Ocean: Topic-Modelling Based Social Media Application to Reduce Stress

Abhishek Pandey
North Carolina State University
aapandey@ncsu.edu

Chengyuan Liu
North Carolina State University
cliu32@ncsu.edu

Piyush Mishra
North Carolina State University
pmishra@ncsu.edu

Xiaorui Tang
North Carolina State University
xtang9@ncsu.edu

ABSTRACT

The stress and anxiety levels among youth and students, in particular, are on an exponential rise. There are a lot of factors all around us like exams, job search, etc. contributing to this problem. But the means to reduce or tackle these troubles are quaint. Social media has become an indispensable part of our lives. But, is it really helping to reduce worries or just piling them up for us in the hindsight?

This paper aims at using these two ubiquitous part of our lives to produce an amalgam that can help us reduce stress via social media. This report tries to find ways, using techniques like sentiment analysis and text-mining, to make our lives less burdensome. We have perused various research papers with relevant citings to draw some conclusions that might help us in developing such an application.

Keywords – Text mining, sentiment analysis, stress reduction, web application, social media

1. INTRODUCTION

"People become attached to their burdens sometimes more than the burden is attached to them."– George Bernard Shaw

The existence of stress and worry can be traced back to the genesis of mankind. And there is no panacea to completely eradicate stress. But, we can surely device some ways to handle stress and keep it in check. It is no surprise that the anxiety levels are the highest among the productive population and, particularly, among college students.[4]According to a recent survey conducted by the American College Health Association under the National College Health Assessment II in 2015 consisting of 19,861 respondents as the Reference group, 57.7% of the students reported feeling "overwhelming anxiety" at least once in the last 12 months.

There are hundreds of other similar surveys present online corroborating this trend. The cost of college continues to skyrocket and the student loans are both more prevalent and more expensive than ever before. Surely, the bills won't ever stop coming, exams and assignments would never end, there will never be more hours in a day and our career and family responsibilities will always be demanding. Most of the time we alone suffer our ordeals and don't get to share it or vent it out in a desirable way due to a social barrier of judgement and negativity. Moreover, the monotonicity of our lives exacerbates the problem.

1.1 Problem Statement

We decided to give our users a break from their routine and let them escape into a completely anonymous world to relax, reflect and rehabilitate. We can tackle stress if we develop a positive attitude towards it. The orthodox methods of tackling stress like listening to music, outdoor activities, etc. may soothe us temporarily but this soothing effect is not long-lasting. We regress back into our bubble of worry as soon as we resume our daily tasks.

1.2 Motivation

We have conducted surveys and cited studies to understand this in a better way. The results of these surveys and studies are included in the subsequent section of this report. We found out that someone who has gone through a similar experience and is willing to help others can play a cameo in such situations. They can empathize and expedite this process of relieving stress in a better and quicker way. This is the basic idea governing the proposal of our application. We have planned to develop an application that lets its users share their feelings and emotions.

1.3 Solution

Firstly, to address this problem we have developed a social-media platform, just like Twitter, primarily focussed on venting out and seeking help from others while maintaining complete anonymity. The anonymity feature is something that we feel is novel in our case and helps people express themselves in a more open way. But, this novel idea has its

own drawbacks which we plan to tackle in the future. As complete anonymity can also be used by few people to offend or discriminate our users who are seeking help. We need to figure out a system to keep a check on such anti-social elements using our application.
Secondly, we have implemented machine learning algorithms to find the keywords from their posts using text mining and save them as tags for future reference. This collected data can be used to find a proper match from the user database. The match would be a person with similar troubles or someone who had faced a similar problem and has overcome it. The user would get few match suggestions to choose from after viewing their profile and ratings. Once the user chooses a match, they can connect to share their thoughts and ideas via a chat messenger incorporated in our application. We plan to record the satisfaction level of a user after talking to a particular match and use it to update the helper's rating. The user has the option to delete any of their posts whenever they want.

2. LITERATURE REVIEW

[1]"The concept of stress may be found as an independent variable, a dependent variable, or a process (Cooper C.L. et al., 2001). According to the HSE (2001) stress is defined as "the adverse reaction people have to excessive pressure or other types of placed on them". The American National Institute for Occupational Safety and Health (NIOSH) (1996) defines work stress as "the harmful physical and emotional responses that occur when the requirements of the job do not match the capabilities, resources or needs of the worker". Stress may be defined as the opposite of relaxation.

