Emotions in UCLA Tweets

Group 9: AXWJ

Alexis Korb 304438802

Xufan Wang 204477479

Wandi Liu 004622731

Justin Xie 404673342

ABSTRACT

Micro-blogging such as Twitter and Weibo has become a dominant part of our daily lives, socially, economically, and even politically. This new form of real time expression with character limits prompts new ways for researchers to analyze the general public's emotions and reactions to various events. In this paper, we will demonstrate how different existing algorithms can be used to classify the happiness of tweets specifically from UCLA. We hope that these results will provide valuable information for the school and helpful insight to the staff, administration, student body, and general public.

1. INTRODUCTION

Twitter is fairly popular on college campuses. In particular, concerning UCLA, there are tweets about academics, student life, rankings, events, athletics, and more. Since we go to UCLA, we thought it would be interesting to study tweets about our school. Namely, we wanted to be able to determine, using tweets, whether students and faculty at UCLA are happy or unhappy. At any given time, if the majority of tweets from UCLA about UCLA are positive, then this would indicate an overall positive mood. Similarly, if most of the tweets are negative, this would signify that students and faculty are unhappy. Thus, we created models to detect the tone of UCLA tweets in order to roughly predict the mood at UCLA. We targeted tweets from UCLA, with regards to location. Then, we determined which features were relevant to predicting the tone of a tweet. Finally, we used knowledge of these features to build appropriate models to allow us to predict the mood of future tweets.

2. PROBLEM FORMALIZATION

2.1 Main Task

The main task is to predict whether the author of a tweet from UCLA is happy or unhappy based on the tone of the tweet.

This task can be broken up into a series of subtasks:

1. Determine the type of data mining problem: Since we are attempting to classify data points based upon predetermined labels (i.e. "happy" and "unhappy"), we are using supervised classification.
2. Determine what type of model to use: There are a number of different types of models we could have used. We chose to implement Linear SVM and Naïve Bayes since these model are good for supervised classification. Naïve Bayes, in particular, is well-suited for text data.
3. Gather training and test data: We collected relevant tweets by using a Twitter crawler with filtering. Next, we performed preprocessing on the data. Then, we performed manual labor in labeling these tweets.
4. Determine which features determine the tone of tweets: We did so by looking at common words and elements among tweets in each class of our training data.
5. Create and test our models: We implemented both Linear SVM and Naïve Bayes. We tested these with 10-fold cross-validation and various evaluation measures, such as accuracy and recall. We also compared the accuracy of these models when we included or excluded different features, such as stop words.
6. Report our findings: We wrote this report to summarize and explain our findings.

2.2 Task Distribution

Task	People
1. Collect and preprocess data	Wandi Liu
2. Label tweets	Alexis Korb, Wandi Lu, Xafan Wang, Justin Xie
3. Determine relevant features	Alexis Korb
3. Implement Naïve Bayes	Wandi Liu, Alexis Korb
4. Implement SVM	Xafan Wang
6. Write report	Alexis Korb, Wandi Liu, Xafan Wang, Justin Xie

3. DATA PREPARATION AND PREPROCESSING

3.1 Tweet Collection

First, we collected relevant tweets using a Twitter crawler. Since our data was concerned with UCLA tweets, we took the following features into consideration:

- **Location:** Only those tweets within a 6 mile radius of UCLA were considered. We determined UCLA's latitudinal, longitudinal coordinates to be 34.065744º N, 118.445647 º W. Ideally, we would have preferred a smaller radius that only encompassed the physical UCLA campus. However, due to the limitations of the Twitter data available to us, we could not get sufficient tweets for data mining with a more restricted radius. Nonetheless, since these tweets originated from locations geographically near UCLA, it is likely that many of these tweets were either written by UCLA students or staff, or pertained to UCLA in its content.
- **Subject:** We wanted to filter tweets by subject matter. For example, we tried excluding tweets that did not contain the keyword "UCLA" or have some related UCLA hashtag. Ultimately, given the limitation on the number of tweets that we could access, these filters resulted in us having far too few tweets to do any real processing. Thus, we removed our filter-by-keyword process from the pipeline.
- **Time:** We accepted tweets from the date range of 12/01/2012 to 11/15/2017. This appeared to be a large enough time range to ensure that tweets would encompass a wide variety of topics and events. (This also diminished the chance of having an event-specific

dominating label in our data because Twitter sentiments are often heavily influenced by regional events that takes place during specific time periods.) We thought it would be interesting to consider tweets from more recent years, since these are more relevant to our current UCLA experience.

