Faculty of Computer Science

Dalhousie University

CSCI 6505 – Machine Learning

PROJECT REPORT

Group - 17

Chatbot

Rishika Bajaj	B00902713	rs348937@dal.ca
Sampada Thakkar	B00893022	sm223034@dal.ca
Suchitra Dhamu	B00897187	sc632007@da.ca

10 April 2022

Abstract

Chatbots are often described as software which chats with people using deep learning. This software is accustomed to performing tasks like quickly responding to users, informing them, helping to urge products and providing better service to customers. Chatbots are a prominent issue, and generative models-based chatbots are gaining popularity. The purpose of this report is to build a chatbot based on 2 approaches – using a feed forward convolutional neural network model and cosine similarity to generate responses. We developed the model and tested it on the CPU-based deep learning framework PyTorch. There were no hand-crafted rules used in the training of the model. The model is created from a tiny dataset and responds to a user.

Introduction

Our DalBot is a chatbot for answering queries related to Dalhousie University courses, payments and societies. We created DalBot because some people face issues while navigating through the sites. They find it much easier to interact with bots and get instant answers rather than navigating and searching for the same. Bots have the capability to give calculated and accurate answers with proper training and testing. To make the model effective, we used two approaches.

In first approach, we used NLP and created a training dataset with PyTorch. After that, we created our model which is a feed forward neural net with 2 hidden layers. Our model would get our bag of words as an input, then we have one fully connected layer which would have different patterns as the input size, then 2 hidden layers and the output size must be the number of different classes.

In second approach, we created data frame from our dataset. Then, we converted the sentences in patterns to vectors. When the user inputs the query, it is vectorized and cosine similarity is calculated between them, and the response is shown with maximum cosine similarity.

Dataset

We have created a synthetic dataset using .json file that contains tags, patterns and responses for the chatbot. We processed the data using NLP Basics: Tokenization, Stemming, Bag of Words.

To create our training data, we considered different patterns for example, “Hi” and “How are you?” and put it under the tag “Greeting”. So, we have X patterns [x1, x2, x3…xn] which are put under the tag Y. But we cannot put string into our training data to train the model. So, we converted our string to a vector that contains numbers. For this we used the concept called “Bag of Words”. To use that we collected all the different words from the patterns and put them into a single array by splitting the string by space. For example,

X	Y
“Hi”	
“How are you?”	Greeting

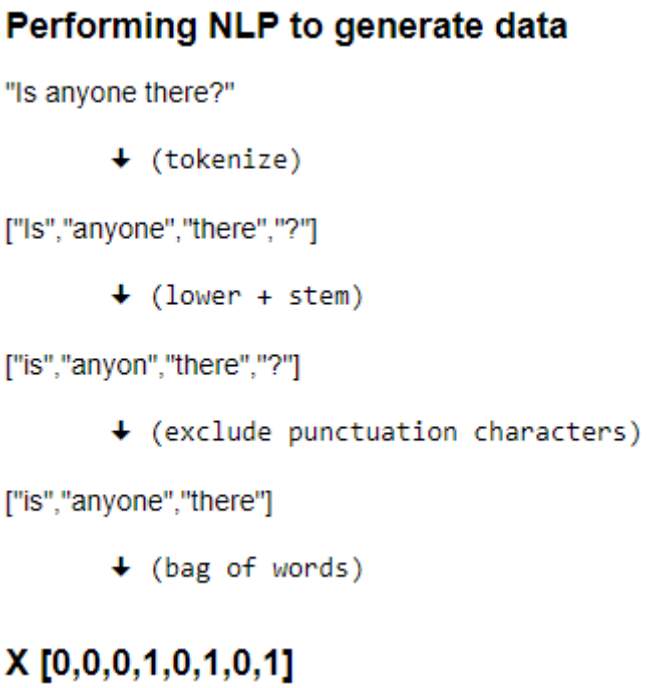
Bag of words-

[“Hi”,” How”,” are”,” you?”]

Then we created an array of numbers, we created a number array for each word of the same size as the array above-

“Hi”	**[1,0,0,0]**	**0**
“How are you?”	**[0,1,1,1]**	**0**

Then we applied Tokenisation so that our training data becomes more refined, and it separated symbols like “?” from the words. We will also use stemming here to generate the root form of the words in order to generate accurate results. To perform all the NLP tasks, used the nltk toolkit.



