COURSE RECOMMENDATION CHATBOT

GROUP 3

Project Mentor: Professor Babatunde Giwa

Project Members:

Shivraj Dhumal

Nonso Ukwuma

Javid Sadr

Vinay Malik

Yukti Kakkar

CONTENTS

INTRODUCTION 3

TECHNICAL DESCRIPTIONS 3

- WEB SCRAPER 4
- INTENT FILE GENERATOR 6
- THE CHATBOT NLP MODEL 7
- DJANGO WEB APPLICATION WITH API - BACKEND 11
- REACT WEB APPLICATION – THE FRONTEND 12
- GANTT CHART 13

CONCLUSION 14

- FUTURE IMPLEMENTATIONS 14
- REFERENCES 14

INTRODUCTION

The project is about an AI chatbot that provides the user with course information when asked with questions related to course names. The chatbot's AI capability is through an NLP machine learning model which uses LSTM as one of the most effective layers. The data for training the chatbot is collected through a custom-made web scraping spider. The chatbot is built using React frontend framework while The NLP model is deployed on a VPS server on a Django web application where the API endpoint is made available to be consumed by the React frontend chat widget. This project manifests an end-to-end application development cycle involving building, training and deploying a machine learning model and the related application for a limited but live use. It contains potential to develop it further for a more sophisticated conversation with the chatbot.

TECHNICAL DESCRIPTIONS

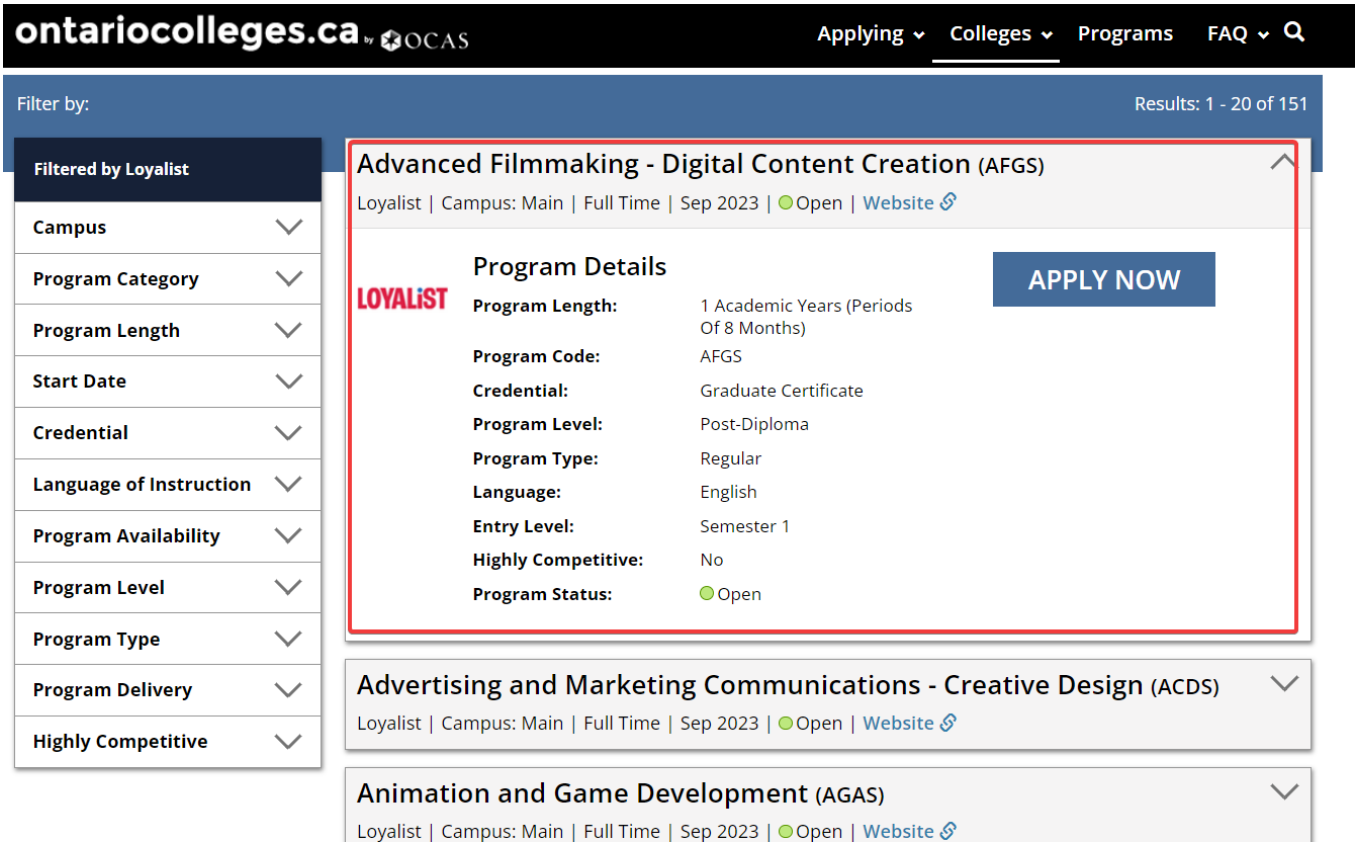
The project is made up of various components that are essential for the end to end implementation of the AI chatbot. These components are as below

1. Web Scraper – To collect Loyalist course information data from https://www.ontariocolleges.ca/en/colleges/loyalist
2. Intent File generator – A custom python script that generates the intent file
3. The NLP Model – a machine learning model which forms the brain of the Chatbot
4. Django Web application with API – The backend
5. React web application – The frontend

All the above 4 main project components are explained in useful detail in the following sub sections.

