Sentiment Analysis on First Person Narratives

Harsh Karia
hkaria@ucsc.edu
University of California, Santa Cruz
Santa Cruz, CA

Bryan Tor
btor@ucsc.edu
University of California, Santa Cruz
Santa Cruz, CA

Sahithi Narla
snarla@ucsc.edu
University of California, Santa Cruz
Santa Cruz, CA

ABSTRACT

Sentiment analysis is a growing application of machine learning. In this report, we employ a variety of methods to classify text as having positive or negative sentiment. Specifically, we analyze a data corpus of first person narratives. We chose to use a variety of existing baseline algorithms, such as SentiWord, Naive Bayes, Logistic Regression. We also opted to use a Word2Vec implementation that is passed through a recurrent neural network, and also applied a bidirectional LSTM recurrent neural network. In addition, we utilized Google Colab to run our models.

1 MOTIVATION AND OBJECTIVE

Individuals writing informally from the first-person perspective typically express emotions indirectly by describing situations from which their audience will infer a general sentiment, instead of explicitly stating the emotion that the individual is feeling. In 2017, the Natural Language and Dialogue Systems Lab (NLDS Lab) of UC Santa Cruz created a model to predict the sentiments of first-person narratives by learning lexio-functional patterns. The goal of our project was to come up with an accurate model that could replicate similar results achieved by the NLDS model without using the AutoSlog-TS pattern learner.

2 DATASET

We used UCSC's Natural Language and Dialogue Systems Lab's collection of sentences written in the first person perspective. The dataset was already split in to a training, test, and dev set with each sentence already labeled with a positive or negative sentiment.

2.1 Data Preprocessing

First, we exported the individual text files in our provided dataset to a single CSV file using a Python script. Then, we chose to read in our data using a Pandas dataframe. Our group had considered directly using a NumPy array. However, we decided to stick with Pandas because we found pandas to be a sufficient wrapper over any num2py operations we would want to execute.

2.2 Data Cleaning

We removed extra whitespaces and punctuation from our data. We removed punctuation because people use punctuation differently online (grammar, emojis, code), there would be too many variations in uses that would make punctuation a meaningless feature.

2.3 Removing Stopwords and Lemmatization

In regards to removing stopwords: stopwords are words that typically do not hold meaningful information for our model. Examples include: "the, a, is." Finally, we lemmatized our data to obtain the base form of words ("bought" and "buys" would become "buy"). In order to feed data to our recurrent neural network, lemmatization allows us to reduce each word into its "lemma" form. For example, "been had done languages cities mice" would become "be have do language city mouse" as most terms can be reduced to a similar "lemma" form. This helped us reduce the number of different words our model would have to account for, as words with the same base form would be reduced to just that base form.

2.4 Data Split

The original NLDS lab data had an existing split of training, validation, and test data provided. The data was split into a training set made up of 46,255 positive and 25,069 negative sentences for a total of 71,324 sentences, a testing set made up of 1,266 positive and 1,440 negative sentences for a total of 2,706 sentences, and a development set composed of 498 positive and 754 negative sentences. In the NLDS model, the validation set was used to tune the parameters of an AutoSlog-TS pattern learner. Because we were not taking a similar approach, we combined the training set with the validation set. Furthermore, we were unconfident in how the NLDS split their data between the training set and testing set, given that the ratio between the training set and testing set was about 3240 : 120. Therefore, we concluded that it was best to evaluate our model in two ways:

- Train off of the training set and test off of the testing set.
- Merge the training and test sets, and split the resulting data in a 66-33 split for the training and test set, respectively.

id	sentence	sentiment
50f70518d7eed88dd8f1de63_2.txt	I remember having an odd dream, but not the whole thing.	0
5101cabcd7eeef2893914552_31.txt	ride my bike maybe?	1
5344221be4b0e2a60cea917d_215.txt	In contrast to what has been written here I am a very optimistic person.	0
50f7476fd7eed88dd8f5072c_11.txt	I can only say and understand a few things, especially the Sono gemeli?	1
50ff4686d7ee253f31855b6c_7.txt	We're basically going to use photoshop and dreamweaver to build a website.	1
53441681e4b0e2a60ce40581_185.txt	I needed you more than anyone else needed you because... news flash...	0
50ff8a69d7ee253f3188413f_11.txt	I don't get all the Oasis haters out there.	1
51030986d7eeef2893967966_8.txt	i think alonso, and massa are good, though massa lost cause of the stupid automated lollipop man and that his team of r	1
50f65fafd7eed88dd8eceee2_1.txt	I suppose people go about securing their critical needs differently when a hurricane threatens.	1
51016475d7eeef28938c62b0_16.txt	u tell me u find out last minute about the sia stuff so u couldnt tell me i go out with my friends and u talk	0

Figure 1: Sentences with sentiments associated with them.

