Final Project CSE 156 Report

Team 26

May.4th /2019

Section 1. Classifier Explanation

1.1 Implementation

For the implementation of the first part, we picked the same implementation as the Programming Assignment 2. The following are the changes that we made. We decided to use the TF-IDF Vectorizer which counts term frequency. This vectorizer counts the number of appearances over a certain length of the text, then TF-IDF Vectorizer uses inverse document frequency to make the weight of the words more proportional and appropriate. The formula for the term frequency and the inverse document frequency formula are here below

$$TF(word,\ sequence) = \frac{number\ of\ word\ appearances\ in\ this\ sequnce}{number\ of\ words\ in\ this\ sequence}$$

$$IDF(word) = log\{\frac{N = number\ of\ sequences}{|d \in D : t \in d| = number\ of\ sequence\ of\ words\ where\ the\ word\ appeared}\}$$

The CountVectorizer and the TF-IDF Vectorizer all have pre-built-in tokenizers. To improve the performance of the classifier, I imported a new tokenizer and a new stemmer from the natural language toolkit (nltk). RegexpTokenizer is used to split the string into regular expressions and I chose to use the default parameter. Then after having all the words in tokens, I need to reduce the number of words with a stemmer. A stemmer is used to remove all the prefixes and affixes of the word and leaving the stem of the word for the machine to process. The SnowballStemmer parameter is set to “English” to allow the pre-built in stemmer to process English. The implementation of the tokenize method will pass in a sentence and the tokenizer along with the stemmer will produce a word; then this word will be checked to see if it is in the middle_word array. The goal is to remove all the words that are ambiguous among classifying to positive or negative reviews. We also use logistic regression**.** One of the fundamental idea to train the logistic regression is to alternate other parameters on the logistic regression function. One of the common practices is tuning the C-value which is the inverse of the regularization strength. I picked C value from [0.01, 0.05, 0.1, 0.15, 0.5, 1, 5, 10, 100, 200, 300, 400, 500] and used the GridSearch to pick the best estimator and set that to be the classifier. The rest of the parameters, I stick with the same as the basic model of the logistic regression. We also selected a batch of middle words which we think they are ambiguous and they cannot be used to do emotion classification. The middle words also have to be selected partially manually, partially by the algorithm based on my implementation. In the end, middle_words is set to be ['and', 'a', 'the', 'am', 'it', 'me', 'with','in','on', 'by', 'near', 'this', 'that', 'an', 'there', 'here', 'those']. Lastly, different language model will generate a different result. Tuning the n-gram parameter in the TF-IDF Vectorizer will generate different results. I set the parameter to be 1, 2 and 3. This means that three different kinds of language models, the unigram model, bigram model, and trigram model will be tested. After tuning, I found out that the unigram model performs poorly because it is taken each word

singularly and did not improve the accuracy at all. The trigram model overfits the dataset. Thus, the bigram model performs the best.

1.2 Examples

Over-Confident Examples:

real answer is: 1
prediction is: 0
We decided to try hash house a go go after reading several positive yelp reviews.We were a little put off when we got to the Imperial Palace location at approximately
[[0.94357441 0.05642559]]
Document id: 12
real answer is: 1
prediction is: 0
Probability(NEGATIVE) = 0.9435744073176413
True class: POSITIVE
[('little', 0.12911499260629666), ('reviews', -0.09530269263594832), ('decided', -0.09503545040244372), ('positive', -0.08299368239504416), ('when', -0.08264266470180585), ('a', 0.0648885361960974)]

real answer is: 1
prediction is: 0
Over all i love the luxor. The rooms are nice and reasonably priced. The only thing bad I have to say is they need to clean the elevators.
[[0.95985091 0.04014909]]
Document id: 15
real answer is: 1
prediction is: 0
Probability(NEGATIVE) = 0.9598509141324436
True class: POSITIVE
[('to', -0.20098989926338703), ('love', 0.20051895499818623), ('bad', -0.16884195271065372), ('The', -0.13638938795125477), ('rooms', -0.11996364856596183), ('reasonably', 0.08983854922453459)]

