. The algorithms are categorized as Rule-based, Automatic, and Hybrid. It also outlines the use cases and applications for sentiment analysis.

An application of sentiment analysis has been described by Ruan [8], which also explains a method for sentiment analysis of text data in Twitter by collecting the tweets based on hashtags and evaluating the sentiment of each tweet, in Node.js. Also, Reeta Rani et al [9] elucidates an approach for Twitter sentiment analysis by the method of classification and proposes an ontology-based technique to generate a summary of input data, implemented in Python. Garg et al [10] present a concept to collect online chat logs for customers to auto predict feedback using textual sentiment analysis. This is achieved by applying the Apriori algorithm for association rules, eliminating the need for manual editing of feedback forms. Also, Don [4] explains in detail all the steps required to develop a sentiment analysis application.

## 3. Method

In this paper, to enable communication between users, and to assess user emotions and user satisfaction, there are two steps. First, integrate FoodStrap with a chatting application, Let's chat. Second, perform a sentiment analysis on chat messages.

### 3.1 Integration of Let's Chat

Let's Chat is an open source and persistent messaging application. It has several features like multiuser channels, private and public channels, private messaging between two users, user mentions using @username, chat history, and file uploads. This chat application is chosen because the application is simple and the technologies on which it is based, matches with that of FoodStrap. Both are built using NodeJS and MongoDB. Hence, other steps or services are not required. This makes the integration and deployment easy.

The various entities in Let's chat are rooms, messages, sessions, and users. Room is used to denote a channel or community group. It stores

all the channel information and has various essential operations like create, find, delete, update, and archive. Messages entity is used to denote the messages sent in channels. It stores the message text along with its channel, sender information, and time. It supports create and find operations. Users entity is used to denote the users in the application. It stores all user information and maintains a list of rooms the user belongs to. It supports create, update, and find operations. Sessions entity denote the user sessions and stores information relevant to that like session cookie and expiration time.

To integrate the application with FoodStrap, initially set up both applications and start their services. To setup Let's Chat, first ensure all the pre-requisite steps of installing NodeJS, MongoDB, and Python are done. MongoDB of version greater than 2.6 is required and service is running. Python version 2.7 or greater is required. Download the code from the GitHub repository [3]. Then install dependencies with npm using the command npm install. Start the service using the command npm start. The service is running in port 5000. Ensure the application is correctly set up by navigating to the URL http://hostname:5000 and checking all the features.

Modification to the FoodStrap application is required to enable the integration. To add the chat feature to the restaurant manager user, certain steps are done. First, add a new entry for chat in the side navigation for the user and link it to the chat page. Side navigation contains links for donation, donations history, process, profile, and log out. This enables the user to navigate to the chats page and chat. Second, create a new chat page for the user. The theme of the page matches with that of other pages, and side and header navigation are also matching. Thirdly, for the content of this page, chats should be seen. To do that, add an iframe of the Let's Chat application *<iframe width=100% height="700" src="http://localhost:5000"></iframe>*. Provide the src attribute of the iframe element as the URL of Let's Chat that was set up earlier. Through this iframe, the chat application will be embedded in the chats page and the user can use the chat application. Perform the same set of steps for other users shelter manager and volunteer also by creating new chat pages for them and linking it to a new entry in the side navigation.

1. Create user in FoodStrap
2. Create same user in Let's chat
    I. Build the user object as per Let's Chat schema
        I. Let user ← {}
        II. user.displayname ← firstname+ ' ' +lastname
        III. user. Lastname ← lastname
        IV. user.firstname ← firstname
        V. user.password ← encrypted(password)
        VI. user.email← email
        VII. user.username ← username
        VIII. user.joined ← now()
    II. Insert user into letschat.users collection in MongoDB
        I. dbo.collection("users").insertOne
            (user, function (err, result) { …. }

