JSS Mahavidyapeetha

JSS SCIENCE AND TECHNOLOGY UNIVERSITY
MYSURU-570006

FINAL YEAR B.E PROJECT REPORT
2019-2020

Legal Chat Advisory

Submitted by

Name	USN	e-mail
Abhishek S Patil	**01JST16EC004**	abhishekspatil92@gmail.com
Anagha Bhupalam	**01JST16EC014**	anaghabhupalam@gmail.com
Ananya N Ambli	**01JST16EC015**	ananya.ambli@gmail.com
Anirudh B M	**01JST16EC017**	anirudh.balaiah@gmail.com

Submitted in partial fulfilment of the requirement of academicevent in BE

Under the Guidance of

Pavithra D R
Assistant Professor
Dept Of Electronics & Communication

JSS SCIENCE AND TECHNOLOGY UBIVERSITY MYSURU-570006

JSS SCIENCE AND TECHNOLOGY UNIVERSITY

DEPARTMENT OF ELECTRONICS & COMMUNICATION
Mysuru 570006

CERTIFICATE

This is to certify that the work entitled LEGAL CHAT ADVISORY is a bonafide work carried out by Abhishek S patil, Anagha Bhupalam, Ananya N Ambli and Anirudh B M. It is certified that all corrections / suggestions indicated during internal assessments have been incorporated in the report. The project report has been approved as it satisfies the academic requirements in respect of project work prescribed for the degree.

Dr. Shankaraiah
Professor and
Head, Dept of E&C
JSSSTU, Mysuru

Pavithra D R
Assistant Professor,
Dept of E&C
JSSSTU, Mysuru

Dr. T.N. Nagabhushan
Principal, JSSSTU Mysuru

External Viva

Name of Examiners	Signature with Date
1.	. . . . .
2.	. . . . .

DECLARATION

We, **Abhishek S Patil (01JST16EC004)**, **Anagha Bhupalam (01JST16EC014)**, **Ananya N Ambli (01JST16EC015)** and **Anirudh B M (01JST16EC017)** students of Bachelor of Engineering in Electronics and Communication, JSS Science and Technology University Mysuru, do hereby declare that this project entitled **"LEGAL CHAT ADVISORY"** has been carried out by us under the guidance of **Prof. D R Pavithra**.

We also declare that, to the best of our knowledge and belief, the matter embodied in this project report has not been submitted by us previously for the award of any degree to any other institution or university.

Place: Mysuru
Date:13.05.2020

ABHISHEK S PATIL
(01JST16EC004)

ANAGHA BHUPALAM
(01JST16EC014)

Ananya N Ambli
(01JST16EC015)

ANIRUDH B M
(01JST16EC017

Acknowledgement

The satisfaction and euphoria that accompanies successful completion of any task would be incomplete without the mention of people who made it possible. We would like to express our deep sense gratitude and indebtedness to the following.

We would like to thank the principal of JSS Science and Technology university Dr. T N Nagabhushan and HOD of E and C Dept. Dr. N Shankaraiah, for encouraging us to carry out our project work in the college.

We specially thank to our guide Assistant Professor Pavithra D R, Dept. of E and C for her precious inputs and suggestions from time to time.

We also thank our panel lectures, Dr M N Jayaram, Dept of E and C, Shivprasad N Assistant Professor, Dept of E and C, M P Madhusudan, Assistant Professor,Dept of E and C, for their support and encouragements in completing our project.

We appreciate the timely help and kind cooperation of our lecturers, other staff members of the department and our seniors, with whom we have come up all the way during our project work without whose support this project would not have been success.

Last but not the least we would like to express our gratitude to our parents and friends for their total support.

Project Members

Abhishek S Patil
Anagha Bhupalam
Ananya N Ambli
Anirudh BM

ABSTRACT

Chatbots have recently become popular due to the widespread use of messaging services and the advancement of Natural Language Understanding. Because of increasing population, all the rules and regulations are made stricter under the constitution. Under this scenario there will be many queries to the citizens regarding the rules and regulations of the country. We have developed the ‘Legal Chat Advisory’ which is a text based chatbot system which helps in answering the basic questions which the citizen of India have. This advisory system will gather all the information in the single platform which saves us from navigating through different websites.

Recently, advocates have found that communicating with their client with all their basic questions is a way lengthy process.The advocate can use this system for the client interation and the questions asked by the client will be recorded for the use. This advisory system can also be suggested to the advocates which helps them by providing all the legal information about the previous judgements and cases. Legal chatbots are great way of optimizing a work process to save lawyer’s time. The chatbot works round the clock to provide customers with legal help whenever they need it.

LIST OF FIGURES

TABLE OF CONTENTS

1 INTRODUCTION

1.1 MOTIVATION

Today, lawyers face the problem of routine and unbilled tasks. Their workdays mainly consist of administrative, financial and advertising task. However, they still have clients that need advice or legal documents created. In a hunt to increase revenue and raise client's satisfaction, most lawyer's work more than they were planning. One of the solutions to reduce this is the chatbot. Legal chatbots are great way of optimizing a work process to save lawyer's time. The chatbot works 24/7 to provide customers with legal help whenever they need it. Chatbots generate and analyse legal documents, perform due diligence and provide legal information to the clients as well as the law firms.

1.2 PROBLEM STATEMENT

It is evident from the research carried out in the literature review that the legal law firms are constantly seeking to expand their technologies, both to improve client service and increase efficiency and revenue through the advancements in technology. A domain specific chatbot will be implemented to assist users with the basic rules and regulations of law and also the IPC sections under the constitution. In order to overcome the client satisfaction issues and also the non-availability of lawyers at any time they need.

The chatbot will provide personal and efficient communication between the client and the law firm in order to manage the basic questions which the client wants to be answered by the law firms and get assistance when needed, such as answering any queries and booking appointments. The chatbot will allow users to feel confident and comfortable when using this service regardless of the user's computer literacy due to the natural language used in messages. It also provides a very accessible and efficient service as all interactions will take place within the one chat conversation negating the need for the user to navigate through a site.

1.3 OBJECTIVES

- As searching information about particular article is time consuming, the system provides concise information of article.
- Constructing a database by gathering information from previous cases and constitutions.
- Training the system using Natural Language Processing and ML algorithms.
- Creating an interactive chatbot using frameworks.
- Integrating our trained model with chatbot for prediction and providing related information.
- Creating a web application for easy interaction.

2 LITERATURE SURVEY

2.1 Previous Research

1. Integrated legal information retrieval: new developments and educational challenges.

Author: Kees (C.) van Noortwijk

The amount of legal information, available digitally, has increased gradually in the past three decades. We are now approaching a situation in which practically all legal information a lawyer needs on a daily basis can be obtained from digital sources. At the same time, powerful retrieval systems capable of integrating these sources and performing more effective search operations have become available. In this paper, new possibilities are outlined that have emerged now that such a large proportion of legal resources have been combined in unified collections. In this paper, some examples were given of new functionalities present in many of these systems. Possibilities occurring when 'semantic search' options would be incorporated were also described. These functionalities are of importance to legal professionals because they can improve the recall rate - and in some cases, the precision as well - of a search operation: a larger proportion of the relevant documents present in the huge, combined data collections can be retrieved with limited 'false hits', which can be vital for any lawyer involved in, for instance, a legal dispute or in litigation. We learned about three databases for legal information retrieval which are publicly accessible materials, commercial publications from legal publishers and know how knowledge of the lawyers.

2. Approaches for Information Retrieval in Legal Documents

Authors: Rachayita Giri, Yosha Porwaly, Vaibhavi Shuklay, Palak Chadhay and Rishabh

With crimes increasing at an alarming rate, it becomes essential to impart justice to the victims readily. To come to a final decision, lawyers need to study several previous judgments for research purposes. Reducing the time spent on research can speed up the judicial process

drastically. The time consumption mostly happens in two areas - searching for the right document and understanding that document. To start with, being able to get hold of the appropriate judgments or other legal documents is the most essential task for any legal professional, especially lawyers. Once a document is obtained, the next most integral and inevitable task is to read and re-read it, and come to necessary as well as needful conclusions after a comprehensive analysis. To resolve the first issue, there is a need for an efficient search system which can provide searching options based upon multiple views. This paper makes an effort at improving the search for users by providing them with search options based upon either the semantics of the word or based upon the IPC sections. It is important that laymen can access all the related judgments by entering just one keyword or phrase without bothering about the legal jargon. Post retrieval of documents, the lengthy texts have to be scrutinized for meaningful inferences. To reduce the time spent in reading texts, the paper intends to present the information in the judgments visually through semantic networks. This IR System provides the features of semantic and IPC section-based search to users, deriving information from semantic networks that are representative of the documents, so that a more efficient search system on legal documents can be put in place.

3. DoNotPay

Founder: Joshua Browder

DoNotPay chat-bot was originally built to contest parking tickets, but has expanded to include other services as well. As a "robot lawyer," DoNotPay is a downloadable mobile app that makes use of artificial intelligence to provide legal services to all users free of charge. It is currently available in the United Kingdom and United States (all 50 states). This app uses Artificial Intelligence - specifically an algorithm which deciphers everyday language. DoNotPay released data showing its software had helped people overturn 160,000 of 250,000 parking tickets since launch.

4. LISA

Owner: Robert Lawyer Limited, a private limited company registered in England.