A situation is considered stressful if it puts pressure on the individual to perform more accurately or faster or differently from his normal mode (Pfaff M.S., 2008). Cooper C.L. et al. (2001) suggest that environmental factors that may function as sources of stress are called stressors and the individual's reaction to the stressors is called strain. Stress is characterized as the relationship between the perceived demands of the current situation and the individual's ability to respond (Pfaff M.S., 2008). Stress may have negative impact or positive impact, which make aware people to stay focus and work in order to obtain performance. This clearly shows that stress has become an inseparable part of our lives and is growing significantly. Battling stress has become the need of the hour and the gravity of this issue needs to be addressed imminently.

Language is the most common and reliable way for people to translate their internal thoughts and emotions into a form that others can understand. Language is comprised of sentences which are made of words which convey some meaning and combined together they can express a variety of emotions. Words and language, then, are the very stuff of psychology and communication. They are the medium by which cognitive, personality, clinical, and social psychologists attempt to understand human beings[2]. The words we use in daily life reflect who we are and the social relationships we are in[2]. This is neither a new nor surprising insight. Words are used to share our feelings and the psychology associated with them.

The problem of sharing an intangible asset such as the knowledge of individuals - can be viewed from many perspectives: psychological, economic, organisational, sociological and technological[3]. The research[3] concerned the reasons for helping other people selflessly.

According to the results, it can be assumed that a worker will share his or her knowledge for motives other than sheer ego, provided that he or she feels empathy towards the knowledge recipient. If a person connects with another person emotionally then he/she is willing to help others. Empathy will be elicited much more by a person who seems similar to us in some way, such as holding similar values or having similar interests and experience. There is an interaction between empathy and liking[3]. When we like someone, it is easier for us to feel empathy towards him or her, and by feeling empathy towards someone, it is easier for him or her to like us. When we feel empathy towards another person, we will share knowledge with him or her to help him or her to, for instance, meet a deadline, without regard for our own interest.

This can be regarded as a genuinely altruistic activity. When we do not feel empathy, we will make our knowledge available with our own interest in mind because the prospective profits exceed the losses[3]. It can be inferred that people who share the same experience be it happy or sad, tend to connect emotionally with each other. If a person who has undergone similar set of circumstances meets another person who is undergoing the same experience then he or she is likely to consult him and help him. This is the basis of our research which is similar to our survey results.

"[5]Online opinions have been recently analyzed using sentiment analysis (SA). This is basically a natural language processing (NLP) application that uses computational linguistics and text mining to identify text sentiment, typically as positive, neutral or negative.

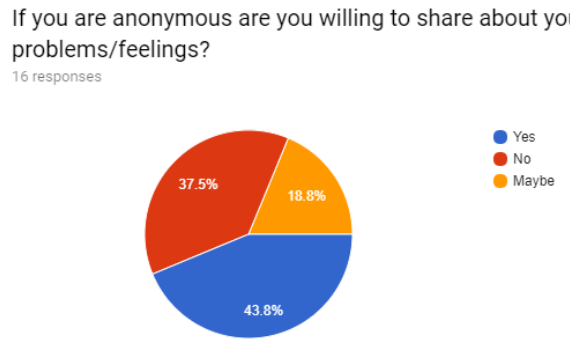
This technique is also known in the text mining literature as emotional polarity analysis (EPA), opinion mining, review mining, or appraisal extraction (Zagal, Tomuro, & Shepitsen, 2012). Thus, SA can be regarded as an automated knowledge discovery technique that aims at finding hidden patterns in a large number of reviews, blogs or tweets. To calculate a sentiment score, the sentiment obtained from the text is compared to a lexicon or a dictionary to determine the strength of the sentiment. For example, the lexical resource SentiWord, which includes around 200,000 entries, uses a semi-supervised method to assign each word with positive, negative and objective scores.

We are going to use this method to analyse the post made by the users and see how negative or positive these emotions are. On the basis of this analysis we will use some algorithms to match the user with potential people who can help them. We are also going to use text mining which is going to help us in extracting keywords which will expedite the process of searching and matching, and would also allow the user to refine their suggestions.

Our initial survey included few crucial questions which helped us render validity to our idea. Below is the description about couple of these questions: If you are anonymous are you willing to share about your problems/feelings?

The question is asking whether people are willing to share their feelings of stress or anxiety to others anonymously. The results showed that, more people were willing to talk to strangers as compared to their near ones.

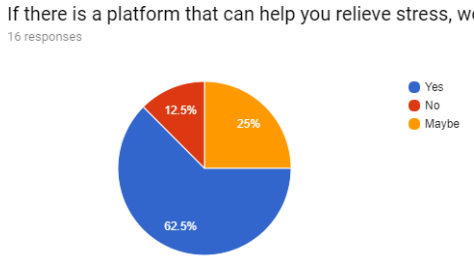
Figure 1: Primary Survey Question



If there is a platform that can help you relieve stress, would you use it?

This question asked whether people would use a platform to relieve stress. The intention here was to predict the acceptance ratio of our application once it is made public. The results showed that more than 60 percent of people showed significant willingness to use, plus the number of potential users, with more than 85 percent of potential users.

Figure 2: Primary Survey Question



3. EXPERIMENT

First of all, we decided to develop our application as a social media platform. Nowadays, there are a lot of social software in the market, most of these softwares provide both web and mobile version. So, the first problem we should consider is that whether we move to a web version or the mobile version of the application. According to the knowledge and skill-set of our team, we decided that the implementation of a web application is easier than the mobile app, so we decided to develop a web application.

Second question we encountered is kind of technical framework that should be used to implement this application. With limited time, it is certainly wise to use the existing maturity framework. At present, the commonly used web framework are: Rails based on ruby, Django based on Python, and Express based on the Node. Js. These three frameworks are widely used in the market, which can rapidly build a web application from scratch. These frameworks are really helpful to work in a restricted time frame, as they can quickly validate our design model's feasibility.

And at this point, another problem which we needed to consider is that our application contains a machine learning module, and as Python is widely used in the field of machine learning, Python was an obvious choice to implement functions related to machine learning. Therefore, microframeworks like Django and Flask were taken into consideration.

At the beginning of the project, we decided to use Django to implement the entire application, but as the team members were not familiar with Django, we encountered a lot of resistance in the initial stage. Apart from machine learning a major part of our code would be for the web platform, so using a single technology was not feasible. After searching for solutions online, we think using the python scripts which include machine learning algorithm to implement a mini web service, and the Rails platform use RESTful API to communicate between passing parameters is a good solution.

4. SOFTWARE REQUIREMENT SPECIFICATION

4.1 Functional and Non-functional requirements

1. **Functional Requirements**

   - **RESTful Interface:** The application should communicate smoothly with the web-service.
   - **Micropost:** User should be able to write and delete their own post.
   - **Match:** Matching algorithm should be accurate and produce desired result every time.

1. **Non-Functional Requirements**

   - **Performance:** The application should be able to perform in real-time without lag.
   - **Reliability:** The match output and tags generated should be consistent and reproducible.
   - **Scalability:** The application should be able to handle a large number of users and simultaneous traffic.
   - **Security:** The application should allow users to securely login and keep their data safe in the database.

4.2 Use Cases

The various use cases for the application :

Use Case Name	User Sign Up
Actors	User (of the application)
Description	The user will visit the application to create to sign up
Trigger	Use case is triggered when the user clicks the new user sign up button
Preconditions	1. User has a PC 2. User has active internet connection
Post conditions	User can successfully sign up or error
Normal Flow	1. User starts the application 2. User selects the new user button 3. Server takes up users' request and renders the new user sign up page 4. User inputs his/her information and valid credentials 5. User submits his sign-up form
Exceptions	Software doesn't start appropriately - Restart software and try it again Server remains unresponsive - Restart server - Try it after some time Interface is unresponsive - Refresh page - Try it after some time
Includes	-
Special Requirements	The software must have a robust interface that can handle user input effectively and is easy for the user to understand it's functionality
Assumptions	- User understands English - Display monitor is active
Notes and Issues	Interfaces is interactive waiting for user to either sign up or for previous users to login.

Use Case Name	Display matches
Actors	Application Server
Description	After finding a match with user generated micro post from other users' micro posts, application server will return the micro posts of the other users
Trigger	Server searches for the matching tags from various micro posts.
Preconditions	1. Match for tags found 2. Software is still active
Post conditions	Interface shows the list of matching posts.
Normal Flow	1. Server finds the result 2. Displays results
Exceptions	Software doesn't start appropriately - Restart software and try it again Server remains unresponsive - Restart server - Try it after some time Interface is unresponsive - Refresh page - Try it after some time
Includes	-
Special Requirements	The software must have a robust interface that can handle user input effectively and is easy for the user to understand it's functionality
Assumptions	- User understands English - Display monitor is active
Notes and Issues	Interfaces is basic.

Use Case Name	User creates a Micro Post
Actors	User (of the application)
Description	The user will log in to the application and create a micro-post
Trigger	Use case is triggered when user logs in to the application and creates a micro-post
Preconditions	1. User has a PC 2. User has active internet connection
Post conditions	User successfully creates his post or error
Normal Flow	1. User starts the application 2. User selects the log in button 3. Server takes up users' request and renders the returning user page 4. User inputs his/her valid credentials 5. User submits the form 6. User is taken to his/her homepage 7. User enters his micro post in the micro post text field area 8. User can view his micro-post along with the tags returned by matching algorithm
Exceptions	Software doesn't start appropriately - Restart software and try it again Server remains unresponsive - Restart server - Try it after some time Interface is unresponsive - Refresh page - Try it after some time
Includes	-
Special Requirements	The software must have a robust interface that can handle user input effectively and is easy for the user to understand it's functionality
Assumptions	- User understands English - Display monitor is active
Notes and Issues	Interfaces is interactive waiting for user to either sign up or for previous users to login.

5. ARCHITECTURE

After further investigation, we decided the framework of the platform should:

Use Rails to develop the major party of social platform, including the basic functions of social media such as authentication system, micropost, following users, etc., and our application's special features: according to the micropost content generation tags (topics), according to the tags match users with the same topic of interest.

Use the python language to implement machine learning algorithms, you can generate tags (topics) based on a microblog post.Use Flask to turn a python script into a micro-web service.

Design a simple communication interface that using RESTful requests to communicate between the main body of the social platform and the Web service.

6. LIFE CYCLE MODEL

6.1 Agile methodology

We use agile methodology for our software development as

it is currently the most widely used software development model, especially when time is limited. Given that we only have one and a half months, agile methodology is the best choice. It allows us to rapidly implement a beta version to verify that the product is feasible and can be incrementally developed in different short cycles.

This time, we have implemented three short development cycles. In the beginning, we estimated that the core of the project and the most difficult part should be the machine learning algorithms, so we set up the first cycle's goal to achieve a basic machine learning algorithms, which can analyze a micropost analogous paragraph, to generate the tags we need (subject).

After that, we turned to social platforms, set the goal of the second cycle to achieve the basic functionality of our social media, .ie., to generate tags for a particular post. Then according to these tags, we went on to matching users and other related functions. At the same time, we still assign one teammate to continue on improving machine learning algorithm.

After the second cycle is completed, the last step was integration. We had to establish the communication between the two modules. The goal of the third cycle is to build a web service for machine learning algorithms and to design and implement the interface and message protocols for communication between two modules. At the same time, we allocated one team member to improve the function of social platform and fix bugs, rest of us were involved in testing of the overall application.

We completed these three cycles successfully. At the end of the third cycle, we realized the basic function of social platform, a preliminary and feasible machine learning algorithm, and realized communication between two modules and simple integration test.

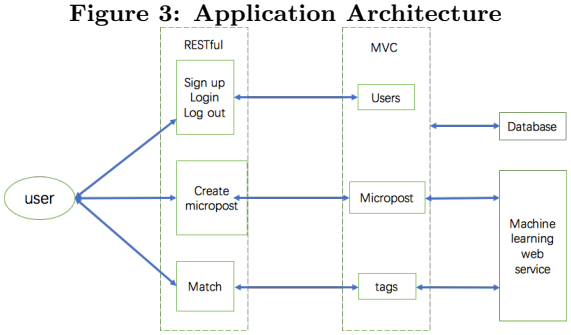
6.2 Pair Programming and code review

During the development period, we basically assigned two team members to development of social software platform, and other two were mainly responsible for the development of machine learning modules. Therefore, we naturally applied pair programming and code review, for example, two members of the team simultaneously engaged in the code of the social platform, and the two members reviewed the code of the machine learning algorithm written by each other.

7. IMPLEMENTATION

The frame diagram of the whole application is shown in the figure below:

Figure 3: Application Architecture



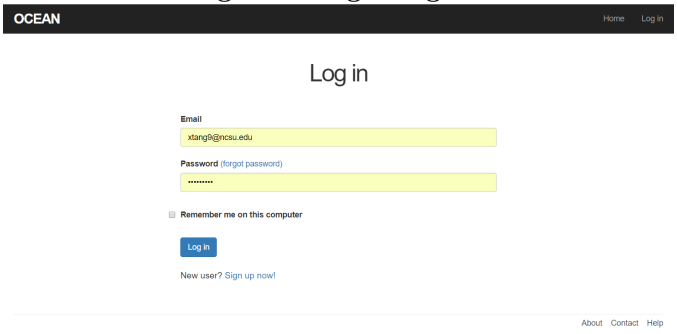
7.1 Social Media Platform

The web app uses the Ruby on Rails framework, and some code references from a book. It conforms to the MVC architecture and use RESTful API. It mainly accomplishes the following functions:

- Authentication system (user registration, login, password change, etc.)
- Micropost
- Use machine learning API to analyze microblogs and generate tags.
- Following/Unfollowing other users and matching with them on the basis of tags generated

7.1.1 Authentication system

Figure 4: Login Page



This application implements a professional and secure authentication system, including user registration, login, email activation account, change password, remember login and other functionalities in the browser. The email function uses the services provided by a third party plug-in – SendGrid. Remember the user in the browser can make the logged user be identified even after rebooting the browser.

