Sentiment Artificial Analysis: Analyzing Emotions in Everyday Life

Jackwin Hui (jkh232), John Guo (jg852), Joseph Choi (jc2493)

Cornell University, CS 4701 FA20

ABSTRACT

Today, young adults more than ever struggle with reflecting on the causes of their own emotions. Through this project, we aim to help users understand themselves by providing them with objective information about the way they are responding to events around them. In this paper, we discuss our Sentiment Artificial Analysis (SAI) project, our attempt at solving this problem.

Our goal in this project was to create an interface in which we could analyze a user's journal entry using three separate NLP techniques-- sentiment analysis, event extraction, and causal inference. The goal of sentiment analysis portion of the analysis is help understand what emotions the user is feeling, while the goal of the event extraction portion was to help a user guess what events may have caused those emotions. Finally, the goal of the causal inference portion of the project was to try and provide a more concrete linking between specific events and emotions.

Our final result was a Python-based application. This journal entry is then analyzed for two important pieces of information. First, the application performs a sentence by sentence sentiment analysis, evaluating each sentence as positive or negative. Then, the application performs an open-domain event detection on each sentence. Finally, the program displays both the overall sentiment of the text, as well as highlighting events which were diagnosed both positively and negatively.

1 SENTIMENT ANALYSIS

The goal of the sentiment analysis portion of the project, as above, is to understand and label the emotions in the user's journal entry. In the end, we created a Python model that, given a piece of text, can analyze whether the overall sentiment of that text is positive or negative. In this section, we will detail our path to creating our final product for the sentiment analysis portion of this assignment.

1.1 EmoBank: The First Attempt

Our first attempt at tackling the issue of sentiment analysis was based on EmoBank[1], an open source corpus built in the JULIE lab at Jena University in 2017. This corpus contains ~10,000 labeled pieces of text balancing multiple genres. Each label was in the form of a Valence-Arousal-Dominance label, which we will discuss now.

1.1.1 The Valence-Arousal-Dominance Scheme

[1] https://github.com/JULIELab/EmoBank

The Valence-Arousal-Dominance (VAD) model is a 3D model representing emotions. The Valence axis represents how positive or negative the sentiment is, the Arousal axis represents how excited or relaxed the sentiment is, and the Dominance axis represents how in control the person feels. In the EmoBank corpus, each individual axis (i.e. valence, arousal, dominance) was labeled in the range [1.0, 5.0]. Thus, our corpus's continuous label space was $[1.0, 5.0]^3$.

The ultimate goal of using the EmoBank corpus was to leverage these VAD labels. Rather than the classical positive or negative sentiment analysis, our hope was to gain a deeper understanding of the user's emotions to provide a more sophisticated end product. For example, using the VAD labels, our product could distinguish a joyful emotion from a surprised one. On the other hand, a traditional sentiment analysis would label both of these emotions as "positive".

Below is a table showing which emotions we assigned to which VAD labels. Note that + represents a value in the range [3.0, 5.0], while - represents a value in the range [1.0, 3.0].

Valence	Arousal	Dominance	Emotion
+	+	+	Joyful
+	+	-	Surprised
+	-	+	Satisfied
+	-	-	Protected
-	+	+	Angry
-	+	-	Fear
-	-	+	Unconcerned
-	-	-	Sad

1.1.2 Our Approach and Results

Here, we discuss the approach that we took to train and test our model. First, we took the text and preprocessed it, performing tokenization (by word), lemmatization, and removing stop words. Then, we preprocessed the VAD labels by mapping values to the emotion given the table above. We first trained our model using NLTK's Naive Bayes classifier, but we later swapped this out for scikit-learn's Naive Bayes classifier, which we found to be objectively better. Finally, we evaluated our model using k-fold

cross-validation (k=10). In the end, we found that, with scikit-learn's Naive Bayes model, we found an accuracy of around ~27%.