Robot Lawyer LISA (Legal Intelligence Support Assistant) - the world's first impartial Artificial Intelligence (AI) lawyer. LISA's first product is a confidentiality agreement or non-disclosure agreement (NDA) which eliminates the need to use lawyers during the negotiation process by starting in the middle ground between both parties. A two way AI app that can be accessed from a computer or smart phone, LISA asks a series of questions to the initiator. The document is created and sent to the reviewer. The reviewer uses LISA to make any changes. Once everyone is happy, the document is ready to be signed and you have a legally binding NDA.LISA has also recently branched out into property tools, such as drafting commercial and business and residential leases.

5. Ross Intelligence

Founder: Andrew Arruda, Jimoh Ovbiagele and Pargles.

ROSS uses the supercomputing power of IBM Watson to comb through huge batches of data and, over time, learn how to best serve its users. The software can sort through something in a matter of seconds that would normally takes a human hours upon hours to review. One of the first places to use ROSS was the law firm Baker Hostetler, where the software handles bankruptcy cases. Employees enter commands into the software in everyday language, like when they need to find examples of precedence for specific cases. ROSS then searches through its legal database to produce the relevant information. In the event a new court decision emerges in the dead of night, ROSS can even send alerts in real-time - without having to pay juniors over-time.

6. Billy Bot**.**

Owner: Clerksroom

Billy Bot is a project by Clerksroom to test artificial intelligence and how it can be used to link lawyers together to find the right Barrister. Described as 'a cheekie chappie who loves to chat', Billy Bot is a junior clerk robot who is programmed to help you find the right barrister

or mediator for your legal problem. Billy Bot will do the work of a traditional barristers' clerk and also provide basic legal information to online users. Named after the clerk in the TV series Silk, the ultimate aim is for the chatbot to point users to free online legal resources and help them decide whether they need legal help and then, if they do, find them a lawyer – either a solicitor or direct access barrister – make appointments and deal with all the tasks a human clerk would.

7. Automio

Owner: Automio

With Automio, lawyers can use their very own lawyer bot to interview clients and create instant, customised advice, contracts and legal documents. Automio's lawyer bot technology reduces the time lawyers spend on the repetitive and lower value work. Clients of law firms are stoked to be able to serve themselves online and get instant legal documents. You can build your Lawyer Bot yourself, or Automio's bot building team can build your Lawyer Bot for you. Automio has a Marketplace where lawyers can create, buy, sell and resell Lawyer Bots. The beauty of this is lawyers are able to create new revenue streams that don't involve billing time.

8. Chatbot Technologies and Challenges

Author: Vagelis Hristidis

Chatbots have recently become popular due to the widespread use of messaging services and the advancement of Natural Language Understanding. This paper gives an overview of the technologies that drive chatbots, including Information Extraction and Deep Learning. It also discusses the differences between conversational and transactional chatbots – the former are trained on free-form chat logs, whereas the latter are defined manually to achieve a specific goal like booking a flight. It also provides an overview of commercial tools and platforms that can help in creating and deploying chatbots.

9. The Potential of Chatbot: Analysis of Chatbot conversations

Authors: Mubashra Akhtar, Julia Neidhardt , Hannes Werthner

The idea of utilizing computers for question answering tasks has been around from the early beginning of these systems. First algorithms with the aim to accomplish this were already implemented in the early 1960s. In recent years, chatbots have been gaining enormous popularity in various fields. In the context of business applications, they are considered as useful tools for improving customer relationships. In this paper, chat conversations between customers and the chatbot of a telecommunication company are analysed to find out if these interactions can be used to determine a) users' topics of interests and b) user satisfaction. To reach this goal, chat conversations were interpreted as sequences of events and user inputs were analysed with the help of text mining techniques. The study showed that based on users written conversational contributions, valuable insights on users' interests and satisfaction could be gained. The majority of users leave the chat conversation after a short period of time if the chatbot was not able to give the desired answer right away. Moreover, a huge number of conversations deal with similar topics.

10. A System of Simple Sentence Parsing Rules to Produce "Answer Matching" Chatbot.

Author: Alan Shaw, Ph.D.

Producing interactive chatbots usually involves complex sentence parsing approaches that are beyond the scope of material that can be handled in introductory CS courses. In this paper, a simple framework is presented for parsing a sentence from a user during a chat with a chatbot using an "answer matching" strategy. Using the three simple sentence parsing rules discussed in this paper, (i.e AND rule, OR rule and AND-OR rule) it is possible to introduce socially interactive computing projects into introductory programming classes helping students to investigate the potential for creating simple and complex chatbot interactions. This, in turn,

provides an opportunity to better understand the compelling nature of building powerful socially intelligent computing applications that engage in interesting social interactions.

11. Enterprise Crowd Computing for Human Aided Chatbots

Author: Alessandro Bozzon

A chatbot is an example of cognitive computing system that emulates human conversations to provide informational, transactional, and conversational services. Despite their widespread adoption, chatbots still suffer from a number of performance issue due to limitations with their programming and training. This paper discusses Human Aided Chatbots, i.e. chatbots that rely on humans in the loop to operate. Human Aided Chatbots exploit human intelligence, brought for instance by crowd workers or full-time employees, to fill the gaps caused by limitations of fully automated solutions.

12. Natural Language Processing based Jaro-The Interviewing Chatbot

Authors: Jitendra Purohit, Aditya Bagwe, Rishabh Mehta, Ojaswini Mangaonkar, Elizabeth George

Recently, recruiters have find the taxing to communicate with all their candidates about the interview process and it results in the hassle of conducting interviews. Also, in cases, where there are large volumes of applicants, communicating with thousands of candidates and conducting other screening duties add to the heap of recruitment problems. The proposed system in this paper, JARO addresses the common concerns that a candidate faces when it comes to attend the mass interviews. Some of the challenges faced are inconsistency in questions, different days, different times of the day, interviewer's mood, venue of the interview and the list goes on.

2.2 Summary of Literature Survey

By conducting Literature Survey we divided our project into three main sub-tasks i.e

1. Information Retrieval from the database:

   There are three main databases which we intend to target in our project namely publicly accessible law materials, commercial publications from legal publishers and know how knowledge of the lawyers.

2. Information processing:

   We found two main methods of processing the data from the data basis which are Content Integration and Content Aggression. Content Integration is a retrieval systems that are, in essence, operating independently of content to be retrieved, but capable of integrating multiple existing databases and retrieving content from these from one central console. To achieve that, the content integration system scans the separate, existing datasets and indexes every document it finds in them. To the user, all content seems to be in one huge database whereas the original documents are still in their respective, original databases. In Content Aggression, it does not actually integrate document collections, but are capable of 'commanding' separate searches in multiple existing document collections, from one central interface. The original database search engines perform the actual searching and results are combined afterwards. Content integration was found to be better. Some other features to improvise the search are semantic search which searches for a document based on concept rather than a keyword , Linked search i.e. if the search engine finds a document containing a link in it then it also checks that link, if it is relevant to the present search than that link is also included in the results.

3 WORKING

Our Legal Case Chatbot is built upon multiple technologies and uses multiple libraries. The main focus will be on the core working of the chatbot where the RASA api is used, Django as the server where they are connected to a front-end web framework. The Rasa implements Natural Language Understanding (NLU) and Deep Learning techniques in order to get the inference for specific tasks. Before feeding the data into the model. Text preprocessing is done in order to maintain generality in all the sentences, reducing overhead for the model and to ensure proper accuracy.

3.1 Text Pre-Processing

All the sentences have to undergo text-preprocessing which includes converting the sentences into lower case, removing the stopwords, removing all kinds of punctuations, HTML tags and lemmatizing all the words in the sentences.

A stop word is a commonly used word (such as “the”, “a”, “an”, “in”) that a search engine has been programmed to ignore, both when indexing entries for searching and when retrieving them as the result of a search query. The punctations and all kinds of tags are removed in order for the model to not get mis leaded by unnecessary characters during training. If the punctuations are present, then they will affect the model’s accuracy in a negative way since they do not have meaning in predicting any use case. Lemmatization is the process of grouping together the different inflected forms of a word so they can be analysed as a single item. Lemmatization is similar to stemming but it brings context to the words. So, it links words with similar meaning to one word. Example of lemmatization is shown in Figure 3.1

	original_word	lemmatized_word
0	trouble	trouble
1	troubling	trouble
2	troubled	trouble
3	troubles	trouble

	original_word	lemmatized_word
0	goose	goose
1	geese	goose

FIGURE 3.1: Lemmatization of words

3.2 Working of RASA

There are multiple policies used by RASA to determine the next action to be taken in a conversation.

Responding to simple intents:

Simple intents include contents like greeting a user, thanking the user or responding to the exit of the user. This is done by Memoization Policy inbuilt in RASA. The Memoization Policy remembers examples from training stories for up to a max_history of turns. The number of "turns" includes messages the user sent, and actions the assistant performed. For the purpose of a simple, context-less FAQ bot, we only need to pay attention to the last message the user sent, and therefore we'll set that to 1.

Responding to more complex intents:

Aside from the more rule-based policies we described above, RASA Core also has some ML policies to use. These come in as an additional layer in policy configuration, and only jump in if the user follows a path that you have not anticipated. It is important to understand that using these policies does not mean letting go of control over your assistant. If a rule based policy is able to make a prediction, that prediction will always have a higher priority and predict the next action. The ML based policies give your assistant the chance not to fail, whereas if they are not used your assistant will definitely fail, like in state machine-based dialogue systems. This type of user behavior is handled by TED Policy.

3.3 TED Policy

When a program can generalize, it is not required to hard-code a response for every possible input because the model learns to recognize patterns based on examples it's already seen. This scales in a way hard-coded rules never could, and it works as well for dialogue management as it does for NLU. This generalization Deep Learning algorithm is implemented using Transformer Embedding Dialogue (TED).