3 MODELS AND ALGORITHMS

3.1 SentiWordNet

The first model we used was a basic SentiWordNet, where we assigned a positive or negative value based off of the average sentiment of each word of the sentence according to SentiWordNet. We used this model to obtain a baseline set of metrics to compare to our other results.

3.2 Multinomial Naive Bayes Classification

Another model we used was a multinomial Naive Bayes Classification, where we used the Bayes classification algorithm from ***sckit − learn***. Naive Bayes is a supervised learning approach for

NLP tasks, assumes that features are independent from one another in accordance with the Bayes Rule. To test this, we used a bag of words (BoW) model, which simply converted the occurences of each word into a matrix. Second, we implemented TF-IDF vectorizer with Bayes Classification.

3.3 Word2Vec

We also used Word2Vec, where we obtained word embeddings of our vocabulary using the model, and then processed that through a densely-connected neural network layer.

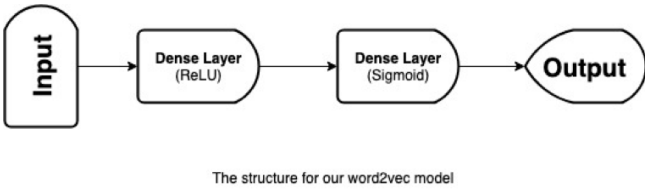

Figure 2: The structure for our Word2Vec model.

3.4 Recurrent Neural Network

Recurrent Neural Networks use the concept of memory to store data throughout time. We used the **Keras** deep learning library for this implementation. An embedding layer is utilized to encode the input into integer form, to account for the vanishing gradient problem, we used a bidirectional LSTM (Long Short-Term Memory) cell, which is composed of 100 neurons that decide how the embedded words indicate positive or negative sentiment. Then, we decided to use sigmoid as our last-layer activation function because it was well suited for binary classification, and because we had previously used a sigmoid function in our logistic regression model and wanted to be better able to compare the results between models. The network had a dropout rate of 0.2 to prevent overfitting, and had a validation split of 0.2.

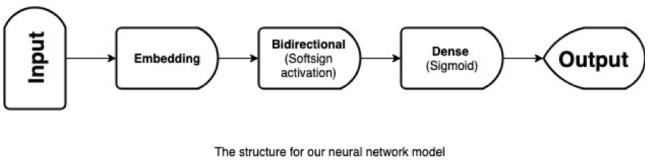

Figure 3: The structure for our recurrent neural network model.

4 RESULTS AND ANALYSIS

As seen in Table 1, our SentiWordNet approach had the worst accuracy at 36.8%. Our Naive Bayes Bag of Words model had the best accuracy at 76.1%. Our Naive Bayes TF-IDf model had the best F1-score at 83.03%. We also noticed that our LSTM model had similar results for accuracy, F1-Score, precision, and recall.

As seen in Table 2, the evaluation results for the SentiWordNet are the exact same as in Table 1 because of how it was generated.

Table 1: Evaluation Results from Merge Set

	Accuracy	F1-Score	Precision	Recall
SentiWordNet	0.3680	0.5382	0.6601	0.4543
Native Bayes (BoW)	0.7605	0.8195	0.8043	0.8353
Naive Bayes (TF-IDF)	0.7531	0.8303	0.7511	0.9280
Logistic Regression	0.7452	0.8094	0.7915	0.8282
Word2Vec	0.6910	0.7882	0.7055	0.9012
LSTM	0.7469	0.7469	0.7459	0.7469

Table 2: Evaluation Results from Original Train and Test Sets

	Accuracy	F1-Score	Precision	Recall
SentiWordNet	0.3680	0.5382	0.6601	0.4543
Native Bayes (BoW)	0.6703	0.6864	0.6198	0.7702
Naive Bayes (TF-IDF)	0.6314	0.6985	0.5655	0.9178
Logistic Regression	0.6603	0.6882	0.6029	0.7704
Word2Vec	0.5674	0.3831	0.4706	0.3248
LSTM	0.6330	0.6331	0.6332	0.6333

We can see that the Naive Bayes Bag of Words model still has the best accuracy at 67.03% and the Naive Bayes TF-IDF model still has the best F1-score at around 69.85%.

4.1 SentiWordNet

The SentiWordNet model consistently had the worst performance results. We expected this. Because the model obtained its sentiment values from SentiWordNet, it was not making classifications that would be specific towards the data.

4.2 Logistic Regression

Our Logistic Regression model had an average performance compared to the other models. Assessing the 10 most informative negative and positive features of the model, terms that appeared frequently in sentences containing a specific sentiment would be assigned a weight associated with that similar sentiment, but there was no clear order in how these weights were assigned. For example, the term “mother” appeared in 472 sentences with a negative sentiment, but was assigned a less significant weight than the term “grandmother,” which appeared in 186 sentences with a negative sentiment. The ratio of negative to positive sentence appearances for a term does not seem to have affected the order either. For example, the term “dentist” has a ratio of approximately 13 negative sentence appearances for every positive sentence appearance, but is more highly weighted than the word “casket,” which has a ratio of approximately 47 negative appearances for every positive appearance. When tested against sentences containing negation, such as “not happy” or “not upset,” the Logistic Regression model would typically misclassify the corresponding sentiment. Because our Logistic Regression model cannot make inferences based off of the structure of a sentence, the model cannot train for these kinds of negations.