Experiments, Results and Discussions

1. **NLP and Feed Forward Neural Network**
   After processing the data using NLP, we created a training dataset with PyTorch. After that, we created our model which is a feed forward neural net with 2 hidden layers and 2 dropout layers with probability 0.2. Our model received our bag of words as an input, then we have one fully connected layer which would have different patterns as the input size, then 2 hidden layers and the output size as the length of tags from out dataset.

   We calculated loss using Cross Entropy Loss and optimized our parameters with Adam optimizer, taking the learning rate as 0.001. With this, we’re getting an accuracy of up to 100% with 1000 epochs while training. This is the results generated after handling the overfitting in the model. As per the graph, the accuracy graph for the testing is fluctuating due to Adam optimizer. We also tried using the SGD optimizer, but the result was the same.

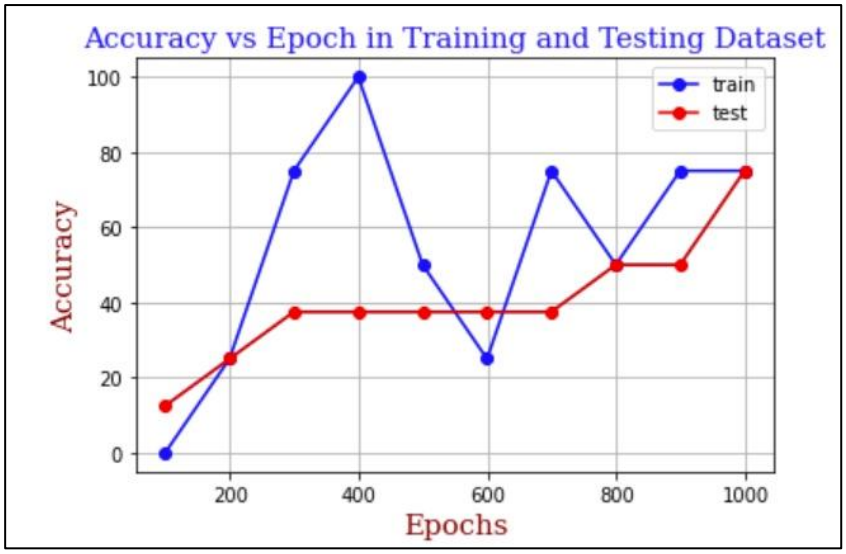


Fig 1. Accuracy vs Epoch for training and testing data

We also tried to optimize our data further by trying different epoch values but the performance of the chatbot degraded when we changed the learning rate or the number of epochs. The model also gave unusual accuracies when we changed the probability of the dropout layer to 0.5. After trying different combinations, we chose the hyper parameters – epoch as 1000, learning rate 0.001 which gave us the maximum and optimized accuracy.

After training the model, we saved it as "data.pth" file and loaded it before testing. We tested the model on validation data which gave the accuracy up to 75% with 1000 epochs.

Once this is completed, we developed an interface to interact with the chatbot by accessing our dataset file and loaded the saved model. Here, we applied the SoftMax function and got probabilities for each class. We're using responses picked from the data set for probabilities greater than 50% otherwise the bot is printing a default message "I do not understand...". The results of this implementation are as below:

```
Let's chat! (type 'quit' to exit)
You: hi
DalBot: Hi there, how can I help?
You: Are you sure you are a robot?
DalBot: Yes I am a robot but I am a smart one!
You: what do you do?
DalBot: I do not understand...
You: what is your job?
DalBot: I am DalBot. Here to help regarding your courses queries.
You: what are the courses?
DalBot: We offer courses like Machine Learning, Visualization, Process of Data Science etc.

You:
```

Fig 2. Screenshot of interaction with chatbot using FFNN

2. Chatbot using Cosine Similarity

Our second approach was building a chatbot using cosine similarity. To do this, we are first creating a data frame from our synthetic data. After that, we are using count vectorizer method to calculate cosine distance for two sentences. One of the two sentences that we are taking as input here is the input query that we take from user (e.g., what are the courses that

you offer?) and for the second input, we are iterating over each pattern that we have in our data frame. Thereafter, we are vectorizing both the input sentences and calculating cosine distance of the input from each pattern. After that we are taking out the pattern with the maximum cosine distance and giving the response corresponding to that output from our data frame.