The transformers layers can also be replaced by Long Short Term networks. But transformers is implemented because, transformer architecture offers two advantages when it

comes to predicting dialogue across more complex, multi-turn conversations. It can decide which elements in the dialogue sequence are important to pay attention to, and it makes each prediction independently from the others in the sequence, so it's able to recover if a user interjects with something unexpected.

At each dialogue turn, the TED policy takes three pieces of information as input: the user's message, the previous system action that was predicted, and any values saved to the assistant's memory as slots. Each of these is featurized and concatenated before being fed into the transformer.

Here's where the self-attention mechanism comes into play: The transformer accesses different parts of the dialogue history dynamically at each turn and then assesses and recalculates the relevance of previous turns. This allows the TED policy to take a user utterance into account at one turn but ignore it completely at another, which makes the transformer a useful architecture for processing dialogue histories.

Next, a dense layer is applied to the transformer's output to get embeddings numeric features used to approximate the meaning of text for the dialogue contexts and system actions. The difference between the embeddings is calculated, and the TED policy maximizes the similarity with the target label and minimizes similarities with incorrect ones.When it's time to predict the next system action, all possible system actions are ranked according to their similarity, and the action with the highest similarity is selected.

This process is repeated across each dialogue turn, as shown in Figure 3.2

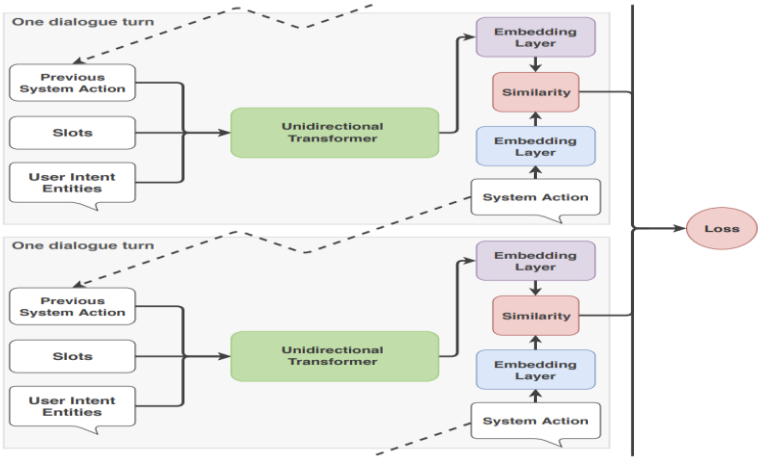

FIGURE 3.2: Two-time steps of transformer embedding dialogue policy

4 SOFTWARE COMPONENTS

4.1 PYTHON

Python is an interpreted, high-level, general-purpose programming language. Python's design philosophy emphasizes code readability with its notable use of significant whitespace. Its language constructs and object-oriented approach aim to help programmers write clear, logical code for small and large-scale projects. Python is dynamically typed and garbage-collected. Python is often described as a batteries included language due to its comprehensive standard library. Python is a language. It also features dynamic name resolution, which binds method and variable names during program execution. Python is a general purpose and high-level programming language. We can use Python for developing desktop GUI applications, websites and web applications. Also, Python, as a high-level programming language, allows us to focus on core functionality of the application by taking care of common programming tasks.

Features of Python:

- Readable and maintainable code: While writing a software application, we must focus on the quality of its source code to simplify maintenance and updates. The syntax rules of Python allow us to express concepts without writing additional code. At the same time, Python, unlike other programming languages, emphasizes on code readability, and allows us to use English keywords instead of punctuations. Hence, we can use Python to build custom applications without writing additional code. The readable and clean code base will help us to maintain and update the software without putting extra time and effort.
- Multiple programming paradigms: Like other modern programming languages, Python also supports several programming paradigms. It supports object oriented and structured programming fully. Also, its language features support various concepts in functional and aspect-oriented programming. At the same time, Python also features a dynamic type system and automatic memory management. The programming paradigms and

language features help us to use Python for developing large and complex software applications.

- Compatible with major platforms and systems: At present, Python is supports many operating systems. we can even use Python interpreters to run the code on specific platforms and tools. Also, Python is an interpreted programming language. It allows us to run the same code on multiple platforms without recompilation. Hence, we are not required to recompile the code after making any alteration. we can run the modified application code without recompiling and check the impact of changes made to the code immediately. The feature makes it easier for us to make changes to the code without increasing development time.
- Robust standard library: Its large and robust standard library makes Python score over other programming languages. The standard library allows us to choose from a wide range of modules according to our precise needs. Each module further enables us to add functionality to the Python application without writing additional code. For instance, while writing a web application in Python, we can use specific modules to implement web services, perform string operations, manage operating system interface or work with internet protocols. we can even gather information about various modules by browsing through the Python Standard Library documentation.
- Many open source frameworks and tools: As an open source programming language, Python helps us to curtail software development cost significantly. we can even use several open source Python frameworks, libraries and development tools to curtail development time without increasing development cost. we even have option to choose from a wide range of open source Python frameworks and development tools according to our precise needs. For instance, we can simplify and speedup web application development by using robust Python web frameworks like Django, Flask, Pyramid, Bottle and Cherrypy. Likewise, we can accelerate desktop GUI application development using Python GUI frameworks and toolkits like PyQT, PyJs, PyGUI, Kivy, PyGTK and WxPython.Python can be used to develop different applications like web applications, graphic user interface based applications, software development application, scientific and numeric applications, network programming, Games and 3D applications and chatbot development.

RASA NLU is primarily used to build chatbots and voice apps. To use Rasa we have to provide some training data. Rasa uses the machine learning to pick up patterns and generalise to unseen sentences. In Rasa action server we need to write the code in python, that is mainly used to trigger the external actions like Google API and Rest API. Python is a robust programming language and provides an easy usage of the code lines, maintenance can be handled in a great way, and debugging can be done easily too. It has gained importance across the globe as computer giant Google has made it one of its official programming languages.

4 .2 Django

Django is a Python-based free and open-source web framework, which follows the model-template-view (MTV) architectural pattern.

It is maintained by the Django Software Foundation , an independent organization. Django's primary goal is to ease the creation of complex, database-driven websites. The framework emphasizes reusability and pluggability of components, less code, low coupling, rapid development, and the principle of don't repeat yourself. Python is used throughout, even for settings files and data models. Django also provides an optional administrative create, read, update and delete interface that is generated dynamically through introspection and configured via admin models .

Despite having its own nomenclature, such as naming the callable objects generating the HTTP responses ,the core Django framework can be seen as an MVC architecture. It consists of an object-relational mapper that mediates between data models and a relational database ("Model"), a system for processing HTTP requests with a web templating system ("View"), and a regular-expression-based URL dispatcher ("Controller").

A Django framework also includes :

- A lightweight and standalone web server for development and testing
- A firm serialization and validation system that can translate between HTML forms and values suitable for storage in the database
- A template system that utilizes the concept of inheritance borrowed from object-oriented programming

- A caching framework that can use any of several cache methods
- support for middleware classes that can intervene at various stages of request processing and carry out custom functions
- An internal dispatcher system that allows components of an application to communicate events to each other via pre-defined signals
- An internationalization system, including translations of Django's own components into a variety of languages
- A serialization system that can produce and read XML and JSON representations of Django model instances
- A system for extending the capabilities of the template engine
- An interface to Python's built-in unit test framework
- Django REST framework is a powerful and flexible toolkit for building Web APIs.

Django's configuration system allows third party code to be plugged into a regular project, provided that it follows the reusable app conventions. The Django philosophy implies loose coupling, the template filters and tags assume one engine implementation, and both the auth and admin bundled applications require the use of the internal ORM. None of these filters or bundled apps are mandatory to run a Django project, but reusable apps tend to depend on them, encouraging developers to keep using the official stack in order to benefit fully from the apps ecosystem. Django officially supports four database backends: PostgreSQL, MySQL, SQLite, and Oracle. Microsoft SQL Server can be used with django-mssql on Microsoft operating systems, while similarly external backends exist for IBM Db2,SQL Anywhere and Firebird.

Django framework runs on any platform like PC, Windows, Mac, Linux etc. It provides a layer between the developer and database called object-relational mapper which makes it possible to move or migrate our applications to other major databases with few lines of code change. As Django is a well-maintained web application framework and widely used across the industries so cloud providers taking all measures to provide support to run Django applications easily and quickly on cloud platforms. It means, once Django applications deployed then it can be managed by an authorized developer with a single command in a cloud environment.

As Django developers are working in the same development environment for a long time so they will grow and expertise in these areas which means applications developed, websites created are getting better day by day, more functional, efficient and reliable. Django framework follows don't repeat yourself principle as it concentrates on getting most out of each and every line of code by which we can spend less time on debugging or code re-orientation etc. In general, Don't repeat yourself code means all uses of data change simultaneously rather than a need to be replicated and its fundamental reason to use of variables and functions in all programming.

With the uses of Django framework, we can develop and deploy web applications within hours as it takes care of much of the hassle of web development. Django is very fast, fully loaded such as it takes care of user authentication, content administration, security as Django takes it very seriously and helps to avoid SQL injection, cross-site scripting etc. We can build different applications from content management to social networking websites using Django framework.

4.3 JSON

JavaScript Object Notation is an open standard file format, and data interchange format, that uses human-readable text to store and transmit data objects consisting of attribute–value pairs and array data types or any other serializable value. It is a very common data format, with a diverse range of applications, such as serving as a replacement for XML in AJAX systems.