Screenshots

Example 1:

```
Enter your sentence: This is a beautiful place with good service


This is a beautiful place with good service
[[0.00351242 0.99648758]]
prediction is:  1
Probability(NEGATIVE) = 0.0035124197416790404
[('is', 0.21519148669440044), ('good', 0.15965135634918867), ('service', -0.07214677384021684), ('beautiful', 0.06991388053723965), ('with', 0.05042448538987358), ('a', 0.04220119246071098)]
Enter your sentence: I thought this is a good place based on the review, but I was wrong


I thought this is a good place based on the review, but I was wrong
[[0.62943337 0.37056663]]
prediction is:  0
Probability(NEGATIVE) = 0.6294333681037536
[('is', 0.2996765377655219 4), ('review', -0.20213537224081357), ('thought', -0.17694173224364299), ('this', 0.1483087318539466), ('I', -0.13067271449106121), ('place', 0.12011730263654194)]
```

Example 2:

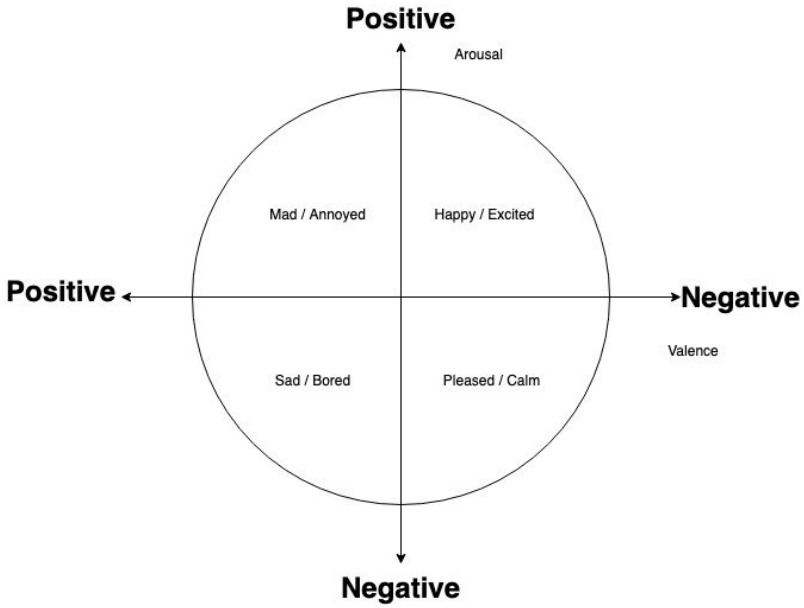
```
 12
We decided to try hash house a go go after reading several positive yelp reviews.We were a little put off when we got to the Imperial Palace location at approximately
[[0.94357441 0.05642559]]
Document id: 12
real answer is:  1
prediction is:  0
Probability(NEGATIVE) = 0.9435744073176413
True class: POSITIVE
[('little', 0.1235281064216365), ('reviews', -0.10000365195366478), ('decided', -0.09011164689458707), ('positive', -0.08244415124082866), ('when', -0.07569176259080428), ('house', 0.06582228777069747)]

 15
Over all i love the luxor.  The rooms are nice and reasonably priced. The only thing bad I have to say is they need to clean the elevators.
[[0.95985091 0.04014909]]
Document id: 15
real answer is:  1
prediction is:  0
Probability(NEGATIVE) = 0.9598509141324436
True class: POSITIVE
[('love', 0.19748023670498324), ('to', -0.19647069585611301), ('bad', -0.1709883843984569), ('The', -0.13592529875721637), ('rooms', -0.11902182296266646), ('nice', 0.09192769262437034)]
```