*Figure 1 Pseudo code for the modified sign up user flow in FoodStrap*

Furthermore, to make the integration smooth, additional changes are required. The Let's chat application is not directly used by the users of FoodStrap and but only through the iframe in chat pages in FoodStrap. Whenever a new user signs up in FoodStrap, code is added to add a corresponding user in Let's Chat also with the same user information. Username and password should match. Figure 1 shows the pseudo code for this. This is done to create the same user in both applications as both use separate collections and to allow the user to use only one application, FoodStrap, and keep Let's Chat away from the user. If this was not implemented, the users will have to go to Let's chat application and create a new account for themselves. To ensure a session of Let's Chat in the chat page, whenever a user signs in FoodStrap, code is added to create a new session for the user in Let's Chat also. Figure 2 shows the pseudo code for this. Now if the user goes to the chat page,

the session is active and chat rooms will be seen. If this was not implemented, the user will see a login page of Let's Chat when the user goes to the chat page. And the user will have to log in first in Let's chat to use the chat rooms.

1. User signs in FoodStrap
2. Create same user's session in Let's chat
    I. Build the session object as per Let's Chat schema
        I. Let sessionObject ← {}
        II. Let session ← {}
        III. Set session.cookie values
        IV. session.passport.user ← user id
        V. sessionObject.session ← session
        VI. sessionObject.expires ← now() + 30 minutes
    II. Insert user into letschat.sessions collection in MongoDB
        I. dbo.collection(" sessions").insertOne
            (sessionObject, function (err, result) { …. }

*Figure 2 Pseudo code for the modified sign in user flow in FoodStrap*

channel. Here restaurants can post messages when they have food available for donation and have submitted the form. This message, seen by shelters can then claim the donation in the app and thank the restaurants for their service. They can also thank volunteers for driving for them. Shelters can ask any queries regarding the donations or seek donations. These channels provide an environment for users to communicate and have conversations. Private messages can also be sent by users. For instance, a shelter can thank the restaurant by sending a message private message instead of posting in the group channel. Figure 3 shows the chat page of a restaurant manager in FoodStrap application integrated with Let's chat.