JSON is a language-independent data format. It was derived from JavaScript, but many modern programming languages include code to generate and parse JSON-format data. JSON was based on a subset of the JavaScript scripting language and is commonly used with JavaScript, but it is a language-independent data format. Code for parsing and generating JSON data is readily available in many programming languages. JSON's website lists JSON libraries by language. While JSON provides a syntactic framework for data interchange, unambiguous data interchange also requires agreement between producer and consumer on the semantics of a specific use of the JSON syntax. While JSON is a data serialization format, it has seen ad hoc usage as a configuration language.

In this use case, support for comments and other features have been deemed useful, which has led to several nonstandard JSON supersets being created. Among them are HOCON and JSON5 which despite its name, isn't the fifth version of JSON.

Features of JSON:

- Standard Structure: JSON objects are having a standard structure that makes developers job easy to read and write code, because they know what to expect from JSON.
- Light weight**:** When working with AJAX, it is important to load the data quickly and asynchronously without requesting the page re-load. Since JSON is light weighted, it becomes easier to get and load the requested data quickly.
- Scalable**:** JSON is language independent, which means it can work well with most of the modern programming language. Let's say if we need to change the server-side language, in that case it would be easier for us to go ahead with that change as JSON structure is same for all the languages.
- Language independent**:** JSON language independent, it represents data in a way that speaks to common elements of many programming languages. With the way that data is represented, such as numbers and words, even the programming languages that aren't object oriented can find this format acceptable.

JSON is considered as a subset of JavaScript but that does not mean that JSON cannot be used with other languages. In fact, it works well with PHP, Perl, Python, Ruby, Java, Ajax and many more. It is also portable, unlike other text formats that require the installation of the DOM (Document Object Model) which is an API for HTML or XML or other platforms.

On the client side, JSON is particularly easy to use with JavaScript, and on the server-side PHP and Java use it through parseurs. On the client side, JSON is particularly easy to use with JavaScript, and on the server-side PHP and Java use it through parseurs. On the other hand, it is strictly forbidden to insert comments, and emphasized characters JSON may be used by direct invocation in the HTML page as a JavaScript file or it may be called by the XMLHttpRequest method; it may also be loaded via a JavaScript command.

The JSON format meets the needs of ease of use, performance and security of a contemporary format. It is best used in a thin client and server environment, and communicates well with the APIs of the digital universe. In XML, JSON, YAML, one creates file trees.

JSON requires double quotes to be used around strings and property names. Single quotes are not valid. Even a single misplaced comma or colon can cause a JSON file to go wrong, and not work. We should be careful to validate any data we are attempting to use although computer-generated JSON is less likely to include errors, as long as the generator program is working correctly. we can validate JSON using an application like JSONLint.JSON can actually take the form of any data type that is valid for inclusion inside JSON, not just arrays or objects. So, for example, a single string or number would be a valid JSON object. Unlike in JavaScript code in which object properties may be unquoted, in JSON only quoted strings may be used as properties.

Uses of JSON:

- It is used while writing JavaScript based applications that includes browser extensions and websites.
- JSON format is used for serializing and transmitting structured data over network connection.
- It is primarily used to transmit data between a server and web applications.
- Web services and APIs use JSON format to provide public data.
- It can be used with modern programming languages.

4.4 Rest API

Representational state transfer (REST) is a software architectural style that defines a set of constraints to be used for creating Web services. Web services that conform to the REST architectural style, called RESTful Web services, provide interoperability between computer systems on the Internet. RESTful Web services allow the requesting systems to access and manipulate textual representations of Web resources by using a uniform and predefined set of stateless operations.

Web resources were first defined on the World Wide Web as documents or files identified by their URLs. However, today they have a much more generic and abstract definition that encompasses everything, entity, or action that can be identified, named, addressed, handled, or performed, in any way whatsoever, on the Web. In a RESTful Web service, requests made to a resource's URI will elicit a response with a payload formatted in HTML, XML, JSON, or some other format. The response can confirm that some alteration has been made to the resource state, and the response can provide hypertext links to other related resources. By using a stateless protocol and standard operations, RESTful systems aim for fast performance, reliability, and the ability to grow by reusing components that can be managed and updated without affecting the system as a whole, even while it is running.

Architectural properties

- Simplicity of a uniform interface.
- Modifiability of components to meet changing needs.
- Visibility of communication between components by service agents.
- Portability of components by moving program code with the data.
- Reliability in the resistance to failure at the system level in the presence of failures within components, connectors, or data.

REST's client-server separation of concerns simplifies component implementation, reduces the complexity of connector semantics, improves the effectiveness of performance tuning, and increases the scalability of pure server components. Layered system constraints allow intermediaries proxies, gateways, and firewalls to be introduced at various points in the communication without changing the interfaces between components, thus allowing them to assist in communication translation or improve performance via large-scale, shared caching. REST enables intermediate processing by constraining messages to be self-descriptive: interaction is stateless between requests, standard methods and media types are used to indicate semantics and exchange information, and responses explicitly indicate cacheability.

Architectural constrains

Six guiding constraints define a RESTful system. These constraints restrict the ways that the server can process and respond to client requests so that, by operating within these

constraints, the system gains desirable non-functional properties, such as performance, scalability, simplicity, modifiability, visibility, portability, and reliability. If a system violates any of the required constraints, it cannot be considered RESTful.

- Client-Server: The principle behind the client-server constraints is the separation of concerns. Separating the user interface concerns from the data storage concerns improves the portability of the user interfaces across multiple platforms. It also improves scalability by simplifying the server components. Perhaps most significant to the Web is that the separation allows the components to evolve independently, thus supporting the Internet-scale requirement of multiple organizational domains
- Stateless: REST is stateless: The client request should contain all the information necessary to respond to a request. In other words, it should be possible to make two or more HTTP requests in any order and the same responses will be received.
- Cacheable: A response should be defined as cacheable or not.
- Layered: The requesting client need not know whether it's communicating with the actual server, a proxy, or any other intermediary.

Working of Rest API

A RESTful API breaks down a transaction to create a series of small modules. Each module addresses a particular underlying part of the transaction. This modularity provides developers with a lot of flexibility, but it can be challenging for developers to design their REST API from scratch. They use GET to retrieve a resource. PUT to change the state of or update a resource, which can be an object, file or block. POST to create that resource and DELETE to remove it.With REST, networked components are a resource the user requests access to a black box whose implementation details are unclear. All calls are stateless; nothing can be retained by the RESTful service between executions.

4.5Amazon web services (S3)

Amazon Simple Storage Service is a scalable, high-speed, web-based cloud storage service designed for online backup and archiving of data and applications on Amazon Web Services. Amazon S3 was designed with a minimal feature set and created to make web-scale computing easier for developers. Amazon S3 is an object storage service, which differs from block and file cloud storage. Each object is stored as a file with its metadata included and is given an ID number. Applications use this ID number to access an object. Unlike file and block cloud storage, a developer can access an object via a REST API.

The S3 cloud storage service gives a subscriber access to the same systems that Amazon uses to run its own websites. S3 enables customers to upload, store and download practically any file or object that is up to five terabytes in size, with the largest single upload capped at five gigabytes .Data can be transferred to S3 over the public internet via access to S3 APIs. There's also Amazon S3 Transfer Acceleration for faster movement over long distances, as well as AWS Direct Connect for a private, consistent connection between S3 and an enterprise's own data center.

An administrator can also use AWS Snowball, a physical transfer device, to ship large amounts of data from an enterprise data center directly to AWS, which will then upload it to S3.In addition, users can integrate other AWS services with S3. For example, an analyst can query data directly on S3 either with Amazon Athena for ad hoc queries or with Amazon Redshift Spectrum for more complex analyses. Amazon S3 Access Points simplifies managing data access at scale for applications using shared data sets on S3. With S3 Access Points, you can now easily create hundreds of access points per bucket, representing a new way of provisioning access to shared data sets.

Access Points provide a customized path into a bucket, with a unique hostname and access policy that enforces the specific permissions and network controls for any request made through the access point. Amazon S3 offers four different storage classes that offer different levels of durability, availability, and performance requirements.

Amazon S3 Standard is the default class. Amazon S3 Standard Infrequent Access is designed for less frequently accessed data. Typical use cases are backup and disaster recovery solutions. Amazon S3 One Zone-Infrequent Access is designed for data that is not often needed but when required, needs to be accessed rapidly. Data is stored in one zone and if that zone is destroyed, all data is lost. Amazon is designed for long-term storage of data that is infrequently accessed and where retrieval latency of minutes or hours is acceptable.

AWS provides a portfolio of data transfer services to provide the right solution for any data migration project. The level of connectivity is a major factor in data migration, and AWS has offerings that can address your hybrid cloud storage, online data transfer, and offline data transfer needs.

- Hybrid cloud storage: AWS Storage Gateway is a hybrid cloud storage service that lets us seamlessly connect and extend our on-premises applications to AWS Storage. Customers use Storage Gateway to seamlessly replace tape libraries with cloud storage, provide cloud storage-backed file shares, or create a low-latency cache to access data in AWS for on-premises applications.

- Offline data transfer: The AWS Snow Family is purpose-built for use in edge locations where network capacity is constrained or nonexistent and provides storage and computing capabilities in harsh environments. The AWS Snowball service uses ruggedized, portable storage and edge computing devices for data collection, processing, and migration. Customers can ship the physical Snowball device for offline data migration to AWS. AWS Snowmobile is an exabyte-scale data transfer service used to move massive volumes of data to the cloud, including video libraries, image repositories, or even a complete data center migration.