1.1.3 Issues and Obstacles

Though our model did perform better than just guessing an emotion label out of 8 (27% versus 12.5%), we decided that this was not good enough for our end result. We identified two main pain points which caused us to eventually scrap this attempt at the project.

The first and most important reason we left this corpus behind is the small amount of data. While 10,000 pieces of labeled text sounded promising at first, it was hard for us to train and evaluate an accurate model, especially with such a diverse label space. For reference, the Twitter database we used in our final product (which will be discussed in the next section), contains 1.5 million labeled tweets-- which is 150 times as large as the EmoBank dataset.

Second, we had a lot of trouble handling the VAD labels datawise. To begin with, our simplistic approach of assigning 8 discrete emotion labels to a continuous space definitely is not perfect, as it did not account for how far along the axis each label was, but rather only which side of positive and negative it was on. However, other analysis attempts, such as mapping emotions to their nearest of Ekman's 6 basic emotions also performed poorly (this approach ended up mapping too many emotions to the same discrete label). Also, there was the question of whether to treat the 3 separate labels as independent or not-- do we know whether valence influences dominance? In the end, these difficulties ended with us looking in a new direction for the sentiment analysis portion of our project.

1.2 Twitter: The Final Result

After deciding to shift from the EmoBank dataset, we found a dataset taken from tweets containing around 1.6 million data points (tweets). I will now walk through the various approaches we took to do sentiment analysis for this new dataset.

1.2.1 Initial Approach

Much of our approach was based off of a blog post titled, "Another Twitter Sentiment Analysis" by Ricky Kim.[2] The goal of this sentiment analysis was to determine whether a sentence could be

[2] https://towardsdatascience.com/another-twitter-sentiment-analysis-with-python-part-11-cnn-word2vec-41f5e28eda74

analyzed as generally positive or negative. The large dataset we had allowed for much more robust testing which will be described in the results section.

1.2.2 Vectorizers: Count vs. TF-IDF[3]

In order to use text in machine learning, the text must be converted to some form of numerical representation. One vital consideration regards the way the vocabulary is to be determined. If every new word the machine encounters is treated as a separate feature, there would be too many features for even small data sets. When possible, it is ideal to combine words with similar semantics and thus reduce the need to take into account esoteric words. The number of features was limited to 80,000 based on recommendations from Ricky Kim's post. Another consideration is whether to include stop words in the vocabulary. Stop words are words such as, "to", "a", "the", that do not have much inherent meaning, but appear frequently in sentences. Accuracy was tested on removing stop words and keeping stop words. The built-in stop words from the SK Learn module were used.

Count vectorizers count the appearance of the words in each text. In a dataset as large as the Twitter dataset we used (~1.5 million tweets), the number of words counted must be reduced to have a reasonable amount of words in the vector.[4]

Example:

- "Once as a joke, James threw something at a clown."
- "That clown told a funny joke."
- "Please Jim. Please, please, please?"

	Once	as	a	joke	James	threw	something	at	clown	that	told	funny	please	Jim
Doc 1	1	1	2	1	1	1	1	1	1					
Doc 2			1	1					1	1	1	1		
Doc 3													4	1

Accuracy with stop words (Count Vectorizer):

```
  n_iter_i = _check_optimize_result(
Creating scikit-learn pipeline...
Training + testing classifier...
Created a classifier with accuracy of 0.7972998371131437
```

Accuracy without stop words Count Vectorizer):

```
  n_iter_i = _check_optimize_result(
Creating scikit-learn pipeline...
Training + testing classifier...
Created a classifier with accuracy of 0.771801779225661
```

[3] https://towardsdatascience.com/another-twitter-sentiment-analysis-with-python-part-5-50b4e87d9bdd

[4] https://towardsdatascience.com/another-twitter-sentiment-analysis-with-python-part-4-count-vectorizer-b3f4944e51b5

TF-IDF stands for Term Frequency-Inverse Document Frequency and is another way to turn textual data into numerical form. The vector value it yields is the product of the two terms TF and IDF. Relative TF and IDF are calculated by the following equations.