Section 2. Valence and Arousal Classifier

2.1 Motivation

This part of the final project is inspired by emotional classification. With the use of logistic regression, it is rather convenient to identify binary cases; for example, the classifier can detect whether the text is happy or not happy. In this case, the ambiguity is rather significant because there are still many other types of emotions other than happy and not happy. Our team, therefore, came up with the idea of doing two binary classifications regarding the input text data. The first classification is detecting the text's valence level. Valence can be interpreted as the subject's pleasant or unpleasant experience regarding the aspect or the topic of the text. If the text is positive in valence, that means the user who inputs the text is having a positive or pleasant attitude towards the subject matter; on the other hand, if the text is negative in valence, this means that the user is displeased or annoyed by the material. The other classification is regarding the arousal level. Arousal means the intensity of the stimulus to the subject. Positive arousal means that the subject matter might boost more



adrenaline or higher blood pressure; negative arousal can be interpreted as tired, calm or other similar emotions. In all, using these two classifications, we can do a multi-class classification. We improved from the traditional binary emotion classification to the better quaternary emotion classification. The classification could be interpreted as the graph here.

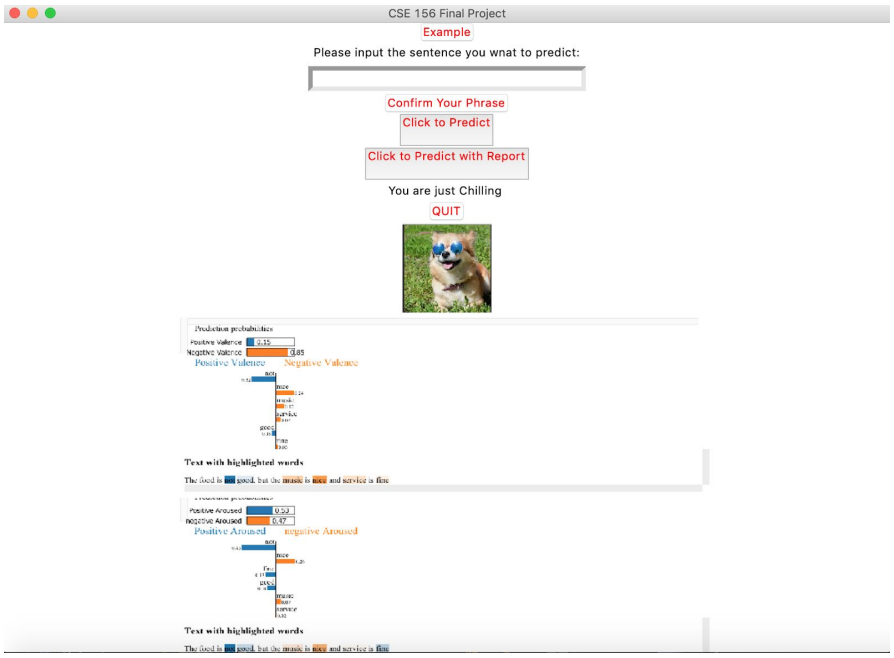
2.2 Implementation and Working Progress

To implement the model above we trained multiple classifiers. We trained bagging classifier with logistic regression, random logistic regression classifier, and the simple logistic regression classifier. The first two actually computes a bit higher accuracy than the simple logistic regression classifier with our testing data, but the first two also requires a significant amount of computing power. Thus, we think the logistic regression classifier will work the best in this case.

