

CS 520/620 Final Report

MoodMinder

Anupama Pasumarth, Aditya Sundarrajan, Yamin Tun

Contents

1. Abstract
2. Introduction
 - a. Goals
 - b. Contributions
3. Approach
 - a. System Architecture
 - b. Sentiment Classification
 - c. Recommender System
 - d. User Interface
4. System Evaluation
 - a. Unit Testing
 - b. Classifier Evaluation
 - c. Recommender Evaluation
 - d. Lessons Learnt
5. Conclusions and Future Work
6. References
7. Appendix

1. Abstract

Mood tracking is useful for people who suffer from depression or mental disorders such as bipolar disorder, where users can keep track of their varying emotions and change their schedules based on positive suggestions by the application. The existing mood tracking applications require manual user's effort to enter their current mood on a daily basis. In this project, we have developed a supervised machine-learning-based web application that automatically detects the user's mood based on the user's Twitter status updates, and alleviates the user's mood in times of distress.

2. Introduction

Mood tracking is a useful application for people who wish to monitor their changes in mood and emotions over time and how they act on it. Moreover, such users are usually interested in receiving positive feedback that would alleviate their mood. For example, if the application detects that a user is feeling stressed out, the app can suggest to display some positive motivational quotes, pictures of nature/cats depending on his or her interest, or play some calming or happy music.

Most existing apps that track emotions, prompt users to provide information about their current mood, and suggest positive actions after. Other apps require its users to complete surveys that are then used to decide the user's mood. This questionnaire based technique could interrupt user's activities and be less useful as time goes on. The idea behind our web app is to automatically observe and determine the user's mood based on the user's social footprint and suggest positive actions when the user experiences a negative mood. This learning based approach does not require the user to constantly answer questions which makes the app frustrating to the user. Past work [1] has shown that twitter provides a lot of information on user sentiments and the availability of twitter data, for instance [2], makes it easy to analyze user moods and sentiments. Hence, in this project, we analyze the user's social behavior on twitter to predict the user's mood. We assume that user's tweets convey either positive or negative mood of the user. Since we focus on providing positive actions when the user's mood is negative, we ignore cases where the user's mood is identified as being positive.

a. Goals

The research questions enumerated below were the driving force behind our project design and development:

1. How can we classify the user's mood as positive/negative from the user's tweets using supervised machine learning?
2. How can we use historical or pre-populated information to make suggestions that would alleviate the user's mood?
3. How do we design a web app that takes in user tweets as input, employs machine learning models that answer questions 1 and 2, and provides useful output to the user?

b. Contributions

1. Analysis and comparison of multiple sentiment classification techniques which led us to choose the most effective one for our application.
2. An implementation of a recommender system based on similarity score of category names and text input.

3. An application design and implementation that is simple, yet flexible and well extensible, that showcases our working pipeline of the entire project.

3. Approach

Our system design is based on a Model-View-Controller architecture. The main components of our system are:

1. Classifier
2. Recommender

both of which constitute the underlying model in the MVC architecture. The Controller is responsible for listening for streaming tweets and fetching the tweets from the Twitter handle (username) that the user specifies in the webapp. The controller then hands the tweet input over to the model which then produces sentiment prediction and recommendation produced by the classifier and recommender systems respectively. The View renders and displays the sentiment classified and recommendation generated from the model on the screen.

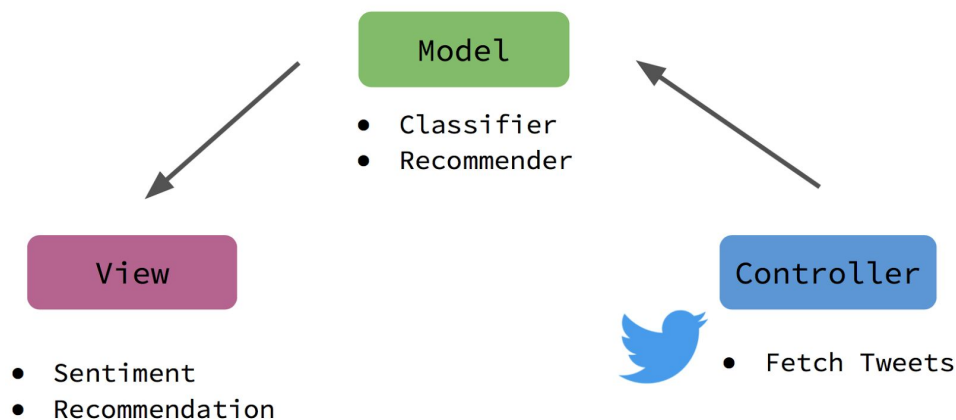


Figure 1. Model-View-Controller System

We use two design patterns to implement the MVC architecture. We use the strategy pattern to implement the interaction between the view and the controller. The strategy pattern is used to specify the type of classifier for sentiment identification that will be used to classify the input tweet. Our `TweetClassify` class encapsulates the type of classifiers, namely SVM or NB. Ideally, we intend users to be able to change different pre-trained classifiers with different features for adjusting accuracy. Depending on the user's selection, a specific classifier will be chosen. The view is updated with sentiment prediction results from the chosen classifier. In our current version, we have set naive bayes model as the default classifier, which is passed into `TweetClassify`. We have left the ability to choose from multiple classifiers for future work. We use the observer design pattern to implement the interaction between the model and view. Every time the model produces an output that include a sentiment and possibly a recommendation, the model notifies the objects in view and update them automatically. In this

project, the view is the webpage that is updated with the result of the classification and recommendation. In the following sections, we describe the different components of the MVC architecture in more detail.