-3.5061	grandmother	3.9776	ride
-3.4825	hate	3.7838	enjoyed
-3.3296	dentist	3.7705	coaster
-3.1087	casket	3.7092	great
-3.0050	funeral	3.6058	fun
-2.9483	died	3.4519	headed
-2.9352	mother	3.4205	camera
-2.9149	relationship	3.3774	park
-2.7686	angry	3.3642	water
-2.7373	exam	3.2725	awesome

Figure 4: Our logistic regression model results.

-11.5172	0000	-5.4411 tohunker
-11.5172	0000025	-5.6094 futon
-11.5172	01500	-5.6126 whooped
-11.5172	06	-5.6222 daze
-11.5172	1000pm	-5.7001 grandchilderen
-11.5172	1015am	-5.7163 recover
-11.5172	10200	-5.7930 gp
-11.5172	1022	-5.8147 linkln
-11.5172	1025	-5.8469 orchestra
-11.5172	1030am	-5.8960 gro

Figure 5: Results of Multinomial Bayes Classification

4.3 Multinomial Naive Bayes Classification

Our Multinomial Naive Bayes Classification model had the best performance compared to the other models. Assessing the 10 most informative and least informative features of the model when sentences have been vectorized through a Bag of Words, it was unclear how the most informative features terms were selected. The least informative features are all numbers, which should not hold any subjective sentiment values. The most informative feature, "tohunker" appeared only once in the dataset, as being a typo of the words "to hunker." The second most informative feature, "futon" appeared three times throughout the data set, but the fifth and seventh most informative features, "grandchilderen" and "linkln" respectively, appeared only once, suggesting no correlation between the frequency of a term and it's relative importance.

-11.5172	0000025	-5.4411 time
-11.5172	008080	-5.6094 fun
-11.5172	02	-5.6126 went
-11.5172	07	-5.6222 day
-11.5172	100am	-5.7001 got
-11.5172	1020	-5.7163 really
-11.5172	1025	-5.7930 good
-11.5172	1027	-5.8147 like
-11.5172	102rose	-5.8469 one
-11.5172	103rd	-5.8960 great

Figure 6: Results of Multinomial Bayes Classification

Assessing the 10 most informative and least informative features of the model when sentences have been vectorized by TF-IDF, all of the 10 most informative features appeared in more positive sentences than negative ones. It is unclear on how informative features have been ranked, as the features are not ordered by frequency. For example, the term "time" is ranked as the most informative feature, but it appears about half as frequently than the term "day," the fourth most informative feature. Similar to the Bag of Words vectorized model, the 10 least informative features once again are all numbers.

Similar to the Logistic Regression model, when tested against sentences containing negation, such as "not happy" or "not upset," the Naive Bayes model would typically misclassify the corresponding sentiment. Because our Naive Bayes model cannot make inferences based off of the structure of a sentence, the model cannot train for these kinds of negations.

4.4 Word2Vec

Figure 7: Results of Word2Vec model.

The Word2Vec model had the second to last worst performance out of all of the models. The basic dense layers were not robust enough to build a good model for our dataset, especially as the model began to overfit after 5 or so epochs. However, our group had not been expecting to get high performing metrics. The main point of this model was to see if the Word2Vec model should be pursued. In the end, however, the group decided to stick with the Keras Embedding layer.

4.5 Recurrent Neural Network

The Recurrent Neural Network had similar metrics for its Accuracy, F1-score, Recall, and Precision, even when trained on the merged and original data sets. Furthermore, the model did not appear to be overfitting after training it for 150 epochs. There is a possibility that the RNN could be improved if it was trained for more epochs.

5 CONTRIBUTIONS

We collaborated on the algorithms and the reports. Our group met with Marcelo to discuss different approaches we could use to test our model. We read various papers to get ideas on approaches to use. We all retrained the models to get better accuracies. Bryan wrote the SentiWordNet model and the Logistic Regression model. Harsh worked on the Bayes Bag of Words model. Bryan worked out that we needed to split the data and Sahithi wrote the code

to split the data for one of our approaches. We all worked on the poster presentation and the final report.

6 FUTURE WORK

In future approaches, we are hoping to balance the data better and train the LSTM for more than 150 epochs. We are also hoping to use WordNet-Affect to analyze emotions instead of just sentiments. Our group is also hoping to be able to create a webpage where a user would be able to input a sentence and get an output of what the sentiment of the sentence is. Another thing we are hoping to accomplish is being able to detect more than positive and negative emotions. We would be able to classify emotions such as anger, depression, happiness, and sadness.