

Revolutionizing Campus Communication: NLP-Powered University Chatbots

Ritu Ramakrishnan, Priyanka Thangamuthu, Austin Nguyen, Jinzhu Gao

School of Engineering and Computer Science, University of the Pacific, Stockton, California, USA.

Abstract—Artificial intelligence (AI) based chatbots leverage programmed software instructions to simulate human speech and user interaction. These versatile tools can be employed in various domains, from managing smart home devices to providing personal virtual assistants. They can also be useful in responding to common queries and can make information easier to access. In response to this need, we developed a specialized chatbot tailored for the academic environment by training an NLP model to answer frequently asked questions (FAQs) the need of searching through the university website. The main goal is to optimize user engagement and streamline information retrieval within a university setting. By employing ML and NLP techniques, we enhance the chatbot's capabilities, enabling it to provide effective and precise answers, contributing to a more seamless and efficient experience for users seeking information about the university. The study discusses the pivotal decision-making process between implementing a custom neural network and the BERT model. Through a comparative analysis, the custom neural network emerges as the preferred solution, displaying efficiency, quick deployment, and superior accuracy in handling task-specific queries. While BERT presents unparalleled versatility in natural language processing, its resource-intensive pre-training, and challenges in adapting to the intricacies of the university-specific dataset limit its efficiency in this application. This research emphasizes the importance of customization to meet the unique demands of a university chatbot, providing valuable insights for developers seeking to strike a balance between efficiency and specialization in similar applications.

Keywords—Artificial intelligence; natural language processing; chatbot; machine learning; recommender systems; neural network; BERT

I. INTRODUCTION

In the evolving landscape of technology, chatbots have emerged as powerful tools that revolutionize the way individuals interact with systems. Automated response systems designed to engage users in natural and conversational dialogue, chatbots have found applications across various industries. Chatbots offer users a natural language interface which allows them to perform tasks easily without the need for direct human interaction through traditional channels like phone calls or emails.

The rise of artificial intelligence based chatbots capable of engaging users in conversations that mimic realistic human interactions have been able to provide useful insights based on user inputs [1]. As such, the main purpose of this project is to build a chatbot for the university that can provide new students with information in a quick and easy-to-understand manner. As

a new student, it can sometimes be difficult to navigate the university website and get information about things quickly. Because of this, modern NLP (Natural Language Processing) techniques can be leveraged to solve this problem in an effective way. Recognizing the repetitive nature of inquiries faced by administrative staff, we wanted to create a tool that could automate responses to frequently asked questions, providing immediate responses for students, handling multiple requests for information, and freeing up valuable time for administrative officers to focus on other crucial tasks.

The implementation of a chatbot can simplify information retrieval for students, proving invaluable in navigating campus life. Another significant benefit of implementing a chatbot for answering questions is being able to integrate it into other domains. Since there is a single pipeline for creating the chatbot, from collecting data to training the model, it would be relatively simple to create chatbots to help provide information on other topics. The developed chatbot employs Natural Language Processing (NLP) techniques, utilizing basic distance metrics to gauge the similarity between questions [2]. Because of this, it is also possible for the chatbot to be programmed to learn from user interactions. Using a chatbot for education can help with personalizing the learning experience [3]. By using a chatbot to inform students and answer certain questions, it is possible to enhance the experience by having the chatbot adapt based on questions that a student is asking as well as student feedback on the quality of the chatbot's answers.

In this work, we shall deep dive into the entire process of making an effective chatbot for university, from data collection, data processing, data analysis, to model construction. This paper will examine the limitations and advantages of using various approaches to create a chatbot based on university information. The results will demonstrate insights and developments for implementing a chatbot in a specialized domain.

II. RELATED WORKS

In recent developments in educational technology, the implementation of virtual teaching assistants by Natural Language Processing (NLP) has gained attention for its potential to enhance student learning outcomes. A notable study examining this technology's impact in Ghanaian higher education illustrates how NLP, a branch of artificial intelligence focused on interpreting human languages can facilitate more effective communication between students and computers [4]. This study underscores the utility of NLP systems in educational settings to bridge the gap between

natural human communication and machine understanding, thereby enabling a more accessible and interactive learning experience for students. The virtual teaching assistant, functioning as a chatbot, not only aids in answering queries but also significantly contributes to a learning environment where students can engage with course material in a conversational and interactive manner [4]. Such findings are instrumental in highlighting the evolving role of AI in education, particularly in regions with unique educational challenges and opportunities.

A significant barrier to student engagement with learning resources is the social apprehension of seeking help. According to findings presented in study [5], students frequently are reluctant to ask for assistance due to concerns over maintaining a positive social image. This phenomenon drives a preference for self-service information retrieval methods, notably through Frequently Asked Questions (FAQs). Since their origin in Usenet groups in 1982, FAQs have been a pivotal self-help tool, compiling common inquiries alongside their answers to facilitate independent problem-solving [5]. The study highlights the relevance of FAQs in educational settings and suggests opportunity for chatbots and conversational interfaces to bridge the gap in student support by offering a non-judgmental, anonymous platform for information access and learning assistance. This underscores the potential of conversational AI to mitigate social barriers to help-seeking, aligning with broader research trends that explore technology's role in enhancing student learning experiences.

Exploring chatbot technology within educational contexts, particularly in facilitating students' transition phases, a distinction is made between two primary types of chatbots: rule-based and AI-based models. As detailed in the study [6], rule-based chatbots function by adhering to pre-defined rules and delivering responses based on these predetermined guidelines. Conversely, AI-based chatbots leverage artificial intelligence techniques, enabling them to learn and adapt from each interaction. This adaptability allows AI-based models to evolve their response mechanisms over time, offering more personalized and contextually relevant interactions [6]. This classification is pivotal for understanding the potential and limitations of chatbot applications in educational settings, as it highlights the technologies that dictate chatbot behavior and efficacy in supporting students during critical transition periods. It is also important to note that chatbots used in educational settings can have significant ethical implications [7].

The evolution of chatbot technology is a critical area of study within the domain of artificial intelligence applications. A seminal work in this field is [8], which provides a comprehensive historical overview, pointing out the creation of ELIZA in 1964 as a foundational moment in chatbot development. ELIZA, designed by Joseph Weizenbaum, represented the first attempt to simulate human-like conversation by analyzing and decomposing user input responses based on predefined rules. This method of operation explains the rule-based approach to chatbot design, laying the groundwork for subsequent advancements in the field [8]. The historical context provided by this study is for understanding the progression of chatbot capabilities from simple rule-based

interactions to the complex AI-driven conversational agents we see today.

Advancements in chatbot technology have increasingly focused on enhancing the accuracy and relevance of their responses. An innovative method for achieving this is highlighted in [9], which discusses the application of deep reinforcement learning (DRL) techniques to improve chatbot interactions. This approach involves the chatbot proactively identifying gaps in the information provided in user inquiries. By engaging users to clarify these ambiguities and gather the necessary details, the chatbot can tailor its responses more accurately and effectively. This method improves the quality of the chatbot's replies and contributes to a more dynamic and engaging interaction between the chatbot and the user [9]. Such a strategy of DRL in refining the capabilities of chatbots, marking a notable change from more traditional, static methods of response generation.

These are some of the related works for chatbots developed for various purposes. Depending on their purpose, the dataset for the chatbot varies and some of the chatbot also use customized datasets for developing the model.

III. CHATBOT DESIGN

In the university website, all the FAQs are in form of documents and other information that a user might need is available on the website. Sometimes this information is outdated so students must verify if the information is correct from the administrative staff. To resolve this issue using NLP, the chatbot design needs to meet the following criteria:

1) *Intent recognition*: The chatbot should recognize intents specific to university-related queries, such as admissions, student services, campus facilities, dorms, hospital facilities etc. User interactions are classified into intents based on the university-related data.

2) *Response generation*: The chatbot should give clear responses relevant to university policies, procedures, and services with up-to-date information, including links or references to official university webpages or other resources for further clarification.

3) *Context handling*: The chatbot should have context management to maintain continuity in conversations related to university-related topics and allow users to ask follow-up questions or seek clarification within the same conversation context.

4) *User interaction*: The chatbot interface should be user-friendly, considering the unique needs and expectations of university students. It also needs to ensure accessibility features to accommodate a diverse student population.

5) *Error handling*: Error messages should be customized to address common issues faced by students. The chatbot should include prompts for users to rephrase or provide additional details if the chatbot encounters ambiguity.

In addition, the chatbot design needs to ensure scalability and consider the diverse sources of university datasets which can be achieved by the following implementations in the chatbot design process:

1) *User base growth*: The chatbot architecture should be designed to handle more student queries, especially during peak periods like enrollment or exam seasons. The infrastructure should scale based on the expected growth in the student population.

2) *Concurrency and load handling*: The concurrent queries mechanisms should be implemented during high-traffic periods, preventing delays in responses. Queries should be distributed evenly across servers to ensure load balancing.

3) *Adaptability to increasing data*: The chatbot should be trained on new FAQs and updates to university policies regularly. The content management system should be implemented for easy addition and modification of FAQs. The FAQs should have updated information and links for easier navigation for the students.

4) *Monitoring and analytics*: The user interactions should be monitored to identify popular queries and areas needing improvement. Analytics should be performed regularly to understand student engagement and continuously enhance the chatbot's effectiveness. If there is any outdated information, then update the database or the university website where the data is updated and train the model again based on the latest information.

IV. IMPLEMENTATION DETAILS

The development of the university chatbot model follows a strategic process separated into distinct stages aimed at optimizing user interactions and addressing the unique needs of university students. The process started with data collection, which focused on gathering valuable insights into frequently asked questions (FAQs) and user queries specific to the university. The subsequent stages include natural language processing (NLP) techniques, neural network implementation, and real-time interaction simulation.

Through this systematic approach, the university chatbot model emerges as a sophisticated tool designed to provide smoother experience in seeking university information instead of navigating a complex university website. Let us investigate the stages of developing the university chatbot in upcoming paragraphs. Fig. 1 explains the flow of the chatbot model.

A. Data Collection

The initial phase in developing our university chatbot model focused on a comprehensive data collection effort. This step was critical for ensuring the chatbot could accurately understand and respond to university student's diverse needs and queries. The data collection process we conducted is detailed below.

1) *Choosing a data format*: The first step of data collection involved deciding on what data format to use for developing the chatbot. After considering multiple file formats, such as plain text, CSV (Comma-separated values), and JSON (JavaScript Object Notation), we eventually settled on using a JSON format so that it could be easier to categorize various pieces of data. The last version of the dataset ended up being organized by having intents to help classify the various

categories of data that were collected. Some examples of these categories would be for international students or orientation for new students. Each category was made to contain a list where each item in the list has a question and an answer along with the corresponding question and answer tags.

2) *Collecting data from webpages*: Data was mostly collected from webpages related to the university. We first started creating a list of URLs (Uniform Resource Locators) from which to get data. This list of URLs is selected based on webpages with valuable information relating to the university and then adding related links from each of those webpages. Since there are many links collected from the webpages, only a fraction (about one-fifth) of all the internal webpage links were added to the list of URLs. After this, data was collected from the list of URLs by parsing through the HTML (Hypertext Markup Language) code from each URL.

The resulting HTML code was then parsed so that the text collected comes from the HTML body tag. Even after doing this step, there were still pieces of extraneous HTML and JavaScript code leftover that had to be cleaned before it could be used for to train the chatbot model. This data cleaning was done with regex functions to help remove certain code and information that was not needed for the chatbot. To collect relevant data in the form of questions and answers, another set of regex functions were used to classify sentences as either questions or answers, identifying the categories that they fall under, and then adding them accordingly to the data set which is related to university data that can help the students.

3) *Collecting data from webpages with frequently asked questions*: The main goal was to develop a chatbot that could respond to frequently asked questions about the university. Thus, we also manually gathered documents with information in the form of questions followed by answers and web pages from the university's website. The chatbot could use this information to learn how to answer frequently asked questions.

The specific web pages were converted into PDF files, which have a more organized layout that is better suited for parsing. Effective question-answer pairs were taken from the PDFs and converted into JSON objects with additional descriptive tags that explained the subject and context of each conversation.

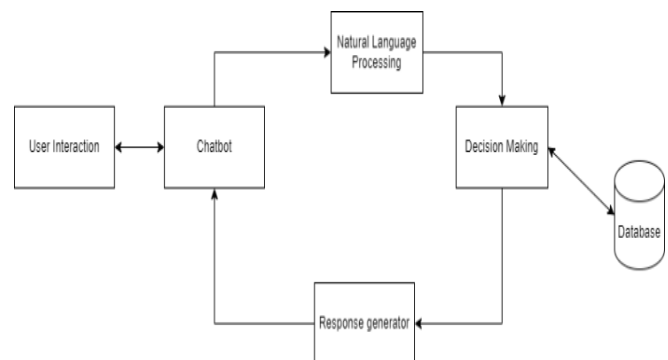


Fig. 1. Chatbot model diagram.

However, most university FAQ pages require users to click on them to reveal the answers, which are initially hidden. As a result, the parsed PDF-derived JSON contained only questions and few answers. The chatbot requires whole sets of questions and their corresponding answers to learn how to answer questions correctly. To close these gaps in the data, more of the question-and-answer data that was previously scraped from websites was added. This resulted in a training data set that was more comprehensive.

B. Data Processing

After the comprehensive collection of the required data, the data processing stage plays a pivotal role in refining and preparing this data for the model building. This stage ensures the data is optimal for training the chatbot model, enhancing its ability to accurately understand and respond to user inquiries. We used the following techniques to pre-process data.

1) *Tokenization*: Tokenization involves breaking down the text into smaller units, typically words. The word tokenize function from NLTK (Natural Language Toolkit) is employed for this purpose for the chatbot. It effectively tokenizes the text into a list of words in the input dataset into meaningful words.

2) *Remove stopwords*: While looking into the dataset we found a lot of stop words. Stop words, such as "the", "and", and "is", are commonly used words that do not carry significant meaning in certain contexts. So, we used the stop words module from NLTK to remove the stop words from the dataset.

3) *Lemmatization*: The next process we used was the Lemmatization process to reduce words to their base or root form. In this case, NLTK's WordNet Lemmatizer module was applied to each word in the list. This helped in reducing inflected words to a common base form, enhancing the accuracy of subsequent analyses.

4) *Calculating the frequency*: The frequencies of words used in meaningful text from the dataset was used to classify it into the intents. These intents were used for the model to process the information. Fig. 2 shows the word cloud of the frequently used words in the dataset.



Fig. 2. Frequently used words in the dataset.

5) *Defining the dataset*: Data was collected in the form of questions and answers and were split based on the intents in the questions. For example, if the questions are related to housing, then the intent is classified as "housing", and all the questions and answers are added to that list in the JSON format. Diverse ways of asking a question were also added to

the data set to have more data to improve the chatbot's accuracy. Fig. 3 shows sample of the intent, questions, and answers which were classified and used in the JSON format as part of the dataset.

```
{
  "tag": "on_campus_housing",
  "questions": [
    "Is on-campus housing available?",
    "What housing options does the University of Pacific's Sacramento campus offer?",
    "How can I apply for on-campus housing?"
  ],
  "answer": "University of Pacific's Sacramento campus offers a variety of on-campus housing opportunities."
},
```

Fig. 3. Sample input dataset.

6) *Tools and Languages*: The tools and libraries used for developing the chatbot model are described in Table I below.

TABLE I. PROGRAMMING LANGUAGE AND TOOLS USED FOR CHATBOT

Tools/Language	Description
Python	Primary programming language for data processing, natural language processing (NLP), and machine learning.
NLTK (Natural Language Toolkit)	Python library for working with human language data. Used for tokenization, stopwords removal, and lemmatization.
re (Regular Expressions)	Python module for working with regular expressions. Utilized for pattern matching and removal of JavaScript functions from the input text.
pickle	Python module for serializing and deserializing objects. Used for saving and loading processed data.
stopwords	A set of common stopwords provided by the nltk library. Employed for removing non-informative words from the text.
Neural Network	Deep learning model used for tasks such as classification, regression, and pattern recognition in the context of natural language processing. It is versatile and can be customized for various applications.
BERT (Bidirectional Encoder Representations from Transformers)	Pre-trained transformer-based model for natural language understanding. BERT is especially powerful in capturing context and semantics, making it suitable for various NLP tasks such as question answering, sentiment analysis, and more.
Jupyter Notebook	Interactive computing environment for creating and sharing documents containing live code, equations, visualizations, and narrative text. Widely used for data exploration, analysis, and presentation.

C. Models

We developed two distinct models to harness the full potential of artificial intelligence for enhancing the user interaction and response accuracy. These models were chosen for their strengths in understanding and processing natural language queries and their ability to learn from interactions to improve over time. Below, we detail the implementation and contributions of each model to the chatbot's development, illustrating how they collectively form a sophisticated system capable of addressing the needs of university students.

1) *BERT model*: The first cornerstone of our chatbot's architecture is based on BERT (Bidirectional Encoder Representations from Transformers), a state-of-the-art model developed by Google. Fig. 4 shows the code we used for the

chatbot using the BERT model and the detailed explanation is listed in the below:

a) *Transformer encoders:* The chatbot utilizes Transformer Encoders, a powerful neural network architecture, to process and understand the input text. This architecture is particularly effective for natural language processing, allowing the chatbot to capture relationships within sentences.

b) *Bidirectional:* BERT is bidirectional. This means it comprehensively considers the context from both the left and right sides of each word in a sentence. This bidirectional approach enhances the chatbot's understanding of the overall meaning and context of user queries.

c) *Masked language model:* During this pre-training phase, the model learns by predicting masked words in sentences based on their surrounding context. This enables the chatbot to grasp nuanced meanings and context in user inputs.

d) *Pretrained model:* The chatbot is built upon a pretrained BERT model, first exposed to a vast amount of diverse textual data. This pre-training equips the chatbot with a solid foundation in general language understanding. The pretrained model is then fine-tuned to specialize in handling specific inquiries relevant to the university setting.

e) *Fine tuning:* Fine-tuning is the process where our chatbot refines its pretrained model for the university context. By training on labeled data specific to frequently asked questions (FAQs) and university-related tasks, the chatbot adapts its language understanding capabilities to better assist students.

f) *Attention layers:* BERT's attention layers are integral to our chatbot's functionality. These layers enable the chatbot to focus on various parts of user inputs, allowing it to comprehend long-range dependencies and relationships between words. This attention to context enhances the chatbot's ability to provide accurate and relevant responses.

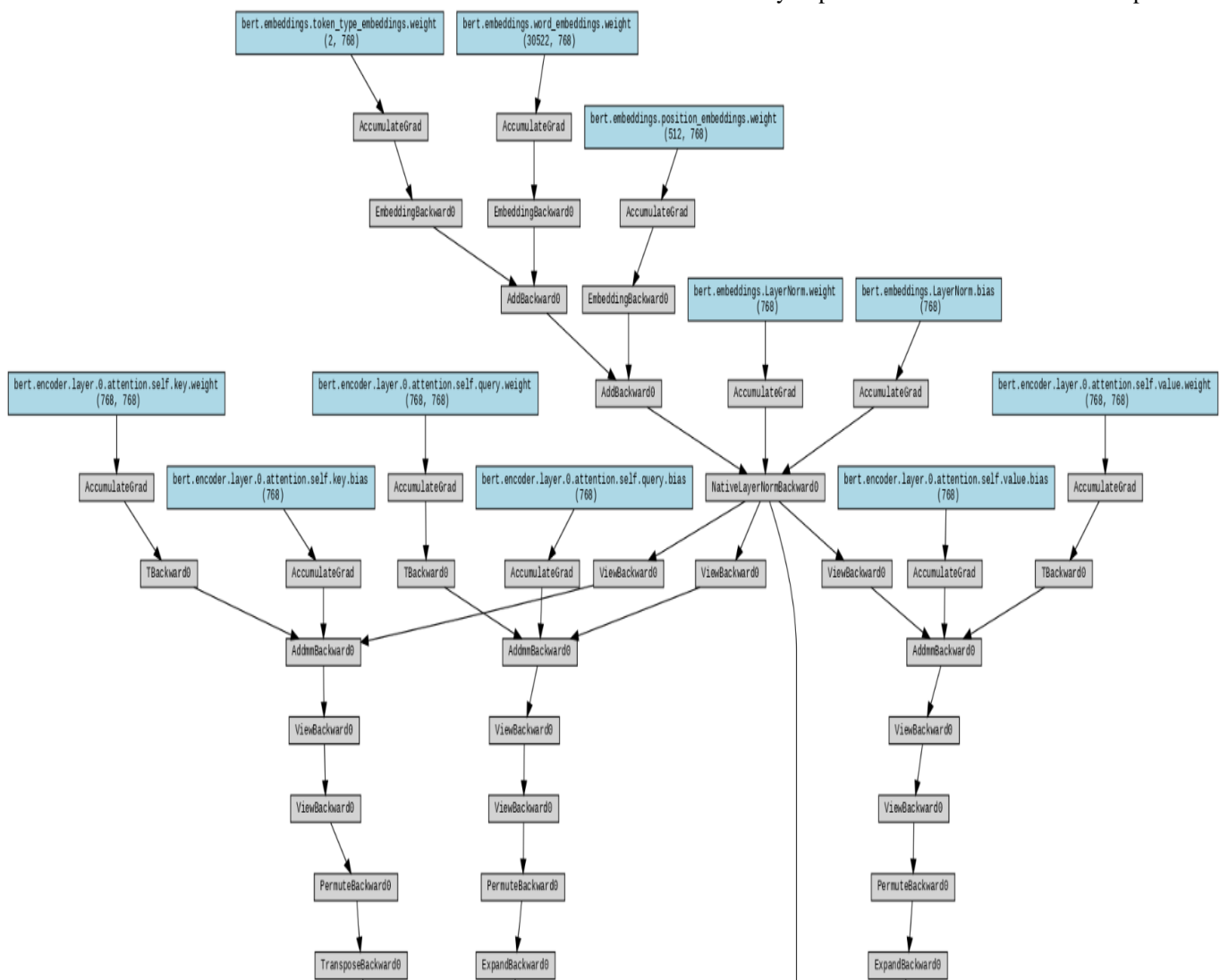


Fig. 4. BERT Model diagram.

```
# Create a Hugging Face datasets.Dataset directly from the list
dataset = Dataset.from_dict({"text": [item["text"] for item in dataset_list],
                             "label": [item["label"] for item in dataset_list]})

# Split the dataset into train and validation (adjust split ratio as needed)
split_ratio = 0.8
train_dataset = dataset.select(range(int(split_ratio * len(dataset))))
validation_dataset = dataset.select(range(int(split_ratio * len(dataset)), len(dataset)))

# BERT model and tokenizer
model_name = "bert-base-uncased"
tokenizer = BertTokenizer.from_pretrained(model_name)

# Tokenize input texts within the datasets with explicit padding
def tokenize_function(examples):
    return tokenizer(examples["text"], padding="max_length", truncation=True,
                    return_tensors="pt", max_length=128)

tokenized_train_dataset = train_dataset.map(tokenize_function, batched=True)
tokenized_validation_dataset = validation_dataset.map(tokenize_function, batched=True)
```

Fig. 5. BERT model code.

Fig. 5 shows part of the visualization of the BERT model for chatbot.

2) *Neural network model*: Building on the foundational capabilities provided by BERT for understanding natural language, our second model employs Neural Networks to further refine and personalize the chatbot's responses to user queries. Neural Networks, with their remarkable ability to model complex patterns and relationships within large datasets, are particularly well-suited for enhancing the predictive accuracy and response quality of our chatbot. This model leverages a sophisticated architecture designed to process, learn from, and respond to the nuanced inquiries presented by university students. Fig. 6 shows the code for the custom neural network model which we used to develop this chatbot.

a) *Three-layered neural network*: As shown in Fig. 6, the neural network architecture comprises of three main layers, which are the input layer, two hidden layers, and an output layer. The input layer receives the features from the preprocessed input data, while the hidden layers process and extract the patterns. The output layer produces the results, representing the probabilities of different classes (e.g., intent tags in the chatbot).

b) *ReLU activations, batch normalization, and dropout*: Rectified Linear Unit (ReLU) activations introduce non-linearity to the model, allowing it to learn complex

relationships within the data. Batch normalization helps in stabilizing and accelerating the training process by normalizing inputs. Dropout is employed during training to prevent overfitting by randomly deactivating a proportion of neurons, enhancing the model's robustness.

c) *Softmax activation*: The SoftMax activation function is applied to the output layer. This function converts the raw output scores into probability distributions across different output classes. In the context of the chatbot, it produces probabilities for the possible intents, helping the model make informed predictions.

d) *Custom dataset class*: A custom dataset class is employed for effective management of input data and corresponding labels as this dataset is used to build the chatbot. This class manages training and testing data, making it compatible with the neural network model. It allows for efficient loading, preprocessing, and batching of data during training.

Overall, the neural network architecture is designed with layers that process input features, introduce non-linearity for better learning, and produce probability scores through SoftMax activation. The model undergoes transfer learning using pre-trained models like BERT for efficiency. The custom dataset class ensures handling of data, contributing to the overall effectiveness of the chatbot in understanding and responding to user queries. The neural network model diagram is shown below in Fig. 7.

```
class NeuralNetwork(nn.Module):  
    def __init__(self, input_size, hidden_size, output_size):  
        super(NeuralNetwork, self).__init__()  
        self.fc1 = nn.Linear(input_size, hidden_size)  
        self.relu1 = nn.ReLU()  
        self.bn1 = nn.BatchNorm1d(hidden_size)  
        self.dropout1 = nn.Dropout(0.2)  
        self.fc2 = nn.Linear(hidden_size, hidden_size)  
        self.relu2 = nn.ReLU()  
        self.bn2 = nn.BatchNorm1d(hidden_size)  
        self.dropout2 = nn.Dropout(0.2)  
        self.fc3 = nn.Linear(hidden_size, output_size)  
        self.softmax = nn.Softmax(dim=1)  
  
    def forward(self, x):  
        x = self.fc1(x)  
        x = self.relu1(x)  
        x = self.bn1(x)  
        x = self.dropout1(x)  
        x = self.fc2(x)  
        x = self.relu2(x)  
        x = self.bn2(x)  
        x = self.dropout2(x)  
        x = self.fc3(x)  
        output = self.softmax(x)  
        return output
```

Fig. 6. Custom neural network code.

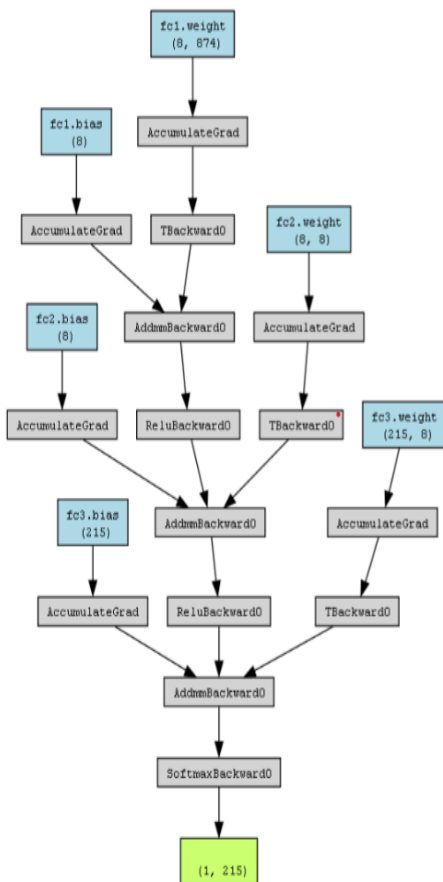


Fig. 7. Custom neural network model diagram.

D. Training and Validation

Transfer learning uses pre-trained models to accelerate the training process. The chatbot benefits from the knowledge acquired by the pre-trained model on vast amounts of general language data. This approach is efficient, especially when dealing with limited labeled data specific to the university context.

1) *BERT model*: We used the BERT uncased model, which makes better decisions on answering questions. In the fine-tuning and training configuration, the BERT undergoes an adaptive training process with epochs ranging from 3 to 50. Each epoch represents a complete pass through the entire training dataset, allowing the model to refine its parameters over multiple iterations. The Adam optimizer, known for its adaptive learning rate and effectiveness in deep learning tasks, is employed to optimize the model during this process. During the training progress, key checkpoints are observed. After 500 steps, the training loss stands at a low value of 0.0137, indicating favorable predictive accuracy on the training dataset. Fig. 8 shows the code for the BERT model training arguments.