Initially, we built our own crawler using the Twitter module and its Basic Streaming API, which allowed us to generate a sample of potential tweets related to UCLA. However, during the actual process of collecting a massive amount of data (our initial goal was more than 6500 tweets), we realized that the Streaming service, which only allows the retrieval of real-time tweets, is far too insufficient and inefficient for our purpose. (Applying the geolocation filter to our stream made the process especially tedious.) Therefore, we resorted to an alternate open-source crawler, created by Github user Jefferson Henrique named GetOldTweets, for our collection process. GetOldTweets is a python project that utilizes pyQuery to retrieve older tweets. We primarily used its interface, Exporter.py, and additional parameters to specify our search location and time period.

We ended up with 20543 tweets total. However, since we chose to use supervised classification, we had to manually label all of the tweets we wanted to use. As a result, we ended up only actually utilizing the first 4000 of these tweets.

3.2 Preprocessing

Next, we performed preprocessing on the tweets. We removed capitalization and urls. We also removed all punctuation, with the exception of exclamation marks. Additionally, we used a Natural Language Toolkit package to tokenize our tweets. Each word and punctuation mark was counted as an individual token.

For labeling, each team member manually labeled 1000 tweets, for a total of 4000 tweets. Labeling was done according to the standard in the table below. Only tweets labeled either happy or unhappy were used as test or training data in our models. Neutral tweets were discarded since we primarily wanted to detect positive or negative emotion. We also did not want to include empty, incoherent, or indeterminate tweets. After discarding all neutral tweets, we ended up with 1495 happy tweets and 557 unhappy tweets for a total of 2052 tweets.

Label	Meaning	Details
0	Neutral	Includes tweets that were • neutral = no overall positive or negative tone • ambivalent = positive and negative influences canceled out • incoherent = we could not determine the meaning of the message • empty = tweet had no text after preprocessing • indeterminate = tweet could have been either positive or negative depending on the context (e.g. if a tweet was sarcastic), but we did not have enough information to determine the context
1	Happy	Overall happy tone; also includes tweets of amusement portrayed through playful sarcasm
4	Unhappy	Overall unhappy tone

Note that some of the inaccuracies in our models could have come from inaccuracies in our tweet labeling. Although we tried to maintain a standard of labeling among all group members, there was still some individual variation. In addition, it was often difficult to classify the tweets based on text alone, particularly since sarcasm was common and we did not take into account any image data that may have existed.

4. METHODS DESCRIPTION

4.1 Gather Tweets

First, we built a basic Twitter crawler that could collect and filter tweets. However, due to the Streaming API limitations of the first crawler, we had to resort to the aforementioned GetOldTweets crawler for sufficient data. In the CLI, we called the Exporter.py script and fed in the following parameters:

```
--near ' 34.065744 -118.445647' --within 6
```

for our geolocation and radius

```
--since 2012-12-01 --until 2017-11-15
```

for our range of timestamps

Through this, we were able to gather 20543 tweets for our study.

4.2 Preprocess Tweets

Using the ntlk and re modules, we passed each raw tweet through the following pipeline (methods):

1. **Tokenization**: We used NLTK's TreeBankTokenizer to split our tweets into a list of individual meaningful units - tokens.
2. **De-capitalization and Punctuation Removal**: All uppercase letters in each token were transformed into their lowercase constituents, and all punctuation tokens, with the exception of exclamation marks, were removed.
3. **URL Removal**: Because embedded URLs provide little information about the sentiment of the mother tweet, we truncated tweets from the first appearance of an 'http' or 'https' token.
4. **Stop Word Removal** (optional): Lastly, for more robust features, we removed any tokens that were likely to have a high frequency in our data but contribute little to the sentiment of the tweet (ie. determiners such as 'the', 'a', and pronouns such as 'he', 'she', 'it').

