Text Analysis of WhatsApp Statuses

Alexandra Keamy (akeamy3@gatech.edu)
Sidharth Parwani (sidhparwani@gatech.edu)
Aaron Adams (aadams61@gatech.edu)

Georgia Institute of Technology

ISYE7406 Data Mining and Statistical Learning

Team101 Final Project Report

Abstract

WhatsApp is one of the most popular messaging apps worldwide. WhatsApp statuses provide a way to share text, photos, and videos with contacts that disappear after 24 hours. This project aims to classify WhatsApp statuses as either sad, angry, or happy. The research questions addressed in this project include investigating the overall properties and characteristics of the data set, assessing the performance of various classification methods in predicting emotion of WhatsApp statuses, and identifying the most important features affecting the prediction of emotional content. To clean the data set, multi-text rows were extracted into individual rows. Non-UTF-8 characters, punctuation, and stop-words were removed. Word clouds were used to visualize the most frequently used keywords in each emotion class, providing valuable insights into the content of the WhatsApp statuses. Various techniques for converting the text column to embeddings were explored, including Count Vectorizer, TFIDF Vectorizer, Universal Sentence Encoder, and Bert Sentence Transformer.

We experimented with various classification algorithms including Logistic Regression, Support Vector Classifiers, Gaussian Naive Bayes, Multinomial Naive Bayes, Bernoulli Naive Bayes, Random Forest, and Neural Networks to determine the most effective approach. Our target variable is the "sentiment" column, and our model uses the vectorized "content" column as a predictor variable to classify whether a WhatsApp Status is happy, sad, or angry. The models were evaluated using accuracy, f1, recall, and precision, and hyperparameters were tuned using GridSearchCV and stratified KFold cross-validation splitting technique. The best-performing model was the Bernoulli Naive Bayes, achieving an accuracy of 73.70% using Count Vectorizer to convert the content column to an embedding. In conclusion, this project developed a robust classification pipeline that allowed us to accurately predict the sentiment of a WhatsApp status message and determine the most important features affecting the prediction of the messages.

Introduction

In an increasingly digitalized world, billions of people everyday use some form of language interfacing technology or software. From using a search engine to perform online research, getting help from a chatbot, or communicating via email or social media platforms, the amount of text data that is generated each day is massively increasing [1]. This data is generally unstructured and requires more preprocessing techniques to make it usable. Text analysis is a growing field that seeks to use this wealth of unstructured data as input into modern machine learning techniques to extract insight and meaning to make data driven decisions.

There are many different types of text analyses. Some of the more popular ones include specific term frequency analysis, document structure analysis, and sentiment analysis [2]. Naturally, both the large amount and content of the data generated by social media platforms make them obvious candidates for sentiment and emotional text analysis applications. As an example of the importance of this field of study, current research is being done to try and predict user depression status by analyzing user's social media posts [3].

WhatsApp is a popular cross-platform, internet-based messaging application that allows users to send messages, make voice and video calls, share files, and more. Owned by Meta, the app boasts over 2 billion users across 180 countries. The app has a feature allowing users to share their "status". This is a text field with a 700-character limit that resets every 24 hours. By their nature, statuses are mainly used to convey one's emotional state on a given day.

However, in their raw form, these statuses can't be used directly in a data mining context. Processing the raw text data into a form that can be input into machine learning models requires additional steps that a more traditional data set doesn't need. First, the text needs to be standardized by removing non utf-8 characters, dealing with emojis, converting to lower case, and removing words with little to no meaning (stop words). At this point the text still can't be used directly in models. Next, features are extracted via vectorization. At this point, models can be built and the focus of the problem shifts to tuning and validating the best model.

The data mining goal of this report is to develop an effective text analysis pipeline that implements modern text preprocessing techniques, feature extraction, and classification model tuning using cross validation techniques. Afterwards, the models will be used both for inference and predictive performance.

