**Abstract**

# **Deep Learning Chat Bot for Universities**

|  |  |  |  |
| --- | --- | --- | --- |
|  | Omar Santos  Texas A&M University  College Station, TX  Priyanka Paul  Texas A&M University  College Station, TX | |  |
|  | |  |  |

*Chatbots are a way for entities to provide information without using many resources. There are two main types of chatbots. The first is the standard rule-based bot that completes scripted actions based on keywords. The second is the AI-powered chatbot, which uses machine learning to converse more naturally. Machine learning has been used to try and improve the AI of a chatbot by learning from the inputs of its experiences. Based on a list of intent and responses, we created a deep learning model to provide useful class information to members of the CSCE 421 class via a chatbot. Our model can learn to converse with users and is able to continue to be improved. We believe a similar chatbot could be used for the benefit of Texas A&M University Students.*

# **Introduction**

Chatbots are computer programs based on Artificial Intelligence designed to simulate conversations with humans, especially over the Internet. They are simple software applications that imitate human or written or spoken human speech for simulating an interaction with an individual. Machine Learning is used to improve these programs.[4] By learning from conversations and building models, the chatbots can improve their communications with people leading to a better user experience.

Business Insider experts predict that by 2020, 80% of enterprises will use chatbots. According to Lauren Foye, by 2022, banks can automate up to 90% of their customer interaction using chatbots. A survey conducted by Spiceworks showed that 40% of large companies employing more than 500 people plan to implement one or more intelligent assistant or AI-based chatbots over corporate mobile devices in 2019. According to Relay, 65.1% of companies using chatbot technology are engaged in web software, and 58% of companies using chatbots technology are focused on business to business communication.

Despite the improvements made in AI and ML technology, chatbot technology is still far from perfect. Companies using chatbots today see many opportunities for improvement. According to Spiceworks, 59% of respondents indicated that chatbots often misunderstood the nuances of human communication, 30% reported that chatbots performed commands inaccurately, and 29% reported difficulties in understanding accents.[4]

Since chatbots are primarily used in business and sales, there is potential to use them in other areas. Our belief is that chatbots can have useful applications regarding university matters. Students often have questions about office hours, assignments, and tests. A dedicated chatbot can save time for both parties - the one who asked the question won’t have to wait for a response and the one that would have been answering won’t have to answer simple questions multiple times.

We believe that the training of a chatbot using machine learning could increase the knowledge of students about a course, and decrease the time professors and TAs will spend answering emails. Although the syllabus is a useful source of information, classes often deal with deadlines that can change and machine learning training will improve the efficacy of the chatbot Even though there are several chatbots used in the market, none of them are specialized for education conversations. The chatbot was tested by students to evaluate its ability to deliver information based on their needs. If the project is successful, we would like to suggest implementing our chatbot to help professors at Texas A&M University.

## **Problem Background**

In college classes, students often have questions about the specifics about the class. While professors try their best to provide their information in a very accessible place, students are either too slothful or have too busy schedules to remember to check the syllabus for the information about the class. Options like class group messages like GroupMe and Slack can be used to help the students find out the answers to their questions faster. The problem with these accessible solutions is that professors and teacher assistants still need to sacrifice their time to answer these questions. After noticing several questions being asked in the class Slack channel, we wanted to work on a project that would help professors in similar situations. With some research done, the topic of Chatbot and Machine Learning was discovered and the team thought it would be a good way to learn about deep learning while finding a useful and fitting real world application to be developed.

## **Retrieval Based Chatbots**

The type of chatbot we are implementing is the closed-domain retrieval-based chatbot. It is a type of AI chatbot that gives specific answers to questions by learning natural language responses and regular updates of the ML model, so it performs better as usage increases. A retrieval based chatbot seemed like the best type of chat bot for a university because it has no language, grammar, or syntax problems since the answers are predetermined and the scope of the questions being asked is not very large. It is closed domain because it is only meant to answer questions about a course or university. Retrieval based chatbots are trained on a file containing questions and their possible answers. It cannot generate completely new answers, but it performs better than the primitive rule-based chatbot because it relies on machine learning, so it can return the most relevant answer based on the query. One of the problems we could run into with a retrieval based chatbot is with irrelevant input. If the user input is not similar to anything in the database, the chatbot will return a random response and seem very redundant. However, since this just bot is for university and course FAQ purposes, there is a lower chance of this occurring. The pipeline of a retrieval based chatbot starts with encoding the questions into vectors, then using a similarity measure to find the most similar question in the database, and returning one of the answers of the chosen question.