WEB SCRAPER

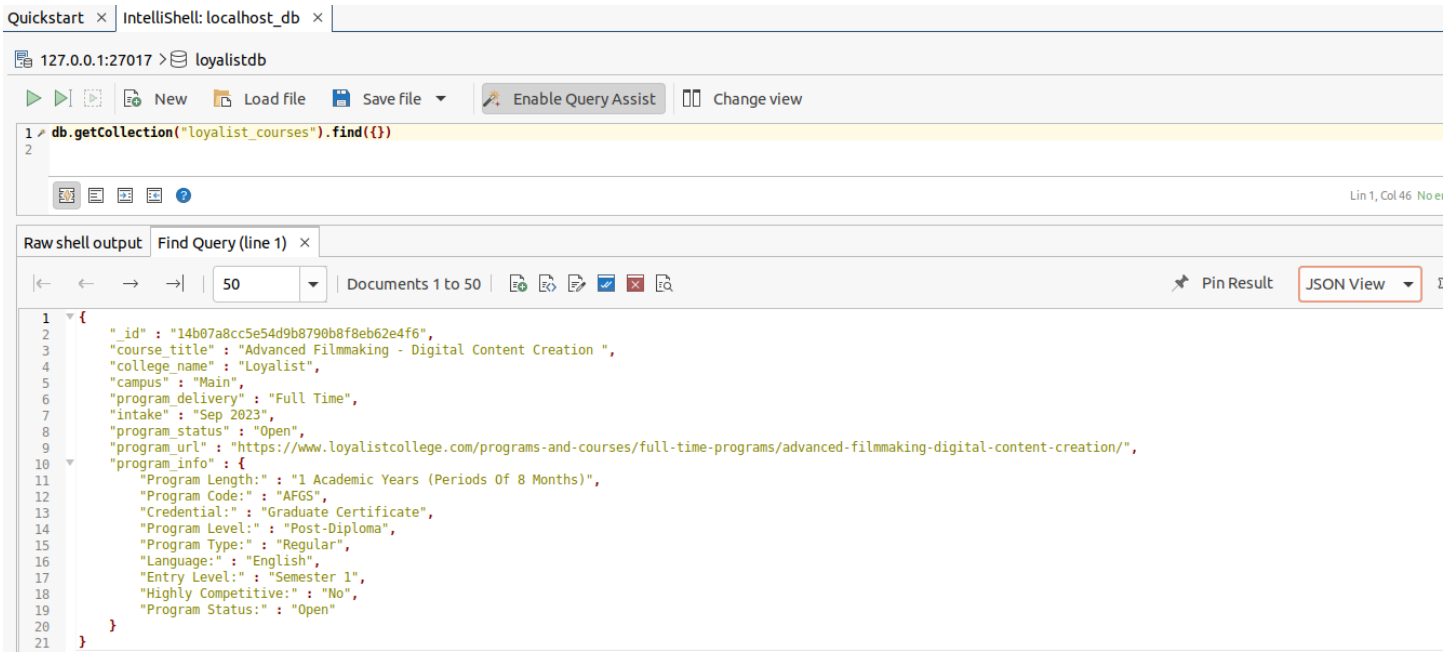
The web scraper also commonly called Spider is built using Python Scrapy. It also makes use of Selenium within the Spider to enable loading of lazy loading data which only loads after a page scroll. The spider starts at the main URL https://www.ontariocolleges.ca/en/colleges/loyalist and scrapes the course information across all the pages by loading one page after the other. The spider scrapes useful information as highlighted in the red box in the screenshot below like the Course Title, College Name, Campus, Program Length, Intake, Program Availability, and all the details from the Program Details section.



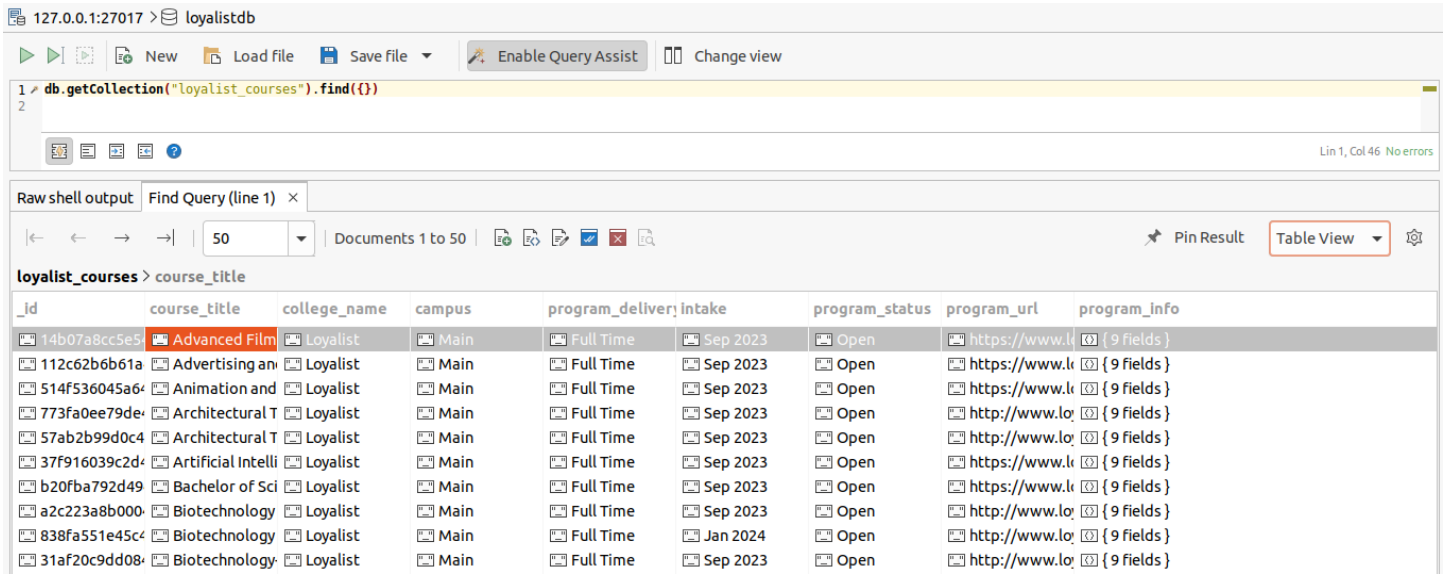
The spider then saves the information in a No-SQL MongoDB database for easy and ready accessibility to the loyalist course data.

The following screenshot gives a snapshot of the data stored in the MongoDB database.

JSON View of the Scraped Data



Tabular view of the Scraped Data



MongoDB provides an easy-to-use shell command to export the database in the JSON format. The JSON data export of the loyalist courses collection is then stored in a JSON file which serves as an input to the next component – The Intent Generator.

INTENT FILE GENERATOR

The Intent File Generator is a custom developed python tool that is used to generate intents JSON file which serves as a corpus of data of the user input patterns and respective responses categorized by several tags. The Intent File Generator takes two input files. The first one is the Manually created Intent file with some basic conversations that the bot should handle. The second one is the courses information JSON file exported from the MongoDB database. These two files go as an input to the Intent File Generator to generate the final intent JSON file.

The Manually created / managed intent file (input file 1) content looks like the screenshot below:


```
{
    "tag": "Identity",
    "patterns": [
        "What's your name?",
        "Who are you?",
        "Can you introduce yourself?",
        "who are",
        "you"
    ],
    "responses": [
        "My name is LChatBot. I am an AI chatbot designed to assist and communicate with
        humans.",
        "I am LChatBot, I can help you with any questions or concerns you may have.",
        "Hi there! I am LChatBot, the friendly chatbot. How can I assist you today?"
    ]
},
```


The course information corpus created by the Intent File generator looks like in this screenshot below


```
"tag": "course_title Advanced Filmmaking - Digital Content Creation ",
"patterns": [
    "tell me more about Advanced Filmmaking - Digital Content Creation ",
    "do you teach Advanced Filmmaking - Digital Content Creation ",
    "i would like to learn Advanced Filmmaking - Digital Content Creation ",
    "Advanced Filmmaking - Digital Content Creation  ",
    "i would like to learn Advanced Filmmaking - Digital Content Creation  ",
    "i would like to learn "
],
"responses": [
    {
        "response": {
            "message": "here you go",
            "course_details": {
                "course_title": "Advanced Filmmaking - Digital Content Creation ",
                "college_name": "Loyalist",
                "campus": "Main",
                "program_delivery": "Full Time",
                "intake": "Sep 2023",
                "program_status": "Open",
                "program_url": "https://www.loyalistcollege.com/programs-and-courses/
                full-time-programs/advanced-filmmaking-digital-content-creation/"
            },
```


The final generated Intent File consists of about **16000** lines of content which will form the corpus of the Chatbot's NLP model.

THE CHATBOT NLP MODEL

The NLP model forms the core component of the chatbot application. Without the model, the chatbot application is just a thoughtless answering machine if at all it could answer the user queries. The model which itself is built in Python essentially makes use of readily available Tensorflow and Scikit-learn Python libraries mainly.

The NLP model contains two main parts or phases, the first one is the **training phase** and the second is the **interaction phase**.

The two phases of the chatbot are explained in detail in the following sub sections.

TRAINING PHASE

Data Loading

The training phase of the model begins with loading the Intents JSON file that was generated by the Intent File Generator tool.

Data Pre-processing and Cleaning

The user input patterns and the tags both form the features to train the model. Firstly, the features corpus is tokenized using Keras Tokenizer and then the method 'fit_on_texts' is called on the tokenizer object that creates the vocabulary index based on the corpus.

```
tokenizer.fit_on_texts(sentence)
tokenizer.word_index
```

```
{'deep': 1, 'do': 5, 'i': 3, 'learning': 2, 'like': 7, 'love': 4, 'you': 6}
```

This is then followed with applying 'texts_to_sequences' on the data to convert texts to a sequence of integers to enable computer to better understand the data.

```
new_sentence=tokenizer.texts_to_sequences(sentence)
new_sentence
```

```
[[3, 4, 1, 2], [5, 6, 7, 1, 2]]
```

We then call the 'pad_sequences' on the data to make the sequences array of same lengths. This is because the machine learning model expects the input sequence of same length.

```
padded=pad_sequences(new_sentence,maxlen=5)
padded
```

```
array([[0, 3, 4, 1, 2],
       [5, 6, 7, 1, 2]], dtype=int32)
```

On the labels parts of the data, we use LabelEncoder class from keras and apply fit_transform method on the 'tags' part of the data to convert the label or target data into values between 0 and (n number of classes -1)

Figure 1: LabelEncoder

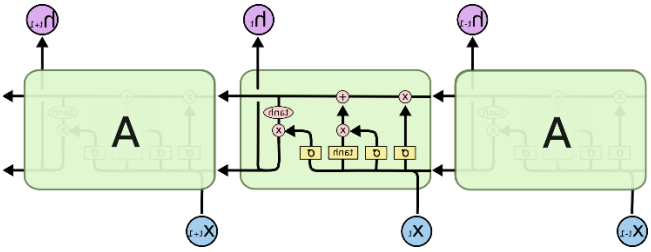
Model Training

We create the model architecture using Functional API implementation of Tensorflow.

The first layer is the Embedding layer. Here the words and sequences of words are converted into dense vectors so that the most related or relevant words are close to each other in a 3 dimensional vector space.

Here is a sample of word embedding visualization on projector.tensorflow.org website.

The model's next layer is an LSTM (Long Short-Term Memory) layer. LSTM is a child of RNN (recurrent neural network) but addresses its drawback of vanishing gradient. LSTM helps retain the sequence of words which is very important when making sense of the user queries or questions that the chatbot will encounter.



The final layer of our model is the Dense output layer with softmax activation to cater to multiple classes prediction in our output.