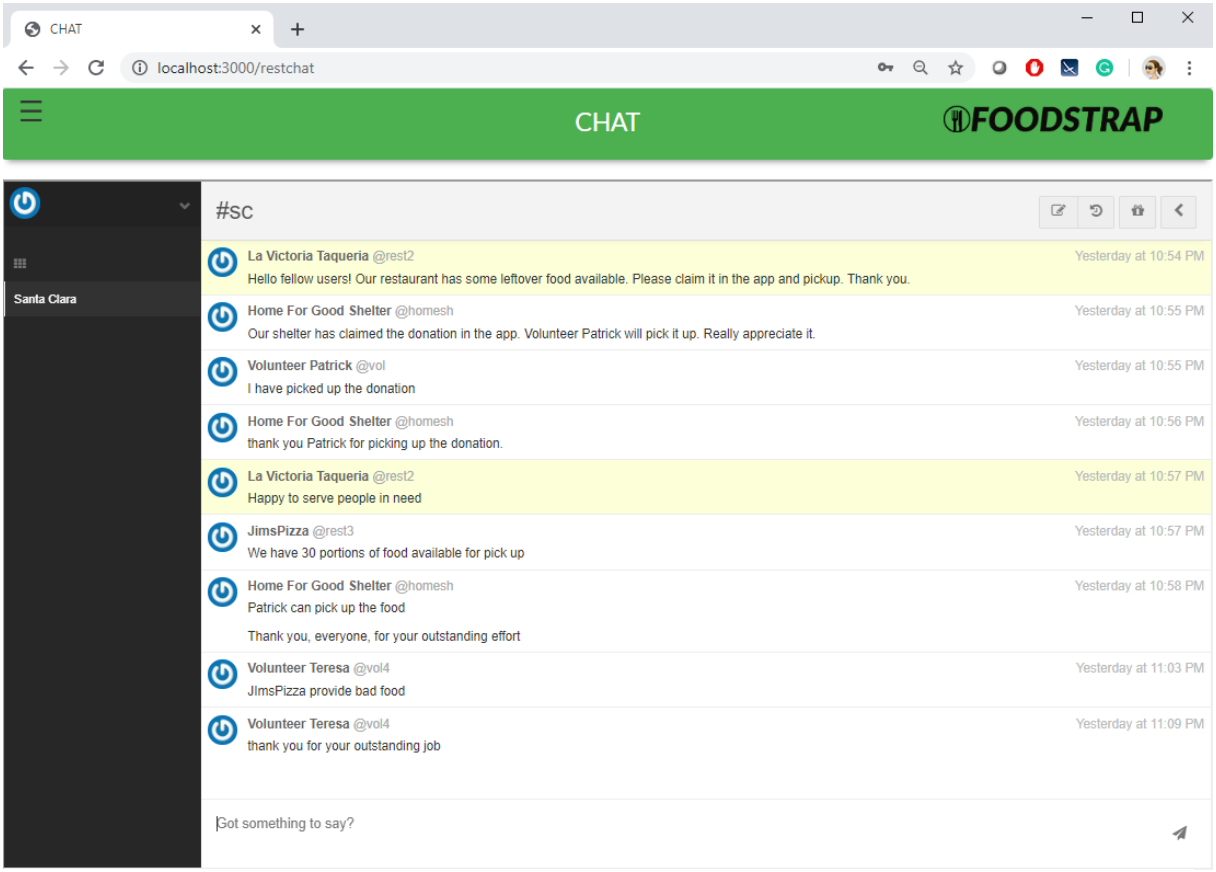



*Figure 3 Let's Chat integrated with FoodStrap*

One public channel per locality is created. For instance, one channel each for Santa Clara and San Mateo counties. The restaurants, shelters, and volunteers in that locality can join this

## 3.2 Sentiment Analysis

Sentiment analysis is the process of analyzing a text and categorizing it as either positive, negative, or neutral. Sentiment analysis on twitter data and product reviews are very popular, and it provides a gist of the users' emotion. The chat messages in the chatting module of FoodStrap provides a dataset of messages sent by different users. This is used to get an overview of the users' emotions and user experience feedback. This overall user emotion rating is helpful to get an idea of how helpful the application and fellow users are. For instance, shelter users use this platform to thank donor restaurants and volunteers for their service. This would yield a high score meaning users are happy. If for instance, users are not happy with the quality of donations made, they would raise their concerns in the app. This will reduce the score meaning the emotion is neutral or sad.

1. Convert text to standard lexicon
2. Convert text to lower case
3. Remove non alphabetical and special characters
4. Tokenize the text – split into a set of words
5. For each word, do a spell check
6. Remove stop words like or, what, a, the
7. Perform Stemming - get root words to get a list of tokens
8. AFINN is a list of words with scores assigned
9. For each word, cross check with AFINN list and get its score
10. The final score for the text is sum of all scores

*Figure 4 High level steps in Sentiment Analysis*

The process of sentimental analysis has several steps. Figure 4 shows the high-level steps. First, the dataset containing the text is required. The chat messages are the dataset here. For each message, the message text is analyzed, and a score is determined. After obtaining the score for all chat messages, the average of that score is taken as the overall score of the chat dataset. Figure shows the steps for getting a score for a message text [4,5]. First convert the text to standard lexicon. This is done to remove any short forms and use a standard lexicon throughout the analysis of all messages. For instance, "I'm very thankful for what you've given us" is converted into "I am very thankful for what you have given us". The second step is to convert the text into lower case. This is done to get the text in a uniform format and do not differentiate between the same text of different cases. For example, good and GOOD should be considered the same. The third step is to remove any non-alphabetical and special characters. This is done as they are not analyzed. Only alphabetical characters and words are analyzed.