a. Sentiment Classification

Twitter Sentiment analysis is a well-studied area in Natural Language Processing as the social media gains popularity among users around the world. Classifying sentiment in tweets is similar to the sentence-level sentiment analysis which is the fundamental step for the classical NLP task such as document level classification. However, the microblogging features (such as all capital letters and character representation) and twitter-specific features (such as hashtags and emoticons) make the task different. Support Vector Classifier (SVC), Naive Bayes (NB) and Maximum-Entropy classifier are three commonly used classifiers for twitter sentiment analysis. [3] [4]

In our project, we have used two different classifiers: Support Vector Classifier (SVC) and Naive Bayes (NB).¹ The current application uses Naive Bayes Classifier trained on 5000 tweets with the use of unigrams and bigrams as features. The app can be switched with a different classifier due to the strategy pattern in our `TweetClassify.py`.

Dataset

A subset of the Sentiment140 tweet corpus was used for training the classifier. The original dataset consists of more than 1 million tweets. The data set was collected by Stanford group by filtering out tweets with positive emoticons such as “:)” and negative emoticons such as “:)” with the use of Twitter Search API. The two classes that we are training to identify are 0 for positive and 4 for negative. [6]

Classifiers

Naive Bayes (NB)

Naive Bayes is a simple probabilistic classifier that uses Bayes’ rule for training. NB is a Bag-of-Word (BoW) model that uses frequency of each word appear in the training set as feature during training process. Naive Bayes makes the ‘naive’ assumption that all features are independent given the label, which is not always true in tweets or any text. Words or phrases in any text may have dependency in a sentence. For example, if unigram is used as features, the unigrams “not” and “happy” in phrases such as “not happy” have dependency on each other. This is one of the major limitations of Naive Bayes. [7]

¹ We have also tried maximum entropy classifier. Maximum entropy classifier select the model parameters by iterative optimization methods. The default 100 iterations gave close to 0.5 classifier accuracy. Since the training time was significantly long (hours long) for higher number of iterations such as 1000 for better optimization, we decided not to use it. [5]

Support Vector Machines (SVMs)

Support vector machines are supervised learning methods, widely used for sentiment analysis. A SVM is a discriminative classifier that produces an optimal hyperplane to separate new instances into different classes given a training data. SVMs can also perform nonlinear classification using the kernel trick which is basically mapping the instances in high-dimensional feature spaces. In our case, we use SVM with linear kernel, assuming that positive and negative classes will be linearly separable in feature space. [8]

Classification Pipeline

Figure 2 shows the pipeline of our classification process. The classifier is pre-trained on training corpus of 5000 tweets before the real-time classification occurs. It is crucial to choose informative features to obtain good classifier accuracy. For the simplicity of feature extraction process, simple features such as unigrams and bigrams are used for sentiment classifiers. We have set up a small experiment with naive bayes classifier to observe the strength and weakness of unigram and bigram features. It was observed that the model trained with bigrams (together with unigrams) correctly classified tweets with specific two-word phrases that the unigram model failed to correctly classify them. For example, with the use of unigrams, “the food is **not good**” was misclassified as positive since “good” was weighed as positive more than other words in the sentence. However, bigram model successfully classified it as negative since “not good” was one of the important features learned by the classifier. According to this result, bigrams were included as additional features to see if there is improvement in accuracy. Results are mentioned in evaluation section.

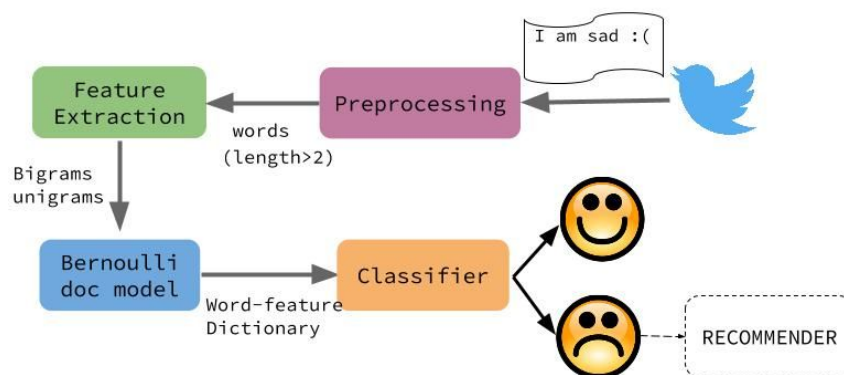


Figure 2 : Classification Pipeline

i. Preprocessing & Feature Extraction

Tweets of 5000 were randomly sampled from Sentiment140 dataset. A raw tweet may contain noisy information that are unrelated to sentiment such as urls, twitter usernames, punctuations with the use of regular expression. All letters in the remaining tweet are converted to lowercase. All adjacent pairs of words were extracted as bigram features of the tweets. Only words with more than 2 characters were extracted as unigram features since most of the words fewer than 2 characters such as preposition and articles usually do not provide useful sentiment information. Though emoticons are key indicators of sentiment, we have excluded emoticons in our sentiment analysis for simplicity of text processing. For future work, we could use a database of emoticons to extract features and use them as additional features in our training process.