Problem Statement and Data Sources

Our data source for this analysis comes from an open source Kaggle competition [4]. The raw data set contains 1592 unique WhatsApp statuses as raw text and a labeled emotion. The emotions represented in this data set are happy, sad, and angry. The process by which these labels were assigned (manually or algorithmically) is not stated. During data exploration, we identified an error in which some rows in the data set contained multiple statuses. After extracting these statuses, we were able to expand the data set by 100 unique angry statuses and 130 sad statuses, bringing the total number unique WhatsApp status texts to 1822, or a 14.5% increase from the raw data. An example of the data structure is given below:

Content	Sentiment
Smile and the world smiles with you.	Happy
Seven billion people on this planet and I have 2 friends	Sad
When life give u lemons, squeeze it in people's eyes ….	Angry

The chart above shows relatively clean text examples, but some of the statuses contained emojis that our text preprocessing steps would need to handle. On top of this, there were a few minor inconsistencies in the data. First, some of the statuses appeared twice in the data set, with two unique emotions assigned to them (Supplemental Figure 1). Second, there were some statuses that seemed to be improperly assigned an emotion (Supplemental Figure 2). These two cases represent a relatively small portion of the data, however it was not feasible to go through every row and check for these cases. As such we left these rows with the originally assigned emotions and acknowledge that this may have a minor impact on model accuracies.

As stated in the introduction, the goal of this analysis is to create optimized machine learning models, using cross validation for selection of hyperparameter. Then, using the best models, address the following inference and predictive objectives of this data set:

1. What are some of the overall properties and characteristics in this data set?
2. How well do various classification methods (Logistic Regression, SVC, Random Forest, Neural Network, etc.) perform in predicting emotion from WhatsApp status?
3. What are the most important features in the data that affect the prediction of emotional content?
4. Apply the classification models to predict the sentiment of new statuses our users generate.

Methodology

The dataset used in this project consists of a collection of WhatsApp statuses labeled as happy, sad, or angry. Our target variable is the "sentiment" column containing labeled data, and our model uses the vectorized "content" column as a predictor variable to classify the sentiment of WhatsApp Statuses. Before training the models, we preprocessed the dataset to remove any noise and improve the quality of the data. We applied several cleaning techniques to the content column, including lowercasing, HTML tag removal, stop word removal, punctuation removal, special- character removal, and lemmatization. These techniques helped to standardize the data and remove any irrelevant information. To transform the text data into a format that can be processed by our chosen classification models, we explored several text embedding techniques. We experimented with Count Vectorizer, TFIDF Vectorizer, Universal Sentence Encoder, and BERT Sentence Transformer. Count Vectorizer is a method that converts text to a vector of token counts. TFIDF Vectorizer converts text to a TF-IDF matrix, which considers the frequency of a term in a document, as well as the frequency of the same term across all documents in the corpus to assign a higher weight to words that are more unique to a specific document. Universal Sentence Encoder and BERT sentence transformer are deep learning models that generate dense embeddings for each text sample. These embedding techniques helped to capture the semantic meaning of the text data and improve the performance of the models.

After preparing our data, we performed multi-class classification using various algorithms. We compared the performance of various classifiers: Logistic Regression, Support Vector Classifiers, Gaussian Naive Bayes, Multinomial Naive Bayes, Bernoulli Naive Bayes, Random Forest, and Neural Networks. These classifiers were chosen because they are commonly used in text classification tasks. Logistic regression is a linear model that is fast to train and interpret but may underfit the data if the relationships between the predictors and the target are complex. We used this as our baseline model. SVC is a more complex model that can capture non-linear relationships but may require more computational resources to train. SVC finds the best hyperplane in a high-dimensional space that separates the classes with the largest margin. Naive bayes is a probabilistic model that is simple and efficient. It calculates the conditional probability of each class and selects the class with the highest probability as the predicted class. The algorithm assumes that the predictor variables are conditionally independent given the class label. We experimented with three different variants of naïve bayes: Bernoulli, Gaussian, and Multinomial. Naïve Bayes is an excellent choice for

our use case due to its simplicity, scalability, effectiveness, and interpretability. Naïve Bayes has been shown to perform well on text classification tasks, often outperforming more complex models. Random Forest is an ensemble method that constructs multiple decision trees and aggregates their predictions to produce a final prediction. Each decision tree is built on a random subset of the input variables and a random subset of training data to help reduce overfitting. Multi-layered perceptron classifier is a neural network model that can capture complex relationships between predictors but may require more data and computational resources to train. It is a feedforward neural network model consisting of multiple layers of nodes. Each neuron receives input from the previous layer and applies a non-linear activation function to produce an output.