4.3 Determine Important Features

We decided to only take into account the text of the tweet. We thought that other features, such as the number of likes, the author's username, and the author's number of followers would not be very influential in determining whether a tweet was positive or negative since this information might not be relevant to the matter, might be similarly distributed between both positive and negative tweets, or could require more information about the user than was available.

For the text features, we looked at the most common unigrams, bigrams, and trigrams for each class of tweets (i.e. happy or unhappy). The tokens from the preprocessed tweets were considered to be unigrams. We also identified these patterns after removing stop words, or commonly used English words. Below are the results and counts of the twenty most common of these patterns in the form (‘pattern’, count).

20 Most Common Unigrams/Bigrams/Trigrams (including stop words):

	In Happy Tweets	In Unhappy Tweets
Unigrams	('!', 591), ('i', 461), ('the', 349), ('to', 300), ('a', 299), ('my', 243), ('you', 222), ('and', 201), ('for', 189), ('is', 156), ('in', 142), ('this', 122), ('of', 122), ('so', 113), ('with', 109), ('me', 104), ('it', 103), ('love', 101), ('hill', 90), ('at', 89)	('i', 251), ('to', 161), ('the', 140), ('my', 114), ('you', 108), ('a', 107), ('and', 93), ('!', 76), ('me', 73), ('it', 67), ('is', 62), ('of', 60), ('so', 57), ('that', 56), ('in', 52), ('on', 48), ('this', 45), ('for', 45), ('have', 42), ('with', 41)
Bigrams	('! !', 180), ('hill high', 63), ('high school', 63), ('i love', 45), ('in the', 37), ('thank you', 34), ('for the', 32), ('and i', 29), ('love you', 23), ('i have', 23), ('to be', 22), ('vista golf', 21), ('to see', 21), ('posted a', 21), ('just posted', 21), ('i got', 20), ('a photo', 20), ('i was', 19), ('casa nieves', 19), ('west palmdale', 18)	('i m', 25), ('! !', 24), ('i have', 19), ('don t', 17), ('i need', 13), ('it s', 12), ('have to', 12), ('going to', 12), ('on my', 11), ('in the', 11), ('i hate', 11), ('and i', 11), ('of the', 10), ('i just', 10), ('i can', 10), ('i am', 10), ('to the', 9), ('to me', 9), ('i was', 9), ('to get', 8)
Trigrams	('! ! !', 63), ('hill high school', 57), ('posted a photo', 19), ('vista golf club', 18), ('vista west palmdale', 17), ('photo hill high', 14), ('a photo hill', 14), ('! hill high', 13), ('http com p', 11), ('who needs a', 7), ('rancho vista golf', 7), ('needs a laugh', 7), ('mom who needs', 7), ('liked a video', 7), ('at rancho vista', 7), ('at godde hill', 7), ('after my own', 7), ('a video http', 7), ('a mom who', 7), ('a laugh !', 7)	('! ! !', 11), ('i have to', 4), ('hanging out with', 4), ('what the fuck', 3), ('i didn t', 3), ('hill high school', 3), ('going to get', 3), ('wish i could', 2), ('will bury the', 2), ('when i was', 2), ('what had happened', 2), ('we let that', 2), ('we have to', 2), ('we don t', 2), ('was in tears', 2), ('up to a', 2), ('u don t', 2), ('too lazy to', 2), ('to their plan', 2), ('to the dmv', 2)