Tokenization of the text message is the next step. It is the process of splitting a text into its individual units like sentences, phrases, words, etc. For example, a paragraph can be split into sentences, sentences into words, etc. Here, the text is split into words. For example, the text message, "We really appreciate your effort thanks." is converted into a list of token or words [We, really, appreciate, your, effort, thanks]. These tokens are then used in the next steps. After tokenization, the tokens or words are checked for any spelling errors and corrected using spelling corrector. This is required as the user is manually typing in all the chat messages and it is error prone. The correction ensures that all the words are considered when analyzing them. For instance, if a word was misspelled, happ instead of happy, it might not make it to the list of positive words as the word happ doesn't exist. Removal of stop words is the next step. Stop words are words that are so frequently used that they can safely be removed from a text without altering its meaning [6]. For example, a, the, an, or, with, etc are stop words. These words are removed from the analysis as their removal doesn't alter the meaning or emotion of the text.

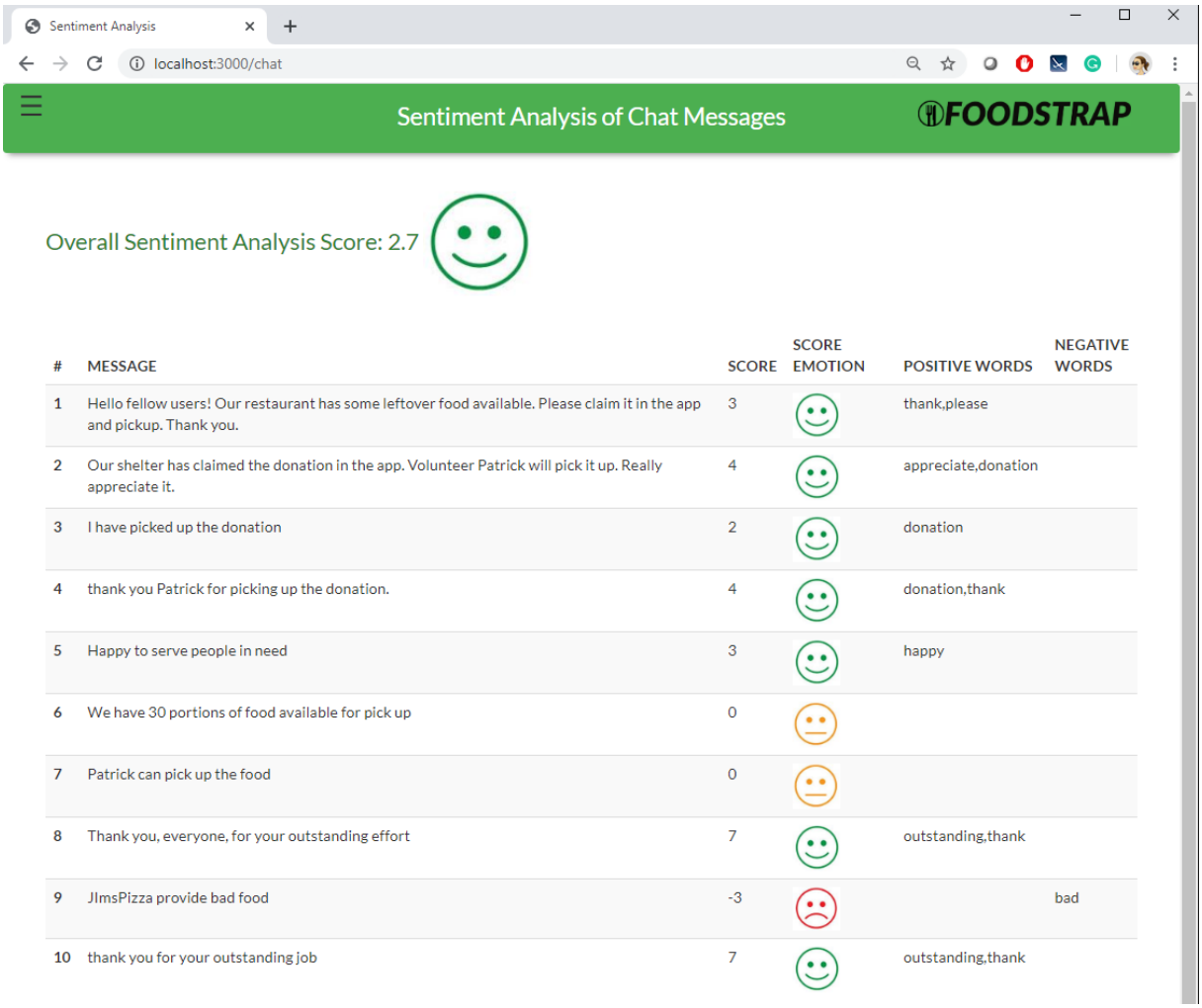
Stemming of tokens is the next step in the process of sentiment analysis. Stemming is the process of obtaining the root word for a given word. For example, the root word for the words thankful, thank and thanks is thank. This is done as words can take many forms and convey the