However, the validation loss at this point is 4.98, suggesting potential room for improvement in generalization to unseen data. Subsequently, after 660 steps, the training loss has increased to 0.28, and the validation loss has risen to 5.04. These increments in loss metrics could signal a risk of overfitting, where the model may be too closely tailored to the training data. The training and validation lost for the BERT model is shown in Fig. 9.

```
# Set up training arguments and trainer
training_args = TrainingArguments(
    output_dir="./results",
    overwrite_output_dir=True,
    num_train_epochs=20,
    per_device_train_batch_size=16,
    per_device_eval_batch_size=16,
    save_steps=10_000,
    save_total_limit=2,
    evaluation_strategy="steps",
    eval_steps=500,
    logging_steps=100,
)

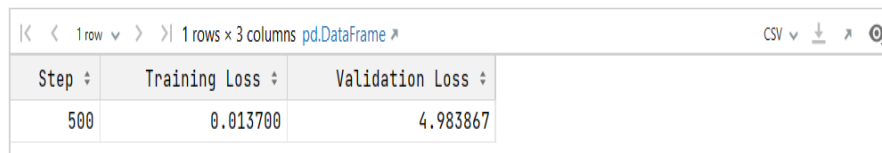
# Ensure labels are properly set
tokenized_train_dataset = tokenized_train_dataset.remove_columns(["text"])
tokenized_train_dataset.set_format("torch", columns=["input_ids", "attention_mask", "label"])

tokenized_validation_dataset = tokenized_validation_dataset.remove_columns(["text"])
tokenized_validation_dataset.set_format("torch", columns=["input_ids", "attention_mask", "label"])

# BERT model and tokenizer
model_name = "bert-base-uncased"
model = BertForSequenceClassification.from_pretrained(model_name, num_labels=len(label_mapping))
```

Fig. 8. BERT model training arguments.

Some weights of BertForSequenceClassification were not initialized from the model checkpoint at bert-base-uncased and are newly initialized: ['classifier.bias', 'classifier.weight']
You should probably TRAIN this model on a down-stream task to be able to use it for predictions and inference.



Step	Training Loss	Validation Loss
500	0.013700	4.983867

```
TrainOutput(global_step=660, training_loss=0.2832737333846815, metrics={'train_runtime':  
73.9241, 'train_samples_per_second': 139.603, 'train_steps_per_second': 8.928, 'total_flos':  
679009370112000.0, 'train_loss': 0.2832737333846815, 'epoch': 20.0})
```

Fig. 9. BERT model training and validation loss diagram.

2) *Neural network model*: The model training configuration involves a batch size of 100, which means that the neural network processes 100 data samples in each training iteration. The architecture comprises two hidden layers with Rectified Linear Unit (ReLU) activations, batch normalization, dropout for regularization, and a SoftMax activation for classification. Training is conducted for 100 epochs, indicating the model iterates over the entire dataset 100 times. The Data Loader is utilized to efficiently handle

batches during training. Fig. 10 shows the training code for the custom neural network in the chatbot with the data loader used for the training and validation.

After completion, the trained model is saved as "model.pth" for future development purposes, allowing easy retrieval and deployment for subsequent tasks without retraining the model. Fig. 11 shows the training and validation loss for the custom neural network model.


```
# Create DataLoader for training and testing data
train_x = torch.tensor(train_x).float()
train_y = torch.tensor(train_y).float()
test_x = torch.tensor(test_x).float()
test_y = torch.tensor(test_y).float()

batch_size = 100
train_dataset = CustomDataset(train_x, train_y)
test_dataset = CustomDataset(test_x, test_y)
train_loader = DataLoader(train_dataset, batch_size=batch_size, shuffle=True)
test_loader = DataLoader(test_dataset, batch_size=batch_size, shuffle=False)

# Define the model, loss function, and optimizer
input_size = len(train_x[0])
hidden_size = 8
output_size = len(train_y[0])
model = NeuralNetwork(input_size, hidden_size, output_size)
criterion = nn.BCELoss()
optimizer = optim.Adam(model.parameters())
```

Fig. 10. Custom neural network data loader for training and testing data.