20 Most Common Unigrams/Bigrams/Trigrams (excluding stop words):

	In Happy Tweets	In Unhappy Tweets
Unigrams	('!', 591), ('love', 101), ('hill', 90), ('school', 74), ('high', 72), ('lol', 60), ('good', 59), ('u', 55), ('happy', 55), ('day', 55), ('like', 48), ('today', 45), ('vista', 44), ('time', 41), ('got', 40), ('see', 39), ('http', 39), ('great', 38), ('one', 37), ('thank', 36)	('!', 76), ('like', 41), ('shit', 23), ('get', 23), ('lol', 22), ('fuck', 22), ('got', 21), ('go', 20), ('u', 18), ('need', 18), ('smh', 17), ('really', 17), ('people', 16), ('http', 16), ('hate', 16), ('going', 16), ('time', 15), ('know', 14), ('fucking', 14), ('see', 13)
Bigrams	('! !', 181), ('hill high', 63), ('high school', 63), ('vista golf', 21), ('posted photo', 19), ('casa nieves', 19), ('west palmdale', 18), ('vista west', 18), ('golf club', 18), ('happy birthday', 15), ('! hill', 15), ('photo hill', 14), ('happy thanksgiving', 12), ('http com', 11), ('day !', 11), ('com p', 11), ('today !', 10), ('good morning', 10), ('! vista', 9), ('rancho vista', 8)	('! !', 24), ('feel like', 4), ('moms house', 3), ('lmao smh', 3), ('hill high', 3), ('high school', 3), ('going get', 3), ('wow go', 2), ('wish could', 2), ('wanna know', 2), ('wanna go', 2), ('wanna cry', 2), ('vista golf', 2), ('u back', 2), ('tired lol', 2), ('thought could', 2), ('speed limit', 2), ('smells like', 2), ('sent twice', 2), ('saturday night', 2),
Trigrams	('! ! !', 63), ('hill high school', 57), ('vista golf club', 18), ('vista west palmdale', 17), ('photo hill high', 14), ('! hill high', 13), ('http com p', 11), ('rancho vista golf', 7), ('needs laugh !', 7), ('mom needs laugh', 7), ('! ! hill', 6), ('nieves http com', 5), ('golf club grill', 5), ('! vista west', 5), ('west avenue n', 4), ('thanks follow !', 4), ('casa nieves http', 4), ('! vista golf', 4), ('! rancho vista', 4), ('! casa nieves', 4), ('today ! !', 3)	('! ! !', 11), ('hill high school', 3), ('thought could trust', 2), ('rancho vista golf', 2), ('protests everywhere december', 2), ('plan kill let', 2), ('morning ! facility', 2), ('let happen protests', 2), ('kill let happen', 2), ('happen protests everywhere', 2), ('everywhere december !', 2), ('debating going get', 2), ('bury backlash plan', 2), ('backlash plan kill', 2), ('zoomie snoozle !', 1), ('youtube channel video', 1), ('young put stupid', 1), ('yo directly fell', 1), ('yesterday wrestling aide', 1), ('yelled dad got', 1)

As evident, both happy and unhappy tweets share some common words and patterns. For example, “!!!” is the most common trigram in both happy and unhappy tweets. Note that the counts may be higher for the happy tweets since there were about three times as many happy tweets. Also, note that local events or interests may skew this data. For example, there were a large number of tweets talking about “hill high school,” leading to this pattern being identified as more common than it probably should have been when considered on a more general scale.

4.4 Model Implementation and Evaluation

The details of this step are found in the next section and the conclusion.

5. EXPERIMENT DESIGN AND EVALUATION

5.1 Naïve Bayes

We chose to use the Naïve Bayes model because it is particularly well-suited to text-based supervised classification. Naïve Bayes makes class predictions based on how often words in the text are found in other instances of each class of data. For reasons described earlier, we only used preprocessed tweets that were labeled as either happy or unhappy. We also did not allow any tweets to be classified as neutral. To split our data between training and test data, we used 10-fold cross-validation and averaged the evaluation results. To construct our model, we referred to the model of Vik Paruchuri for a corpus of movie reviews.