- Online data transfer: AWS DataSync makes it easy and efficient to transfer hundreds of terabytes and millions of files into Amazon S3, up to 10 times faster than open-source tools. DataSync automatically handles or eliminates many manual tasks, including scripting copy jobs, scheduling and monitoring transfers, validating data, and optimizing network utilization.

Amazon S3 Transfer Acceleration enables fast transfers of files over long distances between your client and your Amazon S3 bucket.AWS offers pay-as-you-go approach for pricing for over 160 cloud services. With AWS we pay only for the individual services we need, for as long as we use them, and without requiring long-term contracts or complex licensing.

4.6 Amazon EC2

Amazon Elastic Compute Clouds a part of amazon.com's cloud-computing platform, Amazon Web Services , that allows users to rent virtual computers on which to run their own computer applications. EC2 encourages scalable deployment of applications by providing a web service through which a user can boot an Amazon Machine Image to configure a virtual machine, which Amazon calls an "instance", containing any software desired.

A user can create, launch, and terminate server-instances as needed, paying by the second for active servers hence the term elastic. EC2 provides users with control over the geographical location of instances that allows for latency optimization and high levels of redundancy.

Amazon EC2 provides the following features:

- Virtual computing environments, known as instances
- Preconfigured templates for our instances, known as Amazon Machine Images (AMIs), that package the bits we need our server including the operating system and additional software
- Various configurations of CPU, memory, storage, and networking capacity for our instances, known as instance types
- Secure login information for our instances using key pairs ,AWS stores the public key, and our store the private key in a secure place
- Storage volumes for temporary data that's deleted when we stop or terminate our instance, known as instance store volumes
- Persistent storage volumes for our data using Amazon Elastic Block Store Amazon EBS, known as Amazon EBS volumes
- Multiple physical locations for our resources, such as instances and Amazon EBS volumes, known as Regions and Availability Zones

- A firewall that enables us to specify the protocols, ports, and source IP ranges that can reach our instances using security groups
- Static IPv4 addresses for dynamic cloud computing, known as Elastic IP addresses
- Metadata, known as tags, that you can create and assign to your Amazon EC2 resources
- Virtual networks we can create that are logically isolated from the rest of the AWS cloud, and that we can optionally connect to our own network, known as virtual private clouds (VPCs).

4.7 Amazon webservices cloud9

AWS Cloud9 is a cloud-based integrated development environment (IDE) that lets us write, run, and debug our code with just a browser. It includes a code editor, debugger, and terminal. Cloud9 comes prepackaged with essential tools for popular programming languages, including JavaScript, Python, PHP, and more, so we don't need to install files or configure our development machine to start new projects. Since our Cloud9 IDE is cloud-based, we can work on our projects from our office, home, or anywhere using an internet-connected machine. Cloud9 also provides a seamless experience for developing serverless applications enabling us to easily define resources, debug, and switch between local and remote execution of serverless applications. With Cloud9, we can quickly share our development environment with our team, enabling us to pair program and track each other's inputs in real time.

AWS Cloud9 gives us the flexibility to run our development environment on a managed Amazon EC2 instance or any existing Linux server that supports SSH. This means that we can write, run, and debug applications with just a browser, without needing to install or maintain a local IDE. The Cloud9 code editor and integrated debugger include helpful, time-saving features such as code hinting, code completion, and step-through debugging. The Cloud9 terminal provides a browser-based shell experience enabling us to install additional software, do a git push, or enter commands.

AWS Cloud9 makes collaborating on code easy. we can share our development environment with our team in just a few clicks and pair program together. While collaborating, our team members can see each other type in real time, and instantly chat with one another from within the IDE. AWS Cloud9 comes with a terminal that includes sudo privileges to the

managed Amazon EC2 instance that is hosting your development environment and a preauthenticated AWS Command Line Interface. This makes it easy for us to quickly run commands and directly access AWS services. AWS Cloud9 makes it easy for us to start new projects. Cloud9's development environment comes prepackaged with tooling for over 40 programming languages, including Node.js, JavaScript, Python, PHP, Ruby, Go, and C++. This enables us to start writing code for popular application stacks within minutes by eliminating the need to install or configure files, SDKs, and plug-ins for our development machine. Because Cloud9 is cloud-based, we can easily maintain multiple development environments to isolate our project's resources.

4.8 MySQL

MySQL is an open-source relational database management system. MySQL is free and open-source software under the terms of the GNU General Public License, and is also available under a variety of proprietary licenses.

MySQL can be built and installed manually from source code, but it is more commonly installed from a binary package unless special customizations are required. On most Linux distributions, the package management system can download and install MySQL with minimal effort, though further configuration is often required to adjust security and optimization settings.

Though MySQL began as a low-end alternative to more powerful proprietary databases, it has gradually evolved to support higher-scale needs as well. It is still most commonly used in small to medium scale single-server deployments, either as a component in a LAMP-based web application or as a standalone database server. Much of MySQL's appeal originates in its relative simplicity and ease of use, which is enabled by an ecosystem of open source tools such as phpMyAdmin. In the medium range, MySQL can be scaled by deploying it on more powerful hardware, such as a multi-processor server with gigabytes of memory. There are, however, limits to how far performance can scale on a single server 'scaling up', so on larger scales, multi-server MySQL 'scaling out' deployments are required to provide improved performance and reliability. A typical high-end configuration can include a powerful master

database which handles data write operations and is replicated to multiple slaves that handle all read operations. The master server continually pushes binlog events to connected slaves so in the event of failure a slave can be promoted to become the new master, minimizing downtime. Further improvements in performance can be achieved by caching the results from database queries in memory using memcached, or breaking down a database into smaller chunks called shards which can be spread across a number of distributed server clusters.MySQL Workbench is the official integrated environment for MySQL. It was developed by MySQL AB, and enables users to graphically administer MySQL databases and visually design database structures. MySQL Workbench replaces the previous package of software.

MySQL GUI Tools. Similar to other third-party packages, but still considered the authoritative MySQL front end, MySQL Workbench lets users manage database design & modelling, SQL development and Database administration.

4.9 RASA Architecture

Rasa is an open source machine learning framework for building AI assistants and chatbots. In Rasa Action Serve we need to write code in Python, that mainly used to trigger External actions like Calling Google API or REST API etc.

Rasa has two main modules:

- Rasa NLU for understanding user messages
- Rasa Core for holding conversations and deciding what to do next

- Rasa NLU: This is the place, where rasa tries to understand User messages to detect Intent and Entity in our message. Rasa NLU has different components for recognizing intents and entities, most of which have some additional dependencies. First, we need to build the NLU model. Rasa NLU works using supervised learning model. Therefore, you need training data to train the NLU model. In the training data, we need to specify what is the intent and entity for that data. **Actions** as the name suggest, its an operation which can be performed by the bot. It could be replying something in return, querying a database or any other thing possible by code. **Stories** These are a sample

interaction between the user and bot, defined in terms of intents captured and actions performed. So developer can mention what to do if you get a use input of some intent with/without some entities. Like saying if user intent is to find the day of week and entity is today, find day of week of today and reply

1. Spacy (We need to install it separately)

2. Tensorflow (By Default available with Rasa)

- Rasa Core: This is the place, where Rasa try to help us with contextual message flow. Based on User message, it can predict dialogue as a reply and can trigger Rasa Action Server.Rasa internally uses Tensorflow, whenever you do "pip install rasa".The input message is interpreted by an Interpreter to extract intent and entity. It is then passed to the Tracker that keeps track of the current state of the conversation. The Policy applies Machine Learning algorithm to determine what should be the reply and choses Action accordingly. Action updates the Tracker to reflect the current state.The most significant advantages of Rasa are its sophisticated NLU engine and the open source community that helps power it. If we are poised to build a chatbot capable of processing complex and highly subtle user utterances, whether via text or voice, Rasa is a smart framework to consider.

  Other advantages include the option to build and run our bot on-premises, which could be a deal-breaker for heavily regulated industries. Since Rasa is open source, we also get to see and work with all of the underlying machine learning components. Rasa NLU is probably the framework's most celebrated component. It goes beyond rule-based training and delivers true machine learning that can improve your bot's performance over time. Rasa NLU is written in Python, but we can use it even if Python isn't our go-to language. For example, we can use Rasa NLU via a third-party platform, like Articulate.

5 DESIGN AND IMPLEMENTATION

5.1 Database Design

The below figure 5.1 shows the high-level architectural design of the database. The database we have used is Mysql. MySQL is an open-source relational database management system. Its name is a combination of "My", the name of co-founder Michael Widenius's daughter, and "SQL", the abbreviation for Structured Query Language. We have hosted this on the AWS EC2 Ubuntu instance as the Database server.

