AUBot : A Chatbot for Student FAQs

Ali Hijjawi
Dept. of Electrical and Computer Engineering
American University of Beirut
Beirut, Lebanon
ajh30@mail.aub.edu

Amine Ghoussaini
Dept. of Electrical and Computer Engineering
American University of Beirut
Beirut, Lebanon
ajg11@mail.aub.edu

Fatima Ghadieh
Dept. of Electrical and Computer Engineering
American University of Beirut
Beirut, Lebanon
fjg00@mail.aub.edu

Abstract*—Every year, AUB students experience a never-ending cycle of confusion and panic induced by the stressful environment that college provides. As a result, senior students and university staff are bombarded with messages, questions, and requests, and therefore cannot always guarantee a reply to inquiries. This paper proposes an attempt at easing the student servicing process: a retrieval-based chatbot that responds instantly to student FAQs. Not only would that speed up the process of servicing a new student, but it would also eliminate a lot of redundancy in the questions.

Keywords—Chatbot, Natural Language Processing, Deep Neural Networks.

MOTIVATION

There is nothing more confusing than being a first year AUB student. AUB students are thrust into their academic lives with little guidance on how to carry out their first tasks and responsibilities: registration, student accounts, papers, class locations, and so on. They are overwhelmed by the multiple portals and passwords of the AUB website. And because most AUB offices take days to reply due to the heavy number of inquiries and requests they receive from students, this hinders their productivity and efficiency, making the experience highly stressful for students.

According to the "AUB Student Satisfaction Survey" that was conducted by the Office of Institutional Research and Assessment at the American University of Beirut during fall of 2019-2020, 18% of new students thought that the staff did not provide them with clear answers to their questions. 41% of new students also expressed dissatisfaction with the staff's ability to direct them to other resources if they were unable to assist them. Furthermore, students reported having to seek assistance from other students as well as friends and family [1].

Why not compile the experiences of many students and the common problems they face at AUB to make others' lives easier? How can we make these issues seamless and more enjoyable for a new student overwhelmed by this new experience?

Introducing a custom chatbot to the official AUB website makes information more accessible to students and lessens the pressure on various offices at AUB. Through the automation of the question-answering process, this chatbot enables AUB to improve their students' satisfaction by offering fast and consistent responses to their concerns.

The main question we seek to answer is how can we build the chatbot to successfully and as accurately as possible build a bridge between a student's inquiry and the answer they are seeking using predetermined statements with known answers and linking potential new questions to the initial set of questions?

I. PROBLEM MODEL

The model has one external input and one external output. Naturally, the chatbot receives English sentences sent by the user, and the user, in return, receives an English response from the chatbot. Hence, those externals constitute normal English sentences. However, this input-output relationship becomes more complex when looking at the internals of the chatbot. On a remarkably elevated level, the English sentence sent by the user is mapped into a token or an ID, and then those tokens are mapped back into an English answer. The chatbot may also often reply with images or email addresses of relevant people when necessary. A prompt of satisfaction could also be sent from time to time to rate the performance of the chatbot.

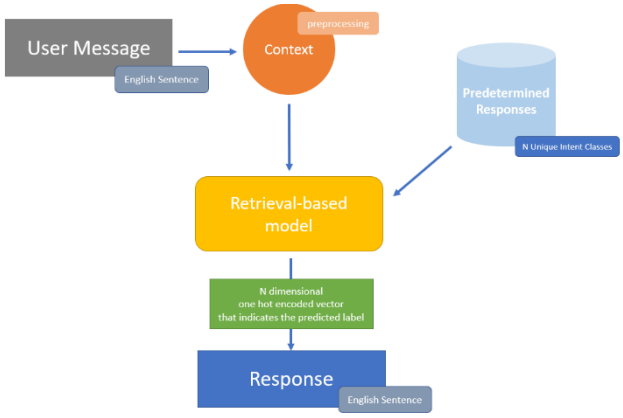

Figure 1: Retrieval-based model architecture [2]

II. MODEL DESCRIPTION

As a reminder, the chatbot is intended to answer student questions and concerns to avoid redundancy and accelerate the process of student servicing. This does not mean that the chatbot is expected to be able to have a normal conversation with students. Hence, the set of answers that the chatbot must be knowledgeable of should be small. We could thus assign a set of tags such that each tag in the set maps to a specific answer. Since the chatbot selects an appropriate response using a repository of predefined responses and a Machine Learning model, this makes it a retrieval-based chatbot [3].

The goal of the chatbot is to be able to convert, as accurately as possible, English messages written by a student, to a specific tag, and then return the response that this tag maps to. This is called word encoding and it will be done by the Natural Language Processing black box that would take the sentences as inputs and encode them into tokens. Data

preprocessing will include text case handling, tokenization, stemming, and generating a bag of words [4]. Similar statements in wording and meaning will get encoded into tokens of close value. We would then use the machine learning algorithm to classify the tokens and map all the possible classifications to different proper answers.

When a new user message is received, the chatbot computes the similarity between the new text sequence and the training data. Based on the confidence score for each tag, it assigns the user's message to the tag with the highest score and sends the predetermined response associated with this tag. Every time the chatbot provides an answer to the student, it inquires them about whether the provided answer was the answer they were looking for. If it was, then everything ran smoothly. If it was not, then it consigns them with the second available answer in the list of sorted answers and so on.

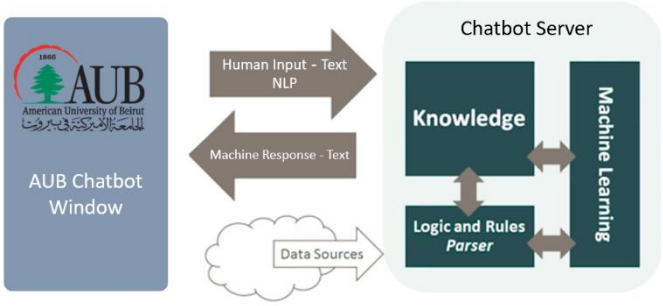

Figure 2: Anatomy of the AUB Chatbot [5]

III. Challenges

Just like any project faces challenges, the chatbot might fail to correctly recognize similar questions if they were phrased differently. As a result, it might not be able to correctly comprehend the student's message. This necessitates ensuring that questions with the same meaning receive the same response. One particularly unfavorable situation is the chatbot stumbling across a question that is not enfolded by its knowledge base. Avoiding this problem requires enriching said knowledge base with as many diverse scenarios as possible so as not to deteriorate the performance of the chatbot if anything outside the programmed script was mentioned. Some problems encountered by students can also be too complex for the chatbot to be able to answer. Finally, the interactions made by the chatbot should appear to be more social and human in nature rather than technological.

IV. Dataset Description

The dataset is formed of 450 questions and responses sent by AUB students on the Facebook group "AUB COURSES/TEACHERS GURU" whereby students ask common questions about issues they encounter during their time at AUB. Students pose their questions as posts within the group, and other students reply to them in the comments. It is a very important platform for student questions and concerns, and it has proven to be beneficial towards making the student's experience seamless and less nerve-wracking.

To collect the data, we built a web scraper that searches the group for specific topics and stores the post along with the comments into a .json file. Each JSON object in the file is formed of three keys: the tag, which shows the main intent behind a user's message, the patterns, which is an array of sample questions posed by students regarding the specific tag, and the responses that the chatbot should output once it identifies the intent behind a user's message.

The questions and responses are in text format and are written in formal American English. A "similar sentence generator" was used as well to augment the collected data and provide additional samples for each question.

Concerning the responses, and to ensure their correctness, we gathered the relevant information directly from the AUB website and asked the relevant offices.

A sample example of the Chatbot dataset is found below:

```
{
    "tag": "Financial Aid Application Date",
    "patterns": [
      "When can I apply for financial aid? ",
      "When can I start applying for financial aid? ",
      "When does the financial aid application come out?
",
      "When can I submit financial aid application for next
fall? ",
      "When do I apply for financial aid for this year?"
    ],
    "responses": [
      "The online application for AY 2020-21 will be
available starting February 10, 2020."
    ]
}
```

V. Experimental Setup

The models will be developed in Python 3.6 using Google Colaboratory environment. These models will be run on a 2 core, 2.30 GHz, Intel® Xeon® CPU with 12 GB RAM and a 12GB Nvidia K80 GPU. Keras and TFlearn packages will be used to build the models.

For data preprocessing, the Natural Language Tool Kit (N LTK) and Lancaster Stemmer will be used. Python Pickle module will be used to create a save of the model. Variables will be loaded into the pickle file so that whenever model prediction is run, the original model predict() function is used. Other crucial libraries such as NumPy, JSON, and Random will be imported as well.

VI. Experimental Assumptions

To ensure that the chatbot understands the user's message, we assume that the student speaks in formal English that is free of any spelling or logical errors. The chatbot has a limited knowledge base; therefore, we also assume that student does not ask questions outside the scope of the chatbot's knowledge. To avoid the chatbot's confusion, the student is expected to send one request at a time in a simple straightforward manner.

VII. Data Preprocessing

After making all the necessary imports, we will load the data from the JSON files. We will then tokenize the words from the JSON files. Tokenization is the process of splitting a sentence into its constituent words. It is a mainstay procedure in both computer and natural language processing and is necessary to treat words as independent entities. We will perform the appropriate stemming. Stemming is important to reduce the size of the vocabulary, by reducing semantically similar words that have been inflected to their common root. Stemming also reduces all characters to lowercase and removes all question marks from the sentences.

The benefits are two-fold, primarily we have a more semantically generalized model that understands similarity between words beyond an exact match, and second, the reduction in vocabulary size decreases the dimensionality of the representation space, which allows for faster learning and convergence with a smaller dataset. This allows us to then vectorize them within a numerical representation for the LSTM model, whereas convert them into one-hot encoding format for the DNN model. The preprocessing stage will be complete, and the input will become suitable for a neural network.

Question: What is the procedure for fulfilling a minor?
Tokenization:
['What', 'is', 'the', 'procedure', 'for', 'fulfilling', 'a', 'minor', '?']
Stemming:
['what', 'is', 'the', 'proc', 'for', 'fulfil', 'a', 'min']
One hot encoding:
[1 0 0 0 0 1 0 1 0 0 0 1 1 1 1 0 1 0]

Figure 3: An example of data preprocessing

VIII. Machine Learning Models

A. Deep Neural Networks

A Deep Neural Network (DNN) is a type of Artificial Neural Network that consists of neurons, layers, and activation functions. The model predicts the intent of some input, based on a set of weights assigned to the features of the problem [7]. In our experiment, the model will map the user's tokenized input to a specific tag and return the corresponding response.

1. Model Architecture

The input layer of the DNN is a layer of neurons whose size is the same as the size of the vectorized user input. The input will be a vector of zeros and ones that indicate whether a word in the vocabulary exists in the text or not. Two additional, fully connected, hidden layers of neurons will be added as well. Each layer takes in the output of the previous layer. Here, we can train the model with multiple added layers and different neurons count on each layer. Those layers will use the Relu activation function. Having a dropout at the layers is used to regularize the parameters of each layer. The output layer of the model will use the SoftMax activation function to perform multiclass classification. Its size corresponds to the number of intents (labels) found in the dataset.

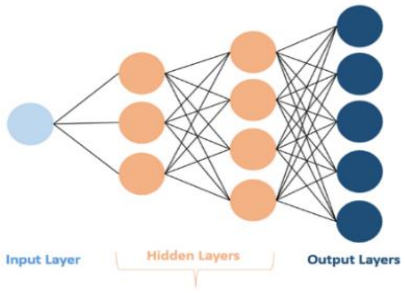

Figure 4: Deep Neural Network model architecture [8]

2. Accuracy Metrics

We will apply a regression function to the model and calculate the optimal weights of the neurons that provide us with the highest accuracy. The regression function will be used to minimize the cost function which measures the closeness of the predicted label to the actual label provided by the dataset. This regression function uses the 'adam' optimizer and categorical cross-entropy as the loss metric.

B. Long Short-Term Memory

Long Short-Term Memory, referred to as LSTM, is a type of recurrent neural network that can learn and store previous information. They have multiple hidden layers and feedback connections, making them particularly good at processing sequences of data. As the data passes through each layer, the irrelevant information is discarded, leaving only the relevant information.

LSTMs do not treat each point of the data independently, but rather try to analyze the relationship between each data point after every iteration. This allows the model to memorize and retain information that can be used to process new data points.

1. Model Architecture

We will implement the model discussed in Jenq-Haur Wang's paper which studies the implementation of an "LSTM Approach to Short Text Sentiment Classification with Word Embeddings".

The first layer will be an embedding layer which takes as input the vocabulary size, the output of the layer, and the input length. This layer transforms the sequence of words within the text into a vector.

We will then add few LSTM layers and set return_sequences to 1. This will make the output of the layer the last hidden state output.

The LSTM outputs will be fed into a Dense layer with a SoftMax activation function because it allows us to perform multi-class classification [9].

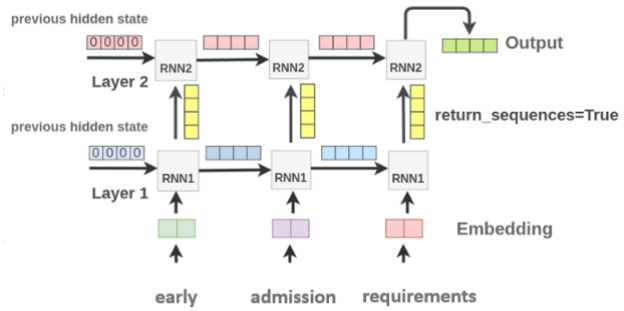

Figure 5: LSTM model architecture [9]

2. ACCURACY METRICS

After performing gradient-based backpropagation algorithm, the weight of edges in the hidden layer will be updated after every iteration. The goal of our model is to have words of similar meaning occupy similar "spatial positions"; hence, the loss will be calculated through cosine similarity. Vectors that relate to such words should be close, so the cosine of the angle between them (1) should approach zero [6].

$$sim(A,B) = cos(\theta) = \frac{A \cdot B}{||A||\,||B||} \qquad (1)$$

IX. PROBLEMS FACED

A. *Data Collection and Organization*

In light of the absence of an AUB-specific dataset of questions, we were thus obligated to obtain a custom-made dataset generated from unstructured data. In order to achieve our goal, we created a python web parser that scraped Facebook posts from a public group that serves as a virtual QA forum for matters concerning AUB policy. After the scraping was complete, it had to be prepared to enable our network to maximize its utility from it. The first step was to parse the data grammatically, which can be done using an automated language system, hence did not consume much time. However, due to several questions concerning policies or courses that were no longer valid, a characteristic undetectable to an automated parser, we would have to do a manual run on the questions, eliminating contextually irrelevant ones. Finally, the data was stored in a JSON format, a human-machine readable store that enables quick, clean access for use in our neural network model.

B. *Model Accuracy*

In order to assess and substantiate both model accuracy and correct operation, we initially tried testing on a small dataset, which contained five intents with four to five different patterns. The results displayed diminutive accuracy levels paired with a high loss. We even tried testing the Chatbot using questions from the training dataset itself and received wrong answers. In order to remedy this issue, we increased the number of epochs. Subsequently, we were able to achieve an accuracy of 1 for each class found in the dataset, the Chatbot was working perfectly, and all questions were correctly classified.