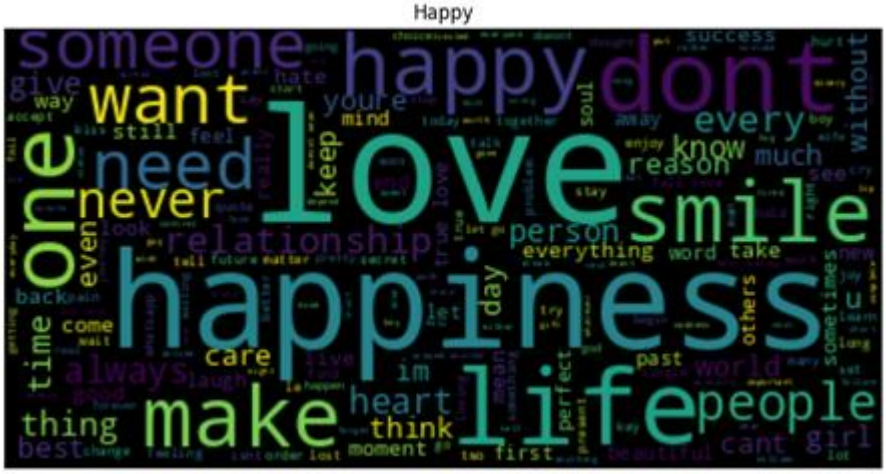
We used grid search CV for hyperparameter tuning of our models. A table of the hyperparameters that we tuned for the final models can be seen in supplemental figure 3. This method helps to find the optimal combination of hyperparameters for each model by searching a range of values. We evaluated the performance of each model using 10-fold cross-validation, which splits the data into 10 equal parts, trains the model on 9 parts, and tests it on the remaining data. This process is repeated 10 times with each part used as the test set once. We used accuracy, precision, recall, and F1-score as performance metrics to evaluate the models. Accuracy measures the percentage of correct predictions, precision measures the proportion of true positives among all positive predictions, recall measures the proportion of true positives among all actual positives, and F1-score is the harmonic mean of precision and recall. The metric we chose to determine the best performing model is accuracy. We chose accuracy as a metric, because we have a balanced dataset meaning we have around the same number of rows for each class. Since we have 3 different classes, accuracy is the most relevant metric because it measures how well the model can classify instances into their respective classes. Accuracy as a metric considers all types of errors and gives equal weight to each class.

Results & Analysis

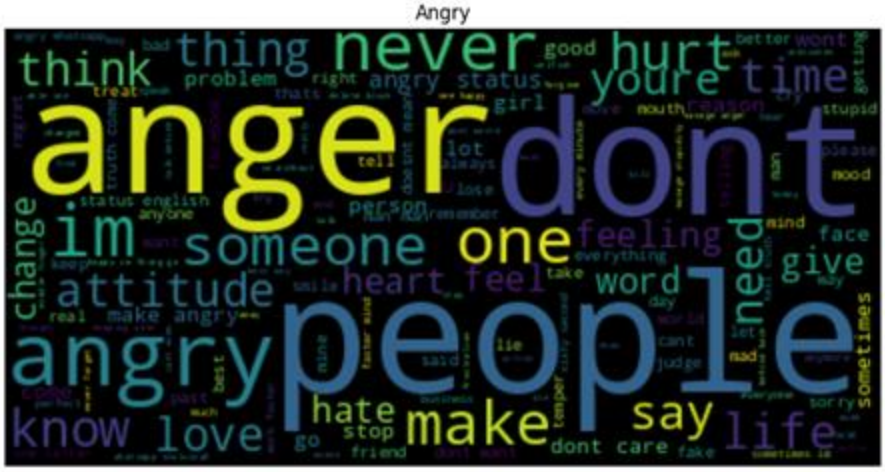
Based on the preprocessing and modeling steps described, we were able to develop a machine learning model for sentiment analysis of WhatsApp statuses. The dataset consisted of statuses labeled as happy, sad, or angry, and we utilized several text preprocessing techniques to clean and standardize the data. We then explored four different text embedding techniques, including Count Vectorizer, TFIDF Vectorizer, Universal Sentence Encoder, and BERT Sentence Transformer, to transform the text data into a format that can be processed by machine learning models.

After data cleaning and preprocessing, we produced word clouds to help us understand the key features of each sentiment. Word clouds are used for visualizing text data by representing the frequency of words in each text through a graphical display. They are often used to identify the most common words and their relative frequencies in a large body of text. Word clouds can be useful for gaining a quick understanding of the most frequently occurring words and themes in a text dataset. Below are the word clouds produced for each sentiment in our dataset.

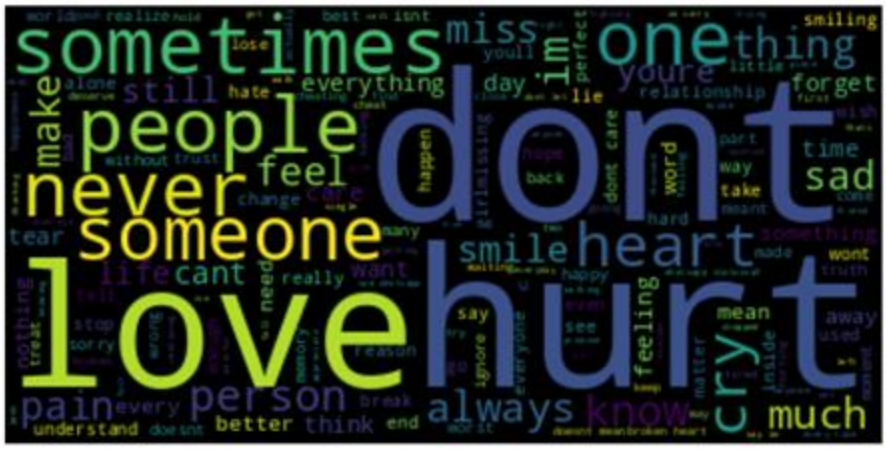
Happy



Angry



Sad



Building off of the word clouds, we also quantified the top 5 important features based on probabilities for each sentiment class. For statuses classified as angry, the most important features were "people", "dont", "angry", "anger", and "im". For statuses classified as sad, the most important features were "dont', "hurt", "people", "love", and "heart". For statuses classified as happy, the most important features were "love", "happiness", "dont", "life", and "one". These important features

reflect the most common and relevant words in each sentiment class that were used to predict the class of each WhatsApp status.

We compared the performance of five different classifiers, including logistic regression, SVC, naive bayes (Bernoulli, Gaussian, and Multinomial), random forest, and multi-layered perceptron classifier. We used grid search CV for hyperparameter tuning and evaluated the performance of each model using 10-fold cross-validation with accuracy, precision, recall, and F1-score as performance metrics.

Based on our evaluation, the Bernoulli Naïve Bayes model achieved the highest accuracy of 73.70% for the classification task. We used Count Vectorizer to convert the 'content' column to an embedding and hyperparameters such as alpha = 0.5 and fit_prior = False were chosen for this model. This model is a probabilistic classifier that assumes the independence of features given the class and calculates the probability of each class for a given sample.

Vectorizer	Model	Accuracy
TF-IDF	Logistic Regression	69.59%
TF-IDF	SVC	67.67%
Universal Sentence Encoder	**Random Forest**	73.15%
Count Vectorizer	**Bernoulli Naïve Bayes**	73.70%
Bert	Neural Network	67.40%

Conclusions

In conclusion, our project aimed to classify WhatsApp statuses based on emotions. We explored various text preprocessing and feature extraction methods to develop a robust classification pipeline and experimented with several classification algorithms to determine the most effective approach for this specific dataset. Our models achieved an accuracy of ~73% using Naïve Bayes classifiers as well as the Random Forest Classifier using Universal Sentence Encoding. However, when fed a deliberately tricky status, our models were split in predicting emotion, highlighting the need for further data gathering and correction of potentially wrongly labeled statuses in the original dataset. On top of fixing some of the mislabeled statuses, future work should also look into added more emotions such as neutral, scared, or silly. Furthermore, looking to expand the model to include other short form social media posts like Tweets could provide a richer data set. Overall, our project

demonstrates the potential of machine learning in classifying emotions in WhatsApp statuses and provides a foundation for future work in this area.

