Mindset in Writing

New Mexico

Supercomputing Challenge

Final Report

April 7, 2021

Team 19

La Cueva High School

Team Members:

Edward Potapov

Joel Gates

Teachers:

Ms. Yolanda Lozano

Project Mentor:

Dennis Trujillo

Executive Summary:

Our project uses natural language processing to analyze large groups of texts for the sentiment. When given a large sample of texts, a prediction of general sentiment is helpful in predicting the popular opinion of a prompt. Using the Python programming languages and the spaCy library, we were able to train a neural network by interfacing the spaCy library to learn from a sample of many polarized reviews from the Stanford Very Large Movie Review Dataset used to train natural language processing AI's for similar tasks. We then evaluated the predicted (by the AI) score of responses collected from a poll to evaluate the effectiveness of the neural network. We would then go through the data ourselves and evaluate them based on their sentiment. Our goal was to check how effective the model was, checking responses based on how we would evaluate them. We would also nullify repeat answers, answers that didn't have an opinion, and random strings that were imputed. After training our model with 20 thousand polarized reviews, we would test the responses obtained from the poll. We were working with CSV files and Pandas to easily check and add new data to a CSV that we would generate. After we personally evaluate the data, we would run our model on it and check how many were correct on the basis of our evaluations.

Project Statement:

For college and class essays, as well as reflections on specific topics, reading and interpreting essays can be time-consuming. We were wondering if there was an easy and effective way to get the tone or sentiment to an answer to a question. This could be used to gauge a class's opinion on a certain subject without wasting a minute of the teacher's time.

Description:

We used an NLP (Natural Language Processing) Library, spaCy, that specializes in production-ready products and is good at evaluating large sets of samples. Since the quality of written text can vary, the use of a neural network is necessary. Our code would read in all the data that we obtained from the Movie Dataset and divide it up into a training and testing dataset. We would divide it up with the ratio of 4/1, so roughly 80% of the specified data was used for training while the rest or 20% was used for testing. The list of entries would also have a label that categorized it as either positive or negative. Right before we divided up the data, it was important to set the seed to 0 because the results were worse without it.

In the spaCy library, we would set up our pipes or classifiers as the text categorizer while removing all other pipes and adding positive and negative as the labels used. As we updated the model with the data, we would also test it by checking for true positives, false positives, true negatives, and false negatives to display the precision, recall, and f-score. These are used in addition to the loss to tell us how well the model is performing with the data as it goes through the training. After it is done training, it generates the model artifact file that we would reuse to evaluate different pieces of writing.

The neural network that we used for the project was CNN (Convolutional Neural Network) which is a commonly used neural network for natural language processing. The neural network was implemented in the spaCy library, so we did not have to build a neural network from the ground up to process the data.

The responses were responses collected from a Google Form which asked two questions. One was “What is your opinion on expanding federally implemented universal health care?”, this question was intended to be the more complicated question that would elicit a more nuanced

response. The other was “What are your thoughts on pineapple on pizza?” which is a very simple yes or no question, do they like it or do they not, that the writer could express in any way, shape or form. The responses were sourced with the help of our computer science teacher Mrs. Yolanda Lozano, who was nice enough to send our form around to both her students and other staff members in La Cueva High School to respond to.

Then we would take the responses in the form of a CSV file and use Pandas to evaluate them right in a duplicate of the file. We first would have to evaluate the writing ourselves and then let the model evaluate it. We personally would add two columns with our opinions and then let the script add two columns that would be the results of the model. The program would go through each entry and analyze them and evaluate them, and put that information into the results column. Pandas were really helpful because we could specify how it would read in the CSV file and how it would spit it out. At the end of the process, we would have a test output with our own evaluations on it.

Verifying Model:

To verify the capabilities of our model, we, as previously said, evaluated the responses ourselves. To be more specific, we looked at the responses to see if they were positive or negative and specifically addressing situations where they were displayed as such. With the Google Form, we got a lot of repeated entries and responses that weren’t direct answers to the question. Answers that were neutral were thrown out because we wanted to test in accordance with an actual class or college essay where you would have to state your claim on a subject. We also had to get rid of some answers because they were random strings of numbers that didn’t

occur often. So to address those responses, we put N/A in our evaluation columns for the affected responses.

To score our model, we went through each response and compared it to our evaluations. Results that were in accordance with our Evaluations would get one point out of the number of responses, and N/A evaluations would decrease the number the score was out of—not giving it a point and decreasing the number of testable responses to make sure that we had the model analyzing the ones we wanted to be analyzed. The program would then give us a score of the different questions.

Results:

Examples of CSV Data, 3 out of 121 entries:

Health Care Question	Pineapple on pizza Question	Q1 Evaluation	Q2 Evaluation	Q1 Model Result	Q2 Model result
I am in favor for expanding universal healthcare.	I throughly enjoy pineapple pizza	Positive	Positive	Negative	Positive
I approve	Disgusting	Positive	Negative	Positive	Negative
I support federally implemented universal health care, people have a right to be able to survive.	Pineapple is good on pizza.	Positive	Positive	Positive	Positive

Results: Question 1: 63/108 Question 2: 73/99

These were the scores when comparing the results from the model to what we evaluated. We had to get rid of repeated answers, answers that didn't have an opinion, and random strings that were imputed.

```
Beginning training
Loss    Precision       Recall  F-score
Training iteration 0
14.880345505895093      0.8673726676708655      0.8431372548978278      0.8550832711864027
Training iteration 1
0.1805236021973542      0.8791374122323012      0.8593137254859838      0.8691125433769503
Training iteration 2
0.08358318791215424     0.8852130325770166      0.8656862745055605      0.8753407682732325
Training iteration 3
0.07044280704394623     0.8861629048042877      0.8852941176427191      0.8857282981810412
Training iteration 4
0.06364498856055434     0.8899355478389196      0.8799019607800005      0.8848903130348292
Training iteration 5
0.04973771890036005     0.8863748155391719      0.8833333333290033      0.884851460835331
Training iteration 6
0.045834960956426585    0.8846908734009584      0.8838235294074323      0.8842569887156241
Training iteration 7
0.03778043760019045     0.8871442590731741      0.8862745097995771      0.8867091711579858
Training iteration 8
0.03496804056737801     0.8837890624956847      0.8872549019564351      0.8855185909937109
Training iteration 9
0.030480689138187245    0.8889432485279406      0.8906862745054379      0.8898139079290411
Training iteration 10
0.02922336872438791     0.8870029097920127      0.8965686274465855      0.8917601170117417
Training iteration 11
0.02530517086256623     0.8881322957155247      0.8950980392112986      0.8916015624956465
Training iteration 12
0.023658611755603687    0.8854924793746459      0.8946078431328696      0.8900268227218239
Training iteration 13
0.022361659436530346    0.8849129593767655      0.8970588235250144      0.8909444985350977
Training iteration 14
0.02168842341245636     0.8844852585747488      0.8970588235250144      0.8907276709618364
Training iteration 15
0.019519418275321776    0.887159533069615       0.8941176470544406      0.8906249999956513
Training iteration 16
0.020349686088382057    0.8857696030934862      0.8970588235250144      0.8913784705265885
Training iteration 17
0.017983669600448948    0.8852300242087883      0.8960784313681566      0.8906211936619216
Training iteration 18
0.015234774563754439    0.8811594202855983      0.8941176470544406      0.8875912408715934
Training iteration 19
0.016584919919956143    0.8817567567525012      0.8955882352897275      0.8886186770384795
Training Time: 1 hours 58 minutes 48.5633 seconds
Predicted sentiment: Positive   Score: 0.9531373977661133
Review test text:
    This movie is the greatest movie I have ever seen. It was better than all the other movies which I
have seen.

Predicted sentiment: Positive   Score: 0.9979650974273682
Review test text:
    I really just thought this movie was bad. It was the worst movie I have ever seen!

Predicted sentiment: Negative   Score: 0.9999545812606812
Review test text:
    This move was okay. I didn't love it, but I also didn't hate it. Pretty mediocre.

Predicted sentiment: Negative   Score: 0.9999545812606812
Results: Question 1: 63/108 Question 2: 73/99
```

Conclusions:

After analyzing our output, we came to several conclusions. First of all, our results were decent with the limitations of the questions we asked. The first question, in particular, could go multiple different ways. In a lot of different cases we saw when people tried to elaborate the pros and cons of the subject, the model would guess the sentiment of which one the person focused on, which could be the opposite from what the person was thinking. The second question pointed out the shortcomings of our model, with some simple questions being deemed negative when they actually weren't. Overall the model proved to be more effective than evaluating them at random but still worse than when a human evaluates them. On average, our model was true. The shortcomings of the project could come in different forms. It could be the lack of data that we could find for this specific task (It was relatively good at evaluating reviews) or our inability to train with datasets that were several times larger. For the most part, it would struggle with the more developed (in addition to the grammatically and spelling error filled) responses and would breeze through the simple answers with, of course, a few hiccups along the way.

Looking back at the initial question of how effective this method is, shows that it is definitely possible to do it with a pretty good efficacy rating, which means that getting a gist of the class opinion is definitely possible, but it might be a little harder for scoring essays. Advanced customizations to the neural network provided by spaCy would most likely aid in language processing and could lead to more refined/developed results. Nonetheless, our model could be a useful item in a teacher's toolbox for quickly determining an estimate of how many people were in favor of one choice (a type of assignment for example) without directly posing the question or in an open ended response.

Achievements:

For both of us, learning more about neural networks and machine learning is an achievement in itself. We are also proud of the fact that we could train a model that was relatively effective in evaluating sentiment. We understand in a lot of different cases, it's hard for the computer to make sense of its surroundings and understand the difference between subtle differences in written language that can even be hard for people to easily understand. With the neural network that was provided by the library, we were able to achieve generally good results, but in future projects we would like to learn more about neural networks. Nonetheless, the effectiveness of the model was an accomplishment as we had no experience with neural networks or natural language processing before beginning this project.

Recommendations:

Initially we had to narrow our research from plagiarism and mindset to just mindset because plagiarism detectors are quite common and it isn't worth it to reinvent the wheel.

Resources:

spaCy - The Natural Language Processing Tool that was used to train and use a language model.

Anaconda - The python data science kit that we used that had all the necessary tools for us.

VScode - Edward's Preferred Text Editor

Spyder - Joel's Preferred Text Editor

Real Python. (2020, November 26). Use Sentiment Analysis With Python to Classify Movie Reviews. Retrieved December 10, 2020, from https://realpython.com/sentiment-analysis-python/

Grayson, Siobhán, et al. "Novel2Vec: Characterising 19th Century Fiction via Word Embeddings." Academia.edu, University College Dublin, www.academia.edu/33141616/Novel2Vec_Characterising_19th_Century_Fiction_via_Word_Embeddings.

Maas, A. (n.d.). Large movie review dataset. Retrieved April 11, 2021, fro https://ai.stanford.edu/~amaas/data/sentiment/

Install spacy · spacy usage documentation. (n.d.). Retrieved April 11, 2021, from https://spacy.io/usage

Acknowledgments:

_____In the beginning of the project, Dennis Trujillo advised us as to which direction we should go with the project and provided us resources where we began our resources from. We would also like to acknowledge and thank our computer science teacher, Ms. Lozano, for encouraging her students to fill out our form so we could have many responses to analyse.