same meaning and obtaining the root word makes the analysis process easy.

```
TEXT: "thank you for your outstanding job"
{
  "score": 7,
  "comparative": 1.1666666666666667,
  "calculation": [
    { "outstanding": 5  },
    { "thank": 2  }
  ],
  "tokens": [
    "thank", "you", "for", "your",
    "outstanding", "job"
  ],
  "words": [  "outstanding", "thank"  ],
  "positive": [ "outstanding", "thank"  ],
  "negative": []
}
```

*Figure 5 Sample sentiment analysis result for a text*

AFINN is a lexicon used for sentiment analysis. It has a list of words rated for valence and a polarity score for each word [7]. For example, thanks has a score of 2, good has a score of 3 and bad has a score of -3. If the score is greater than zero, then the word denotes a positive emotion and a negative score denotes a negative emotion. If the score is zero, the emotion is neutral. The next step in the analysis is to cross check every token in the list to the AFINN list and getting the score. The sum of all the scores is taken. This is the score of the text. This whole process is repeated for all chat messages. The average of the score is the overall average score of the application. Figure 5 shows the sentiment analysis of a sample text.




| # | MESSAGE | SCORE | SCORE EMOTION | POSITIVE WORDS | NEGATIVE WORDS |
|---|---|---|---|---|---|
| 1 | Hello fellow users! Our restaurant has some leftover food available. Please claim it in the app and pickup. Thank you. | 3 | | thank,please | |
| 2 | Our shelter has claimed the donation in the app. Volunteer Patrick will pick it up. Really appreciate it. | 4 | | appreciate,donation | |
| 3 | I have picked up the donation | 2 | | donation | |
| 4 | thank you Patrick for picking up the donation. | 4 | | donation,thank | |
| 5 | Happy to serve people in need | 3 | | happy | |
| 6 | We have 30 portions of food available for pick up | 0 | | | |
| 7 | Patrick can pick up the food | 0 | | | |
| 8 | Thank you, everyone, for your outstanding effort | 7 | | outstanding,thank | |
| 9 | JImsPizza provide bad food | -3 | | | bad |
| 10 | thank you for your outstanding job | 7 | | outstanding,thank | |

*Figure 6 Sentiment Analysis results of chat messages in FoodStrap*

In the FoodStrap application, a new page /chat is created to display the analysis results. Figure 6 shows the screenshot of FoodStrap application sentimental analysis report. It shows the average overall score for the application and the score for each chat message text. AFINN based sentiment analysis library, sentiment, is used for the analysis. If the score is greater than zero, the sentiment is positive, and an image of happy face is displayed. If the score is less than zero, the sentiment is negative, and an image of sad face is displayed. If the score is zero, the sentiment is neutral, and a neutral face is displayed.