## **Methods**

## **Data Handling**

In order to create a model based on a series of intents, the data first needs to be imported and loaded. A file with JSON formatting was loaded with patterns and responses that the Chatbot should respond back to. JavaScript Object Notation (JSON) uses human-readable text to store and transmit data objects consisting of attribute-value pairs and array data types. With the json package in Python, the JSON file was able to be read and be prepared to be processed and used for training. The series of intents consist of tags, patterns and responses. The patterns are the questions that we think a user could ask, the tags are the categories that we place the patterns in, and the responses are the responses that the chatbot will give based on how it classifies the user’s input. We added multiple responses for each tag, so that if the user asks similar questions, the chatbot can give multiple responses instead of the same response over and over again. This will give the chatbot a more human-like feel.

The data read has to be sanitized to be able to create a clean model. Preprocessing the data should be done before creating a deep learning model. In order to make preprocessing easier, we create three arrays for words, classes and documents. Tokenization was applied to break the whole text into small words. In this process, characters like punctuation marks are also removed for simplicity. In our project we iterate through the different patterns in the intents and tokenize each sentence, appending every word into the word array. Lemmatizing is done to convert the words into their lemma forms. The lemma form is basically the base word that is found in the dictionary. We use the NLTK lemmatizer to do this for us. Additionally, duplicate words and classes are then removed to avoid any redundancy. The processed word and classes are stored in .pkl files to be used for predicting in the model. We use the Python Pickle API to serialize and deserialize the words and classes arrays. It converts the arrays to byte streams that can be stored or sent over a network. Then, when we want to predict responses for the user we can retrieve and deserialize the streams back to arrays. We use pickle so that if we want to make predictions at a later time, we don’t have to rewrite or train the model all over again.

The training data is what provides the input and output for the Class Chatbot. The input will be derived from the pattern and the output will be the class that the input pattern belongs to. Using a Bag of Words model, our program creates a bag of words array with a 1 if a word match is found in a current pattern and 0 if it is not. The bag-of-words (BOW) model is a representation that turns arbitrary text into fixed-length vectors by counting how many times each word appears[1]. This process is often referred to as vectorization. This is done with every combination between patterns and intents put together. We chose to use the Bag of Words model to extract features from text because it is simpler and more flexible than the other models. First, we have the bag of words, without any structure or order to them. The only goal we are trying to reach is the occurrence of the words. We went through each pattern-class pair, created the vector arrays and then added them to the training list. After we went through all the patterns, we separated the training data into one list with all the word vectors and one list with all the class vectors. These vectors are what we will use later on to train the model.

## **Model**

Figure 1. Training Output

The deep learning model we used is the Sequential deep learning model from Keras, the Python Deep Learning Library. This type of model is a supervised learning model. To create the Sequential model, we used the adding method to add 3 layers. The first layer has 128 neurons, the second one has 64 neurons and the number of neurons in the third layer is dependent on the number of intents. We use ReLU as the activation function for the first two layers and softmax for the third layer. To improve the model prediction and reduce overfitting, we implemented dropout which randomly picks neurons to be dropped out of the training. Since the probability was set at 50%, one in two inputs would be randomly ignored in each cycle. This made the neural network better at generalization and less likely to overfit the training data. To compile the model, we used the compile function with the following parameters: SGD, stochastic gradient descent, for the optimizer, a loss function, and accuracy metrics.

As mentioned before, Stochastic gradient descent is an optimization algorithm for training machine learning algorithms. We used stochastic gradient descent because for big data, SGD is faster than gradient descent. This is because, in gradient descent, the number of data points for each step is the total number of data points, so for models with a large number of data points, gradient descent is slow. However, for SGD, instead of using all data points for one step, it randomly picks just one sample, so it is faster than gradient descent for large data sets. We enable Nesterov as well, to accelerate gradient descent. The purpose of NAG is to make the gradient descent even faster, by adding an extra step of sliding a bit further in the right direction[5]. To fit and save the model, we used the fit function. The model is trained for 200 epochs to achieve good accuracy and batch size of 5[2]. Figure 1 show how NAG would make the gradient descent process faster.