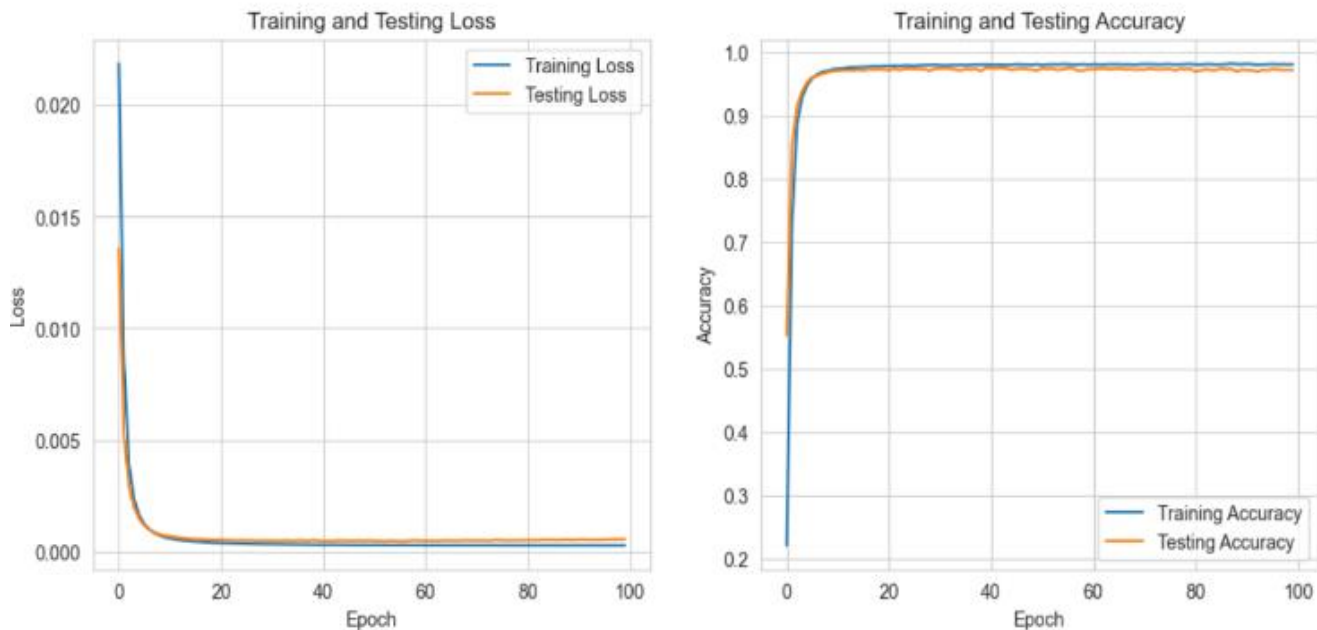


Fig. 11. Custom neural network model training and validation loss.

V. RESULTS

In the comparative evaluation of the BERT model and the neural network, the results reveal that the neural network achieved higher accuracy for the chatbot. The BERT model, while powerful in its natural language processing capabilities, did not perform as well as the neural network. To conclude these results, we collected the dataset from the University of The Pacific website for the training and validating purpose of the chatbot model. As the university data set was small and had customized information, the BERT model was not able to perform well, and the results were not accurate. The accuracy of the BERT model was 0.87 percent. As shown in Fig. 9 after 660 steps, the training loss has increased to 0.28, and the validation loss has risen to 5.04 which were resulting in the overfitting of the model.

Training Accuracy: 0.9811

Testing Accuracy: 0.9715

NeuralNetwork(

(fc1): Linear(in_features=874, out_features=8, bias=True)

(relu1): ReLU()

(bn1): BatchNorm1d(8, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)

(dropout1): Dropout(p=0.2, inplace=False)

(fc2): Linear(in_features=8, out_features=8, bias=True)

(relu2): ReLU()

(bn2): BatchNorm1d(8, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)

(dropout2): Dropout(p=0.2, inplace=False)

Fig. 12. Custom neural network model accuracy.

VI. DISCUSSION

The Flask library from Python is used to connect the model to UI. The chatbot application is developed using HTML, CSS and JavaScript and it is connected to the backend with the Flask library. The model is saved in a particular path and uses the saved model to give a response based on the user interactions. Some of the examples for the user interactions are shown in Fig. 13.

In developing the chatbot for university students, the choice between a custom neural network and the BERT model is important. The research's primary objective is to optimize user engagement and information retrieval within a university setting FAQs. The comparative analysis revealed that the custom neural network outperformed BERT in this specific application. The neural network's efficiency in handling task-specific FAQs and quick deployment made it an ideal choice

for the streamlined nature of the university chatbot. On the other hand, the neural network model training time was less when compared to the BERT model. This outcome emphasizes the importance of selecting the most suitable model for a given task and dataset. While BERT is a state-of-the-art model for various natural language processing tasks, the neural network, tailored to the specific requirements of the chatbot application, emerged as the most accurate solution in this chatbot. The accuracy of the custom neural network model was 0.98 percent and the answers to the queries were accurate while evaluating the model.

Fig. 12 shows the accuracy of the model while training it for the chatbot using the neural network model.

for the streamlined nature of the university chatbot. On the contrary, BERT's versatility and advanced language comprehension capabilities were weighed against its resource-intensive pre-training and challenges in adapting to the university-specific dataset. While BERT is a powerful model with a broad range of applications in natural language processing, the neural network, tailored to the specific requirements of the university chatbot, demonstrated superior accuracy in handling FAQs. The current data set can be improved by adding more information about the university and the general queries of the students. This will increase the dataset and model will have large amount of information about the university. For future improvements we can look more into the dataset and improve it to increase the efficiency of the model. Also, we can have some feedback system from the students to update the information.

