Project Documentation

Team Noobs - Allen Zhang (jiaruiz2), Xueyang Li(xl112)(C), Ziyang Zhan (ziyangz6)

1) Function Overview

We developed a simple Chatbot featuring the function of sentiment analysis, which has the ability to identify the user's sentiment based on user inputs and give a proper response that simulates real life chatting. For example, it will try to comfort the user if s/he inputs "I'm so sad". Also, it can respond based on the topic of a sentence, such as recommending movies if the user inputs "Recommend me a movie".

2) Implementation Details

Sentiment analysis: We used the "Twitter Sentiment Dataset" from Kaggle to train our prediction model. First, we implemented data cleaning by removing null values in the dataset. After that, we created our own functions to remove the punctuations in all sentences.

```
def Text_clean(text):
    text=re.sub(r' \n|\n', ' ', text)
    text = re.sub(r'http\S+', '', text, flags=re.MULTILINE)
    text = re.sub(r'@\S+', '', text, flags=re.MULTILINE)
    text = re.sub(r'#', '', text, flags=re.MULTILINE)
    text = re.sub(r',', '', text, flags=re.MULTILINE)
    text = re.sub(r'!', '', text, flags=re.MULTILINE)
    text = re.sub(r'?', '', text, flags=re.MULTILINE)
    text = re.sub(r'。', '', text, flags=re.MULTILINE)
    text = re.sub(r'“', '', text, flags=re.MULTILINE)
    text = re.sub(r'”', '', text, flags=re.MULTILINE)
    text = re.sub(r'、', '', text, flags=re.MULTILINE)
    text=re.sub('[%s]' % re.escape(string.punctuation), ' ',text)
    text = re.sub('\w*\d\w*', '', text)
    return text

round=lambda x:Text_clean(x)
```

Once the sentences only contain words, we can tokenize them and remove the stopwords. We removed the stopwords because we want to avoid some frequent words misleading our model.

In the model training part, we trained multiple different models for sentiment analysis, such as LogisticRegression, Gaussian Naive Bayes, DecisionTreeClassifier, and RandomForestClassifier. Their accuracy is listed as below:

```
LogisticRegression :  0.87575
GaussianNB :  0.544
DecisionTreeClassifier :  0.41825
RandomForestClassifier :  0.789
```

Then we chose the one with the best performance, Logistic Regression, as our prediction model. However, as we talked in class, accuracy has some disadvantages. For example, it does not have good performance when dealing with skewed data. Therefore, we want to further test this model. We computed its macro precision, recall, f1-score, and micro precision.

```
The Micro Precision score is 0.87475
```

	precision	recall	f1-score	support
-1.0	0.87	0.75	0.81	974
0.0	0.85	0.96	0.90	1353
1.0	0.90	0.88	0.89	1673
accuracy			0.87	4000
macro avg	0.87	0.86	0.87	4000
weighted avg	0.88	0.87	0.87	4000

It turns out that this model performs very well in each aspect. Therefore, we finally chose Logistic Regression as our prediction model.

Chatbot: The Chatbot is implemented with some common NLP techniques and a PyTorch Deep Learning model. It takes in a sentence as input from the user, then preprocesses it (including tokenization, stemming and transforming it into bag-of-words representation). Then, we can apply the deep learning model we trained earlier. It's going to give scores based on predictions over all categories of intents the chatbot supports (such as weather, greetings, etc.) If the category with the highest prediction score exceeds a pre-set threshold (0.8 in our case), the chatbot will randomly select a response from that particular category as a reply for the user. If no category reaches the threshold, the chatbot would give a default response (something like "Sorry, I don't

understand what you said.") The Chatbot's name is also customizable, by changing the value of a certain parameter (bot_name) in chat.py. Also, the Chatbot comes with a simple GUI interface. The user can type in any text they want and click "Enter" to send it in as the input.

Integration: The sentiment analysis model has been integrated into the chatbot as a classifier function. Therefore when a user input sentence comes in, we will first use the sentiment analysis model to generate a sentiment label, and if it's not neutral, then the chatbot will randomly select a response corresponding to the given sentiment label. If the label is neutral, we will apply the usual chatbot routine as described above.

3) Usage Documentation

You'll need Python environment and three packages to run this project: NumPy, Pytorch and NLTK.

Once you have these packages installed, move to the directory of the project, run "train.py" to train the chatbot. (OPTIONAL) Then, run "gui.py" (the specific command will be something like "python3 gui.py") to open the program. Once the window pops up, you can type in anything you can in the bottom text box and click "Enter". Your input and the output response from the chatbot should appear on the screen afterwards. You can close the window to exit the program.

Link to the instruction video: https://www.youtube.com/watch?v=Dl9DuswhkqI

4) Contributions

Allen Zhang: Chatbot (both backend and frontend) and integration of chatbot & sentiment analysis model

Xueyang Li: Dataset selection, Data cleaning, Data processing, Multiple Model training for Sentiment Analysis, Model selection.

Ziyang Zhan: searched for possible implementation methods; Implemented the functions of language recognition, spelling correction and translation.