5.1.1 Procedure

1. Import training dataset from pipeline.py
2. For each tweet, compute the likelihood of it being of a class through the following:
    a. Compute the frequency of the set of words in the tweet using Counter
    b. For each word in the tweet, predict its likelihood of being in the current class by multiplying the frequency of the word in the tweet with the frequency of the word in the entire class of tweets, then dividing by the sum of the size of the current class dictionary and the number of tweets in that class
    c. Multiply the resulting probability with the probability that a tweet is the current class in the given dataset
3. Based on this likelihood function, the maximum of all class likelihoods will be the resulting class prediction for our tweet

5.1.2 Evaluation

1. We used 10-fold cross-validation. First, (data, label) pairs were randomly partitioned into ten equal-sized folds. Then, for each fold, we ran a test with the fold as test data and the other nine folds as training data. At the end, we averaged the results of all ten tests. Cross-validation allowed us to check the accuracy of the model and ensure that we weren't overfitting.
2. We use sklearn.metrics to compute a variety of evaluation metrics for our classification model. This included AUC (Area under the ROC curve), accuracy, precision, recall, and a confusion matrix, whose elements were used to compute specificity and f-measure.

5.1.3 Results

The results of one of our runs is shown below. Here, the "happy" label was considered to be the "positive" value. Note that since the tweet ordering is randomized, different runs will have slightly different values. However, different runs mostly produced similar results. We ran the test both with stop words included and with stop words taken out.

	Including stop words	Excluding stop words
Average Accuracy	0.615035236833	0.612387674913
Average Recall/Sensitivity	0.539421845388	0.539007416459
Average Precision	0.888108945842	0.882657385176
Average Specificity	0.815758643981	0.811666276435
Average F-measure	0.67070361821	0.667977464125
Average area under ROC curve:	0.677590244684	0.675336846447

We observed a consistently higher precision and specificity, and a lower recall score. This indicated that our model is better at avoiding false positives and worse at avoiding false negatives.

5.2 Linear SVM

Since SVM was used in academic papers as the benchmark, we believed that it was useful to implement it in order to compare its performance to that of our other model. Since the test samples may not be linearly separable with our selection of features, we used linear SVM with soft margin (C = 1.0) to deal with noise points. We formulated the calculation as a QP problem and solved its dual form with cvxopt, a python optimization library. We referenced the class structure in (Blondel, 2010). Again, we only used preprocessed tweets that were labeled as either happy or unhappy. We also did not allow any tweets to be classified as neutral.

5.2.1 Procedure

1. Import training dataset from pipeline.py, with stop words removed.
2. Find common unigrams in positive tweets and negative tweets separately.
3. For each tweet, count the number of tokens that belong to the 30 most frequent positive unigrams and the number of tokens that belong to the 30 most frequent negative unigrams.
4. Use linear SVM with soft margin to fit the features and the labels.

5.2.2 Evaluation

1. We used the same 10-fold cross-validation method for evaluation as the one used for Naive Bayes.
2. We use sklearn.metrics to compute a variety of evaluation metrics for our classification model. The metrics are the same as in the Naive Bayes section.

5.2.3 Results

	Excluding stop words
Average Accuracy	0.739747266706
Average Recall/Sensitivity	0.907523279884
Average Precision	0.773747191732
Average Specificity	0.284458099424
Average F-measure	0.835286148341

Average area under ROC curve:	0.595990689654

We did not include the results when we included stop words because including stop words resulted in a significant drop in performance. Futhermore, adding bigrams and trigrams as features did not improve the performance.

The resulting accuracy, precision, and f-measures indicates that the prediction is correct in most cases. The high recall and the low specificity indicate that this algorithm is good at predicting positive tweets and bad at predicting negative tweets.

5.3 Overall

We mainly used common unigrams and short phrases to classify tweets. It's rather surprising that the count of common unigrams alone seemed to be powerful enough to identify the majority of the tweets correctly, as adding bigrams and trigrams did not improve SVM's performance. Evidently, Naive Bayes performs better in area under the ROC curve and precision, while linear SVM excels in accuracy and recall.