$$TF(t,d) = \frac{number\ of\ times\ term(t)\ appears\ in\ document(d)}{total\ number\ of\ terms\ in\ document(d)}$$

$$IDF(t,D) = \log\left(\frac{total\ number\ of\ documents(D)}{number\ of\ documents\ with\ the\ term(t)\ in\ it}\right)$$

$$TFIDF(t,d,D) = TF(t,d) \cdot IDF(t,D)$$

A TFIDF vector value for a word will be close to 0 if it appears often in all documents. This means that the word is not significant in differentiating the meaning between sentences.

Accuracy with stop words (TF-IDF Vectorizer):

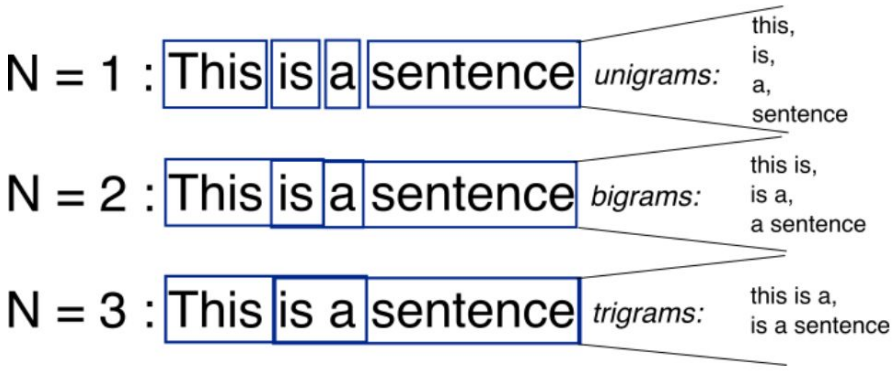
```
  n_iter_i = _check_optimize_result(
Creating scikit-learn pipeline...
Training + testing classifier...
Created a classifier with accuracy of 0.7992106252349329
```

Accuracy without stop words (TF-IDF Vectorizer):

```
  n_iter_i = _check_optimize_result(
Creating scikit-learn pipeline...
Training + testing classifier...
Created a classifier with accuracy of 0.7741511088835985
```

1.2.3 N-grams: Bigram vs. Trigrams

From Wikipedia, "n-gram is a continuous sequence of n items from a given sequence of text or speech." n-grams are all combinations of adjacent words or letters of length n that you can find in your source text. An example of generating n-grams is shown below.



[5]

[5] https://towardsdatascience.com/another-twitter-sentiment-analysis-with-python-part-4-count-vectorizer-b3f4944e51b5

Accuracy with tfidf vectorizer (bigram)

```
Test@DESKTOP-6H1D55P MINGW64 ~/Documents/GitHub/sentiment_artificial_analysis (main)
$ python sentiment_analysis.py
C:\Python39\lib\site-packages\sklearn\linear_model\_logistic.py:760: ConvergenceWarning: lbfgs failed to con
verge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
Creating scikit-learn pipeline...
Training + testing classifier...
Created a classifier with accuracy of 0.8229231925823831
```

Accuracy with tfidf vectorizer (trigram)

```
Test@DESKTOP-6H1D55P MINGW64 ~/Documents/GitHub/sentiment_artificial_analysis (main)
$ python sentiment_analysis.py
C:\Python39\lib\site-packages\sklearn\linear_model\_logistic.py:760: ConvergenceWarning: lbfgs failed to con
verge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
Creating scikit-learn pipeline...
Training + testing classifier...
Created a classifier with accuracy of 0.82530384663576
```