Bag-of-words (BoW) is a list of all unique unigrams and bigrams in training tweets. For each tweet, a word-feature dictionary is constructed by applying BoW. Bernoulli document model is used in this case. Bernoulli document model is the representation of the tweet with a feature vector with binary value of whether the document contains a word in BoW. [9] In our case, a Bernoulli word-feature dictionary has unigrams and bigrams as *keys*, and boolean of whether each word in BoW contains in the tweet as *values*. This feature dictionary will be used as an input into the classifier for training and testing process. [10] [11]

ii. Training

The feature dictionaries corresponding to each training tweet are fed into Naive Bayes (NB) and Support Vector Classifier (SVC) for training purpose. Naive Bayes was trained using nltk library. Linear SVC model was trained using the default parameters of scikit-learn. As future work, hyperparameter selection could be performed to tune the classifier parameters and improve classifier accuracy. [12] [13]

Table 1 and 2 show some of the informative features that Naive Bayes learned during training process. These informative features are used during classification process to identify sentiments. For example, in this tweet, “I **love** today’s weather. Such a **great** day!”, the bolded words are likely to be top-ranked features in identifying positive sentiment.

According to the result, some of the bigram features such as “of my” in Table 2 are noises with no sentiment information. The bigram features such as “I miss” and “miss you” are repeated with no separation from the key sentiment word “miss” with other non-sentiment words such as pronouns in this case. Evaluation section describes more details on pros and cons of bigrams and unigrams.

Table 1: Informative unigram features of Naive Bayes

Positive	Negative
birthday nice video thank amazing best name post song lmao glad dear hehe tweeting thanks ...	sad hurts anymore missed sucks wish forgot sick break left rain feeling cry bad without ...

Table 2: Informative bigram+unigram features of Naive Bayes

	Positive	Negative
Bigram	thank you I like I love ha ha happy to love you glad you happy birthday ...	I miss I want I hate miss you now I wish I of my not good ...
Unigram	thank happy com great hehe ...	sad sorry sick rain nothing ...

iii. Real-time Classification

The tweet posted by the user is fetched and preprocessed to obtain unigrams and bigrams. The BoW from training set is applied on the unigrams and bigrams to build a word-feature dictionary as mentioned in Feature Extraction section. The feature dictionary for the incoming tweet is fed into the pre-trained classifier to classify the sentiment: positive or negative. If the tweet is classified as negative, the recommender system is triggered to provide suggestion for cheerful activity related to the content of the tweet in order to improve the mood.

b. Recommender System

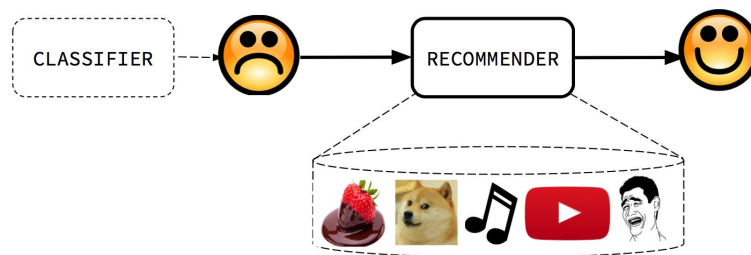


Figure 3 : Recommender system

Recommendation systems such as those used by Netflix, Amazon, etc. [14] have been widely used to suggest actions or choices to users based on their past behavior. Recommendation techniques can be broadly classified into two groups, content-based recommendation and collaborative filtering. Content-based recommendation systems suggest actions based on specific features of past choices made by the same user. For example, these features could

include the genre of movies, the type of products and so on. On the other hand, collaborative filtering suggests actions based on choices made by other similar users. For example, all users that like a certain movie genre are clustered together and suggestions are made based on collective interests. Such a technique is more complicated since a large amount of data, capturing the similarities among different users, needs to be obtained. Netflix uses a hybrid recommendation technique that combines the best features of both content-based recommendation and collaborative filtering based recommendation.

In this project, we use a technique that is similar to content-based recommendation since we want MoodMinder to be customized to a single user's behavior. The recommendation system in MoodMinder is used to suggest positive actions to the user based on his/her tweets. The positive actions suggested to the user are chosen from a pre-populated table, split by categories. As shown in the Figure 3, when the tweet is classified as negative, the recommender suggests some positive action from its pre-populated database. For now, the recommendation system has three categories: food, music and jokes, with each category having one suggested action. For instance, if the user tweets 'This biscuit is bad!', the tweet is classified as negative by the classifier, and the recommendation system recommends something positive from category 'food', perhaps a candy!. If the input tweet is detected to be positive by the classifier, the recommender simply posts an output message 'Be happy' and does nothing more since we focus on alleviating the user's mood when he/she is in a negative mood. In the following sections, we describe the implementation of the recommendation system.