After the model is built and trained, the next step is to predict a response for the user. To do this, the user input has to be processed first. The method we use to process the user input is similar to how we processed the training data for the model. We took in the user input and used the NLTK tokenizer to create a vector of words. Then, we used the Bag of Words method for similarity measure so we can easily compare the user input to the questions in our database. We use the Keras predict function in the model class API to calculate the probability of the user inputted question being in each class in our database. Then we find the class with greatest probability and return one of the responses to the user. During the compiling of the model, it outputs information from each epoch. The loss, accuracy and mean squared error for each epoch is outputted.

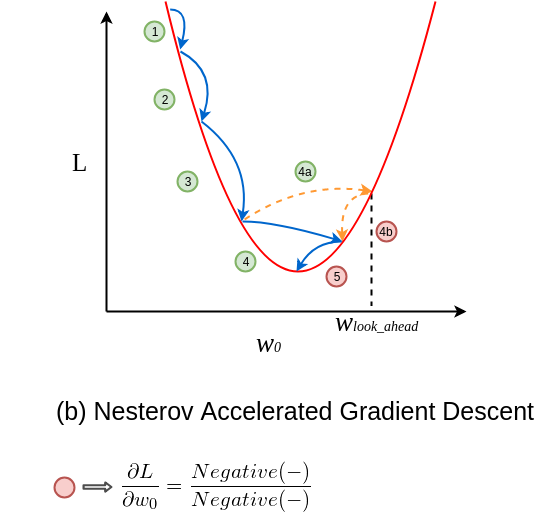


Figure 2. Graphical representation of how NAG works[5]

## **GUI**

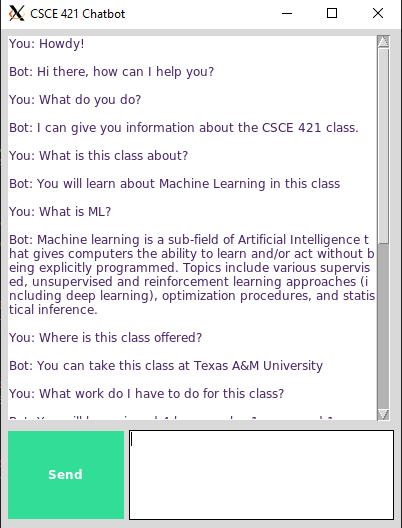


Figure 3. The GUI

To allow the user to interact with our chatbot we created a GUI using the Python tkinter API. The GUI consists of a Send button, a text box and the chatting history. The user will type a question into the text box and then press send. This will add the user input into the chat history and then use that input to predict the response. After the response is predicted it will be added to the chat history as well. We also implemented a scroll bar, so that the user can scroll back through previously asked questions and responses. This easy to use, common format aids users on the usage of our program. Our main intentions with this was that if the users have any dissatisfaction it would not be due to the user experience so we could focus on the reliability of the model.

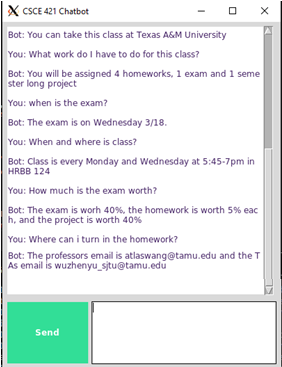


Figure 4. The chatbot conversation thread

# **Experiments and Results**

To validate the results of the chatbot, we used the accuracy and loss metrics to evaluate our model. Our model has 100.00% accuracy and 0.01% loss. Figure 5 shows how the accuracy converges to 1 and Figure 6 shows how the loss function mean squared error converges to 0.