1.2.4 Classifiers: Naive Bayes vs. Logistic Regression

The first classifier we tried was the Naive Bayes classifier. The classifier is “naive” because it assumes that features of a measurement are independent of each other. This is often not the case in real world measurements. NB asks the question, “Give these features, does this measurement belong to class A or B?” The question is answered by “taking the proportion of all previous measurements with the same features belonging to class A multiplied by the proportion of all measurements in class A.“ The same calculation is computed for class B and the larger of the two computations determines the class of the measurement. The results of the Naive Bayes classifier from the scikit-learn library gave us an accuracy of 80.0% using the TF-IDF vectorizer and trigrams.

```
Test@DESKTOP-6H1D55P MINGW64 ~/Documents/GitHub/sentiment_artificial_analysis (main)
$ python sentiment_analysis.py
Creating scikit-learn pipeline...
Training + testing classifier...
Created a classifier with accuracy of 0.800150357098108
```

The second and final classifier we used was the Logistic Regression classifier. Logistic regression is often used when the target variable is categorical. In the case of our sentiment analysis, it was ideal as the classifier needed to decide whether the sentiment was positive (1) or negative (0). The data is fit to a linear regression model and then a logistic function predicts the target variable. A threshold or decision boundary is set to place the estimated probabilities into either of the two classes. The results of the

Logistic Regression classifier from the scikit-learn library gave us an accuracy of 82.5% using the TF-IDF vectorizer and trigrams.

```
Test@DESKTOP-6H1D55P MINGW64 ~/Documents/GitHub/sentiment_artificial_analysis (main)
$ python sentiment_analysis.py
C:\Python39\lib\site-packages\sklearn\linear_model\_logistic.py:760: ConvergenceWarning: lbfgs failed to con
verge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
Creating scikit-learn pipeline...
Training + testing classifier...
Created a classifier with accuracy of 0.82530384663576
```

1.2.5 Final Summary of the Results for Sentiment Analysis

Sentiment Analysis Parameters	Percent Accuracy
Count Vectorizer with stop words (unigram)	79.7
TFIDF Vectorizer with stop words (unigram)	79.9
Count Vectorizer without stop words (unigram)	77.2
TFIDF Vectorizer without stop words (unigram)	77.4
TFIDF Vectorizer with stop words (bigram)	82.3
TFIDF Vectorizer with stop words (trigram)	82.5

2 EVENT EXTRACTION

The goal of the event extraction portion of the project was as follows: given a sentence from the journal entry, analyze what events that a user experienced that could explain their positive or negative emotion towards said event. Our final event extraction product involves an open-source library called GiveMe5W1H, which identifies answers in a text file to the typical “5W1H” questions-- who, what, when, where, why, and how. In this section, we will discuss this software in-depth, as well as our thought process and research in reaching our final product.

2.1 Initial Research and Obstacles

In the research for this potion of the project, we used a 2019 survey of event extraction[6] techniques at the time. In reading this paper, we discovered that a common open-domain event extraction solution is an event extractor. Thus, we aimed to solve our event extraction problem by creating a simple two-part pipeline. First, we would have a pre-processing portion of the pipeline, where commonly used techniques such as tokenization, named-entity recognition (NER), and part-of-speech tagging (POS) would be applied to the text. Then, we would use a supervised machine learning algorithm to train an "Event Extractor", a model which could both detect trigger words, as well as assign text and entities to roles.

With this plan in mind, we quickly ran into a few problems, specifically regarding the corpuses typically used for this task. I discuss these problems below.

2.1.1 Problematic Corpuses

When planning a supervised machine learning task, the most important step was identifying a source of data, i.e. a corpus. However, for event extraction, this task was far more difficult than for sentiment analysis before. First, mentioned in the 2019 survey of event extraction were a few common corpuses used to tackle these tasks-- the ACE Event corpus, the TAC-KBP corpus, and the TDT corpus. However, each of these corpuses were problematic in their own way.

