Conversational Chatbot as an Online Librarian

Shambhavi Sinha; Saurav Suresh; Atharva Belamkar; Akhil Bannur

Abstract - In this Natural Language Processing project, we have attempted to build a Telegram based chatbot for libraries. The convenience of our system is its ability to seamlessly integrate itself with the popular messaging application, Telegram. The chatbot is able to take in the text through Telegram and perform its various functions. The primary features of the bot include retrieving all the information about names of books, bookID and the links to the books , thanks to the Google Books API. The reason for choosing this particular problem is the fact that it is highly scalable.

Index Terms - Chatbot, API, Library, Natural Language Processing

1 Introduction

Chatbots, also known as conversational agents, are designed with the help of AI (Artificial Intelligence) software. They simulate a conversation (or a chat) with users in a natural language via messaging applications, websites, mobile apps, or phones. The book bot is a Telegram bot for answering book-related questions. Users are provided an intuitive interface that allows them to easily access information about books, very similar to asking a bookstore clerk. The bot relies on the APIs of Google Books to extract structured information about books. Using the chatbot is easy, as there are no specific keywords to memorize. Users can simply ask a question in the same way they would also ask a book clerk. This allows for greater flexibility, but also places higher requirements on the natural language understanding component of the system: It needs to be able to deal with certain semantic properties of natural language, in order for communication to be successful. We tried addressing one particularly important property, namely co-reference.

The yearning of humans to converse with a computer system dates back to the inceptive days of computer science. Alan Turing's exposition on imitation games and learning machines, popularly known as the Turing test, provided the embryonic thrust to the possibility of computers having natural and intelligent conversations with humans[1]. Over the years, with the disruptive evolution of Information and Communication Technologies (ICT), computer-based conversational agents, or chatbots in popular parlance, finally became a reality. In simple terms, chatbots are goal-based computer programs developed to imitate and effect intelligent conversational behavior similar to humans, through the integration of an array of suitable computing technologies. Motivations behind developing such systems included real-time intelligent patron engagement and support, austerity in serving costs, and the urge for increased revenues, amongst others. The generic characteristics envisioned and expected from chatbots like intent recognition, entity extraction, dialog automation and management, anthropomorphism and feedback-based correction were in sync with the motivational attributes mentioned before[2]. There have also been numerous parametric classifications of chatbots with respect to their domain of activity, methods of operation, or the kind of services provided. The most significant among them, though, remains the categorization into chatbots powered either by state machines or by artificial intelligence (AI). State machines (also known as a finite automaton) are abstract machines storing a singular state amongst any finite number of states at a given time (essentially, operating through nested conditional programming constructs), and transitioning between states in response to inputs[3]. Artificial Intelligence (AI), on the other hand, implies the implementation of concepts and algorithms from sub-areas within AI (such as Machine Learning, Knowledge Representation, etc.) to achieve a certain degree of learning capability and intelligence to achieve certain stated goal(s) with considerable autonomy. It is but natural to mention that the AI-powered chatbots would be much more inherently adept in handling unpredictable semantics than their state machine-based counterparts. Also, a long-standing matter of concern in the development of conversational software remains the dearth of effective and powerful open-source software platforms. It is needless to mention the exorbitant cost of proprietary conversational AI technologies, in terms of purchase, support, and human resource training. An open-source conversational AI platform will have advantages like community support, customized training datasets, and adaptive integration with backend systems, without any of the extra cost except technical training. Rasa Stack provides a powerful yet easy-to-use python-based open-source software alternative to build contextual conversational agents[2]. The overarching thrust of the paper is to motivate libraries and allied knowledge-intensive institutions to adopt conversational software in general, and the technical finesse of Rasa in specific.

Advantages of Chatbot:

- Facilitate Seamless Live Communication
- Make Customer Service Available 24/7
- Save Time and Money
- Reduce People-to-People Interactions with Customers
- Eliminate Tedious Time-Consuming Tasks
- Offer a Smoother Customer Journey

- Reduce Stress for Consumers
- Eliminate Interactive Voice Response (IVR) Systems
- Humanize Your Brand
- Make Marketing More Targeted
- Help Grow Your Business
- Get Constant Improvement Over Time With Machine Learning

The rest of the paper is organized as follows: Section 2.0 provides a preliminary glimpse of the state-of-the-art implementation of chatbots in various libraries and related issues. Section 3.0 introduces the technical features of the Rasa open stack and elucidates a possible conceptual architecture for a library chatbot utilizing it.

2 LITERATURE SURVEY

Chatbots are normally given a name and represented by an avatar. Having a well-conceived chatbot could transform a library, normally thought of as a place, into something more personal. I have observed a number of people who are regular users of both public and academic libraries that keep going back not to go to the library, but to seek help from a specific person they have developed a rapport with. They are not going to a place for help, but to someone. It is a bit of a stretch -- I admit -- as a chatbot can only take this idea so far, and different people will react differently to chatbots, but it does add some unique personality to the traditional library setting no matter what.