Results and Outcomes

To show the results and outcomes, we are taking the calculated cosine distance of the input sentence to all the patterns and then plotting the corresponding intents along with the cosine distance as shown below.

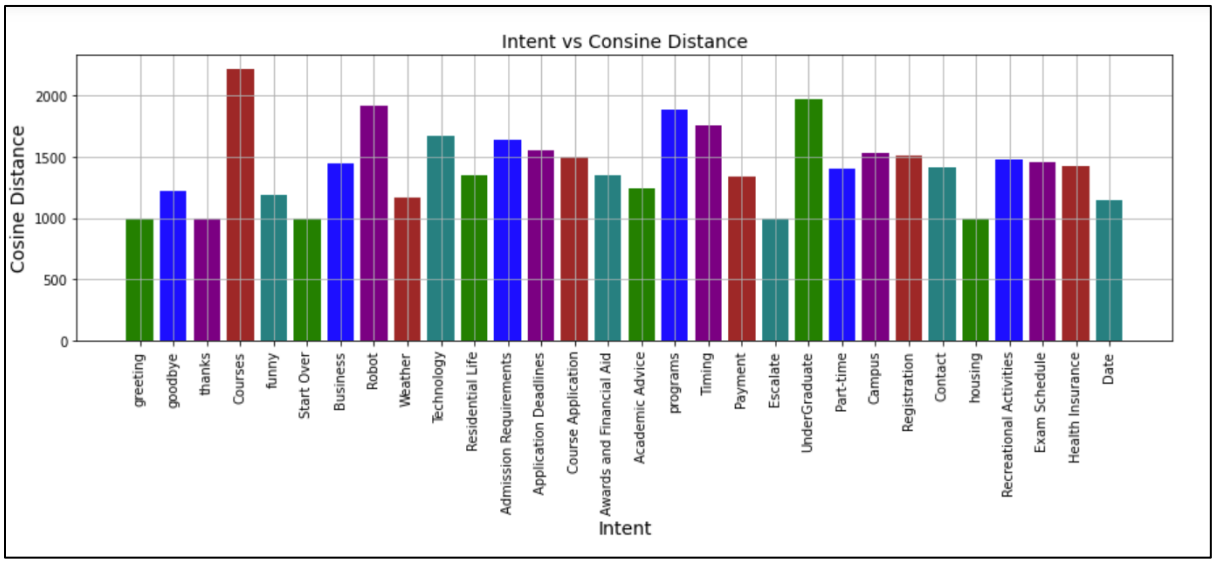


Fig 3. Graph to show the frequency over different intents in Cosine implementation

Secondly, we are making a word cloud with the intents and the cosine distance to show which intent does the input belong to according to the bot and where is it fetching the response from.