Table 7 ACE05 Event Types and Subtypes

Types	Subtype
Life	Be-Born, Marry, Divorce, Injure, Die
Movement	Transport
Transaction	Transfer-Ownership, Transfer-Money
Business	Start-Org, Merge-Org, Declare-Bankruptcy, End-Org
Conflict	Attack, Demonstrate
Contact	Meet, Phone-Write
Personnel	Start-Position, End-Position, Nominate, Elect
Justice	Arrest-Jail, Release-Parole, Trial-Hearing, Charge-Indict, Sue, Convict, Sentence, Fine, Execute, Extradite, Acquit, Appeal, Pardon

First, the ACE and TAC-KBP corpuses are both based off of the 8 event types and 33 subtypes predefined in the ACE corpus. This means that these corpuses are designed for closed-domain event detection. This would be reasonable for our project, except that in the context of journal entries, many, if not all, of these event types are completely irrelevant to our context. The figure to the right shows the ACE corpus's event types, and a quick look should explain why we determined these events were mostly irrelevant to our project.

Additionally, many of these corpuses were provided by the LDC, which, while being a very reputable source for corpuses, is quite expensive (in the range of thousands of dollars just to access) and designed for more intense use cases than our personal project. There was only

[6] https://www.researchgate.net/publication/337638438_A_Survey_of_Event_Extraction_From_Text

one corpus that I could find on LDC for a reasonable price (35$), but with such a limited selection and the reason described above, we decided not to pursue this path. The prices shown are the LDC's prices for accessing the 2014 TAC-KBP corpus.

Finally, touching base on the TDT corpus mentioned in the article, this corpus is a corpus of stories and topics, with labels of YES, BRIEF, and NO, depending on how much of the article discussed the given topic. Given our goal of identifying possible events which could have triggered positive or negative emotions, this seemed irrelevant.

2.1.2 Based on News Articles

This is more of a minor difficulty compared to our difficulty in finding a reasonable corpus, but something we noticed with event extraction is that the majority of work and literature in the field is based on news articles. This meant that, for much of our work, there would be slight discrepancies in the results, given that most extractors and corpuses were designed for news articles. You can even see this in the corpuses mentioned above-- the ACE event type corpus is really designed around major news articles, including event types such as Life-Divorce, Business-Merge-Org, and Justice-Arrest-Jail, which are not really event types found in day to day life.

2.2 GiveMe5W1H

As stated above, GiveMe5W1H is an open-source open-domain event extraction software aimed at answering the "5W1H" questions-- who, what, when, where, why, and how. This software was developed in 2019 by Felix Hamborg at the University of Wuppertal, Germany. The major NLP software utilized in this open-source project is Stanford's CoreNLP.

The reason why we chose to use this open source software in our final application is because it's event extraction software was very similar to what we had originally planned, but with additional features that we never even planned for.

2.2.1 Preprocessing

Just like we had originally planned with our data, the GiveMe5W1H software also does basic preprocessing on the text data, using CoreNLP. This includes tokenization, lemmatization, POS-tagging, NER recognition, among other processes. However, the GiveMe5W1H does another non-trivial process in

the preprocessing step that we never thought to do, which is to use the YAGO knowledge base[7] to do semantic lookup for persons and organizations. This allows the software to recognize semantic relationships between words, as it will map similar words to the same concept.

2.2.2 Event Extractors

Just like our plan, the GiveMe5W1H software trains extractors. However, instead of training one extractor to identify events, as we had initially planned, this trains 4 separate extractors. First, it trains an action extractor. The goal of this extractor is to figure out "who did what" in the main event of the text, which is the most important extractor for our use case, as well as the main extractor we use in our project. GiveMe5W1H also contains 3 other trained extractors, which I will talk about briefly-- an environment extractor (for 'when' and 'where' questions), a cause extractor (for 'why' questions), and a method extractor (for 'how' questions).

However, one issue that GiveMe5W1H does not address is that it still uses news articles, which is not exactly our use case. In this case, the GiveMe5W1H classifiers are trained off of a dataset of 100 news articles from 13 major US and UK news outlets, with an equal amount of news articles in the categories of business, entertainment, politics, sports, and technology. This dataset was customly made for this project, as they hired graduate students to personally do the annotations. Ultimately, we decided that the goals of this program were general enough (i.e. given a piece of text, what are the most reasonable answers to the 5W1H questions) that we were comfortable using a news-based model on our journal-based project.