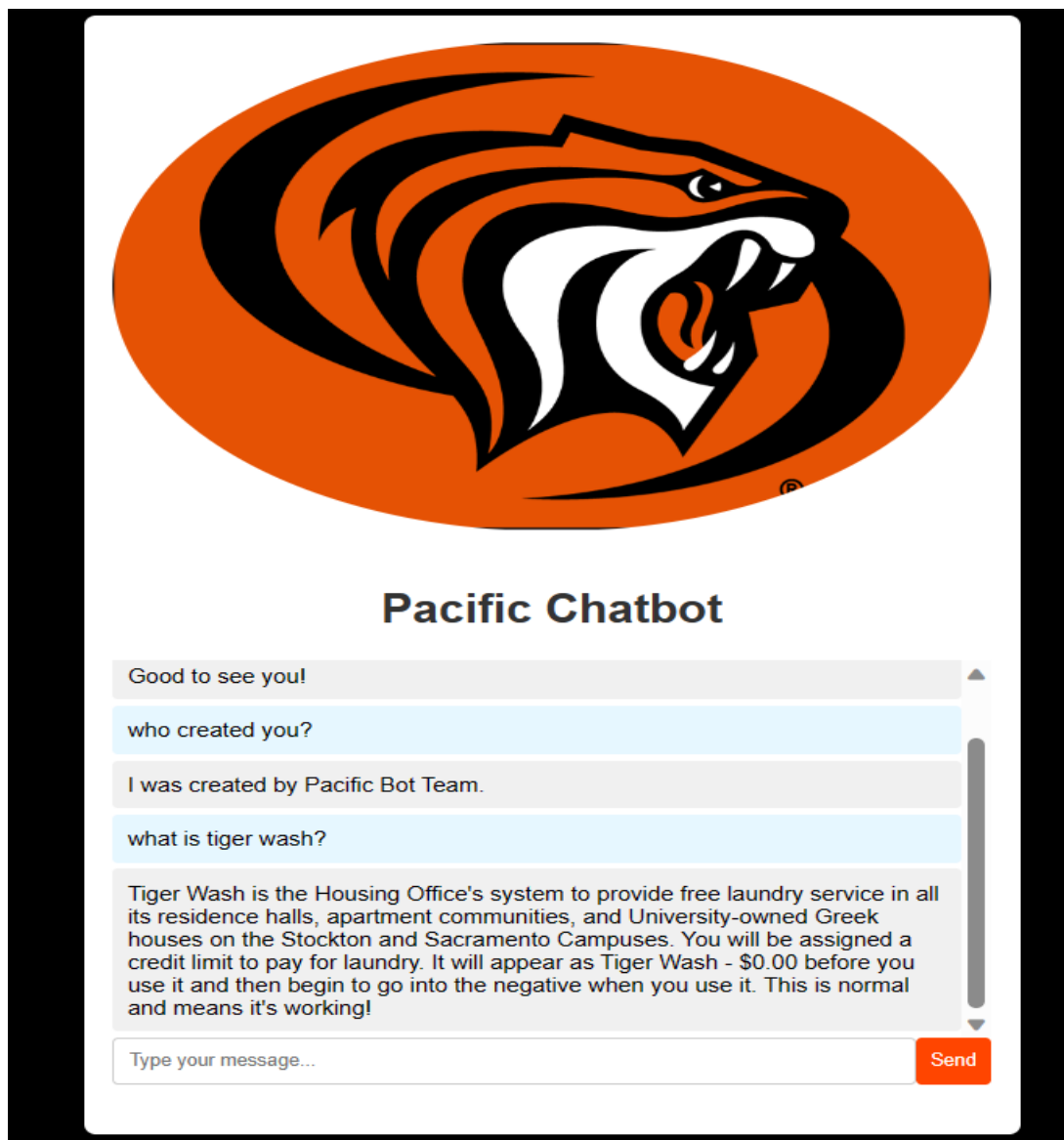


Fig. 13. Chatbot UI application.

VII. CONCLUSION

In conclusion, the development of the chatbot for university students involved a critical decision-making process regarding the choice between a custom neural network and the BERT model. The comparative analysis illustrates the strengths and challenges associated with each model. The custom neural network, tailored for task-specific applications with a small amount of data ended up being suitable for developing this university chatbot. It also demonstrated superior efficiency and accuracy in handling frequently asked questions (FAQs). On the other hand, while BERT exhibited unparalleled versatility and advanced language comprehension, its resource-intensive pre-training, and challenges in adapting to the university specific dataset hindered its performance for this specific task. The research underscores the significance of customization to meet the unique demands of a university chatbot, highlighting the practicality of opting for a more specialized solution over a state-of-the-art model.

To improve the chatbot's understanding and responsiveness to user queries, a hybrid approach that combines custom neural networks and advanced models like BERT could be explored. It is also necessary to focus on improving the adaptability of advanced models like BERT to specific domains such as university administration. Further research could be directed towards refining the chatbot's capabilities based on ongoing user feedback and evolving student needs. Additionally, integrating the chatbot with existing university systems and platforms could enhance its utility and seamless integration into students' daily lives. Future enhancements could extend the chatbot's functionality beyond basic information retrieval to include personalized recommendations, scheduling assistance, and academic support services. Through continuous refinement and innovative approaches, we aim to ensure the chatbot's effectiveness as a valuable resource for university students in the future.

REFERENCES

- [1] E. Abu Shavar, B.A., Chatbots: are they really useful? (2007).Essel, H.B., Vlachopoulos, D., Tachie-Menson, A. et al. The impact of a virtual teaching assistant (chatbot) on students' learning in Ghanaian higher education. *Int J Educ Technol High Educ* 19, 57 (2022). <https://doi.org/10.1186/s41239-022-00362-6>
- [2] Verma, Abhigya and Kuntala, Chandana and Khatri, Pragya and ., Sristi and Kaur, Sukhmani and Mohapatra, A. K. and Singhal, Shweta, University Chatbot System Using Nlp. <http://dx.doi.org/10.2139/ssrn.4255753>
- [3] Hsu, Ting-Chia, Hsiu-Ling Huang, Gwo-Jen Hwang, and Mu-Sheng Chen. "Effects of Incorporating an Expert Decision-Making Mechanism into Chatbots on Students' Achievement, Enjoyment, and Anxiety." *Educational Technology & Society* 26, no. 1 (2023): 218–31. <https://www.jstor.org/stable/48707978>.
- [4] M. McTear, Z. Callejas, D. Griol, The conversational interface: Talking to smart devices (2016).
- [5] S. Valtolina, B. R. Barricelli, S. Di Gaetano and P. Diliberto, "Chatbots and Conversational Interfaces: Three Domains of Use", 2018.
- [6] S. Carayannopoulos, Using chatbots to aid transition (2018).
- [7] Kooli, Chokri. 2023. "Chatbots in Education and Research: A Critical Examination of Ethical Implications and Solutions" *Sustainability* 15, no. 7: 5614. <https://doi.org/10.3390/su15075614>
- [8] R. DALE, "The return of the chatbots," *Natural Language Engineering*, vol. 22, no. 5, pp. 811–817, 2016. doi:10.1017/S1351324916000243
- [9] Serban, Iulian & Sankar, Chinnadhurai & Germain, Mathieu & Zhang, Saizheng & Lin, Zhouhan & Subramanian, Sandeep & Kim, Taesup & Pieper, Michael & Chandar, Sarath & Ke, Nan & Rajeswar, Sai & Brebisson, Alexandre & Sotelo, Jose & Suhubdy, Dendi & Michalski, Vincent & Nguyen, Alexandre & Pineau, Joelle & Bengio, Y.. (2018). A Deep Reinforcement Learning Chatbot (Short Version).