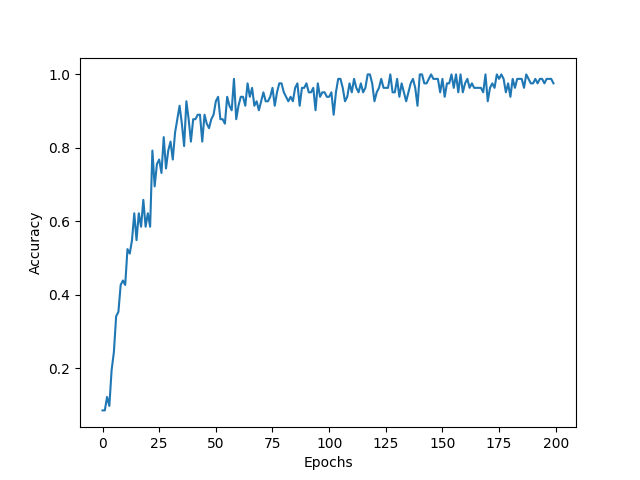


Figure 5. The convergence of the accuracy function

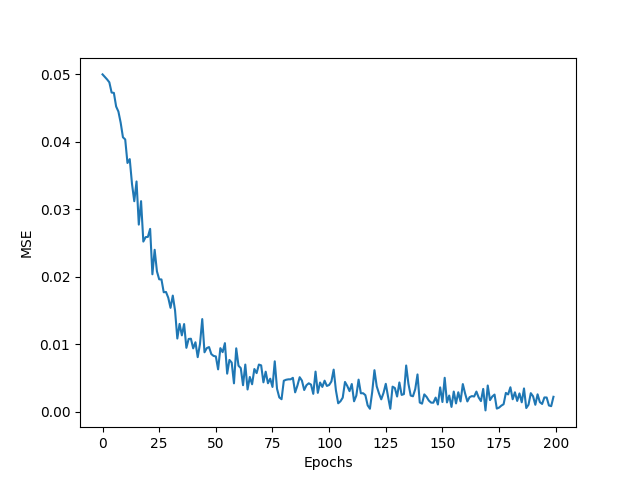


Figure 6. The convergence of the loss function, mean squared error

To evaluate how helpful our chatbot is, we had several college students test it out. They were instructed to ask questions they would have for a college class. After their interaction with the Chatbot, the students completed a questionnaire stating their opinions and experience with the Chatbot. The following are the results from these tests.

The responses indicate a positive response to the quality of the Chatbot responses. The Chatbot’s usability has a positive response as shown in Figure 7. Figure 8 shows that the Chabot has high demand because 100% of the students responded that they would want a Chatbot in all their classes. The responses shown in Figure 9 also show a positive response because 80% of users said that the Chatbot had over 75% of useful responses. Although there were positive responses, there were suggestions for trying to improve the Chatbot performance. A common comment identified was trying to have the Chatbot have more human-like responses. To do this we could incorporate the generative chatbot model with the retrieval model to create more human-like responses. The responses also indicate that the chatbot doesn’t understand acronyms or shortened words. For example, the chatbot didn’t recognize when the user asked a question about “prereqs”, but understood when they expanded “prereqs” to “prerequisites”.

# **Conclusion and Future Expectation**

In this project we created a retrieval based chatbot intended to be used for classroom purposes. With the limited possible answers that a student may have, this was proven to be the correct approach. The surveyed people demonstrated satisfaction with the Chatbot, and an implementation in a class could be beneficial for both the instructor and students. Currently, the chatbot has limited responses aimed for basic questions for the CSCE 421 class. The model could be improved by continuing adding responses and patterns for questions even outside of the scope of the class, including information about other classes and general information about Texas A&M University. Due to the COVID-19 situation, the testing of the program was kept to a minimum due to social distancing. If more time would be allowed to work in this project the current Chatbot would be tested with a greater test population to find all possible fixes that the model or GUI would need. Besides the limited people who were testing, we also conducted experiments to try and break the model to what kind patterns needed to be changed to stop from inflicting with other questions asked. After these experiments, the Chatbot would be ready to be tested in a class setting.

If the Chatbot were to be successful if used for a Machine Learning class it could be expanded and trained for the use in different subjects to increase the scope of our Chatbot. If more time were permitted, integration to website capabilities could be researched and implemented to make user access to the chatbot easier and increase usability for professors. We could also create a mobile application as well, to make user access even easier and to cater more to students. To improve the Chatbot, we would also take into account the student responses that we got from the questionnaire. There was a request to make the Chatbot more human-like. To do this, we would make the chat bot using the generative model, instead of the retrieval model or combine both to create a chatbot that is human-like, but also can answer the specified questions about universities with correct language, grammar, and syntax. We would keep the retrieval model part to ensure that users are receiving the correct information about their university and classes.