7.1.2 Mircopost and Following

Figure 5: User Home Page

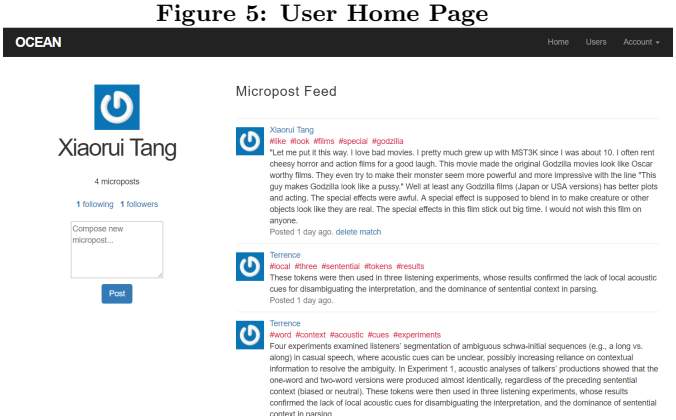
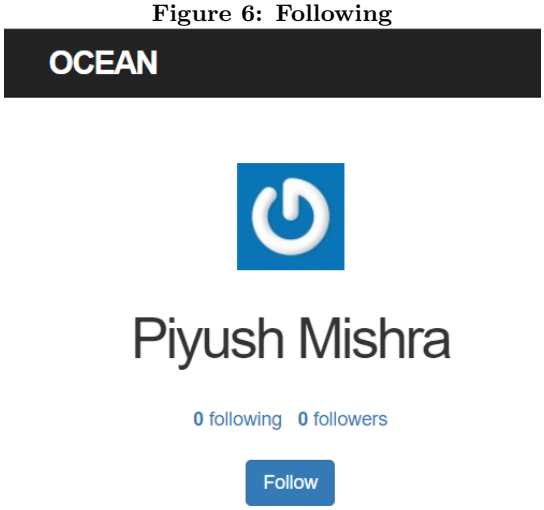


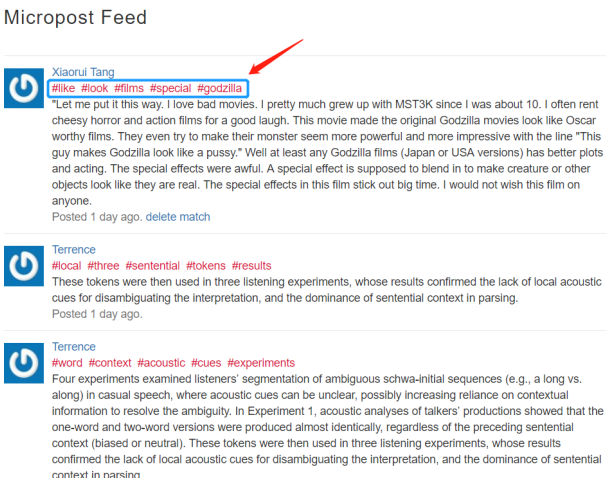
Figure 6: Following



Micropost is a function similar to Twitter posts. On the homepage, users are able to see their own past microposts and display them in reverse chronological order. They can also view the microposts of other users and display the users' microposts dynamically on the home page.

7.1.3 *Use machine learning to analyze microblogs and generate tags*

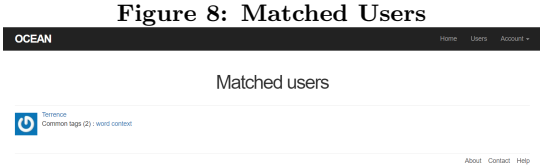
Figure 7: Microposts with respective tags generated using machine Learning



When the user writes a microposts, the system calls a RESTful web-service and generates tags through the machine learning algorithm coded in Python.

7.1.4 *Matching*

Figure 8: Matched Users



Users can choose match on the basis of a specific micropost. When users press the 'Match' button, all of the tags will be searched in proper order, as well as matched with all of tags from other users present in the database. Finally, we can display the matched users with tags successively according to the matching rate(number of matching tags). Users can even look up which blog the tags belong to, by clicking the on a particular tag.