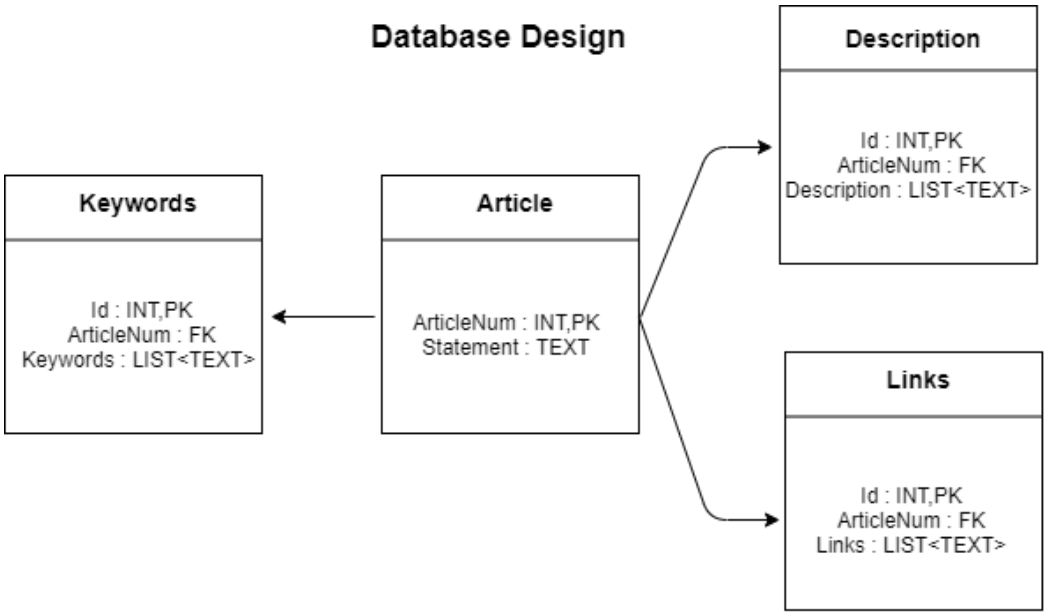

FIGURE 5.1 Database Design

Articles: The main contents of the model are articles, statement descriptions, and links. For this, we have created the main table Article with Article number as primary key and Statement as Text datatype. Which acts as foreign key relationships with all other tables.

Description: The description table consists of all descriptions related to Articles. Which has Id, ArticleNum, Description as members. Id is of type int, ArticleNum has foreign key references with Articles, and Description contains the list of Text datatype.

Links: The Links table consists of all Links related to Articles. Which has Id, ArticleNum, Links as members. Id is of type int, ArticleNum has foreign key references with Articles, and links contain the list of Text datatype.

Keywords: The Keywords table consists of all keywords used for training data related to Articles. Which has Id, ArticleNum, keyword as members. Id is of type int, ArticleNum has foreign key references with Articles, and Keywords contains the list of varchar datatype.

5.2 ML Model

Rasa is an open-source Conversational AI framework. What I like about Rasa is you are not tied to a pre-built model or use-case (Dialogflow etc.). So you can customize it to your use-case which can be a market differentiator. Rasa is not a rule-based framework (e.g. Botkit) and you don't need to worry about putting your data in someone else's cloud as in Dialogflow, Microsoft LUIS, or Amazon Lex.

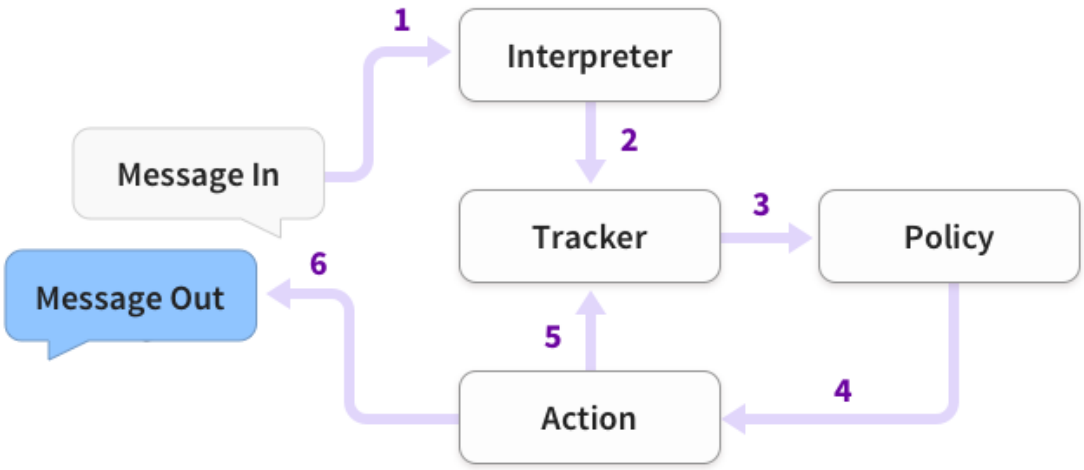

FIGURE 5.2: ML Model

NLU is Natural Language Understanding. Suppose the user says "I want to order a book". NLU's job is to take this input, understand the intent of the user, and find the entities in the input. For example, in the above sentence, the intent is ordering and the entity is a book.

Rasa NLU internally uses Bag-of-Word (BoW) algorithm to find intent and Conditional Random Field (CRF) to find entities. Although you can use other algorithms for finding intent and entities using Rasa. You have to create a custom pipeline to do that. Details about the custom pipeline are out of the scope of this post. If you are interested, check out this link. The job of Rasa Core is to essentially generate the reply message for the chatbot. It takes the output of Rasa NLU (intent and entities) and applies Machine Learning models to generate a reply. We will discuss more about the possible Machine Learning models that you can use later during the API discussion.

The steps are:

1. The message is received and passed to an Interpreter, which converts it into a dictionary including the original text, the intent, and any entities that were found. This part is handled by NLU.
2. The Tracker is the object which keeps track of the conversation state.
3. It receives the info that a new message has come in.
4. The policy receives the current state of the tracker.
5. The policy chooses which action to take next.
6. The chosen action is logged by the tracker. A response is sent to the user.

5.3 Implementation

Django is a Python-based free and open-source web framework, which follows the model-template-view (MTV) architectural pattern. It is maintained by the Django Software Foundation (DSF), an independent organization established as a non-profit.

Django's primary goal is to ease the creation of complex, database-driven websites. The framework emphasizes reusability and "pluggability" of components, less code, low coupling, rapid development, and the principle of don't repeat yourself. Python is used throughout, even for settings files and data models. Django also provides an optional administrative create, read, update and delete interface that is generated dynamically through introspection and configured via admin models.Some well-known sites that use Django include the Public Broadcasting Service, Instagram, Mozilla, The Washington Times.

5.31 The Model Layer

Django provides an abstraction layer (the "models") for structuring and manipulating the data of your Web application.A model is the single, definitive source of information about your data. It contains the essential fields and behaviors of the data you're storing. Generally, each model maps to a single database table.

The basics:

- Each model is a Python class that subclasses django.db.models.Model.
- Each attribute of the model represents a database field.
- With all of this, Django gives you an automatically-generated database-access API; see Making queries.

Once you have defined your models, you need to tell Django you're going to *use* those models. Do this by editing your settings file and changing the INSTALLED_APPS setting to add the name of the module that contains your models.py. The most important part of a model – and the only required part of a model – is the list of database fields it defines. Fields are specified by class attributes. Be careful not to choose field names that conflict with the models API like clean, save, or delete. Each field takes a certain set of field-specific arguments

5.32 The view Layer:

Django has the concept of "views" to encapsulate the logic responsible for processing a user's request and for returning the response. To design URLs for an app, you create a Python module informally called a URLconf (URL configuration). This module is pure Python code and is a mapping between URL path expressions to Python functions (your views).

This mapping can be as short or as long as needed. It can reference other mappings. And, because it's pure Python code, it can be constructed dynamically.Django also provides a way to translate URLs according to the active language.

5.33 The Template Layer:

The template layer provides a designer-friendly syntax for rendering the information to be presented to the user. Being a web framework, Django needs a convenient way to generate HTML dynamically. The most common approach relies on templates. A template contains the static parts of the desired HTML output as well as some special syntax describing how dynamic content will be inserted. For a hands-on example of creating HTML pages with templates.

A Django project can be configured with one or several template engines (or even zero if you don't use templates). Django ships built-in backends for its own template system, creatively called the Django template language (DTL), and for the popular alternative Jinja2. Backends for other template languages may be available from third-parties.

Django defines a standard API for loading and rendering templates regardless of the backend. Loading consists of finding the template for a given identifier and preprocessing it, usually compiling it to an in-memory representation. Rendering means interpolating the template with context data and returning the resulting string.

5.34 Forms

Django provides a rich framework to facilitate the creation of forms and the manipulation of form data. Handling forms is a complex business. Consider Django's admin, where numerous items of data of several different types may need to be prepared for display in a form, rendered as HTML, edited using a convenient interface, returned to the server, validated and cleaned up, and then saved or passed on for further processing.Django's form functionality can simplify and automate vast portions of this work, and can also do it more securely than most programmers would be able to do in code they wrote themselves.

Django handles three distinct parts of the work involved in forms:

- Preparing and restructuring data to make it ready for rendering
- Creating HTML forms for the data
- Receiving and processing submitted forms and data from the client.It is possible to write code that does all of this manually, but Django can take care of it all for you.

5.4 Deployment

Amazon EC2 provides resizable compute capacity in the cloud. It is designed to make web-scale cloud computing easier for developers and allows maximum scalability and availability for websites and web applications.

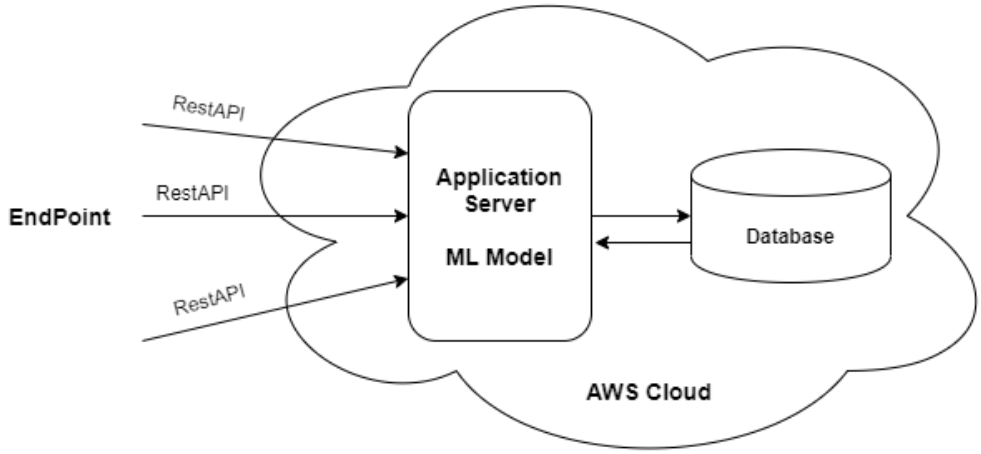

FIGURE 5.3 ML Model Deployment

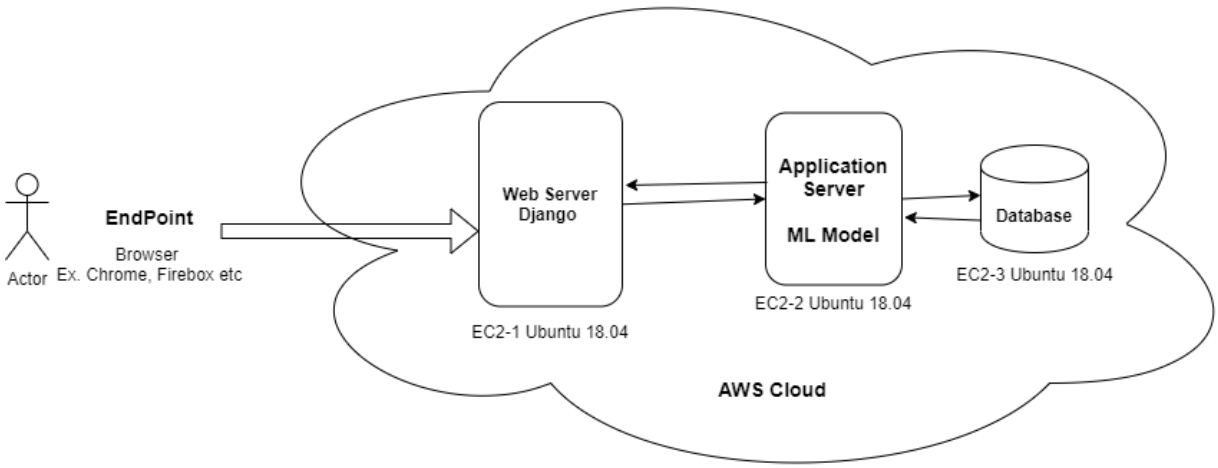

FIGURE 5.4 Web App Deployment

Why use AWS for web hosting?

- Why use AWS for web hosting?
- Datacenters worldwide
- Scalable from day one
- Flexible pricing models

Simple Website Hosting

Simple websites typically consist of a single web server that runs either a Content Management System (CMS), such as WordPress, an eCommerce application, such as Magento, or a development stack, like LAMP. The software makes it easy to build, update, manage, and serve the content of your website.

Simple websites are best for low to medium trafficked sites with multiple authors and more frequent content changes, such as marketing websites, content websites, or blogs. They provide a simple starting point for website which might grow in the future. While typically low cost, these sites require IT administration of the web server and are not built to be highly available or scalable beyond a few servers.

5.5 ML Model Deployment

In this, we have discussed how to use Amazon SageMaker to build, train, and deploy a machine learning (ML) model. We will use the popular XGBoost ML algorithm for this exercise. Amazon SageMaker is a modular, fully managed machine learning service that enables developers and data scientists to build, train, and deploy ML models at scale.

Taking ML models from conceptualization to production is typically complex and time-consuming. You have to manage large amounts of data to train the model, choose the best algorithm for training it, manage the compute capacity while training it and then deploy the model into a production environment. Amazon SageMaker reduces this complexity by making it much easier to build and deploy ML models. After you choose the right algorithms and frameworks from the wide range of choices available, it manages all of the underlying infrastructures to train your model at the petabyte scale, and deploy it to production.

Create a notebook instance

- Prepare the data
- Train the model to learn from the data
- Deploy the model
- Evaluate your ML model's performance

The resources created and used in this tutorial are AWS free tier eligible. Remember to complete Step 7 and terminate your resources. If your account has been active with these resources for longer than two months, your account will be charged less than $0.50.

5.6 High-Level Architecture of the Whole Model

The Below figure shows how the app looks at a high level. Which is based on Client-Server Architecture. It consists of mainly two parts frontend design, backend design. Backend is further classified into Web Server, Application Server, Database Server.

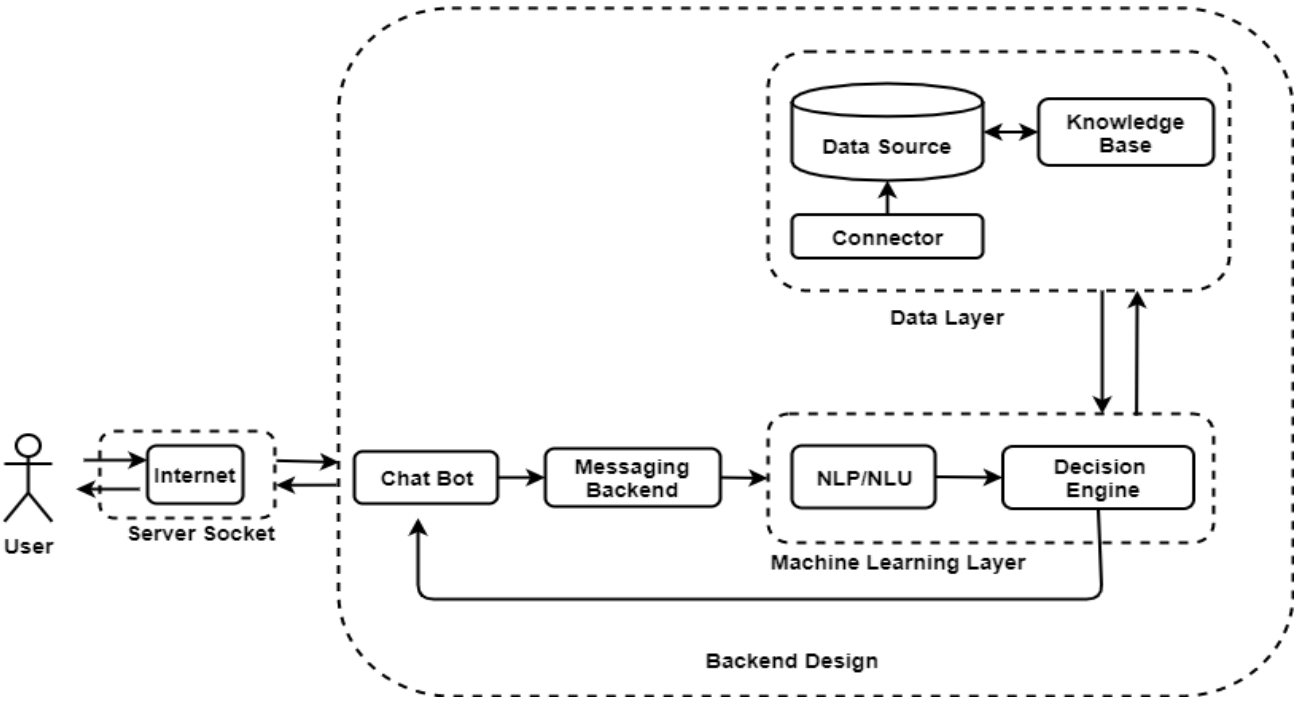

FIGURE 5.5 :Web app architecture

5.7 Security:

Security is a topic of paramount importance in the development of Web applications and Django provides multiple protection tools and mechanisms: Cross site scripting (XSS) protection, Cross site request forgery (CSRF) protection, SQL injection protection, Clickjacking protection are some of them offered by Django.

5.8 Brief explanation of files and their functionality in our project

- nlu.md : It contains all the intents and its respective keyword. Hence if after parsing the request the bot gets any keyword it triggers its respective intent or in our case Article in this file.

- Stories.md : It contains the articles PATH, i.e it contains the list of actions that are function names. Hence this file has the details of which function to call when a particular article is triggered.

- init.py : It is an virtual environment file which is a default blank file but essential for the program to run.There are two of these one for backend and one for frontend.

- config.py : This is also a virtual environment file which has the details of the features used in training the bot for example the languages the bot can understand which in our case is English.

- credential.yml : It is a default file which is used when we want to integrate our bot with other bots for example facebook bot. We are currently not using it.

- domain.yml : This is the file where all the actions or methods are listed and defined for the articles to be executed.

- endpoint.yml : This file contains the URL for Backend call.

- actions.py : This file gets the article data from the outer source. It contains Action Get Article Data method which has tracker information i.e data sent by the user and also the dispatcher or response information which it has fetched from the datastore.

- Models : As explained before all the training data will be here. This makes for the internal datastore.

- settings.py : It is a chat Bot file which s an important config file where we interface CORS with our bot program

- wsgy.py : I is a web server gateway interface. It handles request, response etc

- url.py : There are two of these too. One in chatBot which contains main server hit ad the other file contains url for upload data for training, getdata etc

- models.py : models.py in Basic App directory have an Article table with its number with its column equals to articles name and Statement. There are four tables in our case namely statement , description, resource and keyword with number. When an article is deleted it makes sure that the article is from all four tables.

