# CPSC 599 (Special Topics in Natural Language Processing) Project Final Report

Jason Jiang
Department of Computer Science
University of Calgary
jason.jiang1@ucalgary.ca

## 1. INTRODUCTION

In this project, I explored the field of chatbots. My project is intended to be a customer support chatbot. What this chatbot explores in the field of NLP is sentiment analysis and intent detection. As for intent detection, I created my own model and training data to classify different intents. As for sentiment analysis, I used VADER (Valence Aware Dictionary for Sentiment Reasoning).

The reason I thought of this project was because I realized that customer support staff are very important for Ecommerce websites. In fact, there are whole teams of customer support dedicated to major websites. This doesn't include big websites too, even small businesses have their own customer support staff. So this got me thinking, how can businesses save on human labor cost?

I added my own twist with it as well. As stated above, the chatbot uses intent classification and sentiment analysis, which allows the chatbot to have a better depth of communication. Not only can it react to intent from the user, it can also generate responses based on the sentiment of the user. I thought it would be fun to have the chatbot be able to play nice and mean, so it incentivizes the customer to be nice. The pipeline is as follows

Input (Customer prompt) → Preprocess the input and make it useful (Use intent, get emotion) → Feed it into the model → Get a response from the chatbot

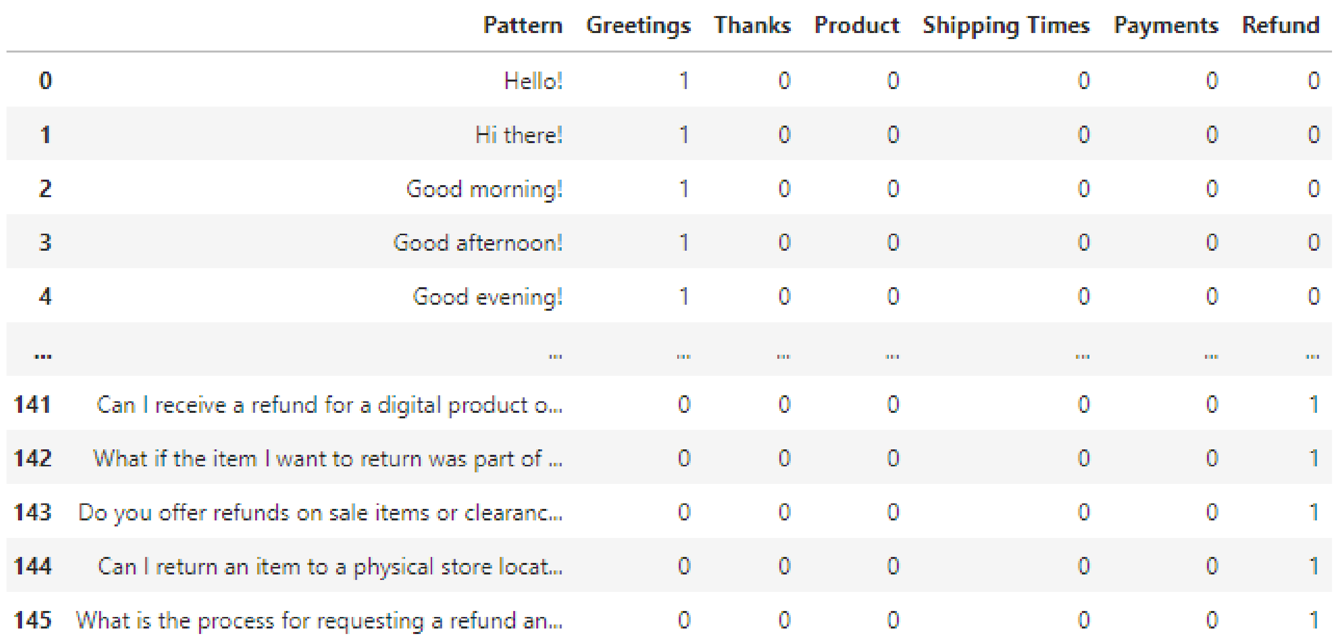
Training of the chatbot was done with a neural network, and the training data was formed from myself and ChatGPT.

As for reference work, there were a few tutorials online that showed you how to make a chat bot

[1]. I watched these tutorials for a brief overview on the correct steps to take to make my own project. I also used the tensorflow documentation [2], namely being the end-end keras example teaching how to build a model. I had used the following documentation in my previous class, so I thought it would be applicable for this application as well.

## 2. MATERIALS AND METHODS

We'll start with generating the training data. The training data is in the form of a csv file, looking like this.



| | Pattern | Greetings | Thanks | Product | Shipping Times | Payments | Refund |
|---|---|---|---|---|---|---|---|
| **0** | Hello! | 1 | 0 | 0 | 0 | 0 | 0 |
| **1** | Hi there! | 1 | 0 | 0 | 0 | 0 | 0 |
| **2** | Good morning! | 1 | 0 | 0 | 0 | 0 | 0 |
| **3** | Good afternoon! | 1 | 0 | 0 | 0 | 0 | 0 |
| **4** | Good evening! | 1 | 0 | 0 | 0 | 0 | 0 |
| **...** | ... | ... | ... | ... | ... | ... | ... |
| **141** | Can I receive a refund for a digital product o... | 0 | 0 | 0 | 0 | 0 | 1 |
| **142** | What if the item I want to return was part of ... | 0 | 0 | 0 | 0 | 0 | 1 |
| **143** | Do you offer refunds on sale items or clearanc... | 0 | 0 | 0 | 0 | 0 | 1 |
| **144** | Can I return an item to a physical store locat... | 0 | 0 | 0 | 0 | 0 | 1 |
| **145** | What is the process for requesting a refund an... | 0 | 0 | 0 | 0 | 0 | 1 |

The training data shown here is used for intent classification. We eventually want the model to recognize different patterns and put responses in the correct category. Data cleaning was minimal, since I created this data myself. Once I had the CSV file, I removed all punctuation, lowercase the words, and lemmatized the words. I used sklearn's CountVectorizer() to generate a bag of words encoding for the data, and resulted in a vocabulary as well. Furthermore, I saved the vocabulary from vect.vocabulary into a text file so we can create a bag of words model when we give it a user input. From this, I split the data into training and testing splits with stratification, then I trained a sequential model.

From this model, we can load it into our chatbot, and use CountVectorizer() again to transform user input into a bag of word encoding, using the vocabulary text file we saved earlier. We finally then get our resulting output from the chatbot, in which chatbot responses are pulled at random from a json file, depending on the user intent, and sentiment.

## 3. RESULTS

I'll start with my early results. Initially, the chatbot performed well. Early results shown below

| **User Input** | **Implied Sentiment/Intent** | **ChatBot Response** |
|---|---|---|
| Hello | Greeting, Neutral | Hello |
| Hi there! How are you my friend | Greeting, Positive | Hello!!!!!!!! |
| You're so useless and not helpful, you are not receiving my thanks. | Thanks, Negative | Bye :\| |
| Hi there friend may I please get a refund | Refund, Positive | Hello!!!!!!!! |

Adding additional training data for the questions (Refund, Shipping)

| **User Input** | **Implied Sentiment/Intent** | **ChatBot Response** |
|---|---|---|
| Hello | Greeting, Neutral | Hi there |
| Hi there! How are you my friend | Greeting, Positive | Hi there! We accept all forms of credit cards. If your credit card is not on there, contact us and we can help sort things out! |
| You're so useless and not helpful, you are not receiving my thanks. | Thanks, Negative | Bye :\| |
| Hi there friend may I please get a refund | Refund, Positive | Refunds are accepted within 30 days usually of a purchase for |

| | | |
|---|---|---|
| | | an unopened, and undamaged item. However if you did open it or anything, we'll see what we can do, no worries! |