7.2 Machine Learning

7.2.1 *Data preprocessing*

In this section, we did some work on the data preprocessing, in each machine learning work, implementing the algorithm as well as the model is not the most significant part, what directly determined the accuracy of prediction is the quality of data, so data preprocessing is quite significant in or job, we have done three steps on that:

- **Tokenization:** Tokenization segments a document into its atomic elements. In this case, we are interested in tokenizing to words. Tokenization can be performed

many waysâĂŞwe are using NLTK's tokenize.regexp module:

- **Stop Words**: Certain parts of English speech, like conjunctions ("for", "or") or the word "the" are meaningless to a topic model. These terms are called stop words and need to be removed from our token list.The definition of a stop word is flexible and the kind of documents may alter that definition. For example, if we're topic modeling a collection of music reviews, then terms like "The Who" will have trouble being surfaced because "the" is a common stop word and is usually removed. You can always construct your own stop word list or seek out another package to fit your use case.

- **Stemming words:** this is another common NLP technique to reduce topically similar words to their root. For example, "stemming," "stemmer," "stemmed," all have similar meanings; stemming reduces those terms to "stem." This is important for topic modeling, which would otherwise view those terms as separate entities and reduce their importance in the model.

7.2.2 Applying LDA Model

corpus is a document-term matrix and now we are ready to generate an LDA model:

The LdaModel class is described in detail in the gensim documentation. Parameters used in our example:
Parameters:

- **Num_topics:** required. An LDA model requires the user to determine how many topics should be generated. Our document set is small, so we're only asking for three topics.

- **Id2word:** required. The LdaModel class requires our previous dictionary to map ids to strings.

- **Passes:** optional. The number of laps the model will take through corpus. The greater the number of passes, the more accurate the model will be. A lot of passes can be slow on a very large corpus.

7.3 Deployment:

1. **Web Application:** As the Heroku offers good support for deploying Rails app easily, we chose to deploy our web platform on Heroku.

2. **RESTful Web API:** The Flask-based Python webservice is deployed on PythonAnywhere (uses AWS behind the scenes) which allows easy hosting of Python based web applications.

8. ALGORITHM COMPARISON

8.1 Text rank

Graph-based ranking algorithms are essentially a way of deciding the importance of a vertex within a graph, based on global information recursively drawn from the entire graph. The basic idea implemented by a graph-based ranking model is that of "voting" or "recommendation". When one vertex links to another one, it is basically casting a vote for that other vertex. The higher the number of votes that are cast for a vertex, the higher the importance of the vertex. Moreover, the importance of the vertex casting the vote determines how important the vote itself is, and this information is also taken into account by the ranking model. Hence, the score associated with a vertex is determined based on the votes that are cast for it, and the score of the vertices casting these votes. Formally, let be a directed graph with the set of vertices and set of edges , where is a subset of . For a given vertex, let be the set of vertices that point to it (predecessors), and let be the set of vertices that vertex points to (successors). The score of a vertex is defined as follows (Brin and Page, 1998)

$$S(V_i) = (1-d) + d * \sum_{j \in In(V_i)} \frac{1}{\left|Out(V_j)\right|} S(V_j)$$

where L is a damping factor that can be set between 0 and 1, which has the role of integrating into the model the probability of jumping from a given vertex to another random vertex in the graph. In the context of Web surfing, this graph-based ranking algorithm implements the "random surfer model", where a user clicks on links at random with a probability L, and jumps to a completely new page with probability MON L. The factor L is usually set to 0.85 (Brin and Page, 1998), and this is the value we are also using in our implementation

8.2 Graph-based Ranking Algorithms

To enable the application of graph-based ranking algorithms to natural language texts, TextRank starts by building a graph that represents the text, and interconnects words or other text entities with meaningful relations. For the task of sentence extraction, the goal is to rank entire sentences, and therefore a vertex is added to the graph for each sentence in the text. To establish connections (edges) between sentences, we are defining a "similarity" relation, where "similarity" is measured as a function of content overlap. Such a relation between two sentences can be seen as a process of "recommendation": a sentence that addresses certain concepts in a text, gives the reader a "recommendation" to refer to other sentences in the text that address the same concepts, and therefore a link can be drawn between any two such sentences that share common content. The overlap of two sentences can be determined simply as the number of common tokens between the lexical representations of the two sentences, or it can be run through syntactic filters, which only count words of a certain syntactic category. Moreover, to avoid promoting long sentences, we are using a normalization factor, and divide the content overlap of two sentences with the length of each sentence.