- views.py : It is one of the most important file which consists of triggering info in URL. It gets the front end requests and returns in HTML format because we are doing web development and if it was a REST API it would return in Jason format.
- migration : This directory contains roll back files of database. Hence if we mess up somehow with the database we can roll back to previous states present in this directory.
- home.html : It is an html file which define the front end UI working. We have kept an online template as basis and have built extra features as needed. This gets the user query and makes appropriate backend calls.

Hence when the user enters the query, the home.html collects I and makes a call to backend url which is in endpoints.yml. From which the query is parsed according to config file and a particular keyword triggers an intent in nlu.md. Following which it getsthe path of triggered action from Stories.md and gets the action defination from domain.yml. The program gets the respective articles from the database with the help of models, url.py etc and the response is sent back to home.html.

5.9 RESULTS

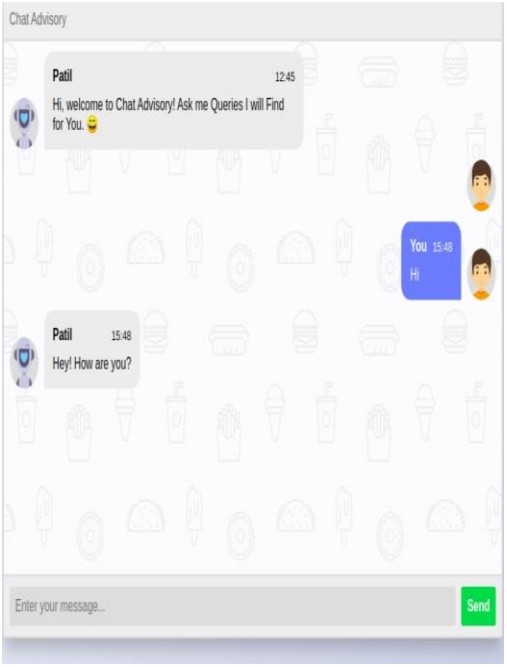

FIGURE 5.6: Front end screen

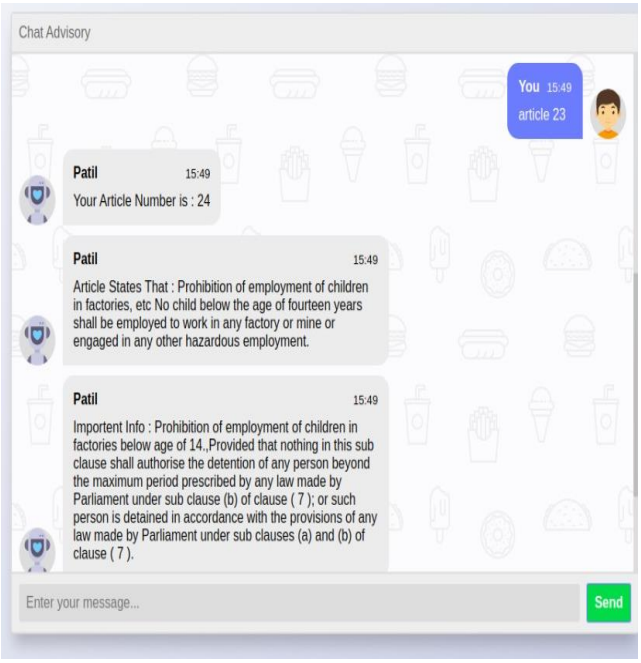

FIGURE 5.7: Chatbot reply

To demonstrate our approach we have created a website and the website can be used through the text.The natural language is analysed by the chatbot and the proper reply has been sent to the user.Here in the text based chatting we have to initiate the process and ask for the bot a information .The bot will work based on the NLU processing and send the desired reply to the use

6.CONCLUSIONS AND FUTURE ENHANCEMENTS

The Legal Chat Bot web application was successfully implemented using the technology stack as explained in the above sections. The user can chat with the chatbot if he wishes for some time and can start asking his legal related issues. The chatbot picks up important phrases from the text and is made sure that the query of the user is tagged to the related article and providing him with a brief description about the article, previous related similar cases and also important links pertaining to that article. With the help of previous cases, the user can link his issue with the one already happened and can take necessary actions required to his problem. The machine learning algorithm in the backend makes sure that the articles are properly mapped to the user queries.

This web application is made blazingly fast so that the user will be able to get his results in fraction of seconds. With the help of this speed, the user will need not waste his precious time in querying his problem in google and search multiple websites for the solution. There will be also no need in searching for lawyers and spending huge amount of money to figure out a solution for his problem. The web app is also made interactive to the user, so that the user feels comfortable and connected in chatting with the application.

6.1 ADVANTAGES

- Our chatbot is made interactive and very user friendly so that any type of person having any level of knowledge will be able to comfortably use the application to get response to any of his queries.
- User need not waste lot of his time in researching about his issue and looking up into multiple links.
- Since the website is hosted in AWS EC2 instance, it can be accessed from anywhere using just the IP address.
- The website can be accessed by anyone without any legal restrictions.

- The users need not spend their huge amount of money in going to a lawyer and asking for a solution, whereas same amount of information is provided in our website.
- The users can learn from previous cases which are related to their case and can take necessary actions in tackling their issue.
- Our website helps in providing concise information related to the users and in a way which is easily understood by them.
- The response time of the application is very quick and smooth.

6.2 DRAWBACKS

- The current implemented app is limited to a few articles which can be later expanded to a bigger domain.
- This application is country specific, where the implementation of this app in different countries requires adhering to their constitution rules and regulations.
- The chatbot will not be able to answer questions other than legal case issues.
- If the entered user text is either not with a clear explanation or involves with lot of jargon, the issue may not be mapped to the actual article sometimes.

6.3 APPLICATIONS

- Useful when most queries/requests are common but varied in querying or fetching of results.
- Finding a human customer service assistant.
- Resolving complaint or problems.
- Specific content delivery.
- Automated customer support for similar queries.
- Restaurant applications, Flight and Bus ticket booking, Content delivery, Market and Research analysis, Health care, E-Commerce and Legal information.

6.4 FUTURE ENHANCEMENTS

The Legal Chat Bot currently developed already surpasses some of the currently existing applications. But, a lot of improvements can be done with additional features to this existing web application chatbot.

- The currently developed website can be made as an Android or IOS app so that it will be more accessible and convenient for the user to interact.
- The mobile app can be made such that, the user can interact with chatbot to get his basic queries cleared (like getting to know the article number or general interactions etc.) without connecting to the internet.
- The machine learning model can be further developed using the feedback from users so that the model becomes much more accurate in predicting.
- Multiple access level can be given to the users so that the common people will not be having access to sensitive information which only lawyers can see.
- The application can be made much more intelligent where the bot can even provide some suggestion to the users using AI techniques.
- Can create a community where the people with similar issues can have a conversation with each other to resolve their legal case issues much faster.
- Our application can be tied with some professional lawyers so that the lawyers can quickly take up the case in case of urgency from the user.

7 REFERNCES

[1] Noortwijk, K. van, "Integrated legal information retrieval: new developments and educational challenges", in European Journal of Law and Technology, Vol 8, No 1, 2017.

[2] Rachayita Giri, Yosha Porwaly, Vaibhavi Shuklay, Palak Chadhay and Rishabh Kaushal "Approaches for Information Retrieval in Legal Documents" Proceedings of 2017 Tenth International Conference on Contemporary Computing (IC3), 10-12 August 2017, Noida, India

[3] https://donotpay.com

[4] https://robotlawyerlisa.com

[5] https://rossintelligence.com

[6] http://www.billybot.co.uk

[7] https://autom.io/about

[8] Vagelis Hristidis, " Chatbot Technologies and Challenges" 2018 First International Conference on Artificial Intelligence for Industries

[9] Mubashra Akhtar, Julia Neidhardt , Hannes Werthner, "The Potential of Chatbot: Analysis of Chatbot conversations" 2019 IEEE 21st Conference on Business Informatics (CBI)

[10] Alan Shaw, Ph.D, "A System of Simple Sentence Parsing Rules to Produce "Answer Matching" Chatbot." 2014 11th International Conference on Information Technology: New Generations

[11]Alessandro Bozzon, "Enterprise Crowd Computing for Human Aided Chatbots" 2018 ACM/IEEE 1st International Workshop on Software Engineering for Cognitive Services

[12] Jitendra Purohit, Aditya Bagwe, Rishabh Mehta, Ojaswini Mangaonkar, Elizabeth George, "Natural Language Processing based Jaro-The Interviewing Chatbot" Proceedings of the Third International Conference on Computing Methodologies and Communication (ICCMC 2019) IEEE Xplore Part Number: CFP19K25-ART; ISBN: 978-1-5386-7808-4

[13] https://rasa.com/docs/getting-started/

[14] https://itnext.io/building-a-chatbot-with-rasa-9c3f3c6ad64d

[15] https://www.djangoproject.com/start/

[16] https://restfulapi.net/

[17] https://en.wikipedia.org/wiki/WebSocket

[18] https://en.wikipedia.org/wiki/Chatbot

[19] https://en.wikipedia.org/wiki/Machine_learning

[20] https://en.wikipedia.org/wiki/Natural_language_processing

[21] https://en.wikipedia.org/wiki/Information_retrieval

[22] www.tutorialspoint.com/natural_language_processing/natural_language_processing_information_retrieval.htm

[23] https://en.wikipedia.org/wiki/Web_crawler

[24] https://en.wikipedia.org/wiki/Database