## 4. Results and Evaluation

The sentiment analysis done evaluates each chat message and provides a sentiment score to it. And the overall score is calculated as the average of the individual scores. The analysis is based on the AFINN-165 wordlist. AFINN is a widely used lexicon for sentiment analysis and has more than 3000 words. The scores are calculated based on this list. A total of sixty message texts were analyzed and scores obtained. On the manual evaluation of the texts and their corresponding sentiment score, the results are accurate. The tokenization was done correctly for all texts. Positive and negative words identification, based on the AFINN list, was done correctly. Scoring based on these identifications was also done accurately.

Even though the AFINN-165 list is an improvement over its previous version AFINN-111 and has more than 3000 words, there are thousands of words still not considered. They could be positive, negative, or neutral. And its points will be ignored if those words are used in the text. Certain negative words can convey a positive sentiment when used in phrases. However, the AFINN list has only a limited number of phrases. The analysis has other limitations also when dealing with irony, sarcasm, slangs, and tone. For instance, the model will not recognize that the text “yeah sure thanks for nothing” as being sarcastic and that the user is not thankful.

## 5. Conclusion and Future Work

Restaurants throw away pounds of perfectly edible and nutritious food every day. At the same time, millions of Americans are food insecure. Recovering a small percent of the wasted food can help serve millions of needy people. FoodStrap aims to bridge this gap between hunger and food waste and to make food donation easy.

Integration with Let’s Chat provides a platform for users to communicate with each other. It enables users to have conversations with fellow donors, shelters, and volunteers in the community. They can express gratitude for one another for their service. This acts as a motivation for everyone. The sentiment analysis on the chat messages provide an insight into user sentiment and emotion. The analysis provides a near accurate sentiment score and is a good parameter to gauge user experience and user satisfaction. In the future, sentiment analysis for a user can be done by analyzing all the messages by that user. Testing for texts with irony, sarcasm, and double negations can be done.

## References

[1] Foodprint.org, “The Problem of Food Waste”, *foodprint.org* [Online]. Available: https://foodprint.org/issues/the-problem-of-food-waste/ [Accessed May 10, 2020].

[2] Monkeylearn.com, “Sentiment Analysis”, *monkeylearn.com* [Online]. Available: https://monkeylearn.com/sentiment-analysis/ [Accessed May 10, 2020].

[3] Github.com, lets-chat, *github.com* [Online]. Available: https://github.com/sdelements/lets-chat [Accessed May 10, 2020].

[4] E.Don, “Building a sentiment analysis app with Node.js”, *blog.logrocket.com* [Online]. Available: https://blog.logrocket.com/sentiment-analysis-node-js/ [Accessed May 10, 2020].

[5] NPM, “sentiment”, *npmjs.com* [online]. Available: https://www.npmjs.com/package/sentiment [Accessed May 10, 2020].

[6] NPM, “stopword”, *npmjs.com* [online]. Available: https://www.npmjs.com/package/stopword [Accessed May 10, 2020].

[7] D.Sarkar, "Emotion and Sentiment Analysis: A Practitioner’s Guide to NLP", *kdnuggets.com* [Online]. Available: https://www.kdnuggets.com/2018/08/emotion-sentiment-analysis-practitioners-guide-nlp-5.html [Accessed May 10, 2020].

[8] B.Ruan, “Twitter Sentiment Analysis with Node.js”, *towardsdatascience.com* [Online]. Available: https://towardsdatascience.com/twitter-sentiment-analysis-with-node-js-ae1ed8dd8fa7 [Accessed May 10, 2020].

[9] R.Rani and S.Tandon, “Chat Summarization and Sentiment Analysis Techniques in Data Mining*”, IEEE 4th International Conference on Computing Sciences (ICCS)*, 2018. Available: https://ieeexplore-ieee-org.libaccess.sjlibrary.org/document/8611043 [Accessed May 10, 2020].

[10] S.Garg and S.N.Singh, "Auto Predictive Customer Feedback from Textual Analysis of Online Chat Logs", IEEE 4th International Conference on Computational Intelligence & Communication Technology (CICT), 2018. Available: https://ieeexplore-ieee-org.libaccess.sjlibrary.org/document/8480350 [Accessed May 10, 2020]