Adding additional training data for greetings

| **User Input** | **Implied Sentiment/Intent** | **ChatBot Response** |
|---|---|---|
| Hello | Greeting, Neutral | Hi there |
| Hi there! How are you my friend | Greeting, Positive | Hi there! How are you my friend |
| You're so useless and not helpful, you are not receiving my thanks. | Thanks, Negative | Bye :\| |
| Hi there friend may I please get a refund | Refund, Positive | Refunds are accepted within 30 days usually of a purchase for an unopened, and undamaged item. However if you did open it or anything, we'll see what we can do, no worries! |

## 4. DISCUSSION AND CONCLUSION

As we can see with the early results, it seems fairly accurate for simple prompts. The big issue I was encountering was the fact that the chatbot recognized some phrases incorrectly. For example, *"Hi there friend may I please get a refund"* is supposed to refer to refunds, but because the word "Hi" is in there, the chatbot recognizes it as a greeting.

I attributed this fact to the small training size the chatbot had to learn from. Therefore, an improvement that would help the chatbot would be to increase the training size. Therefore, when I increased the amount of training data for the questions relating to refunds and payment, it helped solve that issue.

However, another issue was encountered, relating to *"Hi there! How are you my friend"* being treated as a question for a payment instead of a greeting. To solve this, I increased the training data for the greetings set. It seems that once we increase the training data set size, the results seem a bit better. Another proposed solution to problems like these was from Dr. Ovens. The suggestion was to treat every prompt with key words (ie, refund, payment) as a payment or refund question. This would prevent incorrect responses, but could have the issue of making the chatbot too hard wired (for example, should the prompt *"Hi there! My refund was received. Thank you!"* be treated as a refund question? Or a thanks?

When compared to other results, I noticed that they had a confidence threshold. Since the model outputs an array of confidence values, the other results had it so that if the confidence was below a certain threshold, then the chatbout would just respond with "I don't know". This could be something that I could implement, although my concern would be that if there are certain prompts that are very closely related (so all categories have similar confidence), the output of "I don't know" might not be suitable. For a simple example, the prompt "Hi there my friend, may I please have a refund?" may trigger 49% greetings, and 51% refund. In the other results, their confidence threshold was 75%, therefore the following prompt would trigger "I don't know". Additional research is required to figure out how to implement something like this.

The biggest limitation was training data size. At the end, there were only 347 rows of training data. We saw above in the results that more training size led to more accurate classification by the chatbot, so therefore, if I had a hypothetical training data set that included 20000 customer inputs for each pattern, then the chatbot would be better suited to handle more complex prompts.

As for future improvements, one improvement from our classmate, *Tait Wiley*, suggested that I could keep a memory of the conversation, so that if the customer starts out negative, and then becomes positive, the chatbot could have a quirky response such as “Now you’re being nice?”, or interpret the overall sentiment as neutral. This would improve the depth and complexity of the conversations with the chatbot, which would be interesting to see.

Another improvement is to definitely include more training data from an external source. If there are websites that share customer input data, that would be good for training the chatbot for that unique website application.

Another suggestion was from Dr. Ovens, and that was to allow the chatbot to create more fluid and dynamic responses (not pulling from a json file for pre-generated responses), kind of like ChatGPT or the chatbot exercise in class. This is also a good suggestion to allow more depth into conversations.

Overall, the chatbot performs well for the given limitations. The project was a good way to learn NLP applications and the methodologies we used in class. From text-preprocessing, to word embeddings, and eventually to model training. As for future work, and the future improvements, it would be interesting to see how far I can take the chatbot with those suggestions.

## 5. REFERENCES

[1] Patrick Loeber. 2020. *Chat Bot with PyTorch - NLP and Deep Learning - Python Tutorial* *https://www.youtube.com/watch?v=RpWeNzfSUHw&ab_channel=PatrickLoeber*

[2] 2022. *Training a neural network on MNIST with Keras* *https://www.tensorflow.org/datasets/keras_example*