8.3 LDA Topic Modeling(what we choose in this project)

Blei et al. (2003) introduced a new, semantically consistent topic model, Latent Dirichlet Allocation (LDA), which immediately attracted a considerable interest from the statistical machine learning and natural language processing communities. The basic generative process of LDA closely resembles pLSI. In pLSI, the topic mixture is conditioned on each document. In LDA, the topic mixture is drawn from a conjugate Dirichlet prior that remains the same for all documents. The process of generating a corpus is as follows (we consider the smoothed LDA here):

1) Pick a multinomial distribution Φ *z for each topic z from a Dirichlet distribution with parameter* β *;*

2) For each document d, pick a multinomial distribution θ *d from a Dirichlet distribution with parameter* α *,*

3) For each word token w in document d, pick a topic z $\in$ *1...K from the multinomial distribution* θ *d*

4) Pick word w from the multinomial distribution Φ *z .*

Thus, the likelihood of generating a corpus is:

$$= \iint \prod_{z=1}^{K} P(\phi_z \mid \beta) \prod_{d=1}^{N} P(\theta_d \mid \alpha) (\prod_{i=1}^{N_d} \sum_{z_i=1}^{K} P(z_i \mid \theta) P(w_i \mid z, \phi)) d\theta \, d\phi$$

Compared to the cluster model, LDA allows a document to contain a mixture of topics, relaxing the assumption made in the cluster model that each document is generated from only one topic. This assumption may be too limited to effectively model a large collection of documents; in contrast, the LDA model allows a document to exhibit multiple topics to different degrees, thus being more flexible.

9. CHALLENGES

Every project has its own set of challenges. Our project is no exception. During the development process, we encountered all kind of problems, but through active efforts, most of us found a solution. Here are some major questions:

1. **Rails app can't accept CORS Post requirement**
   During the process of integrating the web platform and the RESTful web API, we encountered the problem that applications cannot accept POST requests across domains. After searching about this problem online, we found that the solution to this is adding "skip_before_action: verify_authenticity_token" in the code to enable cross-domain request and solve the problem.

2. **Dropdown menu in the header doesn't work**
   When the dropdown menu was implemented using Bootstrap, it didn't work at first. After looking into this matter, we found that this was a relatively common problem. Rails does not allow JQuery by default, and the dropdown menu for Bootstrap requires JQuery support. So we introduced enabled JQuery in Rails application.js, and we were able to overcome the problem.

3. **Email can't be delivered because of incorrect domain**
   After finishing the email authentication function, we realized that we cannot get the user activation email. The error code received was: 550 5.7.1 Unauthenticated email from domain.tld is not accepted due to domain's DMARC policy. After rigorously checking the settings of the application, we found that we had set the default email from address as "noreply@example.com", which leads to the error mentioned above. By changing that to a valid email address, we were able to close the issue.

4. **The tags can't be displayed without refreshing after the app receive them from ML web service**
   At first, the way we implemented tags was to send an HTTP POST request to the Machine Learning web service when the user created a microblog, and then to receive the HTTP JSON response in return as a result from the RESTful web service. But in this way, users don't get tags immediately when they create a micropost, and there is a short delay.This is an asynchronous operation to some extent, so we need to implement CORS's asynchronous request and update the tag on the microblogging page. After to the various attempts failed to resolve this issue, and render the changes synchronously in real-time, we changed our plan and settled for a temporary workaround where we wait for the response from the web-service and then render the post and tags at the same time. Obviously, this is not a good solution as the users have to wait before they could see their post on the screen.

5. **The email sending takes almost 1 minute to process**
   This problem has not been resolved yet. Problem can be simply because we are using SendGrid. The third-party plug-in email response is slow, the email is not sent to the user immediately, instead it takes a long delay (30 seconds to 1 minute).The cause of the problem is unclear. One may be able to re-configure the email system to solve problems by not using SendGrid.

10. EVALUATION

We developed a form and floated it that procured users' opinions based on three factors:

- User Sign up : whether the users were successfully able to sign up and login into the application.
- User able to post : This question is based on the functional aspect of the project.
- Ease of use of the application : whether it helped the users understand the system and how to use it in an easy way.
- Matching of users.

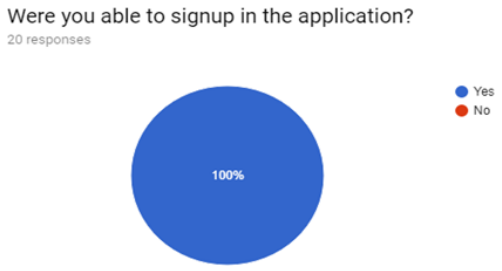
- Their opinion on whether the application was a good idea. This could in a way describe the ”cool” factor.

- Further improvement ideas were gathered from them that could be looked forward to in the future scope.