Design and implementation

The recommendation system is composed of two components, a keyword extractor and a category generator. Both of these components are described in detail below.

i. Keyword extractor

We use a simple keyword extraction technique based on parts-of-speech tagging [15]. POS works well for short text inputs such as tweets. In this technique, every incoming tweet is classified into different parts of speech using standard POS taggers (nltk). Before the parts of speech is determined, the input tweet is filtered to remove stop words such as 'this', 'that', etc. Once the tweet is tagged, we extract only the nouns, adjectives and adverbs, as useful features and discard the rest. Future work could look at incorporating other parts of speech as well. Once the keywords are extracted, they are passed on to the category generator which maps the tweet to one of the available categories and chooses an action from that category.

We illustrate the keyword extraction procedure through an example. Let us consider the following tweet 'Both the music and the dance were horrible'. The keyword extraction procedure is explained in the following steps.

1. Stop words 'the', 'and' and 'were' are filtered from the input tweet.

2. The remaining words are tagged as 'Both'=determiner, 'music'=noun, 'dance'=noun and 'horrible'=adjective by the POS tagger.
3. Words that are not nouns, adjectives or adverbs are removed. Hence, the keywords extracted from the tweet are 'music', 'dance' and 'horrible'. These three keywords are then sent as input to the category generator.

ii. Category generator

The category generator works by calculating similarity scores (between 0 and 1), using standard libraries such as nltk wordnet [16], between the different keywords extracted from the incoming tweet and the different category names. The cumulative similarity score for a specific category is then calculated as the sum of all similarity scores between all the extracted keyword and that category name. The category with the highest cumulative similarity score is then chosen and any random action from that category is suggested to the user, provided the cumulative score is greater than a threshold, currently fixed at 0.2. Random actions from the category 'jokes' is suggested if all the cumulative similarity scores are less than 0.2. The value of this similarity score threshold can be reduced as the recommender becomes more accurate; this is left as future work.

We illustrate the entire procedure though the same example as before. Let us consider the tweet 'Both the music and the dance were horrible'. From the previous section, the keywords extracted are 'music', 'dance' and 'horrible'. In the following steps, we explain how category generation works.

1. For each category name in ['food', 'music', 'jokes'], we calculate the similarity with all the keywords extracted above and obtain the cumulative similarity score. The cumulative similarity scores are as follows: 'food'=2.56, 'music'=9.20, 'jokes'=3.93.
2. Category 'music' is chosen since it has the highest cumulative similarity score and it is greater than 0.2.
3. In our current implementation, the category 'music' has only one suggestion, coldplay, and this is suggested to the user.

c. Web Application

Django

We chose Django web framework to create our web app. Django is a high-level Python Web framework that encourages rapid development and clean, pragmatic design.

The core Django framework can be seen as an MVC architecture. It consists of an object-relational mapper (ORM) that mediates between data models (defined as Python classes) and a relational database ("Model"); a system for processing HTTP requests with a web templating system ("View") and a regular-expression-based URL dispatcher ("Controller").

The core framework also includes a lightweight and standalone web server for development and testing and an interface to Python's built-in unit test framework which was one of the main factors that made us choose Django.

Bootstrap

Our template design was a simple one created using Bootstrap classes for styling the webpage. Bootstrap is a free and open-source front-end library for creating websites and web applications. It contains HTML and CSS-based design templates for typography, forms, buttons, navigation and other interface components, as well as optional JavaScript extensions. It aims to ease the development of dynamic websites and web applications.

d. UI Design

To demonstrate the working of our complete system, we designed a web app with a simple user interface with one page. The page contains a form along with a submit button which allows a user to enter any valid Twitter handle. Upon form submission by clicking the 'Get Latest Tweet' button in the form, the value from the text field will be sent through a POST request to the method that contains the logic for our sentiment classifier and recommendation system. This will return the set of values in a data structure that would be parsed by Django's View component and displayed as a table to the user. The entire web application runs locally on the Django server. The following screenshots are from a test-run of our application.



User Handle	Tweets	Sentiment	Recommendation
-------------	--------	-----------	----------------


Screenshot 1: The initial form prior to entering a Twitter handle and form submission



Get Latest Tweet

User Handle	Tweets	Sentiment	Recommendation
UmassAmherst	Ribbon cutting today at #UMass Amherst renovated College of Educ Furcolo Hall w/ Foster Furcolo's sons and @SenStan. https://t.co/1LkzpNpBCa	😊	Be happy
sadquotes	RT @relatablequote_: Sometimes you have to go through the worst in order to get to the best.	😞	COLDPLAY
realDonaldTrump	We are now at 1001 delegates. We will win on the first ballot and are not wasting time and effort on other ballots because system is rigged!	😊	Be happy
sosadtoday	i maybe actually hate everything	😞	KNOCK KNOCK.. WHO IS

Screenshot 2: The table after form submission of multiple twitter handles (most recent: UmassAmherst)




Get Latest Tweet

Refresh!

User Handle	Tweets	Sentiment	Recommendation
-------------	--------	-----------	----------------

Screenshot 3: The initial form of the streaming version of our webapp prior to entering a Twitter handle and form submission



Get Latest Tweet

Refresh!

User Handle	Tweets	Sentiment	Recommendation
cs_loves_kim_k	Today is a nice day!	😊	Be happy
cs_loves_kim_k	Both the music and the dance were horrible :(😞	COLDPLAY
cs_loves_kim_k	Make America great again :)	😊	Be happy
cs_loves_kim_k	I hate this food!	😞	CANDY

Screenshot 4: The streaming version of our webapp prior after testing with the handle 'cs_loves_kim_k'. Hitting "Refresh!" automatically loads the most recent tweet after specifying user handle

4. System Evaluation

a. Unit testing

We perform unit testing for the relevant functions in `views.py` in `moodminder_test/app` and the functions in `TweetProcess.py`, `TweetRecommend.py` and `TweetView.py`. The unit tests can be found in `moodminder_test/app/tests.py` for `views.py` and `moodminder_test/app/code/Tests.py` for the functions in `Tweet*.py`.

Unit tests were also written for the functions in feature extractors that strip off urls and punctuations and extract bigrams and unigrams. The functions in `Util_data.py` for splitting dataset and sampling tweets were also unit-tested. Minimal number of inputs are used in the unit tests.

b. Classifier Evaluation

Classifier Evaluation is performed to mainly observe the impact of different feature sets and performance of different classifier types. Two popular classifiers for sentiment analysis such as Naive Bayes (NB) and Support Vector Classifier (SVC) were evaluated in comparison in terms of training time, accuracy, precision and recall. The classifiers were trained using two different feature sets to evaluate the impacts of unigrams and bigrams. Below are four different models that we are evaluating:

1. NB with unigrams only
2. SVC with unigrams only
3. NB with unigrams and bigrams
4. SVC with unigrams and bigrams

Experiment 1: Cross-Validation

One potential issue in training supervised machine learning model is that the model could be “memorizing” or fitting over the insignificant noise in the given training data instead of the underlying pattern of features. Such process is called overfitting. The possibility of overfitting exists due to the difference between training and test criteria. The model is trained by maximizing its performance on some part of training data. However, the actual performance of the model is evaluated based on unseen data. To reduce overfitting, the cross-validation was performed to assess how generalizable our model is to an independent data set. [17]

The 3-fold cross-validation was performed by holding out 500 tweets for testing while training classifier with 1000 tweets. Table 3 shows the average accuracy of Naive Bayes and Support

Vector Classifier, for two different features: only unigrams and both bigrams and unigrams. Overall, SVC has slightly higher accuracy (about 1 %) for both feature sets than NB.

The results show that using bigrams as additional features decreases the overall accuracy of the classifiers slightly rather than improving. The decrease in accuracy of models with bigrams is explained in the confusion matrix in Figure 5. For both NB and SVC, more false positives and fewer true negatives were classified after the addition of bigrams as features. Although bigrams capture sentiment in certain two-word phrases that unigrams fail to capture such as “not good”, bigrams could be populated with irrelevant words. The main reason is that in bigram features, the key sentiment word are not isolated from irrelevant words such as prepositions and articles. This is partly because bigrams were extracted without removing words fewer than 2 characters. For example, the bigram “I hate,” does not isolate the negative keyword “hate” well from the irrelevant information such as “I.” Unlike unigram, this bigram will not be a useful feature for classifying tweets that consist “we hate” despite the same content. The above hypothesis cannot be generalized to all datasets unless we observe the results for cross-validating a bigger data set with a higher k-fold value.

Table 3: Average accuracy of 3-fold cross-validation. Unigrams and bigrams are extracted as features from the data set of 1500 tweets

Classifiers	Average accuracy	
	Unigrams	Bigrams+Unigrams
Naive Bayes	0.667	0.666
Support Vector Machine	0.682	0.679

Experiment 2: Performance Comparison of Classifiers

The goal for this experiment is to compare two classifiers that are SVC and NB in terms of different performance criteria such as training time, accuracy, precision and recall. For this experiment, 2000 tweets were randomly subsampled from the Sentiment140 dataset. The data set of 2000 tweets is further split into 0.8-0.2 ratio for train-test sets.

In this context, precision is the fraction of positively classified tweets that are true positive. In other words, $\text{precision} = \frac{\text{true positives}}{\text{true positives} + \text{false positives}}$. The higher precision value implies fewer false positives. Recall is the fraction of true positive tweets that are actually classified as positive. Recall can be written as: $\text{recall} = \frac{\text{true positives}}{\text{true positives} + \text{false negatives}}$. The higher recall means fewer false negatives and vice versa. There exists trade-off between recall and precision: increase in recall correlates with decrease in recall and vice versa.