Libraries have always been at the forefront of recognizing, absorbing and effectuating technology into their folds as compared to other academic and memory institutions. Some have even considered adopting artificial conversational entities for purposes like reference interviews, in an attempt to re-attract users to utilize their services.

In fact, the University of Nebraska-Lincoln library's Artificial Intelligence Mark-up Language (AIML) based Pixel chatbot (essentially, based on pattern matching) was among the first such chatbots to go live in the USA5 . There has also been a serious deliberation among Canadian libraries about incorporating chatbots and aligned Human Computer Interaction (HCI) techniques in facilitating dedicated library services such as web site navigations, digital reference interviews and virtual story narrations6. A collaborative project shared amongst select Swiss public libraries, enterprises and information science student groups developed the first Swiss library chatbot – Kornelia (also the first public library chatbot in the world)7. A library chatbot prototype has been developed at the University of Technology Sydney (UTS) based on a comprehensive explanatory study of the role of librarians in ensuring the friendly and trustworthy conversational design of a library chatbot[8] .

The possible utilisation of chatbots in special libraries such as law libraries have also been discussed and researched upon . The role of chatbots as instructional and learning technologies employed in different institutional arenas has also been enthusiastically debated by the scientific community10.

Generally speaking, there are several insights from the studies above which necessitate the adoption and implementation of AI-powered open-source conversational software platforms like Rasa in libraries, and reinforces its superiority and novelty.

Firstly, almost all of the existing library chatbots have been developed on platforms which are essentially either tuned for small-scale use (lacks scalability) or designed for exclusive research purposes. None of the chatbots cited above can satisfactorily handle complex, contextual dialogue management scenarios (which requires a strong embedded grounding of Machine Learning techniques) and composite Application Programming Interface (API) interactions. Professional documentation and ease of learning are the other critical features which many of these chatbot development platforms lack. . Finally, most of the chatbot development software tools above are not completely platform independent.

2 METHODOLOGY

2.1 PYTHON ENVIRONMENT

A virtual environment is a Python utility for managing dependencies and isolating projects. They enable Python site packages to be deployed locally in an isolated directory for a specific project rather than globally. We will be able to construct a distinct location for all of the libraries related to this project with the aid of this Python environment.

For this project, we will need to install the following dependencies/ libraries and APIs.

- Telebot
- NLTK
- OS
- sys
- Telegram API
- Google Books API

2.2 TELEBOT

Telebot is a bot framework for Telegram Bot API. This package provides the best of its kind API for command routing, inline query requests and keyboards, as well as callbacks.

2.3 NLTK

NLTK (Natural Language Toolkit) Library is a suite that contains libraries and programs for statistical language processing. It is one of the most powerful NLP libraries, which contains packages to make machines understand human language and reply to it with an appropriate response.

2.4 OS

The OS module in Python provides functions for interacting with the operating system. OS comes under Python's standard utility modules. This module provides a portable way of using operating system dependent functionality. The os and os.path modules include many functions to interact with the file system.

2.5 SYS

The sys module in Python provides various functions and variables that are used to manipulate different parts of the Python runtime environment. It allows operating on the interpreter as it provides access to the variables and functions that interact strongly with the interpreter. Let's consider the below example.

2.6 TELEGRAM API

A simple, but extensible Python implementation for the Telegram Bot API that supports both sync and async ways. In general terms, an implementation of an HTTP-based API written in a particular language simplifies building your program by allowing you to specify your ID's and credentials, and then simply calling functions to get things done, instead of having to engineer every call and include the parameters manually or repetitively. It also takes care of constructing the requests and passing the parameters in a compatible format to the server, while it may also provide some additional parsing to the parameters provided to the abstracted functions.

2.7 GOOGLE BOOKS API

Google Books is Google's effort to make book content more discoverable on the Web. Using the Google Books API, your application can perform full-text searches and retrieve book information, viewability, and eBook availability. You can also manage your personal bookshelves. Google Books has a mission to digitize the world's book content and make it more discoverable on the Web. The Books API is a way to search and access that content, as well as to create and view personalization around that content.

2.8 Creating the Bot

Before we begin developing the bot, let's take a brief view at how the bot will function. The bot should be able to handle commands like "/greet" "/hello" and "/start" and regular text like "find Harry Potter books," There is no specific format that the input messages have to conform to making is very user friendly and accessible to those who are not familiar with the bot interface.

In this project, we have used the pyTelegramBotAPI 3.7.7. his API is tested with Python 3.6-3.10 and Pypy 3. There are two ways to install the library:
Installation using pip (a Python package manager)*:

```
$ pip install pyTelegramBotAPI
```

Installation from source (requires git):

```
$ pip install git+https://github.com/eternnoir/pyTelegramBotAPI.git
```

It is presumed that you have obtained an API token with @BotFather. We will call this token TOKEN.

We have used the message handler function that is decorated with the message_handler decorator of a TeleBot instance. Message handlers consist of one or multiple filters. Each filter must return True for a certain message in order for a message handler to become eligible to handle that message. A message handler is declared in the following way (provided bot is an instance of TeleBot):

```
@bot.message_handler(filters)
def function_name(message):


    bot.reply_to(message, "This is a message handler")
```

Here function_name is not bound to any restrictions. Any function name is permitted with message handlers. The function must accept at most one argument, which will be the message that the function must handle. filters is a list of keyword arguments. A filter is declared in the following manner: name=argument. One handler may have multiple filters.

In order to fetch the data about the books we have used the google-api-python-client.
Installation using pip (a Python package manager)*:

```
$ pip install --upgrade google-api-python-client
```

Using the google books api, the user input is sent as a search query to google books and the top 5 search results are returned along with the image of the book which is the first search result as it is the most accurate match.

The functions used in the bot are as follows:-

1) Functions of the PyTelegramBot:
   a) reply_to: It replies to the user query. We have used pre-programmed responses.
   b) send_message: It sends a message back to the user after the user sends a query.
   c) send_photo: It sends a photo back to the user as a message, the caption feature of this function is used to respond with information that the user requested.
   d) message_handler: It waits for user input in the chat.
2) User defined functions:
   a) Bot commands: We are using 3 pre-programmed bot commands (/greet,/hello and /start) which respond with fixed responses.
   b) send_book: This is a function which has a dynamic response. It responds with the top 5 search results based on the user input.
3) Other Functions:
   a) word_tokenize: This is a function from the nltk library used to tokenize a sentence.

In order to make our bot easy to use we have used the NLTK python library and performed some preprocessing on the input text message. NLTK is a natural language toolkit in which we have imported modules like stopwords which is nothing but "dictionary" and PorterStemmer to generate a root word. From the NLTK library, we have to download stopwords for text cleaning and then tokenization is performed. The processed text is now sent to the google-api-python-client as a request which is integrated with pyTelegramBotAPI to send the top 5 books search results containing an image of the first book and the names, book ids and the links to all 5 books as a message to the user.

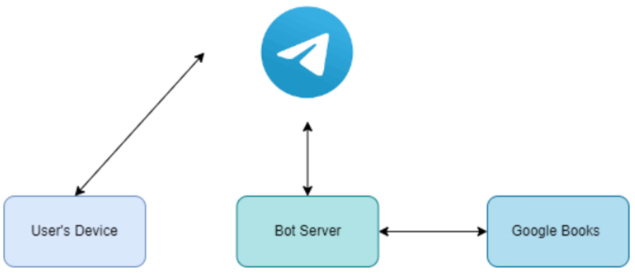
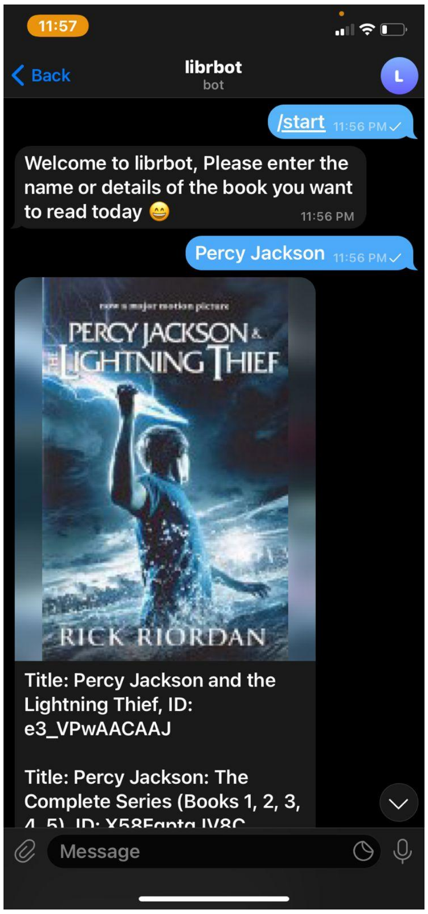


Figure 2 : System Block Diagram

3 Result

This section presents the results obtained from the project.