# **Task Assignment and Acknowledgement**

This project was completed by the cooperation of Priyanka Paul and Omar Santos, under the instruction of Dr. Zhangyang (Atlas) Wang and Tianlong Chen. Priyanka Paul took charge of intents data and GUI, Omar Santos took charge of the data processing and model, and they collaborated on this report. The contribution ratio of this project was 50% for Omar Santos and 50% for Priyanka Paul as each member worked equally hard and contributed equally.

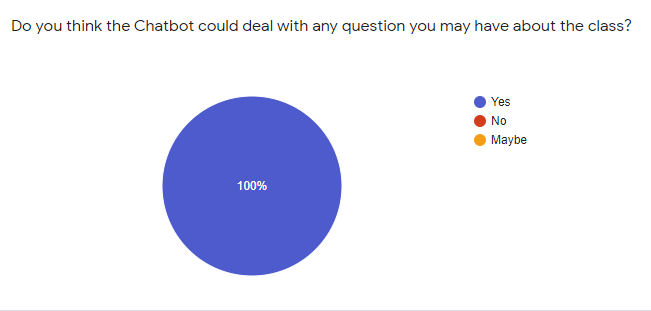
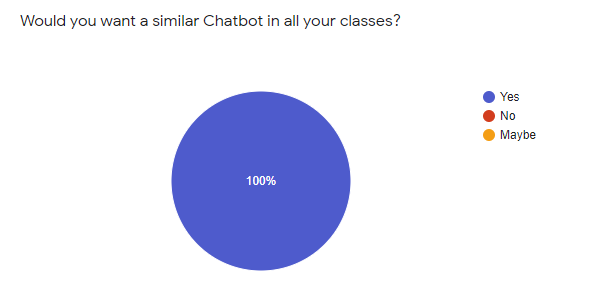


Figure 8. Questionnaire Response

Figure 7. Questionnaire Response

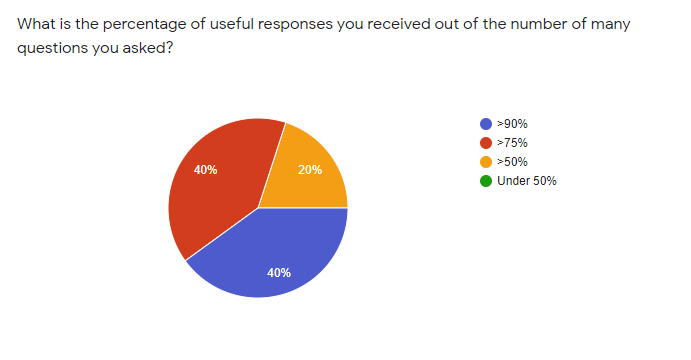


Figure 9. Questionnaire Response

**References**

Brownlee, J., 2020. A Gentle Introduction To The Bag-Of-Words Model. Machine Learning Mastery. Available at: <https://machinelearningmastery.com/gentle-introduction-bag-words-model/>

Brownlee, J., 2020. How To Reduce Overfitting With Dropout Regularization In Keras. Machine Learning Mastery. Available at: <https://machinelearningmastery.com/how-to-reduce-overfitting-with-dropout-regularization-in-keras/>

Brownlee, J., 2020. Your First Deep Learning Project In Python With Keras Step-By-Step. [online] Machine Learning Mastery. Available at: <https://machinelearningmastery.com/tutorial-first-neural-network-python-keras/>

Medium. 2020. Chatbot Report 2019: Global Trends And Analysis. [online] Available at: <https://chatbotsmagazine.com/chatbot-report-2019-global-trends-and-analysis-a487afec05b

Medium. 2020. Learning Parameters, Part 2: Momentum-Based And Nesterov Accelerated Gradient Descent. [online] Available at: <https://towardsdatascience.com/learning-parameters-part-2-a190bef2d12>