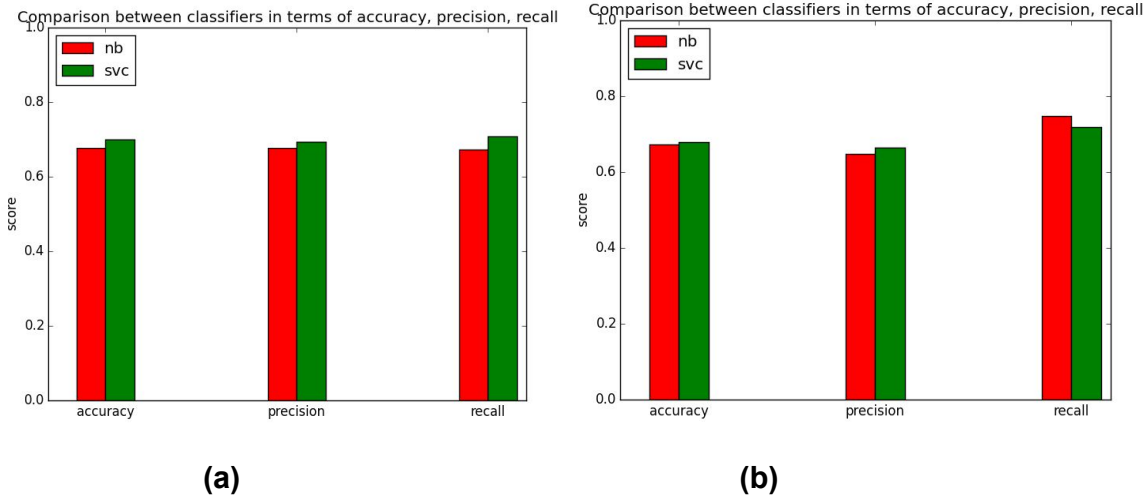


Figure 4. Comparison of training time, accuracy, precision, recall for 0.8-0.2 split for train-test with the data set of size 2000 tweets (a) unigrams only (b) unigrams and bigrams as features of classifiers

Figure 4 shows that the accuracy, precision and recall are close to 70% for both unigram-only and unigram-bigram models. For this specific training and testing dataset, with unigrams, SVC shows about 2-3% higher accuracy, precision and recall than NB. With the combination of bigrams and unigrams, SVC and NB perform similarly in terms of accuracy, precision and recall. SVC training process was about 1.4 times faster than NB's with unigrams as features. With both unigrams and bigrams, SVC was trained 1.3 times as fast as NB. The additional bigram features increase the training time significantly since there are more features to fit the model over.

Table 4: Comparison of training times for Naive Bayes and Support Vector Machine with unigrams as the only features, and the combination of unigrams and bigrams as features

Classifiers	Training time	
	Unigrams	Bigrams+Unigrams
Naive Bayes	33 seconds	2.8 minutes
Support Vector Machine	24 seconds	2.25 minutes

Figures 5 (a) and (b) show that both SVC and NB produce almost the same confusion matrices. However, after the addition of bigrams as features, Figures 5 (c) and (d) shows that both NB and SVC give more false positives and fewer true negatives than the unigram models. Considering the fact that simple features (unigram and bigrams) were alone used, the classification performances of SVC and NB are relatively sufficient enough for our prototype.

Either SVC or NB could be used in our web application since SVC and NB performs similarly. In this case, our web application uses NB classifier with bigrams and unigrams trained on 5000 tweets as default classifier. As future work, the false positives and false negatives could be reduced by using better features such as lexicons and phrases, as well as by tuning classifier parameters better.