C. *Accuracy with a larger Dataset*

i. DNN

Upon increasing the size of the dataset by adding more intent within the json file, we noticed a huge decrease in the accuracy of the model (around 0.012). This Is expected due to an increased output dimensionality meaning a lower probability of success. In order to increase the flexibility of our model to accommodate more scenarios, we started incrementing the number of neurons and tuning the model to achieve optimal accuracy. We tested several depths, from initially 3 to 6, and then to 10, each layer ranging from 500 to 1200 nodes. The accuracy increased greatly for 6 layers, but then decreased by 10, due to an overfitting situation. Also, increasing the number of epochs gradually from 150 to 1500 increased accuracy due to obtaining a finer fit to our data. We also tried adding over 2000 nodes for most of the layers but that only achieved overfitting.

Due to the size of our dataset, training took hours, but then fixing the batch size to around 150 immensely helped model fitting, decreasing time for each epoch, without sacrificing any prediction accuracy.

With 6 layers and a quadratic shape in number of neurons, we achieved optimal accuracy of 0.69-0.7 at around 1500 epochs. Increasing accuracy further came through a breakthrough in the data rather than the model. By increasing the coverage of the question space by adding more variations of questions for the intents present in our data, we exposed the network to a more diverse range of inputs, and the accuracy rapidly exceeded 0.9.

The bot instantly started replying correctly and more frequently to challenging questions. It could also identify the correct classification with new perturbations that were unlike those it had been shown, demonstrating strong semantic understanding.

ii. LSTM

The accuracy of the LSTM model was very low initially, so we tried to fix the tokenization by tokenizing according to the size of the vocabulary of the dataset. In turn, we also adjusted the input of the embeddings layer by making it equal to our vocabulary size. This resulted in a much higher accuracy per epoch.

Moreover, seeing as the model was underfitting the data, we decided to tune the model by adding neurons and layers which lead to a much higher accuracy. The model was no longer underfitting the data.

X. RESULTS REPORTED

We observed the metrics provided by the output from the model.fit() function when implementing the LSTM and the DNN models. The values of accuracy and loss were 1.33751 and 0.9109 (figure 6) in the DNN model compared to -0.7157 and 0.6832 (figure 7) in the LSTM model.

```
Training Step: 929  | total loss: 1.47647 | time: 0.150s
| Adam | epoch: 310 | loss: 1.47647 - acc: 0.9062 -- iter: 300/303
Training Step: 930  | total loss: 1.33751 | time: 0.260s
| Adam | epoch: 310 | loss: 1.33751 - acc: 0.9109 -- iter: 303/303
```

Figure 6: DNN Chatbot Metrics

```
- 4s 14ms/sample - loss: -0.7449 - acc: 0.7129 - val_loss: -0.7006 - val_acc: 0.6222

- 5s 15ms/sample - loss: -0.6803 - acc: 0.6436 - val_loss: -0.7061 - val_acc: 0.6222

- 4s 15ms/sample - loss: -0.7157 - acc: 0.6832 - val_loss: -0.7161 - val_acc: 0.6444
```

Figure 7: LSTM Chatbot Metrics

a. Accuracy Curves

After training the models on the same dataset and to be able to compare their performances, we plot their accuracy curves with respect to the number of epochs. We notice that the LSTM model (figure 8) reaches its maximum accuracy much faster than DNN (figure 9), however, the DNN model has a much higher value of accuracy compared to the LSTM model.

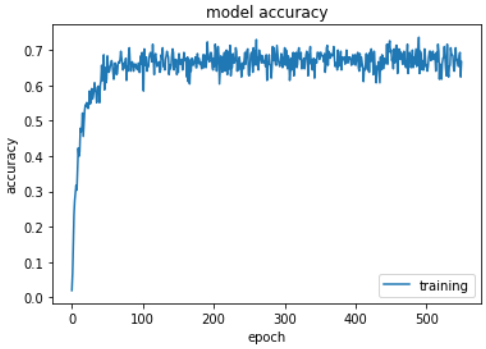

Figure 8: LSTM Accuracy Curve

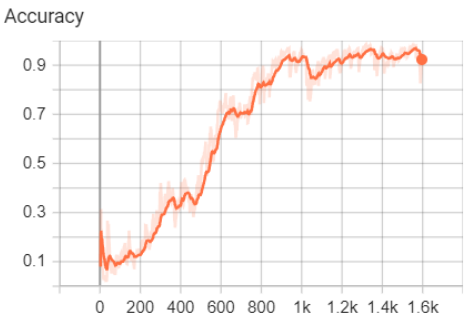

Figure 9: DNN Accuracy Curve

b. Loss Curves

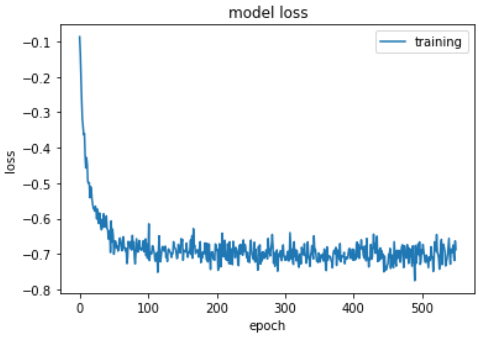

Figure 10: LSTM Loss Curve [10]

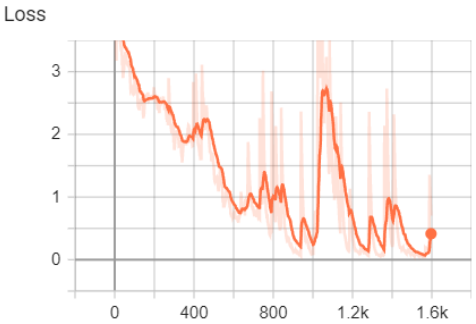

Figure 3: DNN Loss Curve[10]

Similarly, when comparing the loss curves for both models, we notice that the loss curve for the LSTM (figure 10) is much more stable and not as fluctuating as the loss curve of the DNN model (figure 11). Also, the LSTM model reaches the minimum at a much faster rate compared to the DNN model.

c. Testing the Chatbots

We tested both chatbot models using the same set of questions and compared how accurately they were able to predict the correct responses. Both gave us good performance, however, the DNN was able to make much more accurate predictions about the questions it received than the LSTM model. It answered 99% of the questions correctly whereas the LSTM was only able to answer 75% of the questions correctly.

Question: when are the opening hours of the gym?

LSTM Bot: AUB Bookstore is open Monday to Friday, 8:00 am till 5 pm. [Wrong Prediction]

DNN Bot: CHSC opening hours are Monday to Friday from 7:00am till 7:00pm and Saturday from 9am till 5pm. Note that the gym is closed on Sundays. [Correct prediction]

Question: How do I submit a petition?

LSTM Bot: Google 'aub petitions' and it should be the first link you see. [Correct prediction]

DNN Bot: go to the following link: https://epetitions.aub.edu.lb/. [Correct prediction]

Figure 12: Testing the chatbot models

XI. Other Alternatives

With the inefficient use of physical hardware, DNN and LSTM models train sequentially: they process each word in the input on its own, which makes them very slow. Transformers, however, can take all the input in one go, and then using encoders and decoders, figure out the relation between the input and actual output. This makes classification and training much more efficient. A chatbot using transformers will have the questions added as inputs as

a whole, and then each iteration would be the number of questions the dataset has, instead of each word.

The model would take those sentences and have the words mapped to an embedding space where words with similar meanings are physically close to each other. Then to differentiate the context of the words according to their position in a question or answer, the words run through sine and cosine functions (equations (2) and (3) respectively – these equations are plausible proposals to the positional split function). leading to a vectorized form of each question/answer.

$$PE_{(pos,\ 2i)} = sin\left(\frac{pos}{10000^{\frac{2i}{d_{model}}}}\right) \quad (2)$$

$$PE_{(pos,\ 2i+1)} = cos\left(\frac{pos}{10000^{\frac{2i}{d_{model}}}}\right) \quad (3)$$

$$where\ pos = \ the\ position\ of\ the\ word\ in\ the\ sequence$$
$$d = the\ encoding\ vector\ dimension$$
$$and\ i = the\ index\ of\ the\ encoding\ vector$$

Both questions and their actual answers individually run through an "attention block" to give out a vector showing self-attention, or self-relevance, of the words in the same sentences. Only the actual output is decoded in the attention block in reverse. This is important so that the model actually predicts the relations and not just spit out the answers deterministically. The model then catches the relevance of the words of both the input and the actual output, learning higher values in the vector for correct input-output relation, and lower ones for wrong pairs.