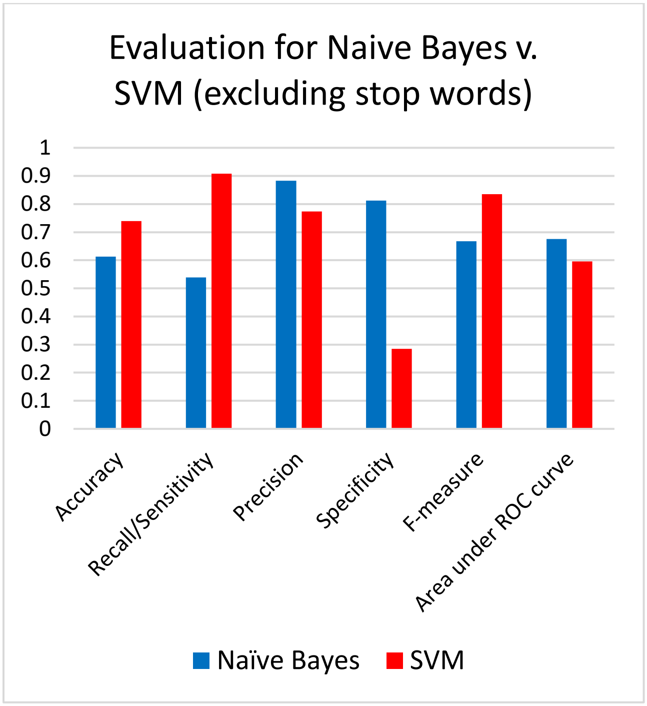

Figure 1. Evaluation for Naive Bayes v. SVM

6. RELATED WORK

The age of social media has sparked many research interests in sentiment analysis in areas such as novels (Mohammad, 2011), e-mail (Mohammad and Yang, 2011), news headlines (Strapparava and Mihalcea, 2008), and suicide notes (Pestian et al., 2011). In our project, we looked specifically at sentiment analysis in tweets, a form of microblogging. The most relevant work to this area is probably that of Robert et al, as we based most of our procedures from that paper. We also acknowledge that there are other related papers in this areas such as those listed in the reference section.

7. CONCLUSION

Although identifying the emotion of a tweet can be a highly subjective task based on context and unique human personalities, our algorithms seemed to be able to classify the majority of tweets correctly with good accuracy and precision. Since our algorithms relied mostly on the counts of common unigrams, it was rather remarkable that short phrases alone were enough to fairly accurately identify the emotion of tweets. We conjecture that this could be due to a similarity between this method and how humans identify tweets when labeling them at a fast pace. It is also possible that our reading of previous academic papers on this topic influenced our perception of how to identify the emotion of tweets. In short, we might have been biased to classify tweets in a certain way. Still, we are very satisfied with the accuracy of our models, since the algorithms seemed to identify the majority of emotions correctly, despite individual variances in tweet labeling

8. REFERENCES

[1] Kirk Roberts et al., *EmpaTweet: Annotating and Detecting Emotions on Twitter.* Proceedings of the Eighth International Conference on Language Resources and Evaluation, 2012, pp. 3806–3813., Istanbul. http://www.hlt.utdallas.edu/%7Ekirk/publications/robertsLREC2012_2.pdf.

[2] Shrivastava, Sameeksha, and Pramod S. Nair. "Mood Prediction On Tweets Using Classification Algorithm." International Journal of Science and Research (IJSR), vol. 4, no. 11, May 2015, pp. 295–299., doi:10.21275/v4i11.nov151169.

[3] Henrique, Jefferson. *Get Old Tweets Programmatically*, (2016). https://github.com/Jefferson-Henrique/GetOldTweets-python.

[4] Blondel, Mathieu. Support Vector Machines, (2010). https://gist.github.com/mblondel/586753.

[5] Non-standard Python libraries used: nltk, numpy, sklearn, cvxopt.

[6] Vik Paruchuri. 2017. Naive bayes: Predicting movie review sentiment. (December 2017). Retrieved December 10, 2017 from https://www.dataquest.io/blog/naive-bayes-tutorial/