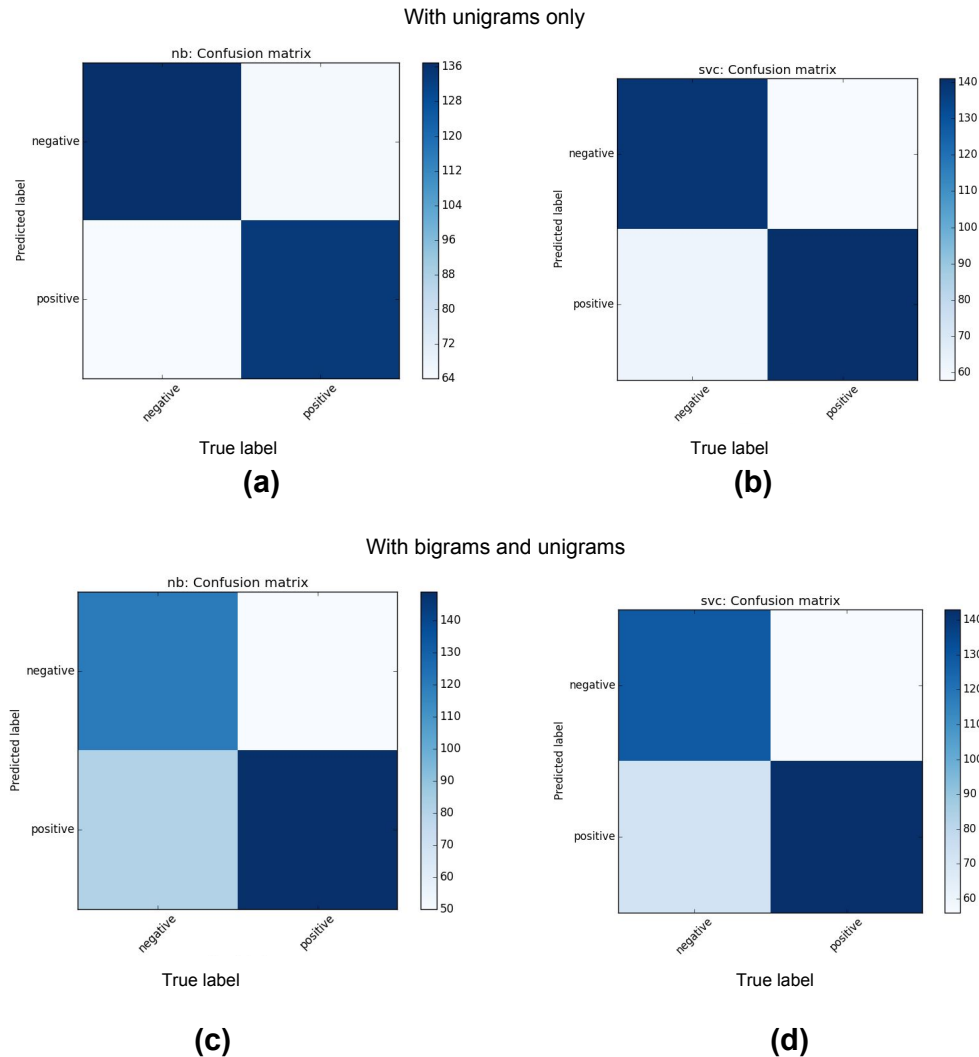


Figure 5 : Unnormalized confusion matrices for (a) Naive Bayes with unigram (b) Support Vector Classifier with unigram (c) Naive Bayes with unigram and bigram (d) Support Vector Classifier with unigram and bigram. The scale bar represents the number of tweets classified.

c. Recommender Evaluation

The accuracy of the recommendation system is relative to the user using MoodMinder. For instance, if the user is experiencing a bad day, MoodMinder might recommend a song based on

the features extracted and the category generated, while the user might have instead preferred a candy or a joke to cheer him/her up. With this in mind, we evaluate the accuracy of the recommender system using 30 tweets (a combination of tweets generated by us and by other users). The current recommender has three categories music, food and jokes, with a single action in each category namely, coldplay, candy and knock knock.. who is, respectively. Some of the tweets and the actions suggested by the recommender can be seen in Table 5. The complete result can be seen in the Appendix. The recommender is fairly accurate in 29 out of 30 cases. In one scenario, the recommender system suggests a joke when the user tweeted 'I am starving :('. This is because the keyword extractor doesn't extract verbs. Techniques like lemmatization could be used to improve the accuracy of the recommender but we leave this for future work.

Table 5: Recommender evaluation sample out

Tweet	Action
Anything funny or humorous?	KNOCK KNOCK.. WHO IS
Both the music and dance were bad	COLDPLAY
I hate tea and coffee	CANDY

d. Lessons Learnt

The final prototype of our project was quite similar to our initial design proposal except for a couple of changes.

Our initial design consisted of charts that helped a user visualize the time-series mood patterns of the user across different timescales. After a design iteration, we decided to add a learning component to our system and focus on improving the recommender system design to make it more intelligent (as described in the earlier section).

Some of the noteworthy points from our project are:

1. An MVC architecture made our design and development process smooth and flexible
2. Using third-party libraries imposed some constraints that were not very easy to comply with through the entire development cycle
3. One of the challenges we faced was to define a concrete testing and evaluation approach e.g. how do we quantify the relevance of our recommendation, since it is a subjective aspect that depends and varies from user to user

5. Conclusions and Future Work

MoodMinder is a django based web application that can be used by people who suffer from depression or bipolar disorder and so on to automatically their moods and emotions and act on suggestions made by the app to alleviate their mood. Unlike previous applications that depend on questionnaires to provide positive suggestions, MoodMinder monitors the user's tweets and uses supervised machine learning to classify the tweets as positive or negative and recommend positive actions based on pre-populated table. The overall system has a fairly reasonable accuracy rate of about 65% based on tests performed on real tweets.

Future work

In this project, we have presented a simple classification and recommendation system that is reasonably accurate but there are several opportunities for future enhancements. We list some of possible future directions below.

1. More effective preprocessing procedures before training and predicting could be performed to improve the effectiveness of feature extraction. Stemming in which tense and plurality are removed, could be performed to correlate related features and improve the effectiveness of training process.
2. Better features such as lexicons could be extracted. One of the widely-used features for BoW models is term-frequency times inverse document-frequency (tf-idf). This will decrease the impact of less informative tokens that frequently appear in the entire training set such as "the", "in", "a" and "is." Chunking or shallow parsing could be performed to extract Parts-Of-Speech and obtain short phrases relevant to sentiment such as noun phrases. One example of chunking is Named Entity Recognition. For example, the phrase "Adolf Hitler" is related to negative emotion, while "Disneyland Resort" is related to positive emotion. [12] [18]
3. Feature selection could be done by selecting k best features according to chi-squared statistics of. Chi-squared score is used to weed out the features that are irrelevant for classification purpose such as articles and pronouns. [8]
4. Emoticons could be identified and considered as features in our training process since they are widely used by Twitter users to express emotions.
5. Hyperparameter selection could be performed using grid search technique for SVC model to better tune parameters such as penalty parameter for error and hinge loss.
6. MoodMinder currently supports only three categories, which is suitable for a proof-of-concept implementation. A more practical application should support more categories and this should be easy to extend.
7. Currently, the recommended actions are chosen from a pre-populated table. A simple extension could be a hybrid approach: a pre-populated table along with real-time updates. In other words, user feedback could be used to update the actions that would

be recommended to the user. For instance, every time a recommendation is made, be it positive or negative, the user gives some form of feedback (like or dislike). The recommender system records the dislikes and actions that receive a certain number of dislikes are removed from the table. Also, the user should be able to add new positive actions to the table over time.