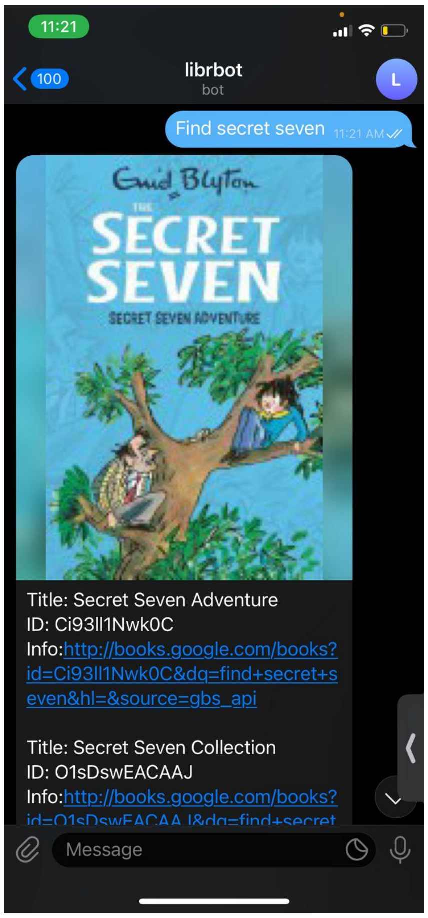
/start 11:56 PM

Welcome to librbot, Please enter the name or details of the book you want to read today 😄 11:56 PM

Percy Jackson 11:56 PM

Title: Percy Jackson and the Lightning Thief, ID: e3_VPwAACAAJ

Title: Percy Jackson: The Complete Series (Books 1, 2, 3, 4, 5), ID: X58Fqntq IV8C



Find secret seven 11:21 AM

Title: Secret Seven Adventure
ID: Ci93ll1Nwk0C
Info:http://books.google.com/books?id=Ci93ll1Nwk0C&dq=find+secret+seven&hl=&source=gbs_api

Title: Secret Seven Collection
ID: O1sDswEACAAJ
Info:http://books.google.com/books?id=O1sDswEACAAJ&dq=find+secret

Figure 2 : Result Screenshots of the Chatbot Functioning

The chatbot is activated using a keyword- /start. On sending the code-word a generic message is popped out by the chatbot regarding the usage of the chatbot and what are the different features of the chatbot.

From Figure 2, it can be seen that the chatbot is returning the name of the author as well as listing various books from the same author or books from the same series.

4 Conclusion

In this mini project, we have created a simple Telegram chatbot that returns the library equivalent name of the books when the name of the author is mentioned and vice versa. It is also capable of mentioning all the books from a particular series with the help of Google Books API. The convenience of this bot is that it can be used seamlessly with Telegram with the help of Telegram API

References

[1] Allison, D. (2012), "Chatbots in the library: is it time?", Library Hi Tech, Vol. 30 No. 1, pp. 95-107.

[2] J. Purohit, A. Bagwe, R. Mehta, O. Mangaonkar and E. George, "Natural Language Processing based Jaro-The Interviewing Chatbot," *2019 3rd International Conference on Computing Methodologies and Communication (ICCMC)*, 2019, pp. 134-136, doi: 10.1109/ICCMC.2019.8819708.

[3] P. Voege and A. Ouda, "A Study on Natural Language Chatbot-based Authentication Systems," 2021 International Symposium on Networks, Computers and Communications (ISNCC), 2021, pp. 1-4, doi: 10.1109/ISNCC52172.2021.9615767.

[4] Bagchi, Mayukh. (2020). Conceptualizing a Library Chatbot using Open Source Conversational Artificial Intelligence. DESIDOC Journal of Library & Information Technology. 40. 329-333. 10.14429/djlit.40.06.15611.

[5] Matteo Carisi, Andrea Albarelli, and Flaminia L. Luccio. 2019. Design and implementation of an airport chatbot. In <i>Proceedings of the 5th EAI International Conference on Smart Objects and Technologies for Social Good</i> (<i>GoodTechs '19</i>). Association for Computing Machinery, New York, NY, USA, 49–54. DOI:https://doi.org/10.1145/3342428.3342664

[6] An Intelligent Chatbot System Based on Entity Extraction Using RASA NLU and Neural Network Anran Jiao 2020 J. Phys.: Conf. Ser. 1487 012014

[7] Heo, Miri & Lee, Kyoung. (2018). Chatbot as a New Business Communication Tool: The Case of Naver TalkTalk. Business Communication Research and Practice. 1. 41-45. 10.22682/bcrp.2018.1.1.41.

[8] P. Srivastava and N. Singh, "Automatized Medical Chatbot (Medibot)," 2020 International Conference on Power Electronics & IoT Applications in Renewable Energy and its Control (PARC), 2020, pp. 351-354, doi: 10.1109/PARC49193.2020.236624.

[9] Library Chatbots? Ratledge, David.Tennessee Libraries; Memphis Vol. 62, Iss. 3

[10] Adamopoulou E., Moussiades L. (2020) An Overview of Chatbot Technology. In: Maglogiannis I., Iliadis L., Pimenidis E. (eds) Artificial Intelligence Applications and Innovations. AIAI 2020. IFIP Advances in Information and Communication Technology, vol 584. Springer, Cham. https://doi.org/10.1007/978-3-030-49186-4_31