The model then uses the softmax activation so that it can be human digestible in probabilistic form. In this way, the chatbot would be able to swiftly and accurately classify the questions to their proper answers, and with a good dataset, make the chatbot successful and efficient for new unseen input. This would definitely mean much faster training times and higher accuracy, for transformers treat all words in the sentence equally while the DNN and LSTM model only see the input position of the words, making words that are far apart in the sentences hard to connect.

We did not use transformers due to the problem requiring a very narrow, but accurate subset of answers. Powerful transformers excel at approximate mappings to larger subspaces of answers, but in domain-specific, narrow mapping situations, a classification of question to answer is a more suitable paradigm for the task at hand. Technically, transformers are LSTM for both input and output, hence making them a bridge between natural language processing and natural language generation. Since our objective is not to generate, but to match, the architecture is not entirely relevant to our needs and hence omitted.

XII. Insights from experiments

While both chatbots were able to provide us with good results, the DNN model had a much better performance compared to the LSTM model. We believe that this might be due to the tuning of the LSTM model, and this can be fixed by adding more neurons or layers to the system, but it will take much more resources and time to test. And because our application is simple and the questions are very repetitive and predictable, it shouldn't need a complex model to implement. Therefore, it is evident that the DNN model should be adopted as a solution to our problem.

XIII. Conclusion and Future work

In this paper, we propose a chatbot as a solution for a very common problem that AUB students face when they first join the university. We tested two chatbot models: the LSTM model and the DNN model. The DNN had a much better performance compared to the LSTM model. Future work might include adjusting learning rates for the models and enhancing the quality of the dataset by allowing it to cover many more topics with more training data. We can also test different embedding methods or even use transformers.

XIV. Acknowledgments

We would like to thank Professor Mariette Awad for explaining the basics of machine learning which was important to our understanding of how building the project should be. Also, Ms. Julia El Zini played a critical role in highlighting the proper way to deal with and build this project, as well as her constant care and effort in helping and aiding in making it successful.

XV. References

[1] Office of Institutional Research and Assessment at the American University of Beirut, *Student Satisfaction Feedback Report*, September 2019. Beirut, Lebanon.

[2] Bhagwat, V. A., Deep Learning for Chatbots. pp. 24–25. San Jose State University

[3] Britz, D., *Deep Learning for Chatbots, Part 1 – Introduction.* Retrieved from: http://www.wildml.com/2016/04/deep-learning-for-chatbots-part-1-introduction

[4] Great Learning Team, *Basics of building an Artificial Intelligence Chatbot.* Retrieved from: https://www.mygreatlearning.com/

[5] Dass, R., *Create your chatbot using Python NLTK.* Retrieved from: https://medium.com/@ritidass29/

[6] Karani, D. (2018). *Introduction to Word Embedding and Word2Vec*. TowardsdataScience

[7] Montavon, G., Samek, W. & Müller, K.-R.(2018). Methods for Interpreting and Understanding Deep Neural Networks. *Digital Signal Processing*. DOI: 10.1016/j.dsp.2017.10.011

[8] Ye, A. (2020) *Essential Neural Network Architectures, Visualized & Explained.* medium.com

[9] Wang, J., Liu, T. (2018) *An LSTM Approach to Short Text Sentiment Classification with Word Embeddings.* The 2018 Conference on Computational Linguistics and Speech Processing ROCLING 2018, pp. 214-223

[10] Tensorboard Results: https://tensorboard.dev/experiment/0IYJzeoKSJWwfaNKboBYOw/#scalars