Laugh Out Loud: Developing a Joke Android App using OpenAI GPT

Sushma Yadav Duvvi
Department of Computer Science and Information Systems
Texas A&M University - Commerce
Commerce, Texas, USA
sduvvi@leomail.tamuc.edu

ABSTRACT

Android devices have become ubiquitous today and have revolutionized the way we interact with technology. With a large and growing user base, the demand for Android applications has never been higher. As a result, there has been a significant increase in the number of developers creating apps for the Android platform. This project focuses on developing a daily joke Android application that utilizes the OpenAI GPT API and GPT-3.5-Turbo AI model to generate humor-based content. The app allows users to select from five categories, including book, food, animal, holiday, and nature, and generates a joke based on the selected category using the API and AI model. The generated jokes are then displayed to the user within the Android Studio emulator or on a physical Android mobile device. To evaluate the effectiveness of the OpenAI GPT API and AI model, an assessment was conducted, ranking the generated jokes from 1 to 5 based on their humor quality. The findings suggest that the GPT Turbo model can efficiently generate high-quality jokes for certain categories, such as animal jokes, while other categories, such as food jokes, may require further fine-tuning to enhance diversity and quality. Overall, this project demonstrates the potential of utilizing OpenAI GPT API and GPT-3.5-Turbo for developing humor-based content on the Android platform.

1 INTRODUCTION

Humor is an important aspect of human communication that can build social bonds, reduce stress, and entertain. With the advancement of mobile applications and machine learning technologies such as natural language processing (NLP), it has become possible to develop more sophisticated humor-based apps that can generate creative and contextually appropriate content in response to user input. The OpenAI GPT-3.5-Turbo model is one such model that has shown potential in generating high-quality and engaging responses for humor-based applications, without requiring training on a specific dataset. However, using GPT models for joke generation comes with challenges such as the need for fine-tuning the model for the specific application and potential biases in the generated content.

Despite the challenges, using machine learning-based approaches to generate humor content offers numerous benefits such as adaptability to user preferences, generating content on demand, and personalized user experiences. With the increasing demand for novel and engaging mobile apps, the development of a humor-based app that leverages machine learning technologies presents new opportunities for social interaction, entertainment, and creativity. Our project aims to create a joke Android app that integrates machine learning with humor to deliver a unique and engaging user experience that enhances social interaction while providing humor-based entertainment. This project has the potential to contribute to the growing field of computational humor and provide a new form of mobile-based entertainment for users[1].

2 LITERATURE REVIEW

Humor has always been a vital aspect of human life, but its importance has grown even more pronounced in the aftermath of the COVID-19 pandemic. By providing a mental escape, fostering community and connection, and promoting physical and emotional health, humor has become an essential tool for navigating the challenges of the modern world. Humor has been found to have various benefits, such as improving mood, strengthening relationships, enhancing creativity, reducing stress, and promoting resilience. In the context of joke apps, humor is a key driver of user engagement and retention[4].

2.1 OVERVIEW OF HUMOR-BASED ANDROID APPS

Humor-based Android apps have become increasingly popular in recent years. These apps are designed to provide entertainment to users with jokes, funny images, and humorous videos. One of the main types of humor-based Android apps is the joke app, which provides users with a wide range of jokes and puns to share with their friends and family. Other humor-based apps include funny image and video apps, which provide users with a variety of humorous pictures and videos to enjoy and share. These apps have become very popular among users of all ages, as they provide a quick and easy way to add humor and laughter to their daily lives[1].

While humor-based Android apps have become very popular, developing such apps can be a challenge. One of the main challenges is developing content that is truly funny and engaging for users. This requires a deep understanding of the type of humor that will appeal to the app's target audience, as well as the ability to create content that is both creative and humorous. In addition, app developers must consider factors such as user engagement, user retention, and monetization strategies to ensure the long-term success of their apps. However, with the help of advanced AI models such as OpenAI GPT-3.5-Turbo in specific, developers can now create humor-based Android apps that are more engaging and entertaining than ever before[3].

2.2 PREVIOUS JOKE GENERATION MODELS

In the early days of joke generation in mobile applications, developers primarily relied on rule-based systems and Markov chains to generate jokes. Rule-based systems are software programs that generate output based on a set of predefined rules, while Markov chains are mathematical models that use probability theory to generate output based on a given input. While these systems could generate jokes, the quality and novelty of the output were often limited, as the algorithms were not able to generate truly creative or unexpected jokes.

Another approach used in joke generation was the use of pre-written jokes or scripts, which were often manually curated and added to the app. While this approach allowed for higher quality and more engaging content, it was time-consuming and limited the app's ability to generate new and original content on a regular basis.

With the development of advanced machine learning algorithms such as OpenAI GPT-3.5-Turbo, developers now have access to more powerful tools for joke generation in mobile applications. These algorithms can generate more creative and engaging content that is tailored to the app's target audience, making them a valuable tool for developers looking to create high-quality humor-based Android apps.

2.3 OpenAI GPT API AND AI MODEL

In this paper, we focus on the use of OpenAI API and OpenAI GPT-3.5-Turbo model for developing a humor-based Android application that generates jokes. The GPT-3.5-Turbo model is a language model developed by OpenAI, which is designed to generate human-like natural language text in response to prompts. It is a state-of-the-art language model that utilizes deep learning techniques, including deep neural networks, to generate high-quality text.

We explore the advantages of using GPT Turbo for generating humorous content and discuss how it compares to older models used in joke generation, such as rule-based systems and Markov chains. We also discuss the challenges and limitations of using GPT Turbo in joke generation, and how these can be addressed to build a high-quality, engaging humor-based Android app[5].

2.4 USE OF OpenAI GPT IN ANDROID APPS

The OpenAI GPT API and GPT language model has been used in a variety of AI-based Android apps, ranging from language translation to chatbots and text summarization. Its advanced natural language processing capabilities have made it an ideal tool for developing humor-based Android apps that generate jokes. Using the GPT API, developers can train their models on large datasets of text to generate highly contextual and relevant jokes that can keep users engaged and entertained. Additionally, the API's flexibility allows developers to fine-tune their models for specific use cases, ensuring that the generated jokes match the intended tone and style of the app. The use of GPT in Android apps has shown great potential for improving the quality and interactivity of the user experience, and it is expected to become an increasingly popular tool in the development of AI-based Android apps in the coming years[2].

2.5 CHALLENGES OF GPT FOR JOKE GENERATION

While the OpenAI GPT-3.5-Turbo model offers many advantages for joke generation in Android apps, there are also some challenges that developers need to be aware of. One of the primary challenges is the size of the model, which can be quite large and require significant computational resources to train and run. This can be a limiting factor for developers who have limited computing power or limited access to cloud-based resources.

Additionally, the accuracy and quality of the generated jokes can vary based on the quality and quantity of the training data used, as well as the specific parameters and settings of the GPT (Generative Pre-trained Transformer) model. Finally, as with any machine learning model, the GPT API may generate inappropriate or offensive content, which can be a concern for developers who want to ensure that their apps are appropriate for all users. Addressing these challenges requires careful attention to the training data, model settings, and content filtering mechanisms, as well as ongoing monitoring and refinement of the model over time.

3 METHOD

3.1 PROBLEM IDENTIFICATION

The COVID-19 pandemic has brought about unprecedented levels of stress and anxiety worldwide. With millions of people forced to stay indoors due to social distancing measures, there has been a surge in demand for entertainment and distraction. While there are many apps that provide entertainment, there is a lack of apps that specifically address the need for humor and levity during this difficult time.

Research shows that humor can have a positive impact on mental health, reducing stress and anxiety levels. However, many people are struggling to find ways to access humor and joy in their daily lives. There is a clear need for an app that provides a steady stream of humorous content, tailored to the interests and preferences of individual users[4].

This app aims to fill this gap by providing users with a platform to access a wide range of jokes in different categories. By leveraging the capabilities of the OpenAI GPT API and GPT-3.5-Turbo AI model, the app generates humorous content that is tailored to the user's interests and preferences. The app aims to help users alleviate stress and anxiety by providing a much-needed source of humor and levity during the ongoing pandemic.

3.2 COMPETITIVE ANALYSIS

To better understand the market and potential competitors, we conducted a thorough analysis of existing apps and solutions that aim to address the same problem. Our research revealed that there are several apps available that offer similar functionality to our joke app, one such example is Jokester.

However, upon closer examination, we found that these apps are either limited in their joke categories, have a smaller database of jokes, or lack the ability to generate new jokes on demand. Our app aims to stand out by offering a wider range of joke categories as well as the ability to generate new jokes using the OpenAI GPT API and GPT-3.5-Turbo model[5]. Additionally, our app has a user-friendly interface that allows users to easily navigate and request jokes.

We believe that our app offers a unique and valuable solution to the problem of providing users with fresh and funny jokes on demand and will stand out in a crowded market of joke apps.

3.3 TECHNOLOGY SELECTION

Our app leverages a variety of technologies to deliver a seamless and engaging user experience. The following are the key technologies we are using in the development of our app:

1. **Android Studio:** We used Android Studio, the official Integrated Development Environment (IDE) for Android app development, to build our app.

2. **Kotlin:** We are using Kotlin, a modern programming language that is fully supported by Android Studio, to write the code for our app[6]. Additionally, in 2017, Google announced that Kotlin would be an official Android development language, which has contributed to its popularity and widespread adoption in the Android development community. As such, we are confident in Kotlin's stability, reliability, and continued support from Google, making it an excellent choice for our app development needs[7].

3. **OpenAI GPT:** We are utilizing the OpenAI GPT API and GPT-3.5-Turbo model to generate jokes in real-time for our users. This state-of-the-art API uses deep learning algorithms to generate natural language responses to user prompts[5].

4. **Retrofit:** We are using Retrofit, a type-safe HTTP client for Android and Java, to handle network requests and API calls in our app.

5. **Material Design:** We followed Material Design guidelines to create a beautiful and intuitive user interface for our app[8].

6. **Git/GitHub:** In terms of version control, we used Git and GitHub to manage the source code of our app, and this allowed us to track changes and manage versions.

By leveraging these cutting-edge technologies, we can create an app that is fast, reliable, and easy to use. Our app leverages the power of deep learning algorithms to generate jokes in real-time, making it an innovative and engaging addition to the Android app market.

3.4 DEVELOPMENT PROCESS

The development process of our app involves several stages, starting with research and planning, followed by design, coding, and testing. We decided to use Kotlin as the primary programming language for our Android app and integrate the OpenAI GPT API and GPT-3.5-Turbo model to generate jokes[9].

We surveyed potential users to better understand their preferences and expectations from such an app. After gathering requirements, we designed the app's interface and functionality. We decided to keep the interface simple and user-friendly, with multiple buttons to generate jokes.

3.4.1 Integration of OpenAI GPT API: We integrated the OpenAI GPT API and GPT-3.5-Turbo AI model into our joke generator app using Retrofit, a popular REST client library for Android. We used Retrofit library to send HTTP requests to the API endpoint using the library's built-in functionality. The response from the API was parsed using JSON to extract the generated joke, which was then displayed to the user in the app's interface[5,10].

3.4.2 User Experience Design**:** The user experience design for the joke generator app involved creating a wireframe to map out the basic layout and functionality. The app was designed to be simple and intuitive to use, with a clean and minimalist design. The app features a single main activity and multiple fragments(screens) design that includes one fragment with buttons for different categories such as “book”, “food”, “animal”, “holiday”, “nature”, and “random”. When the user clicks on a specific button, a prompt is sent to the OpenAI GPT API in the backend, and the generated joke is displayed is displayed in a new page [11,12].

The goal is to create an app that provides users with a fun and entertaining way to generate jokes on-the-go while being user-friendly and functional. The use of technologies such as Kotlin and the OpenAI GPT API was essential in achieving these goals.

3.5 TESTING AND VALIDATION:

To ensure the quality and reliability of our joke generator app, we implemented a comprehensive testing process. Our testing plan includes both automated and manual testing methods. We used JUnit testing frameworks to test the app's functionality and identify any issues or bugs in the code.

In addition to automated testing, we also conducted manual testing to evaluate the user experience and usability of the app. We requested a group of users to provide feedback on the overall experience and to evaluate different design and functionality options, and the quality of the jokes. [9, 11].

Overall, our testing and validation process helped us to ensure that our app is functioning correctly and meeting the needs of our users. We will continue to monitor the app and make improvements as necessary to ensure that it remains reliable and effective.

4 IMPLEMENTATIONS

In this section, we will discuss the implementation details of our joke generator android app. The development process involved several stages, starting with research and planning, followed by design, coding, and testing. We utilized Kotlin as the primary programming language and Android Studio IDE for our Android app and integrated the OpenAI GPT API and GPT-3.5-Turbo model to generate jokes. The integration of the API and model was done using Retrofit, a popular REST client library for Android. The user experience design was kept simple and intuitive, with a clean and minimalist design. Multiple rounds of user testing were conducted to refine and enhance the design further.

4.1 APP UI DESIGN

The app is designed to be simple and intuitive to use, with a clean and minimalist design. The app features a single main activity and multiple screen UI design, and each screen is known as a fragment[14,15].

The application consists of single main Activity and three fragments(screens) that are each designed to provide a unique user experience:

1. **Main Activity Screen:** The main activity screen contains the navigation component called “navHostFragment” which helps to host and navigate between multiple screens.

2. **Button Fragment:** This fragment utilizes a Linear Layout to display a set of buttons, each corresponding to a different category of jokes such as “book”, “food”, “animal”, “holiday”, “nature”, and “random”. When the user clicks on a specific button, a prompt is sent to the OpenAI GPT API in the backend.

3. **Joke Fragment:** This fragment utilizes a Relative. Layout to display the generated joke in a TextView. Additionally, the fragment contains two buttons: "more" and "rate". The "more" button navigates the user back to the Button Fragment, allowing them to generate more jokes in the selected category. The "rate" button navigates the user to the Rate Fragment.

4. **Rate Fragment:** This fragment utilizes a Linear Layout to display a set of image views corresponding to each rating option. The layout is designed to be intuitive and user-friendly, with each image view corresponding to a different rating option. When the user clicks on an image view, the rating is displayed on the screen such as “Good Joke, rated 4” or “Hilarious Joke, rated 5” etc. and user is navigated back to the Button Fragment to generate more jokes.

4.2 APP ARCHITECTURE

For our Android app, we implemented the Model-View-Controller (MVC) architecture pattern to ensure a clear separation of concerns and improve the maintainability and scalability of the app[16].

In the MVC architecture pattern, the app's components are divided into three layers:

1. **Model:** The Model layer of the app handles the retrieval, storage, and manipulation of data, which in this case involves sending requests to the OpenAI GPT API to generate jokes.

2. **View:** The View layer represents the user interface of the app. Here the Joke Fragment which has the TextView for displaying the generated jokes represents the View.

3. **Controller:** The Controller layer acts as a bridge between the View and Model layers. It receives user input from the View and translates it into actions on the Model. It also updates the View with data from the Model. In this application, the Button Fragment acts as a controller, which observes the live data. Once the joke is retrieved from the API, the TextView is updated with the joke.

Communication between the layers is implemented through interfaces and callbacks, ensuring that each layer is responsible for its specific task.

4.2.1 Retrofit Library and API Integration

We used the Retrofit library to simplify the process of making API requests for our app. Retrofit is a type-safe HTTP client library for Android and Java that makes it easy to consume RESTful web services. By defining an interface with methods for each API endpoint, specifying the request method, headers, query parameters, and request body, we were able to send HTTP requests to the base URL “https://api.openai.com/v1/” and “/chat/completions” API endpoint. The response from the API was parsed using JSON to extract the generated joke, which was then displayed in TextView[17].

4.2.2 API Key

To use the GPT-3.5-Turbo model in our app, we needed to obtain an API key from OpenAI. An API key is a unique identifier that provides access to a specific set of APIs. We used the OpenAI API key to authenticate our app with the OpenAI servers and to issue HTTP requests to the GPT-3.5-Turbo model to generate jokes in response to user prompts.

The use of an API key allows OpenAI to manage and control access to their resources, including the GPT-3.5-Turbo model.

This ensures that only authorized users can access the model and helps to prevent misuse or abuse of the API.

4.2.3 Prompt Design

When a user clicks a button, a “prompt” is sent to the GPT-3.5-Turbo model as a query to generate a joke in a specific category. This is because the model functions as a chat completion model. To enhance the user experience, we designed specific prompts for our joke generator that allowed users to customize their joke requests. Our prompts included a user role and specific category requests such as "Tell me a funny joke about different animals”. This ensured that users could generate new and unique jokes in specific categories[12].

4.2.4 Input Parameters for API Call

In addition to the API endpoints, API Key, and prompt, we added a temperature parameter to our API endpoints that allows users to control the level of randomness in generated jokes. This made our joke generator stand out as a fun and engaging app, allowing users to easily generate unique and entertaining jokes in various categories[10].

5 JOKE EVALUATION

To evaluate the effectiveness of the joke generation system, we conducted a survey with a few participants who were asked to rate the generated jokes from 1 to 5 based on their humor quality. The survey included a mix of age groups and genders to ensure diversity in the sample. We also recorded the time it took for the app to generate and display the joke to the user.

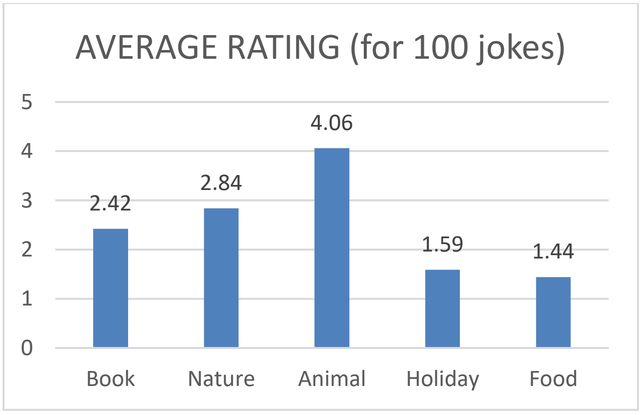

Figure 1: Joke Quality Evaluation (Average rating for 100 jokes in each category)

The survey conducted among the users of the app revealed that the animal jokes generated by the joke generator were the most popular among the respondents, receiving an average rating of 4.06 out of 5. This indicates that users found these jokes to be the most entertaining and enjoyable. On the other hand, the food jokes generated by the app received the lowest average rating of 1.44 out of 5, indicating that they were the least popular among the users.

It is worth noting that the rating scale was subjective and may have varied among the users, but these results provide an insight into the preferences of the users and can be used to improve the app's performance and user satisfaction.

6 APPS PERFORMANCE AND CHALLENGES

During the implementation of our app, on average, the response time to generate the joke was around 3 seconds, which was acceptable for our use case. However, we noticed that continuous clicks on the same button resulted in a delay of around 20 seconds before making another API call. This is due to the OpenAI API's restriction on the number of requests that can be made in a short period.

Moreover, we realized that the quality of the generated jokes can be heavily influenced by the prompts used to generate them. To avoid repetitive jokes, it is crucial to use specific prompts that provide context to the OpenAI GPT Turbo model. However, it is equally important to avoid prompts that are too specific or contain biased language, as they can lead to inappropriate or offensive jokes. Therefore, we carefully curated and refined the prompts used in our app to ensure they are entertaining and appropriate for a diverse audience.

6.1 API KEY SECURITY

In addition to these challenges, we also had to consider the security of the OpenAI API key used in the app. Firstly, we stored the key in a local properties file and accessed it through code, rather than hardcoding it into the app. This minimized the risk of the key being exposed through version control or other means. Additionally, we made sure to limit the access to the API key only to the necessary parts of the app and did not expose it in any user-facing parts of the application[18].

7 MAINTENANCE AND UPDATES

We recognize that maintaining and updating our app is crucial to providing an exceptional user experience. Our comprehensive plan includes ongoing monitoring, bug fixing, content updates, and incorporating user feedback. By continuously monitoring the app's performance and promptly addressing any issues or bugs that arise, we ensure that users have a seamless experience. Additionally, we will regularly update the app's content to keep it engaging and relevant. We value user feedback and will actively seek it out to ensure that we are meeting their needs and expectations. We also plan to stay ahead of the curve by incorporating industry trends and user needs into future enhancements. Our commitment to this plan is a testament to our dedication to providing the best possible experience for our users.

8 CONCLUSIONS

Our evaluation of the OpenAI GPT API and GPT-3.5-Turbo AI model showed promising results in generating high-quality jokes for certain categories, such as animal jokes. However, the generated jokes in some categories, such as food jokes, were less diverse and may require further fine-tuning. Overall, our project demonstrates the potential of utilizing the OpenAI GPT API and GPT-3.5-Turbo for developing humor-based content on the Android platform. Further experimentation and optimization could be done to enhance the quality and diversity of the generated jokes and expand the range of categories covered. We also suggest incorporating more objective measures of humor quality, such as user ratings and feedback, to improve the evaluation process.

9 FUTURE DEVELOPMENT

In the future, we plan to further enhance the app's functionality by incorporating more joke categories, advanced features such as a share feature and a database room to store favorite jokes. Additionally, we will integrate more sophisticated machine learning algorithms and natural language processing techniques to generate more nuanced and personalized jokes that can adapt to individual user preferences over time. We also plan to expand the app's user base through targeted marketing and outreach efforts. Finally, we will continue to incorporate user feedback and suggestions into future updates to ensure that the app remains relevant and engaging for joke lovers everywhere.

REFERENCES

[1] Md Romael Haque and Sabirat Rubya. 2022. “For an App Supposed to Make Its Users Feel Better, It Sure is a Joke” - An Analysis of User Reviews of Mobile Mental Health Applications. *Proc. ACM Hum.-Comput. Interact.* 6, CSCW2 (November 2022), 421:1-421:29. DOI:https://doi.org/10.1145/3555146

[2] Oğuzhan Katar, Dilek Ozkan, GPT, Özal Yildirim, and U Rajendra Acharya. 2022. *Evaluation of GPT-3 AI language model in research paper writing*. DOI:https://doi.org/10.13140/RG.2.2.11949.15844

[3] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language Models are Unsupervised Multitask Learners.

[4] humor_in_medicine.pdf. Retrieved February 17, 2023 from https://www.utmb.edu/gem/pdfs/humor_in_medicine.pdf

[5] OpenAI API. Retrieved February 17, 2023 from https://platform.openai.com

[6] Kotlin for Android | Kotlin. *Kotlin Help*. Retrieved March 10, 2023 from https://kotlinlang.org/docs/android-overview.html

[7] Android Announces Support for Kotlin. *Android Developers Blog*. Retrieved March 10, 2023 from https://android-developers.googleblog.com/2017/05/android-announces-support-for-kotlin.html

[8] Material Design for Android. *Android Developers*. Retrieved March 10, 2023 from https://developer.android.com/develop/ui/views/theming/look-and-feel

[9] The activity lifecycle | Android Developers. Retrieved March 10, 2023 from https://developer.android.com/guide/components/activities/activity-lifecycle

[10] https://platform.openai.com/docs/api-reference/chat/create.

[11] Add buttons to your app | Android Developers. Retrieved March 10, 2023 from https://developer.android.com/develop/ui/views/components/button

[12] https://platform.openai.com/docs/guides/completion/prompt-design.

[13] Design for Android. *Android Developers*. Retrieved March 10, 2023 from https://developer.android.com/design

[14] https://developer.android.com/guide/fragments.

[15] https://developer.android.com/guide/navigation/navigation-getting-started.

[16] https://developer.android.com/topic/architecture.

[17] https://platform.openai.com/docs/api-reference/making-requests.

[18] https://help.openai.com/en/articles/5112595-best-practices-for-api-key-safety.