Finally, when training the model, 80% of the data was used for training, and 20% was used for testing. As in a typical supervised machine learning problem, the goal of training was to minimize error. In this case, error was measured using WMD, which is a generic measure of the semantic similarity of two phrases. WMD is certainly an interesting technology, but for the most part, this training process is pretty similar to what we had planned.

2.2.3 Final Remarks

In the end, GiveMe5W1H was a good solution for our problem. Though it is based on news articles from the U.S. and UK, we believe that the question it is trying to solve is broad enough that it can be applied to our use case. We performed some basic tests on dummy journal entries and found that the software was, with reasonable accuracy, extracting events from text as we intended.

[7] https://www.mpi-inf.mpg.de/departments/databases-and-information-systems/research/yago-naga/yago

3 CAUSAL INFERENCE

The goal of the causal inference portion of the project is to determine if a cause-effect relationship exists between the extracted events and information (who, what, where). From such causal relationships, a user would be able to establish if there are certain recurring events happening in their daily lives and whether those events may be linked to certain moods. While sound in theory, producing true causal explanations requires capabilities not currently available with modern AI. We faced numerous problems, specifically with training the data to understand and ascertain which events from journal entries were causes and which were effects, finding effective and efficient reasoning and explanations for why one event may cause another event.

3.1 Research

3.1.1 Granger Causality

Granger causality is a probabilistic statistical hypothesis test, finding patterns of correlation between two time series events. Two time series events are considered to be Granger caused if event X "causes" another event Y to improve its values as the values of X change. The issue with Granger causality is that it does not determine true cause-and-effect relationship, but rather if one time series event occurs before the other time series event occurs. This assumes that the cause occurs before the effect and can correctly predict the effect in a causal relationship. With the given event extractions from our project, this is not always the case and thus, the null hypothesis given from a Granger causality that independent time series X does not Granger cause independent time series Y cannot always be rejected.

3.1.2 Feature extraction from events

From the events that we have extracted and the who, what, where information, we must further extract three features: the N-gram words, the related topics, and the sentiments. From the previous parts of the project, we have already extracted two of the three features, N-gram words and sentiments. However, the obstacle that arises is connecting different topics obtained from event extractions and understanding how they are causally related. Creating a causal graph of tuples with relations, cause, and effect from the extracted features is difficult as we cannot necessarily discern such data given the information we have previously extracted.

3.2 DoWhy

The extraction of causation from data is a complicated process and one that still requires a lot of research in the AI field. Two variables X and Y can be said to be correlated such that P(X|Y) can be found, but in statistics, correlation does not mean causation, as two variables can be correlated but which causes the other cannot be determined. To bridge the gap between correlation and causation, Microsoft has created the DoWhy library, named after the related probability function P(X|do(Y)) given two variables X and Y. DoWhy is a well-developed library as it also takes into consideration confounding variables in statistical data, which is important to take note of when analyzing causal relationships. However, the DoWhy library is limited in its data application, and we found it difficult to bridge the gap between our event extraction and the statistical data that the DoWhy library takes in.

3.3 Obstacles

While there has been some progressment in causal inference methods in AI software, it continues to be a relatively unexplored and difficult aspect of AI. Right now, most causal inference software such as DoWhy focus on narrow examples of data and are often unable to fully understand causal relationships. While from a statistical vantage, we can find causal interpretations of data, it is still unclear how to find causal relationships in general sets of information, which is needed for our project. Moreover, it is hard to account for confounding variables, especially with the extracted events we currently have obtained. Hopefully as general artificial intelligence continues to be explored and the understanding of how the brain creates cause-and-effect relationships are furthered, causal relationships will develop and become available to use.