Lessons Learned & Course Recommendations

Overall, this project was a great opportunity to reinforce many of the key takeaways from this course. We needed to find a data set and formulate a research question and then develop a road map to get to an answer. This required working with a real-world data set that needed cleaning and preprocessing. Afterwards, we were able to apply 5 of the classification models we learned this semester for both predictive and inference problems. Using cross-validation and hyperparameter tuning, we were able to get the best performance from our models and feel confident in the results/takeaways they offered.

On top of this, the final project allowed us to explore new topics and expand our skill set. Working with text data was not taught in this course, but our team had an interest in this type of analysis. This project allowed us to take a deeper dive into techniques such as stemming, lemmatization, and vectorization. This project was coded in python, which required members who had previously done the homeworks in R to also learn how to implement the models in python.

Throughout this project, we encountered several challenges and learned valuable lessons regarding text analytics. Some of the key lessons included the importance of data preprocessing, the value of cross-validation, and the challenge of classifying ambiguous text. Data preprocessing was critical for achieving high accuracy. Specifically, removing stop words, stemming the data, and converting text to embeddings were all important steps in improving the performance of our models. Furthermore, we found that cross-validation was an effective way to optimize our model hyperparameters and avoid overfitting. Lastly, we found that some status updates were difficult to classify, even for human annotators. For example, some status updates contained sarcasm or other forms of irony, which made it difficult to determine the intended emotion, which highlights the need for more sophisticated models and techniques that can handle ambiguous text.

The main constructive feedback for the course organization would be to invest in providing sample code/solutions to homework in both R and python. These are both super useful tools and I feel students would greatly benefit from learning how to implement the data mining techniques in both languages.

References

[1] Marr, B. (2018, May 21). How much data do we create every day? the mind-blowing stats everyone should read. Forbes. Retrieved April 12, 2023, from https://www.forbes.com/sites/bernardmarr/2018/05/21/how-much-data-do-we-create-every-day-the-mind-blowing-stats-everyone-should-read/?sh=4d0b8bf60ba9

[2] van Loon A. Three families of automated text analysis. Soc Sci Res. 2022 Nov;108:102798. doi: 10.1016/j.ssresearch.2022.102798. Epub 2022 Oct 1. PMID: 36334926.

[3] Babu NV, Kanaga EGM. Sentiment Analysis in Social Media Data for Depression Detection Using Artificial Intelligence: A Review. SN Comput Sci. 2022;3(1):74. doi: 10.1007/s42979-021-00958-1. Epub 2021 Nov 19. PMID: 34816124; PMCID: PMC8603338.

[4] Mondal, S. S. (2020, July 20). Emotion data for WhatsApp Status. Kaggle. Retrieved April 12, 2023, from https://www.kaggle.com/datasets/sankha1998/emotion

Appendix/Supplemental

	content	sentiment
1041	I still care, that 's the problem	happy
1258	I still care, that 's the problem	sad

Supplemental Fig. 1 – Same status, 2 labels.

```
Content: 204 countries, 805 Islands, 7 seas, 7+ Billion people and I'm still single..
Sentiment: happy
```

Supplemental Fig. 2 – Status and emotion don't seem to match.

Model	Parameters	Values
Logistic Regression	solver	newton-cg, lbfgs, liblinear
	penalty	l2
	C	0.01, 0.1, 1, 10, 100
Bernoulli Naïve Bayes	alpha	0.5, 1.0, 1.5, 2.0, 5
	fit_prior	True, False
Random Forest	n_estimators	100, 500, 1000
	max_features	sqrt, log2, None
Neural Net	alpha	0.00001, 0.0001, 0.001

Supplemental Fig. 3 – Hyperparameter tuning space for final models