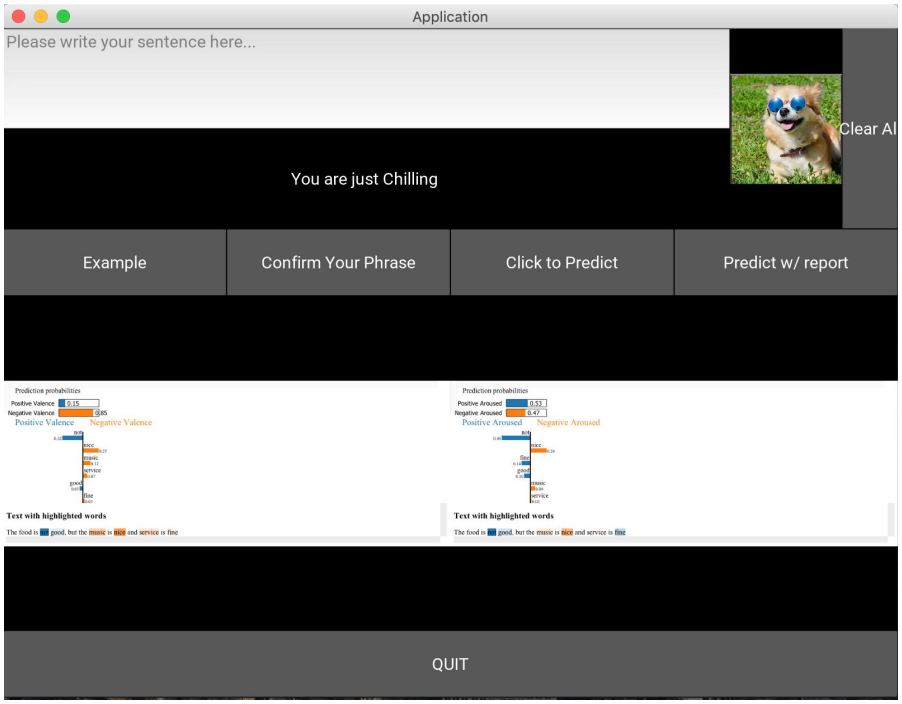
We picked two datasets online. Both of them are texts with emotion-related data. The first one is in the format of one sentence following a valence value and an arousal value both which are classified with a threshold of 3. If the valence or arousal is above 3, then it is positive; if it's below 3 then it's negative. 3, in this case, is neutral and I classify them into positive for binary classification. The second file is more complicated than the first. The second file is in the format of one sentence and followed by an actual emotion. I followed the Valence-arousal dimension model and asserted the valence and arousal value by hand. Different from the first file, if the text is marked with'Neutral' emotion, my algorithm chose to ignore the text. After loading the data, the training process starts with logistic regression and grid search. Majority of the parameter tuning process is the same as part 1 above. I tried different n-gram language models and a different set of ambiguous words. In the end, the trained logistic regression should be the most time efficient with reasonable accuracy result.

To finish the project and showcase our predictions, we also tried multiple python GUI toolbox. We attempted both Tkinter and Kivy. For the project, we decided to have Tkinter as the demo and Kivy as the final product. The following screenshots are the Tkinter and Kivy GUI.

Tkinter GUI screenshot:

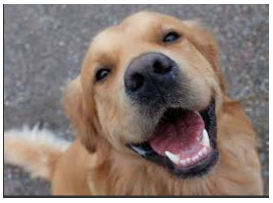


Kivy GUI screenshot:



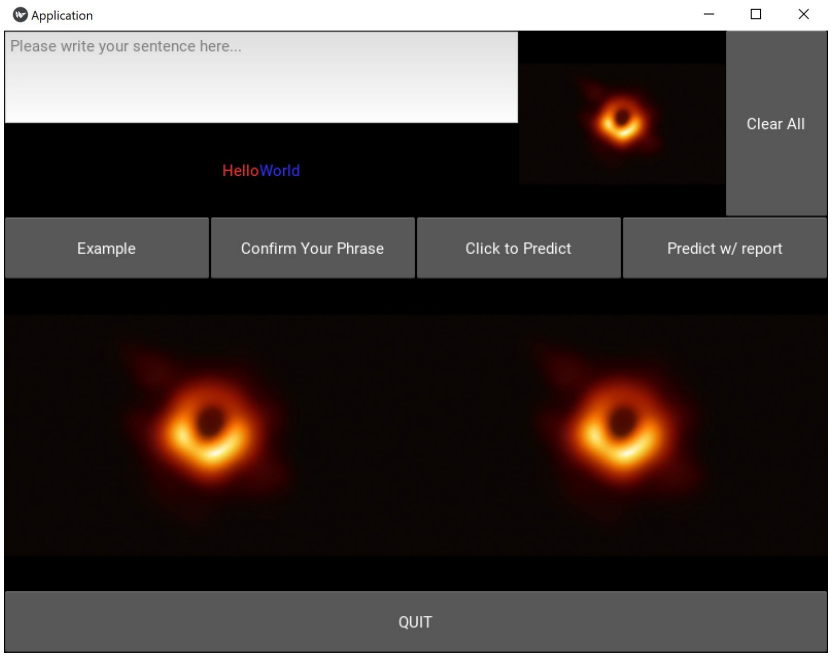
To use the program, the user may click example and see what kind of outputs will be generated from the program. The program will classify the text's emotion into four categories: happy, chilling, mad and sad. To input a new sentence into the program, the user should simply type into the white text box. After inputting the sentence, please click the button 'Confirm Your Phrase'. This way, the application will update in the back that this sentence is going to be predicted. Then, click on the 'Click to Predict' and the application/program will generate the result in the black section with a sentence description and a corresponding picture. 'Predict w/ Report' will generate a prediction and two pictures explaining why the input phrase is predicted that way. To quit the program, please click the red button in the top left corner.

The table below is the four picture representation of the emotion categories.

 Happy	 Chilling
 Sad	 Angry

The Following are Screenshot Examples

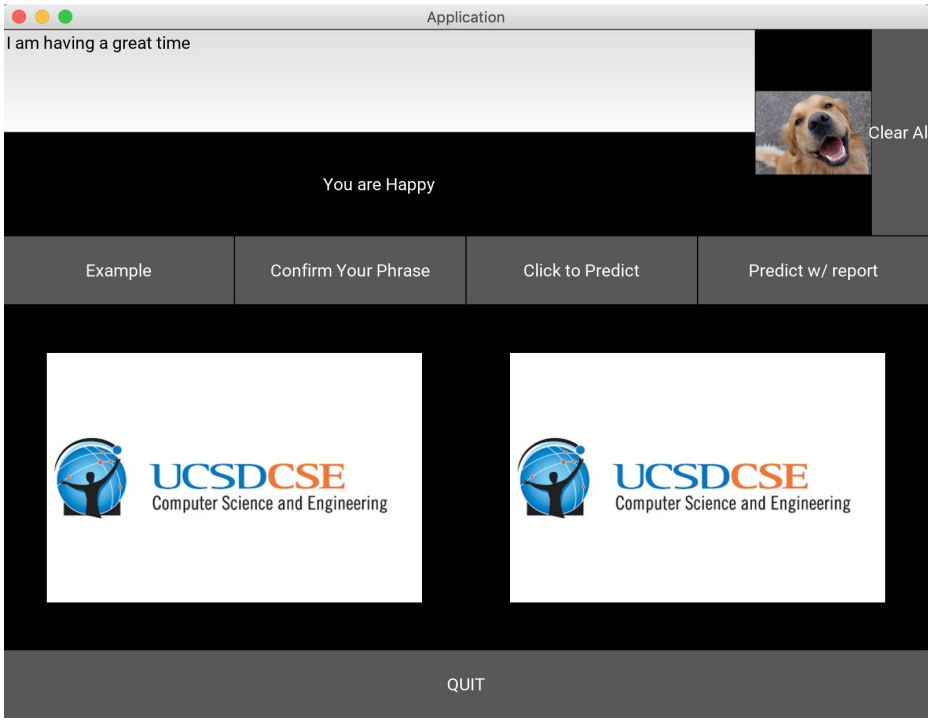
Initial State:



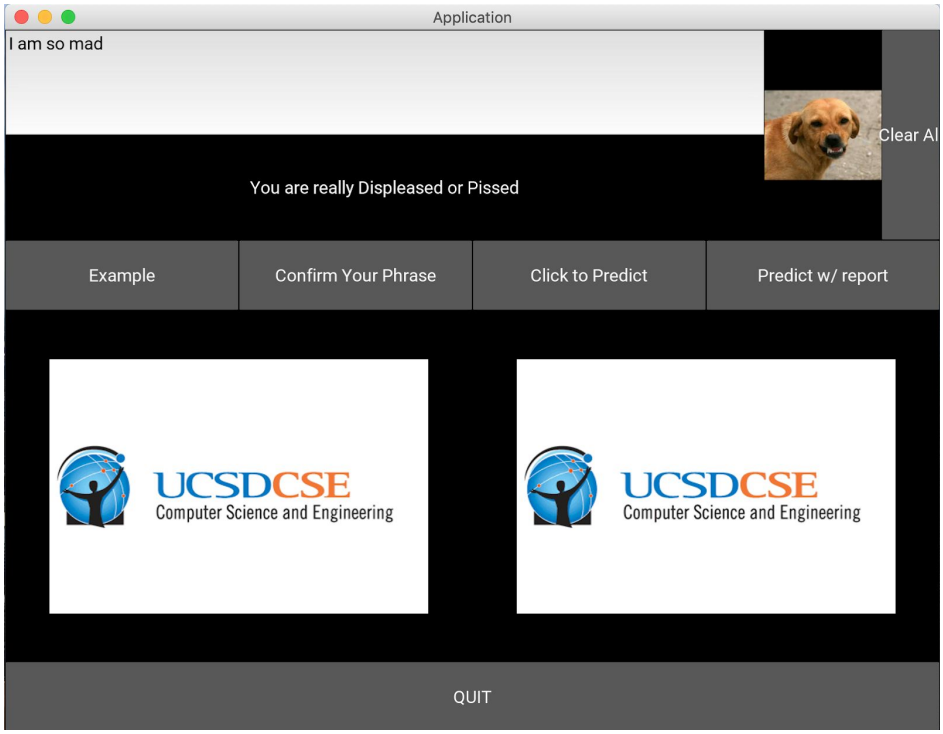
We will start with some easy examples:

Input: I am having a great time

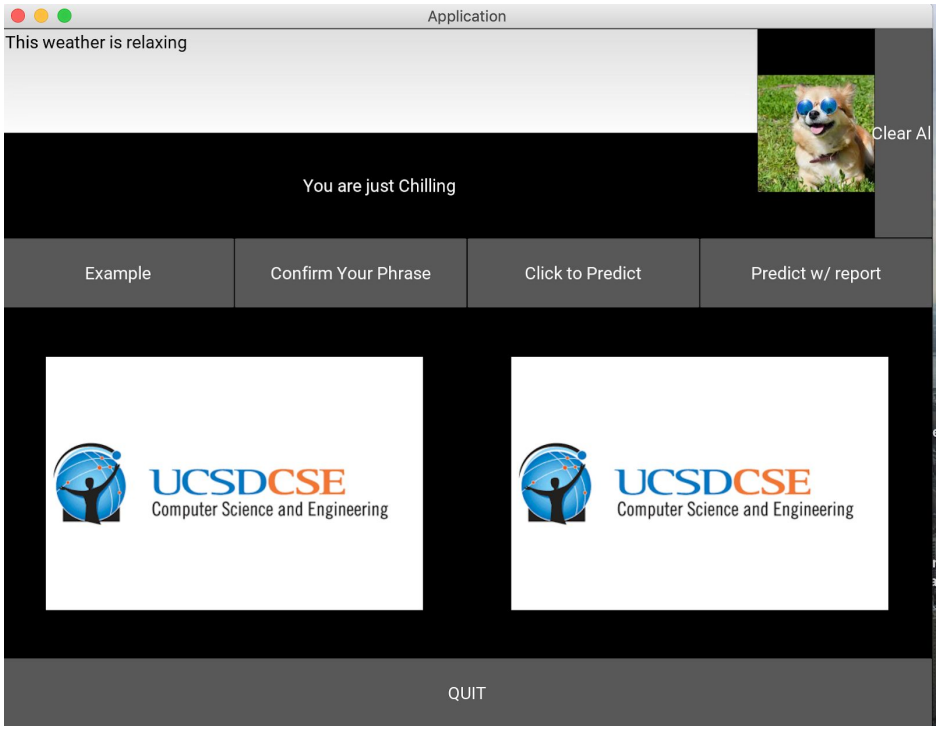
Output:



Input: I am so mad
Output:

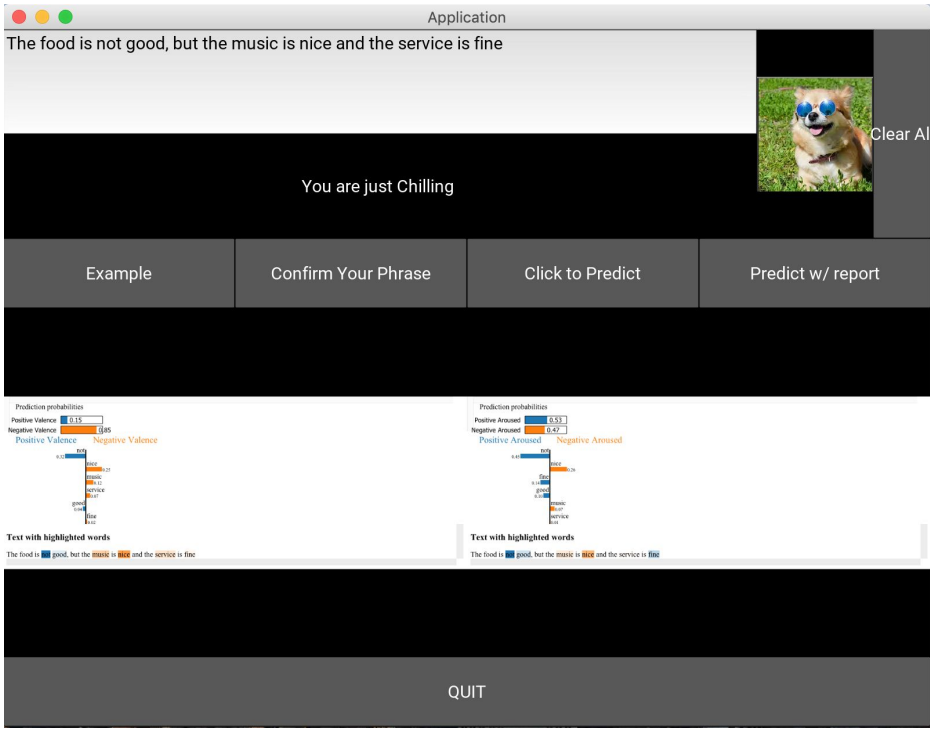


Input: This weather is relaxing
Output:

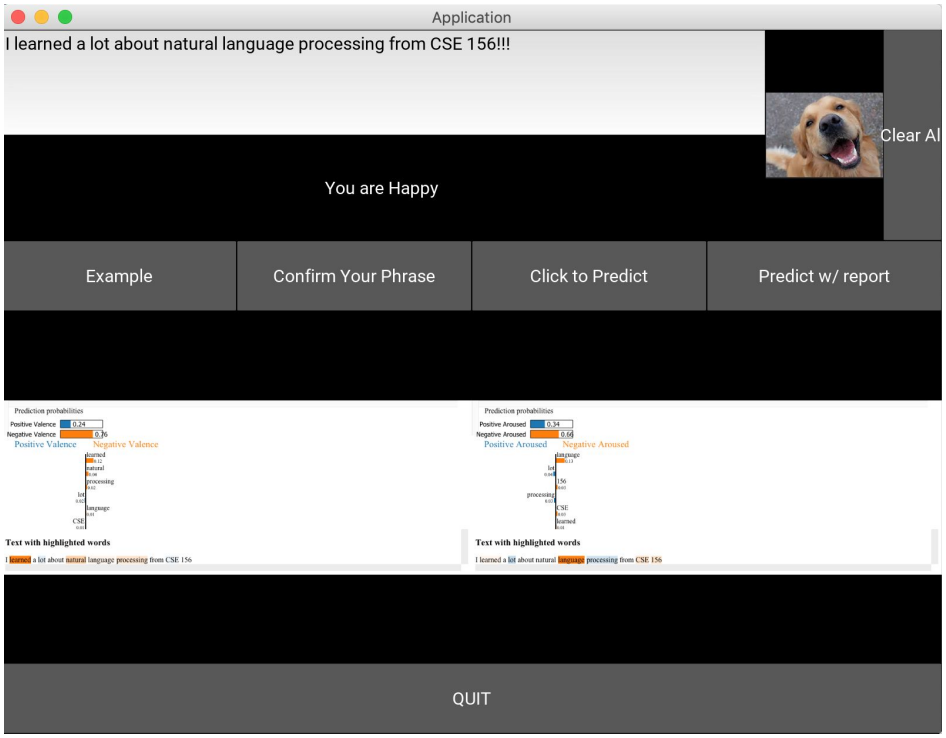


Then, here are some tricky examples with the report:

Input: The food is not good, but the music is nice and the service is fine
Output:

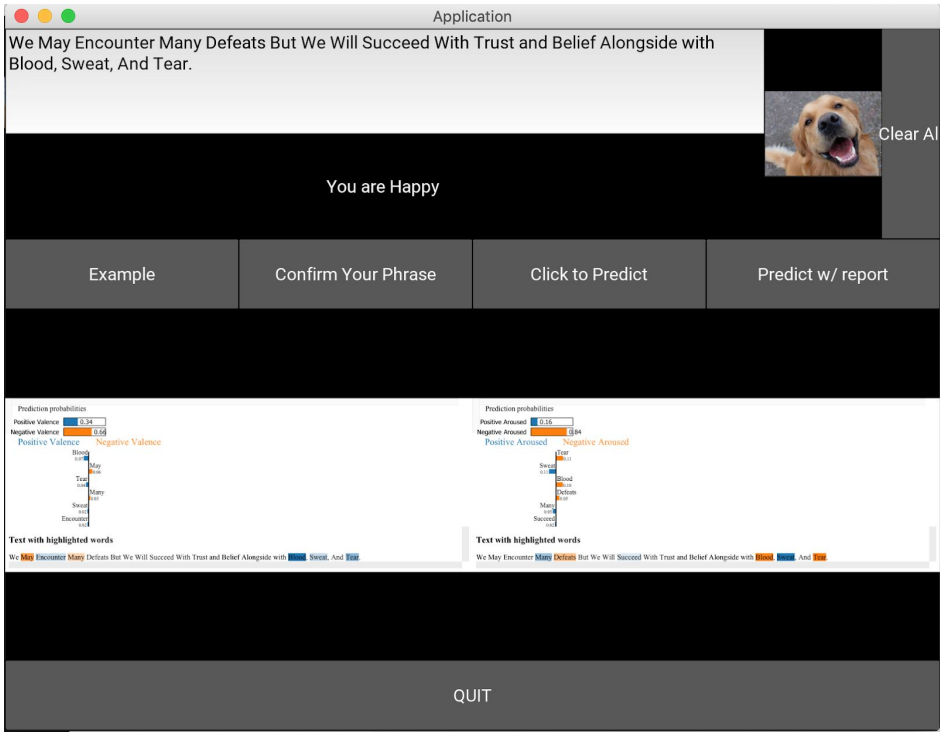


Input: I learned a lot about natural language processing from CSE 156
Output:



Input: We May Encounter Many Defeats But We Will Succeed With Trust and Belief Alongside with Blood, Sweat, And Tear.

Output:



Explanation Details: