Text

Description automatically generated with low confidence

**Department of Computer Science, University of Leicester CO7201 Individual Project**

**JokeBot: An AI Comedian Twitter Bot**

Aakash Chahal

[ac879@student.le.ac.uk](mailto:ac879@student.le.ac.uk)

**Final Report Template**

**Project Supervisor:** Dr Paula Severi

**Second Marker:** Muhammad Kazim

**Word Count:**

**Date of Submission:**

**Declaration**

All sentences or passages quoted in this report, or computer code of any form whatsoever used and/or submitted at any stages, which are taken from other people’s work have been specifically acknowledged by clear citation of the source, specifying author, work, date, and page(s). Any part of my own written work, or software coding, which is substantially based upon other people’s work, is duly accompanied by clear citation of the source, specifying author, work, date, and page(s). I understand that failure to do this amount to plagiarism and will be considered grounds for failure in this module and the degree examination as a whole.

**Name:** Aakash Chahal

**Date:**

**ABSTRACT**

In this dissertation, the JokeBot: AI Twitter Bot Comedian project is presented, which aims to develop an AI-powered bot that can generate humorous content on Twitter. The project's aim is to provide users with a fun and entertaining experience, improve the quality of generated content over time, and promote the bot's launch on social media platforms. The project team faced several challenges while getting a language model to generate jokes, such as the large dataset, limited access to Twitter API endpoints, and technical limitations in using deep learning techniques. To mitigate these challenges, the team used the GPT-3.5 model, which is pre-trained on a large dataset, to generate jokes in multiple languages. The bot is working fine on Twitter and is tweeting jokes in multiple languages, but because of several challenges bot is only able to post tweets (jokes) on Twitter but to make it more engaging and functional it was integrated into a web interface which allowed it function as a chat bot that users can easily talk to, on the web app users are able to chat with the bot, or generate images that looks like a twitter post by providing any username, amount of likes, and retweets they want along with either a randomly generated joke, a joke of their choice or any text that the user would want which might particularly not be a joke. All these challenges and having to think how to make it better despite the challenges made the project really interesting to work on.

**INDEX**

Chapter 1: Introduction

* 1. Motivation
  2. Objectives
  3. Challenges
  4. Technical Specifications
  5. Background Material
  6. Detailed Work Plan
  7. Risk Management Plan

Chapter 2: Literature Review

2.1 NLP Techniques

2.2 Deep Learning Techniques

2.3 Open AI API and Different Language Models

2.4 Humour Generation

2.5 Twitter API and Bots

Chapter 3: Methodology

3.1 Data Collection and Pre-processing

3.2 Language Model Training

3.3 Bot Development

3.4 Integration and Testing

Chapter 4: Results and Analysis

4.1 Quality of Generated Content

4.2 User Engagement and Satisfaction

4.3 User Feedback

Chapter 5: Evaluation and Future Scope

5.1 Comparison with Existing Bots

5.2 Limitations and Challenges Faced

5.3 Future Work and Improvements

Chapter 6: Conclusion

Chapter 7: Source Code with Explanation

Chapter 8: Bibliography (References)

**Chapter 1: Introduction**

* 1. **Motivation**Humour is an essential aspect of human communication, and social media platforms like Twitter provide an excellent opportunity for people to share and enjoy humorous content. However, generating humorous content consistently and effectively could be and is a challenging task. The JokeBot: AI Twitter Bot Comedian Project aims to develop an AI-powered bot that can do just the same i.e., generating humorous content (jokes) in 5 of the most spoken languages in the world namely, English, Mandarin, Spanish, French and Hindi for and on Twitter, providing users with a fun and entertaining experiences.
  2. **Objectives**The objectives on higher level of the projects were simple to create an AI-powered bot that generates and tweets jokes on Twitter, following are the objectives that were written down at the beginning of the project and also including some that were included or updated during the development when faced with some challenges to make sure that the project is developed and delivered successfully:   
     -> Research about AI bots, NLP techniques, and Language Models.   
     -> Gather and pre-process a large amount of jokes data.   
     -> Train a language model like GPT3 using the dataset.   
     -> Test the trained and fine-tuned languages model using appropriate technologies, preferably sentiment analysis.   
     -> Use Pre-trained Language model (GPT-3.5) to generate better jokes and in multiple languages. (Adapted Objective)   
     -> Integrate the language model with Twitter.   
     -> Develop a web app to integrate the bot into. (Adapted Objective)
  3. **Challenges**Several challenges were faced during research and development of the JokeBot, this section lists all the challenges that were faced:   
     -> **Gathering and Using Dataset:** There were few challenges while gathering dataset and then when having to use it to train the language model. At the beginning the challenge was to gather a large amount of data which can be used to train the language model to be able to get a decent output from it, but after the data has been collected and pre-processed to be used to train the language model few challenges emerged as the train a language model on a large amount of data we’d need a lot of resources including computer power, and it’d also cost a lot of money and when the dataset was reduced to a fraction of its original size, I was able to train/fine-tune the language model but the result generated using that fine-tuned model wasn’t presentable at all.   
       
     -> **Technical Limitations:** As described above, training or fine tuning a language model would usually require a lot of computer resources and being a student I did not have access to resources that could be used to train a language model on a large amount of data which would allow for the language model to be able to generate good results.  
       
     -> **Twitter API:** Having used the twitter API before this wasn’t supposed to be challenge but when I started to write code and use the new Twitter API, they had implemented new changes which includes introduction of access levels which limits the endpoints we can access using the Twitter API, which initially after creating a new account for the bot and then creating a developers account for the same to get API keys gave errors while authenticating and trying to access an endpoint to fetch the tweets, upon reading more into the errors and docs it became clear that the new (and free) accounts had basic access to the Twitter API which only allowed some read access to a limited amount of endpoints and in order to get access to more endpoints you need at least elevated access. As the project was under development, Twitter depreciated these access levels and I was only able to either POST or DELETE tweets on and from twitter, which made it difficult or rather impossible to implement other features of the project including replies, likes, following back users, retweeting, and replying to direct messages.
  4. **Technical Specification**

The JokeBot is a Twitter Bot Comedian and a chatbot interface that lets in users to interact in actual-time conversations with the bot. To make certain the bot's excessive performance and efficiency in producing textual content in more than one language, the JokeBot leverages the GPT-3.5 language model from OpenAI. This model is trained on a massive dataset and can generate exquisite jokes on numerous topics.

The JokeBot is developed using Python programming language and numerous libraries, along with Tweepy, Pandas, and NumPy. Tweepy library is used to engage with Twitter's API v2 endpoints, Pandas is used to pre-process the records, and NumPy is used to handle numerical calculations.

Although the JokeBot isn't always presently deployed or hosted, it can easily be hosted on a cloud server like AWS.

To make certain the non-stop operation of the JokeBot without interruptions, the utility uses a queue-based structure. The incoming requests are introduced to a queue and processed sequentially. The queue-based totally architecture permits the JokeBot to deal with a big wide variety of requests without any overall performance degradation.

To sum it up, the technical specifications of the JokeBot encompass:

**Programming languages used:**

* Python: Python is the main and only programming language used to develop the Twitter Bot, to train the language model, to make calls to the pre-trained language model that is used to generate jokes, to make calls to Twitter API, and in the backend of the web app.
* JavaScript: JavaScript is used to develop the UI for the web app, the bot is integrated with and allows user to chat with the bot.
* HTML: HTML is used to generate an element that looks like a tweet which is then used to generate an image.

**Libraries and Frameworks used:**

* ReactJS: ReactJS is JavaScript library that is used to develop the frontend of the web app.
* Material UI: Material UI provides easy to use react components that are used to design and add styles to UI.
* Flask: Flask is a micro web framework that is built in python, it is used in this project to setup a backend server that can be used to get requests from frontend, generate a response using the bot’s language model and send a response back.

**APIs used:**

* Twitter API
* OpenAI API

The JokeBot is integrated into a web app, which makes use of React in the front end and flask in the backend. This integration permits users to chat with the bot in actual time thru a user-friendly interface.

* 1. **Background Material**The development of this project (JokeBot) is based on extensive research and analysis of various information related to NLP, ML, and the social media platform Twitter along with it’s API. The following are some of the key sources of information that were used during the research phase and the development phase of the project JokeBot:   
     -> “Natural Language Processing with Python” by Steven Bird, Ewan Klein, and Edward Loper.   
     -> “Python Machine Learning” by Sebastian Raschka and Vahid Mirjalili.  
     -> “Deep Learning with Python” by Francois Chollet.   
     -> Twitter API documentation.   
     -> OpenAI’s GPT-3 documentation.   
     -> “What is a chatbot” by IBM (https://www.ibm.com/topics/chatbots).  
     These sources of information provided valuable knowledge and insights into the architecture and implementation of the JokeBot project. Libraries were like Pandas and NumPy were initially used to clean data, and then NLP techniques, algorithms were used to pre-process the data to finally train or fine tune a language model based on GPT-3.5 model of Open AI. The Twitter’ API documentation was used to interact with Twitter API endpoints , by posting tweets or extracting information from twitter as required.
  2. **Detailed Work Plan**The project involved several tasks and sub-tasks that had to be completed to ensure that the project was successfully completed. A detailed work plan was created and updated along the way as and when required to ensure that project remained on track and was completed within the set deadline. The work plan was divided into several phases, each with specific tasks and sub-tasks.   
     The phases and tasks included in the work plan were as follows:   
     **Phase 1: Project Planning and Research** ->Task 1.1: Define the project scope and objectives   
      -> Task 1.2: Research and select appropriate language models for training and generating jokes.   
      -> Task 1.3: Develop a detailed work plan, and risk management plan.   
       
     **Phase 2: Data Collection and Pre-processing** -> Task 2.1: Research and gather a dataset of jokes that can be used to train the language model.   
      ->Task 2.2: Clean the collected data using appropriate libraries like Pandas and NumPy.   
      -> Task 2.3: Pre-process the dataset using NLP techniques which includes tokenisation of different jokes.   
      **Phase 3: Model Training and Fine-tuning:** -> Task 3.1: Train / Fine tune the GPT-3.5 based model on the collected dataset.   
      -> Task 3.2: Evaluate the performance of the fine-tuned model using various metrics.   
      -> Task 3.3: Generate jokes using the fine-tuned model and perform manual quality check on the output generated.   
       
     **Phase 4: Twitter Bot Development:** -> Task 4.1: Develop a Twitter bot using Python using Tweepy library which provides abstraction over twitter api’s endpoints by providing us with different easy to call methods.   
      -> Task 4.2: Integrate the fine-tuned language model with the twitter bot.   
      -> Task 4.3: Test the bot and fix any issues.   
      -> Task 4.4: Deploy the bot on a cloud server. (Optional)   
       
     **Phase 5: Bot Maintenance and Optimization:** -> Task 5.1: Monitor the performance of the bot and fix any issue if any.  
      -> Task 5.2: Perform regular backups and updates of the bot and its dependencies.  
     The detailed work plan helped in breaking down the project into manageable tasks and sub-tasks, making it easier to track progress and identify any issue that arose during the development of the project in time.
  3. **Risk Management Plan**A risk management plan was also developed to identify and mitigate risks that could affect the project’s timeline and success. The risks identified and their respective mitigation strategies are summarised below:  
     **Risk:** Insufficient computational resources for training large language models.   
     **Mitigation Strategy:** Use pre-trained models and cloud-based services for training and fine-tuning.   
     **Risk:** Twitter’s API restriction and changes.   
     **Mitigation Strategy:** Monitor the twitter’s API updates regularly and adjust the bot’s functionality accordingly.   
       
     **Risk:** Bot account suspension or termination by Twitter.   
     **Mitigation Strategy:** Follow Twitter’s rules and guidelines and comply with the terms of services of Twitter.   
       
     **Risk:** Inaccurate response or offensive joke generation.   
     **Mitigation Strategy:** If using personally trained language model, implement content filtering algorithms and perform manual quality checks. Or use a pre trained language model that would not generate any offensive response or jokes, and also perform manual quality check on the response generated.   
       
     The risk management plan helped in identifying potential risks and implementing mitigation strategies to reduce their impact on the development and outcome of the project.   
     Overall the detailed work plan and risk management plan helped in ensuring the project was completed successfully and within the set timelines, while also mitigating potential risks that could have affected the success of the project.

**Chapter 2: Literature review**The field of AI and NLP has been growing and gaining significant attention in the recent years, and that has led to the development of a wide range of applications that use NLP techniques. One such application is the development of chat bots, which are designed and developed to interact with users in natural language which is the basic idea of developing this JokeBot as it’ll in its roots act as a chat bot. In this section, we review some of the literature related to chat bots, their use in social media platforms like Twitter for this project and how chatbots relate to the project JokeBot.   
  
**2.1 Chatbots in social media** Chatbots have become increasingly popular in recent years, with many businesses using them to provide customer service and support. Social media platforms such as Twitter have also started to integrate bots (chatbots) into their platform, allowing business to interact with customers in real time.   
 One of the key benefits of using chat bots in social media is their ability to automate interactions with users. This can help to improve response times and reduce the workload for customer service teams. Additional chat bots can provide a personalised experience for each and every user by analysing their previous interactions and tailoring their responses accordingly and get better as they keep interacting.   
 However, the effectiveness of such chatbots in social media platforms depends on their ability to understand natural language and respond appropriately to the user’s query. This requires the use of NLP techniques such as sentiment analysis and entity recognition, and intent detection.

**2.2 NLP Techniques for chatbots** NLP techniques are essential for developing effective chatbots, as they enable the bot to understand and interpret natural language input (as we would except from any user). One of the key challenges in NLP is the ambiguity of natural language, which makes it very difficult for a machine to understand the meaning of a sentence in natural language.   
 One approach to addressing this challenge is to use ML algorithms to train a chat bot on a large corpus of text data. This would enable the chat bot to identify similar patterns and relationships between words and phrases, and then use this knowledge to generate appropriate responses to user queries in natural language.   
 Another important aspect of NLP for chatbots is the ability to perform sentiment analysis, which involves identifying the emotional tone of a user’s message or query. This can be used to tailor the bot’s response in the future accordingly, for example by providing a more empathetic response a user who is expressing frustration or dissatisfaction or providing with an energetic response to user showing happiness or similar emotions.

**2.3 Existing chatbot applications**

There are several existing chat bot applications that have been developed for use in social media. One such example in the recent times is ChatGPT, which uses OpenAI’s GPT-3.5 language model to interpret user message or query and give an appropriate response accordingly. The AI chat bot is capable of generating response on a wide range of topics (almost everything) and the GPT-3.5 model that it uses can be trained on any specific dataset for personal use and tailored experience.   
 Another example is the Mitsuku chat bot, which has also won several awards for its performance in the annual Loebner Prize competition, it has won the Loebner award in the years 2013, and then 2016 to 2019. Mitsuku uses a combination of rule-based and ML approaches to generate responses to user input.

**2.4 Evaluation metrics for chat bots**

The effectiveness of a chatbot can be evaluated using a range of metrics, such as accuracy, response time, and user satisfaction. Accuracy measures how well the chatbot is able to understand and interpret user input, while response time measures how quickly the chat bot is able to generate a response.

User satisfaction is a really important metric for chat bots, as it reflects how well the bot is able to meet the requirements of its users/ This can be measured using surveys or by analysing user feedback and reviews.

**2.5 How does a chat bot relate to our JokeBot.**

As discussed above chat bots have been gaining a lot of interests as they are helping companies’ setup their customer service on social medias using chat bots and as we use NLP techniques to build these bots they are easily able to understand the natural language and provide user with appropriate response.

Now, the project JokeBot’s main and most important function is its ability to tweet jokes in multiple languages and tweet jokes in relevance to what’s trending (in the UK for now) on twitter. But, in order to make it more functional and entertaining for users the bot was also then planned to be integrated into a web application or interface where users would be able to have a normal chat with the JokeBot and ask it to generate jokes and different comedic content in different styles, genres and categories. The bot when used as a chat bot would also be able to handle any other user queries like any other chat bot would do. And that’s why to successfully develop the project it was really important to study chat bots and understand how we could make a better user experience which is engaging and entertaining.

**Chapter 3: Methodology**

In this section, the methodology of the project will be discussed. The methodology of the project involved several steps including data collection, data pre-processing, language model training, pre-trained language model selection, developing a web app. The detailed steps involved in each of these processes have been discussed in the previous section. In this section, the rationale behind selecting these specific methodologies for the project will be thoroughly discussed.

**3.1 Data Collection**

Data collection is the first and foremost step in any machine learning project. In this project, we collected a large amount of text data (jokes) from various sources including social media, jokes websites, and other online sources. The data was collected using web scraping techniques and APIs. The data collected was in just one language because it was really difficult to gather a large amount of data in different languages especially if it comes to a very particular category like comedy or jokes per say.

The rationale behind a large amount of dataset was so that the language model can be trained on as much data as possible because it’d help the model to learn the relationships between the words and phrases and would allow to generate better results or in this case funny jokes.

**3.2 Data Pre-processing**

Data pre-processing is an essential step in any machine learning project. In this project, we used several techniques to pre-process the data, starting with cleaning the data using libraries Pandas and NumPy which allowed to remove any empty rows, any characters (other than alphabets or numbers) that aren’t as important for the jokes, removed any jokes that had used either some sort of URL (which could’ve been for any image or videos). And, after that as there were several language models that were tried to get a better result and data pre-processing was done differently for each of them for example while training a language model provided by OpenAI we used their own API to pre-process data without having to write much code, but when training a language model using TensorFlow we used techniques like tokenisation. Tokenisation is essentially a breaking down a string of text into words or phrases.

During tokenization, a text document is divided into individual tokens or words, which can be further used for text analysis, classification, or modelling. The tokens can also be normalized by converting them to lowercase or removing punctuation marks, stop words, or other unwanted characters.

Tokenization helps to reduce the complexity of text data and makes it easier to process and analyse. It is a critical step in building models that rely on natural language processing, such as text classification, language translation, or sentiment analysis.

The rationale behind using these techniques was to ensure that the data was in a format suitable for training the language model. Pre-processing the data helped to reduce noise in the data and made it easier for the language model to understand the meaning of the text.

**3.3 Model Selection**

Model selection is a crucial step in any machine learning project. In this project, we experimented and trained several language models on our dataset of jokes including GPT-3, GPT-3.5, and TensorFlow and then also tried experimenting with pre-trained language model gpt-3.5-turbo. After experimenting with these models, we decided to use the gpt-3.5-turbo model not only because it provided the best results in terms of generating jokes and different responses in multiple languages and on various topics but also because as the training of those language models on a large dataset was not possible due to lack of computer resources and costs if using cloud services.

The rationale behind choosing the pre-trained gpt-3.5-turbo model was that it was a pre-trained language model on a large dataset and was able to generate high-quality text in multiple languages. The model was also available on a cloud-based platform which made it easy to deploy and use.

**3.4 Web App**

As the project was coming towards its end, twitter decided to depreciate all their previous access levels and made it into two main tiers **free**, and **basic**. Free tier didn’t allow to use any endpoints apart posting tweets and deleting tweets (it also allows lookup for users) which meant that more than half of the functionalities developed for the bot wasn’t useful anymore which included liking tweets, retweeting tweets, replying to comments, and replying to direct messages. So, to include or somehow give these features to user we created a web app that would allow a user to directly chat to the bot and get jokes on variety of different topics.

The rationale behind choosing to create a web app and integrate JokeBot into it was to give users the feature where they can talk to the bot and get jokes that they can use.

**Chapter 4: Results and Analysis**

This chapter presents the results and analysis of the JokeBot project. The project is implemented successfully and the bot was able to generate and tweet jokes in multiple languages on random topics and in random styles of jokes, it also tweets a joke in relation to what’s trending in the UK for now as there is not really much data to use and know what countries could the bot be used from, but as it is now also integrated into a web chat app the users can easily provide the JokeBot with what’s trending in their location and it’d be able to generate a joke about that. In this chapter, we will discuss the performance of the bot and evaluate its effectiveness in achieving the project objectives.

**4.1 Performance Metrics**

To evaluate the performance of the JokeBot, we used several metrics, including joke quality, engagement rate, and user feedback. Ten random jokes generated by the bot were used in a survey and users were asked to give every joke a rating from 1 to 5, 1 being not funny at all and 5 being really funny. The web app was also tested with a few users and feedback was recorded if it makes the bot better to use or if it adds any feature to the project.

**4.2 Joke Quality**

The quality of the jokes generated by the bot was evaluated based on their humour and relevance to the topic. We used a survey to collect feedback from users on the quality of the jokes generated by the bot. The survey consisted of a set of ten jokes randomly selected from the bot's tweets. The users were asked to rate the humour and relevance of each joke on a scale of 1 to 5, with 1 being not funny or relevant at all, and 5 being very funny or relevant.

Table below shows the results from the survey which shows rating for every joke that was in the survey.

**4.3 User Feedback**

\*This section would be completed as soon as the development of web app is complete and a user test has been conducted for the same\*

**Chapter 5: Evaluation and Future Scope**

\* This section would be completed as soon as the development of web app is complete and it would contain the following sections  
1. Comparison with existing bots

2. limitations and challenges faced

3. future improvements and scope\*

**Chapter 6: Conclusion**

\* this section would also be completed when the development of the project is completed and it would contain a thorough conclusion for the project and this report \*

**Chapter 7: Source code Explanation**

\* this section would also be written after the completed development of the project and it would contain explanation for different terms used in the report and it would also display different import parts of the code that makes the bot work \*

**Bibliography**

[1] GPT-3.5. OpenAI. Retrieved from

<https://platform.openai.com/docs/models/gpt-3-5>

[2] Gehrmann, S., & Mercer, R. E. (2019). Natural language processing for social media: A systematic review. Social Science Computer Review, 37(4), 470-500.

[3] Hossain, M. S., & Muhammad, G. (2021). Deep learning-based automatic question answering system: A comprehensive review. Information Processing & Management, 58(1), 102316.

[4] Ranganath, T., Choudhury, M. D., & Roy, A. (2020). Natural Language Processing with Python and NLTK: Analyzing Text with the Natural Language Toolkit. Packt Publishing Ltd.

[5] Sharma, A., & Dhir, A. (2020). Social Media Analytics using Python: Extracting Data from Facebook, Twitter, and Instagram. Apress.

[6] Zhang, Y., & Wallace, B. (2017). A sensitivity analysis of (and practitioners' guide to) convolutional neural networks for sentence classification. arXiv preprint arXiv:1510.03820.

[7] Twitter API v2 Documentation. Retrieved from

<https://developer.twitter.com/en/docs/twitter-api>

[8] OpenAI API Documentation. Retrieved from

<https://platform.openai.com/docs/introduction>

[9] Jokes One-liners dataset. Retrieved from

<https://www.kaggle.com/datasets/abhinavmoudgil95/short-jokes>

[10] Jester Jokes dataset. Retrieved from

<https://www.kaggle.com/datasets/vikashrajluhaniwal/jester-17m-jokes-ratings-dataset>