Graphs related to our evaluation form:

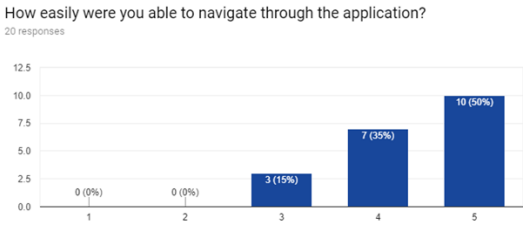
Here we asked our evaluators, if they were able to signup and login into our web application so as to confirm the non-functional requirements of our project. Naturally, the security of user account is of great concern to us, hence we have implemented hashing based security measure for password. All of our users were able to successfully signup and login with no issues found so far.

Figure 9: Evaluation Question about Accessability



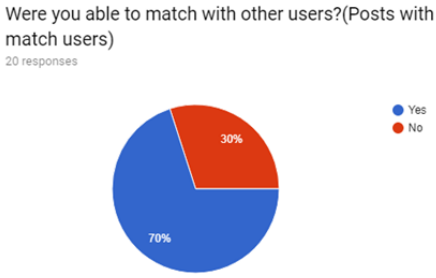
This question helped us determine how our users felt about the overall user experience. 50% of our users felt at ease and comfortable using our application. We tried a few tweaks with the UI to keep it as simple as possible.

Figure 10: Evaluation Question about Ease of use



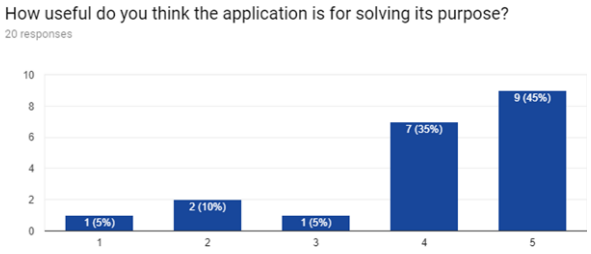
After the users got accustomed and moved around to figure out the application. We then asked the primary feature of our application in the present scope to them. To our surprise, 30% of our users weren't able to test it out clearly. This is mainly due to the lack of posts and users in our application to match the initial users' post with some other post. After a few posts and guidance on our side, 70% were able to match later.

Figure 11: Evaluation Question validating Matching Functionality



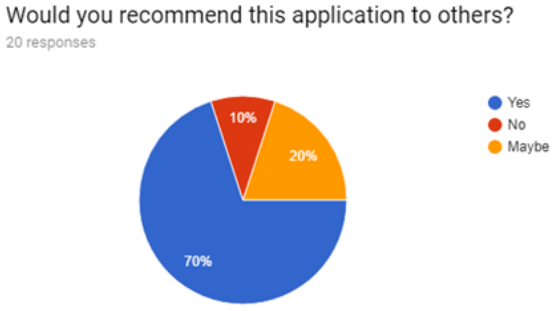
80% of our evaluators think that stress can be relieved through the use of our project. This is a big difference compared to the initial user feedback prior to the implementation of the application.

Figure 12: Evaluation Question validating overall Functional Requirements



This is with reference to the previous question, majority of the evaluators would recommend our application to others. 30% of our users are unsure or deny sharing it with others. This is in sync with our initial findings.

Figure 13: Evaluation Question checking the willingness to use



11. CONCLUSION

We were able to successfully implement all of the proposed measures in the given time frame. We implemented these major functionalities:

- Social media based web application which would allow user to post stuff and see others' posts on the home page
- Accurately assign tags(keywords) to each micropost using machine learning algorithms
- Successfully matching users on the basis of common tags.

12. FUTURE SCOPE

Having implemented all the proposed functionality requirements, we feel that given more time duration we can still improve our application and a few more functionalities like:

- Make the users completely anonymous
- Let users connect on a chatting platform
- Improve the user-matching algorithm
- Android/iOS Mobile platform to complement our web platform

13. REFERENCES

[1] Ways of reducing the impact of stress on human capital performance Lecturer PhD Nicoleta Valentina FLOREA Valahia University of Targoviste, Romania. Professor PhD Constanta POPESCU Valahia University of Targoviste, Romania.

[2] The Psychological Meaning of Words: LIWC and Computerized Text Analysis Methods Yla R. Tausczik1 and James W. Pennebaker1 Journal of Language and Social Psychology

[3] Willingness to Share Knowledge Compared with Selected Social Psychology Theories- Ewa Krok

[4] American College Health Association. American College Health Association-National College Health Assessment II: Reference Group Executive Summary Fall 2015. Hanover, MD: American College Health Association; 2016.

[5] More than words: Social networks' text mining for consumer brand sentiments Mohamed M. Mostafa Instituto Universitario de Lisboa, Business Research Unit, Avenida das Forcas Armadas, Lisbon, Portugal

[6]Ruby on Rails Tutorial: Learn Web Development with Rails by Michael Hartl