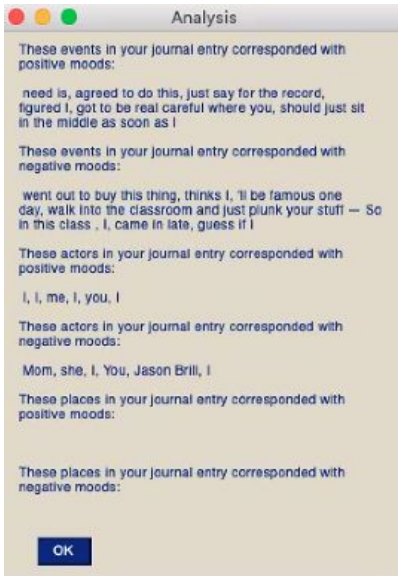
4 CONCLUSION

4.1 Final functionality

While we were unable to accomplish many of the features for the final result, we learned much about the current landscape of AI and the technology that exists. Combining what we learned about sentiment analysis and using the 5W1H library, we created a text editor that allows the user to analyze their journal entries to see what percentage of their sentences were positive or negative and correlates positive and negative sentiment in relation to what they were doing, who they were doing it with and where they were doing it. Due to the limitations of the sentiment analysis classifier and the 5W1H library, there was often false or mislabeled information. However, it was able to reliably measure the overall

sentiment of a journal entry to 82.5% accuracy, and we were able to use the 5W1H library for some interesting applications in the form of five buttons.

- **Analyze Text:** Groups events based on positive and negative sentiment and displays the results on a popup window



- **Analyze Sentiment:** Calculates the percentage of positive sentiment sentences and displays the results on a popup window
- **Analyze What, Who, Where:** Highlights in green and red the respective “questions”

4.2 Future Steps

One of the clear limitations that we saw was the existence of both positive and negative sentiments in complex and even simple sentences. Currently, our program does not handle this well. Another consideration is the addition of a "Neutral" label for sentiment in addition to positive and negative. This would become helpful in not cluttering the data, thereby not skewing the results, with sentences that do not hold significant sentiment value.

Another improvement would be to allow for more complex emotion labeling apart from positive and negative. This would require significantly more corpuses with data that includes more complicated emotions.

Finally, the initial goal of this project was to have an AI that continues to learn from the user in a way that compounds journal entries in order to give better and more refined analysis. This was ultimately difficult to do because of the lack of data we had to test on. The future of this project will depend heavily on interdisciplinary cooperation with clinical psychologists, neuroscientists, philosophers and many others to gather the data sets necessary to create an accurate machine learning classifier that could analyze sentiment at a meaningful level.

REFERENCES

- Sven Buechel and Udo Hahn. 2017. EmoBank: Studying the Impact of Annotation Perspective and Representation Format on Dimensional Emotion Analysis. In EACL 2017 - Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics. Valencia, Spain, April 3-7, 2017. Volume 2, Short Papers, pages 578-585. Available: http://aclweb.org/anthology/E17-2092
- Ricky Kim. 2018. Personal project on sentiment analysis using twitter datasets. Available: https://towardsdatascience.com/another-twitter-sentiment-analysis-with-python-part-11-cnn-word2vec-41f5e28eda74
- Hamborg, Felix and Breitinger, Corinna and Gipp, Bela. 2019. Giveme5W1H: A Universal System for Extracting Main Events from News Articles. Copenhagen, Denmark. Sept 2019. Available: http://www.gipp.com/wp-content/papercite-data/pdf/hamborg2019b.pdf
- Wei, Xiang & Wang, Bang. (2019). A Survey of Event Extraction From Text. IEEE Access. PP. 1-1. 10.1109/ACCESS.2019.2956831. Available: https://www.researchgate.net/publication/337638438_A_Survey_of_Event_Extraction_From_Text
- Dongyeop Kang, Varun Gangal, Ang Lu, Zheng Chen, & Eduard Hovy. (2017). Detecting and Explaining Causes From Text For a Time Series Event. Available: http://www.cs.cmu.edu/~dongyeok/papers/emnlp17_explain.pdf