The model is compiled using **Adam** optimizer and loss function of **sparse categorical crossentropy**

```
i = Input(shape=(input_shape,))
x = Embedding(vocabulary+1,10)(i)
x = LSTM(30,return_sequences=True)(x)
x = Flatten()(x)
x = Dense(output_length,activation="softmax")(x)
model  = Model(i,x)
model.compile(loss="sparse_categorical_crossentropy",optimizer='adam',metrics=['accuracy'])
```

We train the model for 100 epochs.

After 100 epochs, The model reaches an accuracy score of around 69% with loss of around 90%.

```
324/324 [==============================] - 2s 6ms/step - loss: 0.9031 - accuracy: 0.6858
Epoch 94/100
324/324 [==============================] - 2s 6ms/step - loss: 0.9033 - accuracy: 0.6842
Epoch 95/100
324/324 [==============================] - 2s 6ms/step - loss: 0.9025 - accuracy: 0.6888
Epoch 96/100
324/324 [==============================] - 2s 6ms/step - loss: 0.9021 - accuracy: 0.6897
Epoch 97/100
324/324 [==============================] - 2s 6ms/step - loss: 0.9028 - accuracy: 0.6874
Epoch 98/100
324/324 [==============================] - 2s 6ms/step - loss: 0.9024 - accuracy: 0.6865
Epoch 99/100
324/324 [==============================] - 2s 6ms/step - loss: 0.9019 - accuracy: 0.6891
Epoch 100/100
324/324 [==============================] - 2s 6ms/step - loss: 0.9023 - accuracy: 0.6881
[19/Apr/2023 12:30:22] "GET /bot/train HTTP/1.1" 200 18
```

The entire trained model is then saved in an object file to be used during the interaction using Tensorflow's **save** method on the model object. The model could be trained on the server which hosts the Django application or on a higher performing GPU server or computer with the model then saved and transferred to the Django application folder.

INTERACTION PHASE

This is the phase where the saved NLP model is used by the application to predict the output class and then the output response to be generated as a reply to the user for the query.

This phase also first begins with cleaning and pre-processing the user input data into series of steps like tokenization, texts_to_sequences and padding the sequences.

The **predict** method of the model is then applied to the cleaned and pre-formatted user input data to predict the output class.

The LabelEncoder inverse_transform method gives a way to get the text values of the labels based on the previously encoded integer values.

Emotional_State	Salary	Purchased	Job	Job_Encoded	Job_Invers_Transformed
good	4700	Yes	Doctor	0	Doctor
bad	2400	No	Farmer	2	Farmer
neutral	4500	No	Electrician	1	Electrician
very_good	2500	Yes	Teacher	4	Teacher
excellent	3500	No	Pilot	3	Pilot

The response is then encoded in a json format to be sent back as a reply from the bot

```
#tokenizing and padding
prediction_input = tokenizer.texts_to_sequences(texts_p)
prediction_input = np.array(prediction_input).reshape(-1)
prediction_input = pad_sequences([prediction_input],input_shape)

#getting output from model
output = model.predict(prediction_input)
output = output.argmax()

#finding the right tag and predicting
response_tag = le.inverse_transform([output])[0]
print("Loyalist Bot : ",random.choice(responses[response_tag]))
chat_resp = random.choice(responses[response_tag])
```

DJANGO WEB APPLICATION WITH API - BACKEND

Django is a high-level python web framework for fast web development. It also provides Django REST a toolkit to build web APIs.

The chatbot backend is built using Django and hosted on a VPS server.

The backend consists of two main apps namely '**bot'** and '**botapi'**.

The bot app is where all the business logic regarding the training the model and generating user response is located. It consists of two main methods called **train_model** and **generate_response**.

The **train_model** method forms the **training phase** where it trains and saves the NLP model. This method is to be called the first time to generate and save the NLP model and thereafter can be called whenever there are any changes or optimizations in either the model architecture or the Intents JSON file.

The **generate_response** method forms the **interaction phase** where it takes in a text (ideally a query from the user) and generates the bot response to be sent back to the user.

```
from django.urls import path
from . import views

urlpatterns = [
    # frontend URLs
    path('train', views.train_model, name='train_model'),
    path('generate_response', views.generate_response, name='generate_response'),
]
```

The botapi app is the implementation of Django REST toolkit which provides API endpoints to be consumed at a frontend application.

The API endpoint **/askthebot/** is where the frontend application can send POST requests with the user input data and which also sends back the bot response. In the background, the **askthebot** method of the bot api simply communicates with the **generate_response** method of the bot app to get the response generated by the NLP model. However, one important task that the **askthebot** method performs is that it serializes the response to be sent back to the frontend application. Serialization of response is done so that the entire response can be transported back to the frontend in a reliable and persistent manner.

```
router = routers.DefaultRouter()
router.register('askthebot',
                views.askthebot, 'askthebot')

urlpatterns = [
    path('', include(router.urls)),
]
```

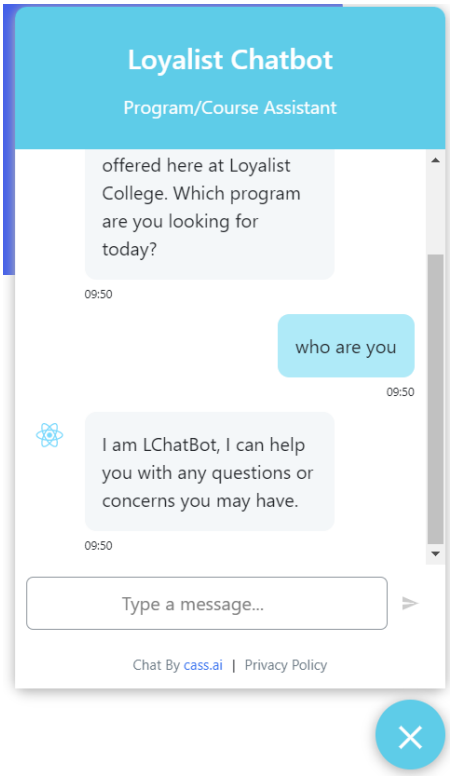
```
from rest_framework import serializers


class BotResponseSerializer(serializers.Serializer):
    botresponse = serializers.CharField(max_length=600)
```

REACT WEB APPLICATION – THE FRONTEND

React is one of the most popular a javascript framework for creating web interfaces. It provides features such as virtual DOMs and reusable components to cut down development time and built a UI efficiently. It also helps easier state management of the web elements as compared to vanilla javascript.

On the frontend, the react based web application chatbot widget looks like this (just like any other chat widget but powered with our NLP based AI Chatbot).



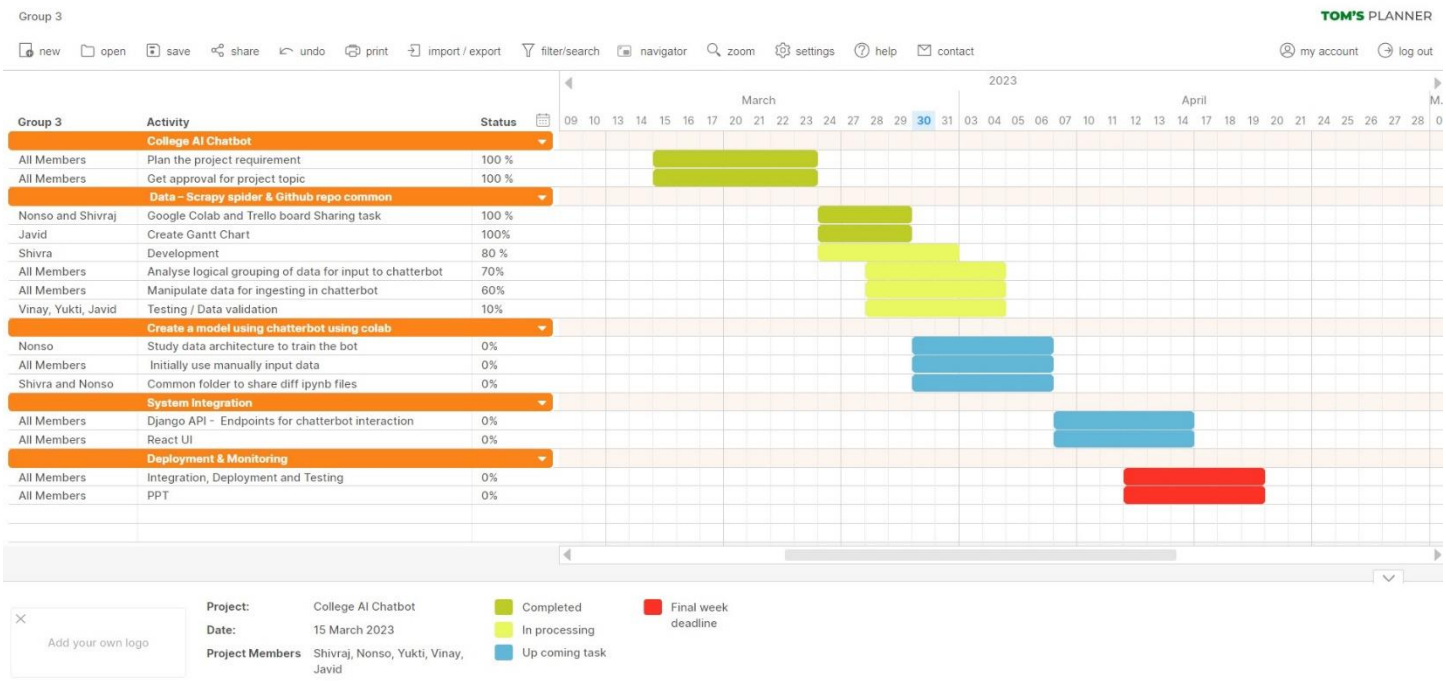
Mainly the React frontend sends the user input to the django API endpoint **/askthebot/** through the implementation as below:

```
const handleNewUserMessage = (newMessage) => {
  console.log(`New message incoming! ${newMessage}`);
  const user_message = { user_msg: newMessage}
  const post_header = { headers: { Authorization: "Token 765748[redacted]5cdc"}

  axios.post('http://137.184.170.69/botapi/askthebot/', user_message, post_header)
       .then(res => {
         addResponseMessage(res.data);
         console.log(res.data);})
       .catch(err => console.error(err));
};
```

And once the response is received from the Django server, displays information on the chat widget on the Frontend UI seamlessly without reloading the page (by just updating the DOM)

GANTT CHART


Group 3
TOM'S PLANNER
new
open
save
share
undo
print
import / export
filter/search
navigator
zoom
settings
help
contact
my account
log out
2023
March
April
Group 3
Activity
Status
College AI Chatbot
All Members
Plan the project requirement
100 %
All Members
Get approval for project topic
100 %
Data – Scrapy spider & Github repo common
Nonso and Shivraj
Google Colab and Trello board Sharing task
100 %
Javid
Create Gantt Chart
100%
Shivra
Development
80 %
All Members
Analyse logical grouping of data for input to chatterbot
70%
All Members
Manipulate data for ingesting in chatterbot
60%
Vinay, Yukti, Javid
Testing / Data validation
10%
Create a model using chatterbot using colab
Nonso
Study data architecture to train the bot
0%
All Members
Initially use manually input data
0%
Shivra and Nonso
Common folder to share diff ipynb files
0%
System Integration
All Members
Django API - Endpoints for chatterbot interaction
0%
All Members
React UI
0%
Deployment & Monitoring
All Members
Integration, Deployment and Testing
0%
All Members
PPT
0%
Add your own logo
Project:
College AI Chatbot
Date:
15 March 2023
Project Members
Shivraj, Nonso, Yukti, Vinay, Javid
Completed
In processing
Up coming task
Final week deadline

CONCLUSION

This project gives an end-to-end lifecycle of NLP ML model including data extraction, data cleansing and pre-processing, model training and its implementation for a real production use on a cloud-based web server with its capabilities consumed over a frontend user facing application.

FUTURE IMPLEMENTATIONS

Although the components that form the entire application are separated in the way they are executed, they could be connected through a process pipeline to automate the entire lifecycle. The model training could be moved to a cron based job where the new model will only be generated when either the data or the architecture changes. The application which is currently served through http could be served through https for security purposes. Currently due to server limitations, the model is trained on a local GPU machine and then deployed seamlessly through GIT on command line to the server where it is made available for the frontend to consume its capability. However, the model training could be deployed on a separate GPU based server and interaction could be served from a high CPU cloud-based server. This project mainly aims to provide a roadmap for future more serious implementations of the AI powered Chatbot application.

REFERENCES

https://stackoverflow.com/questions/51956000/what-does-keras-tokenizer-method-exactly-do

https://machinelearningknowledge.ai/keras-tokenizer-tutorial-with-examples-for-fit_on_texts-texts_to_sequences-texts_to_matrix-sequences_to_matrix/

https://medium.com/analytics-vidhya/understanding-nlp-keras-tokenizer-class-arguments-with-example-551c100f0cbd

https://ekababisong.org/gcp-ml-seminar/scikit-learn/

https://torontoai.org/2019/12/07/multi-class-text-classification-with-lstm-using-tensorflow-2-0/

TABLE OF CONTENTS

Introduction: 16

Project Description/Problem Statement: Motivation/Rationale 16

Project Objectives/Benefits (Research) Questions (Highlight contributions) 16

Deliverables/Outcomes 17

Scope/Constraints/Risks/Current Issues (for risks, propose mitigation measures) 17

WBS/Gantt Chart with milestones 17

Description of Team (Group Members) and matching skillsets of Project Members for the realization of project deliverables/outcomes 19

Methodology: 20

Tools/Resources: 20

Summary/Conclusions: 21

Overall Report Quality: 22

References: 22

INTRODUCTION:

Our proposal aims to address the issue of poor user experience on the Loyalist College website by developing and integrating a chatbot powered by natural language processing (NLP) and artificial intelligence (AI) technology. The chatbot will be able to answer frequent questions and provide assistance to users in real-time, thus improving the overall user experience and reducing response times. Our project will involve developing the chatbot using Django, collecting, and preprocessing training data using Scrapy web API, and integrating the chatbot into the Loyalist College website using the React JavaScript framework. We will evaluate the performance of the chatbot and gather feedback from users to continuously improve its functionality and usability.

PROJECT DESCRIPTION/PROBLEM STATEMENT: MOTIVATION/RATIONALE

The Loyalist College website currently lacks interactivity and relies on feedback forms to interact with users, resulting in a poor user experience for prospective students and other website users. This can lead to frustration and a negative perception of the college, potentially impacting enrollment, and reputation. To address this issue, our project aims to develop a chatbot powered by natural language processing (NLP) and artificial intelligence (AI) technology that can answer frequent questions and provide assistance to users in real-time.

The motivation behind this project is to improve the user experience on the Loyalist College website and enhance the image of the college. By providing a chatbot that can answer questions and provide responses in real-time, we can reduce response times and increase user satisfaction. This can also potentially increase enrollment and improve the college's reputation by demonstrating its commitment to providing a high-quality user experience. Furthermore, this project can contribute to the field of AI and data science by developing a chatbot that can assist users in real-time and by providing preprocessed training data and an evaluation report.

PROJECT OBJECTIVES/BENEFITS (RESEARCH) QUESTIONS (HIGHLIGHT CONTRIBUTIONS)

The specific objectives of our project are:

To develop a chatbot using TensorFlow LSTM module, a Python NLP library, and Django, a web framework for Python.

To collect and preprocess training data from the Loyalist College website using Scrapy web API.

To train the chatbot using The Scrapy WebCrawler generated data modules with custom data.

To integrate the chatbot into the Loyalist College website using the React JavaScript framework.

To evaluate the performance of the chatbot and gather feedback from users.

The research questions that our project aims to answer are:

How effective is the chatbot in answering frequently asked questions and providing assistance to users in real-time?

How does the chatbot impact the user experience of prospective students and other website users?

What improvements can be made to the chatbot to enhance its functionality and usability?

The specific contributions that our project will make to the field of AI and data science are:

Development of a chatbot powered by NLP and AI technology that can assist users in real-time.

Collection and preprocessing of training data using Scrapy web API.

Integration of the chatbot into the Loyalist College website using React JavaScript framework.

Evaluation of the chatbot's performance and gathering feedback from users.

DELIVERABLES/OUTCOMES

The specific deliverables and outcomes of our project are:

Chatbot software that can be integrated into the Loyalist College website.

Preprocessed training data in JSON or text format.

Evaluation report containing performance metrics and user feedback.

SCOPE/CONSTRAINTS/RISKS/CURRENT ISSUES (FOR RISKS, PROPOSE MITIGATION MEASURES)

The scope of our project is limited to the development and integration of a chatbot into the Loyalist College website.

Constraints include limited access to training data and the need to train the chatbot using custom data.

Risks associated with our project include technical issues during development and integration, user dissatisfaction with chatbot performance, and potential legal and ethical issues related to data privacy.

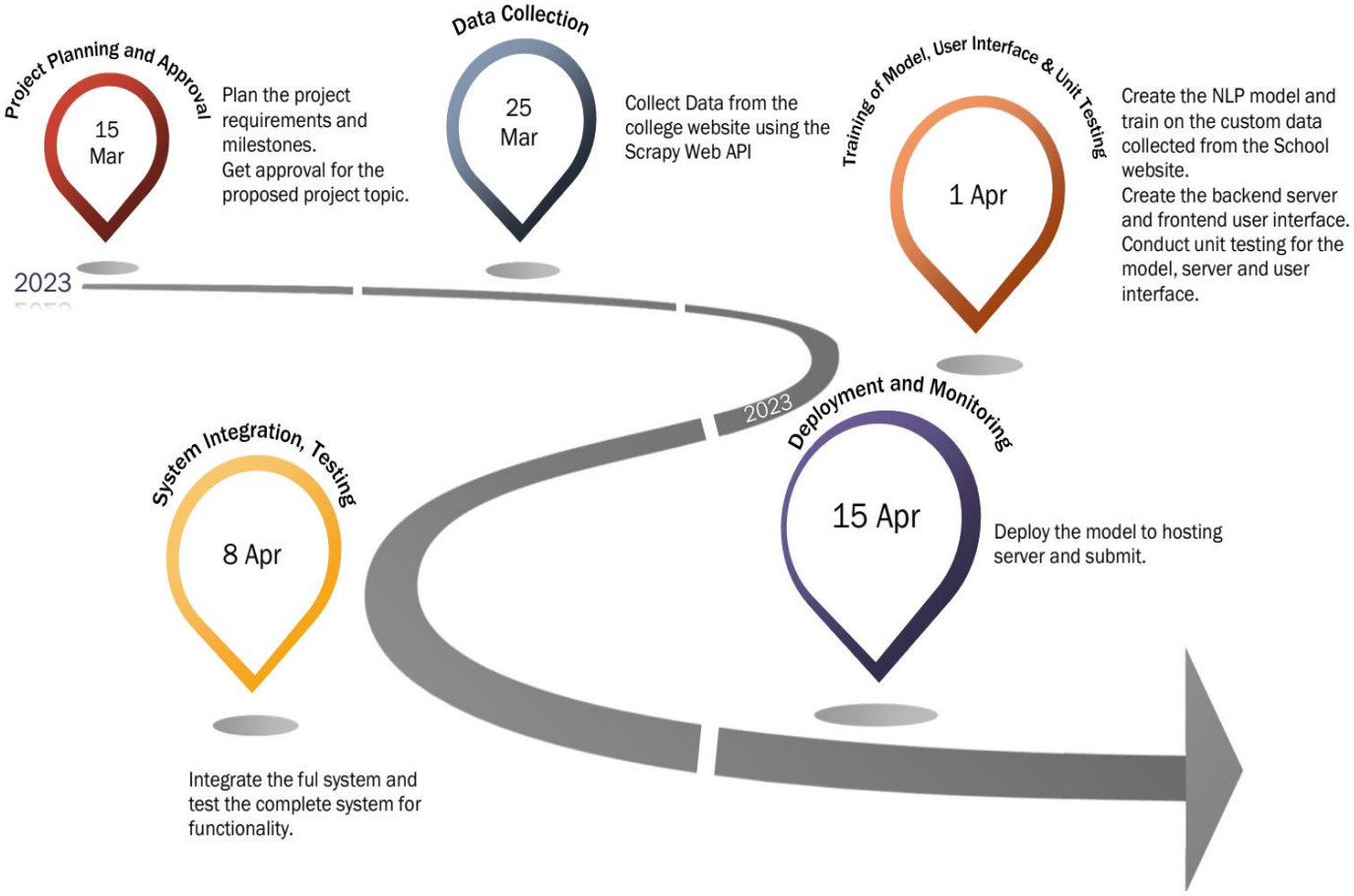
Mitigation measures for these risks include thorough testing and user feedback.

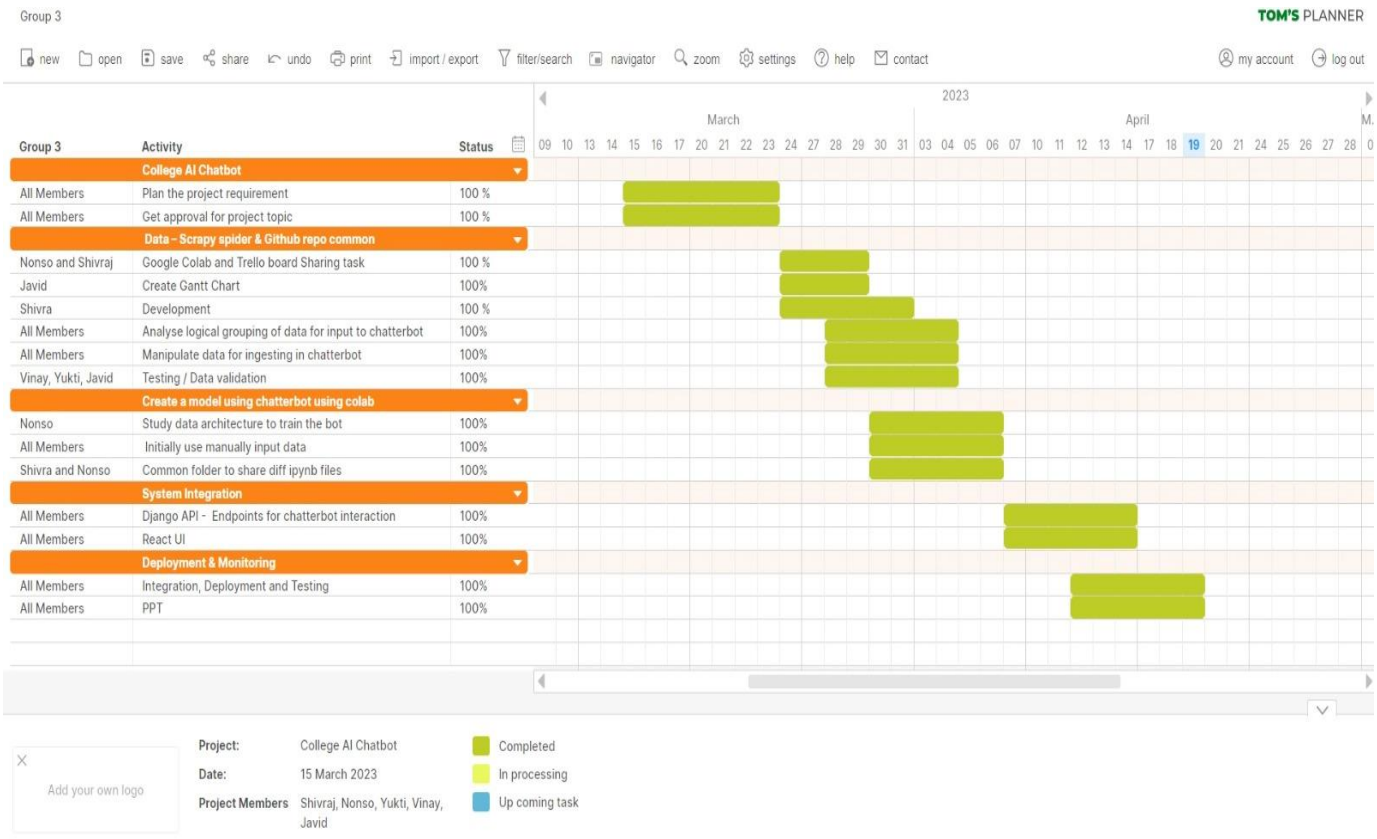
WBS/GANTT CHART WITH MILESTONES

Our project will be managed using a Work Breakdown Structure (WBS) and Gantt chart to show the major milestones of the project.

Milestones include data collection and preprocessing, chatbot development and testing, integration into the website, and evaluation and feedback.

The WBS and Gantt chart will be used to ensure timely completion of each milestone and to track progress towards the overall project objective.


Project Planning and Approval
15 Mar
Plan the project requirements and milestones.
Get approval for the proposed project topic.
Data Collection
25 Mar
Collect Data from the college website using the Scrapy Web API
Training of Model, User Interface & Unit Testing
1 Apr
Create the NLP model and train on the custom data collected from the School website.
Create the backend server and frontend user interface.
Conduct unit testing for the model, server and user interface.
2023
2023
Deployment and Monitoring
15 Apr
Deploy the model to hosting server and submit.
System Integration, Testing
8 Apr
Integrate the ful system and test the complete system for functionality.



DESCRIPTION OF TEAM (GROUP MEMBERS) AND MATCHING SKILLSETS OF PROJECT MEMBERS FOR THE REALIZATION OF PROJECT DELIVERABLES/OUTCOMES

Our team consists of Yukti, Javid, Nonso, Shivraj and Vinay

Shivraj and Nonso – Actively participate for the development of the chatbot and integration into the Loyalist College website. They will use their coding expertise to develop the chatbot using TensorFlow LSTM, a Python NLP library, and Django, a web framework for Python. They will also collect and preprocess training data using Scrapy web API, train the chatbot using The Scrapy WebCrawler generated data and other custom data, and integrate the chatbot into the website using the React JavaScript framework.

Yukti – Reference Links Share, Proposal Draft, Meetings, Testing Help, primary responsibility will be to manage the team's people, oversee meetings, maintain communication, and ensure that everyone is on track. She will also provide support in drafting proposals, share reference links, and assist in testing the chatbot's performance. Collaborate with Javid to gather feedback from users and make design improvements to the chatbot. Javid's expertise in UX/UI design will ensure that the chatbot is easy to use, aesthetically pleasing, and meets the needs of Loyalist College's prospective students and other users.

Javid – Javid will be responsible for creating the chatbot's visual design and ensuring a seamless user experience. He will work closely with Nonso and Shivraj to ensure that the chatbot's integration into the website aligns with its visual elements.

Vinay – Making presentation, participation report and assisting in testing the chatbot and provide references and help that team needs. Vinay will also actively check our current website for data for programs and courses fees etc. and maintain a worksheet for any relevant information.

METHODOLOGY:

Our methodology for completing this project involves the following steps:

Conduct a literature review to identify best practices for developing a chatbot using NLP and machine learning techniques.

Gather and preprocess data from the Loyalist College website using the scrapy web API.

Train the TensorFlow LSTM model using the Scrapy WebCrawler generated data and other custom data.

Build the Django backend server and React-based user interface for the chatbot.

Test and validate the chatbot's performance with real users and iteratively improve its accuracy and effectiveness.

TOOLS/RESOURCES:

For our project, we will be using the following tools and resources:

Python programming language for developing the chatbot.

TensorFlow LSTM library for implementing the natural language processing capabilities of the chatbot.

Django framework for building the backend server.

React framework for developing the user interface.

Scrapy web API for gathering and preprocessing data from the Loyalist College website.

Git for version control and collaboration.

Google Cloud Platform or Heroku for hosting the chatbot.

We believe that these tools and resources are appropriate for our project because they are widely used in the development of chatbots and web applications, and they provide the necessary functionality and scalability for our project.

SUMMARY/CONCLUSIONS:

In summary, our project aims to develop an NLP-backed chatbot for the Loyalist College website to improve the user experience for prospective students and other users. Our objectives include gathering and preprocessing data, training the TensorFlow LSTM model, building the backend server and user interface, and testing and validating the chatbot's performance. Through our project, we hope to contribute to the field of artificial intelligence and data science by demonstrating the practical application of NLP and machine learning techniques for developing chatbots.

We are confident in the success of our project, given our team's experience and expertise in software development and NLP, and the availability of appropriate tools and resources.

OVERALL REPORT QUALITY:

We have ensured that our proposal is well-organized with a clear table of contents, visuals, and page numbers. We have used a consistent and professional layout throughout the document to enhance readability and comprehension.

REFERENCES:

1. https://stackoverflow.com/questions/51956000/what-does-keras-tokenizer-method-exactly-do
2. https://machinelearningknowledge.ai/keras-tokenizer-tutorial-with-examples-for-fit_on_texts-texts_to_sequences-texts_to_matrix-sequences_to_matrix/
3. https://medium.com/analytics-vidhya/understanding-nlp-keras-tokenizer-class-arguments-with-example-551c100f0cbd
4. https://torontoai.org/2019/12/07/multi-class-text-classification-with-lstm-using-tensorflow-2-0/