8. The current recommender system suggests positive actions when negative tweets are detected but does nothing intelligent when positive tweets are detected. A feedback based recommender system could detect positive actions from positive tweets and add it to the database if the positive action doesn't already exist. This is of course subject to the accuracy of the classifier used.

6. References

1. Alexander Pak and Patrick Paroubek. Twitter as a corpus for sentiment analysis and opinion mining. In *Proceedings of the Seventh International Conference on Language Resources and Evaluation (LREC'10)*, 2010.
2. Brede Wiki. Sentiment analysis- 8 Apr. 2016
<http://neuro.compute.dtu.dk/wiki/Sentiment_analysis#Corpora >
3. Apoorv Agarwal, Boyi Xie, Ilia Vovsha, Owen Rambow, and Rebecca Passonneau. Sentiment Analysis of Twitter Data. LSM '11 Proceedings of the Workshop on Languages in Social Media(2011): 30-38. Web.
4. Efthymios Kouloumpis, Theresa Wilson and Johanna D. Moore. Twitter Sentiment Analysis: The Good the Bad and the OMG!. Proceedings of the Fifth International Conference on Weblogs and Social Media, Barcelona, Catalonia, Spain, July 17-21, 2011. AAAI Press, 2011. p. 538-541.
5. Natural Language Processing with Python. NLTK Book. <<http://www.nltk.org/book/>>.
6. For Academics - Sentiment140 - A Twitter Sentiment Analysis Tool
<<http://help.sentiment140.com/for-students/>>.
7. Natural Language Toolkit — NLTK 3.0 Documentation. <<http://www.nltk.org/>>.
8. Scikit-learn: Machine Learning in Python — Scikit-learn 0.16.1 Documentation.<<http://scikit-learn.org/>>.
9. Shimodaira, Hiroshi. Text Classification Using Naive Bayes.
<<http://www.inf.ed.ac.uk/teaching/courses/inf2b/learnnotes/inf2b-learn-note07-2up.pdf>>
10. Text Classification for Sentiment Analysis – NLTK Scikit-Learn. StreamHacker. 2012.
<<http://streamhacker.com/2012/11/22/text-classification-sentiment-analysis-nltk-scikitlearn/>>.
11. Laurent Luce. Twitter Sentiment Analysis Using Python and NLTK.
<<http://www.laurentluce.com/posts/twitter-sentiment-analysis-using-python-and-nltk/>>.
12. Python NLP - NLTK and Scikit-learn.
<<http://billchambers.me/tutorials/2015/01/14/python-nlp-cheatsheet-nltk-scikit-learn.html>>.

13. The Glowing Python: Combining Scikit-Learn and NLTK.
<<http://glowingpython.blogspot.com/2013/07/combining-scikit-learn-and-ntlk.html>>.
14. Jure Leskovec, Anand Rajaraman, and Jeffrey David Ullman, Mining of massive datasets, Cambridge University Press, 2014
15. Categorizing and Tagging Words. <<http://www.nltk.org/book/ch05.html>>
16. WordNet Interface. <<http://www.nltk.org/howto/wordnet.html>>
17. Overfitting: When Accuracy Measure Goes Wrong.
<<http://blog.lokad.com/journal/2009/4/22/overfitting-when-accuracy-measure-goes-wrong.html>>.
18. Basic Statistical NLP Part 1 - Jaccard Similarity and TF-IDF.
<<http://billchambers.me/tutorials/2014/12/21/tf-idf-explained-in-python.html>>

7. Appendix

Complete results from testing the recommender with 30 different tweets (irrespective of sentiment).

Table 6: Recommender evaluation complete out

Tweet	Action
Today is a bad day	COLDPLAY
Anything funny or humorous?	KNOCK KNOCK.. WHO IS
Make america great again	COLDPLAY
I miss how happy I used to be	KNOCK KNOCK.. WHO IS
I maybe actually hate everything	KNOCK KNOCK.. WHO IS
I keep making the wrong choices	COLDPLAY
Every body is evil!!	COLDPLAY
Physically mentally and emotionally tired!	KNOCK KNOCK.. WHO IS
Anything to cheer me up?	COLDPLAY
No joke in the tv show is funny	KNOCK KNOCK.. WHO IS
Both the music and dance were bad	COLDPLAY
I am always depressed at the end of the day	COLDPLAY

This is the worst lunch I have ever had	COLDPLAY
Justin Bieber is a horrible singer	COLDPLAY
Which is the best music show in town?	COLDPLAY
Jazz is the worst kind of music	COLDPLAY
I don't know what songs to listen to	COLDPLAY
Who has the best voice?	COLDPLAY
All songs are bad	COLDPLAY
nothing is melodious	COLDPLAY
Any song suggestions while exercising?	COLDPLAY
I hate tea and coffee	CANDY
Don't go to this restaurant! the food is horrible	CANDY
I want some dessert	CANDY
I don't like this food	CANDY
Indian food is very spicy	CANDY
I want a snack :(CANDY
I am starving	KNOCK KNOCK.. WHO IS
What is the best food around?	CANDY
I need to eat healthy vegetables!	CANDY