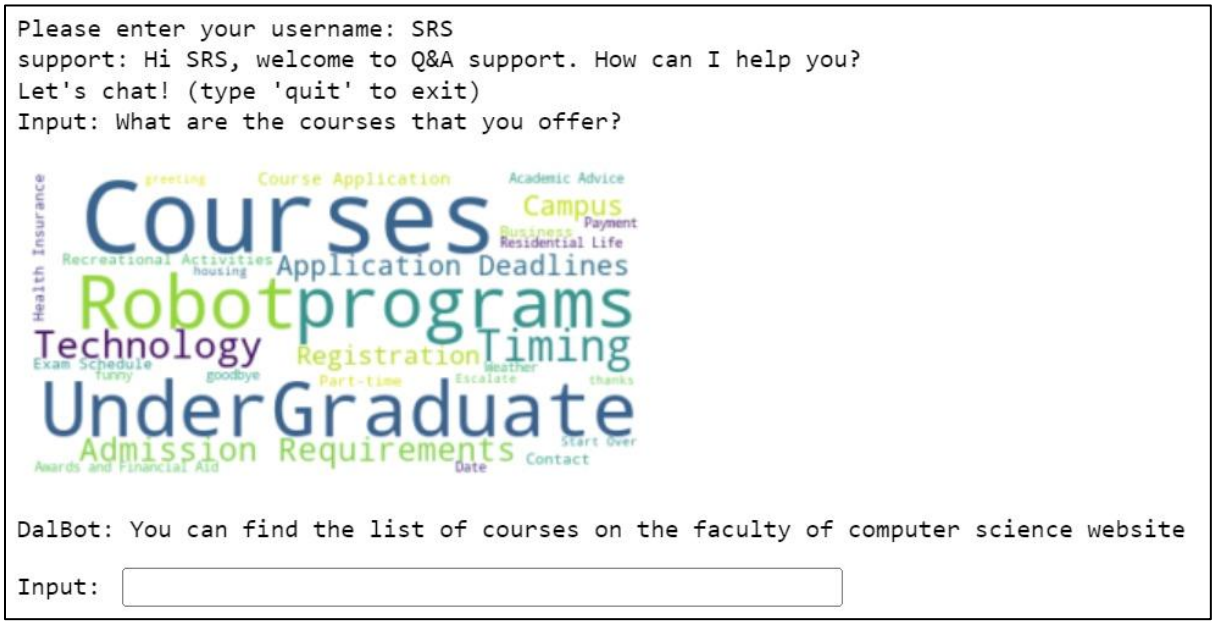


Fig 4. Screenshot of interaction with chatbot using Cosine Similarity along with word cloud

Conclusion

We found the accuracies using the feed forward neural network to be varying when testing the model with validation data. In cosine similarity, we used a data frame that stores the tags, patterns and responses and the outcome from the chatbot is consistent on each run. After building with both approaches, we captured individual pattern and response and found that the NN chatbot mapped with its ground truth and gave the correct probability for each class. However, there is still scope for more improved in the future, but we were able to use the parameters that optimized the results.

In future, more data can be generated to evaluate how good it fits the data to predict the outcome. A basic voice I/O system can also be introduced to make it more user-friendly.

References

[1] Medium. 2022. *Innovative Chatbot using 1-Dimensional Convolutional Layers*. [online] Available at: <https://towardsdatascience.com/innovative-chatbot-using-1-dimensional-convolutional-layers-2cab4090b0fc> [Accessed 27 March 2022].

[2] Pytorch.org. 2022. *CosineSimilarity — PyTorch 1.11.0 documentation*. [online] Available at: <https://pytorch.org/docs/stable/generated/torch.nn.CosineSimilarity.html> [Accessed 2 April 2022].

[3] Youtube.com. 2022. [online] Available at: <https://www.youtube.com/watch?v=m_CooIRM3UI> [Accessed 1 April 2022].

[4] Brownlee, J., 2022. *How to Avoid Overfitting in Deep Learning Neural Networks*. [online] Machine Learning Mastery. Available at: <https://machinelearningmastery.com/introduction-to-regularization-to-reduce-overfitting-and-improve-generalization-error/> [Accessed 21 March 2022].

[5] Machine Learning Plus. 2022. *Cosine Similarity - Understanding the math and how it works? (with python)*. [online] Available at: <https://www.machinelearningplus.com/nlp/cosine-similarity/> [Accessed 28 March 2022].

[6] Pytorch.org. 2022. *Chatbot Tutorial — PyTorch Tutorials 1.11.0+cu102 documentation*. [online] Available at: <https://pytorch.org/tutorials/beginner/chatbot_tutorial.html> [Accessed 10 March 2022].

[7] Python-engineer.com. 2022. *Chat Bot With PyTorch - NLP And Deep Learning | Python Engineer*. [online] Available at: <https://www.python-engineer.com/posts/chatbot-pytorch/> [Accessed 3 March 2022].

[8] T. Nguyen and M. Shcherbakov, "A Neural Network based Vietnamese Chatbot," 2018 International Conference on System Modeling & Advancement in Research Trends (SMART), 2018, pp. 147-149, doi: 10.1109/SYSMART.2018.8746962.

[9] Medium. 2022. *Don't Overfit!—How to prevent Overfitting in your Deep Learning Models*. [online] Available at: <https://towardsdatascience.com/dont-overfit-how-to-prevent-overfitting-in-your-deep-learning-models-63274e552323> [Accessed 25 March 2022].