

Ava, Question-Answering Bot

Candidate no.: XXXX

Supervisor: XXXX

Statement of Originality

This report is submitted as part requirement for the degree of BSc (Hons) Computing for Digital Media at the University of Sussex. It is the product of my own labour except where indicated in the text. The report may be freely copied and distributed provided the source is acknowledged.

.....

Summary of Report

This report discusses the use of natural language processing techniques for the production of Ava, a question-answering bot that can answer simple natural language questions pertaining to information contained within the Spotify music library database.

Table of Terminologies

Abbreviation	Full form	Page no. of first mention
NLP / NLE	Natural Language Processing / Engineering	8
NERD	Named Entity Recognition and Disambiguation	8
NLTK	Natural Language Toolkit	10
POS tagging	Part-of-speech Tagging	14
RegEx	Regular Expression	18
Fast NP Extractor	Fast Noun Phrase Extractor	20
gTTS	Google Text-to-Speech	27
GUI	Graphical User Interface	33

List of Figures

Figure 1: System Pipeline	15
Figure 2: Kaggle Datasets	17
Figure 3: Question Analysis pipeline.....	17
Figure 4: Sample input - original	18
Figure 5: Sample input - "cleaned"	19
Figure 6: Sample input - intent classified.....	20
Figure 7: Sample input - NER performed	21
Figure 8: Sample input - Entities linked	23
Figure 9: Hypothesis Generation pipeline	23
Figure 10: Sample input - final output.....	24
Figure 11: System Flow Diagram	30
Figure 12: User Feedback - Q1	31
Figure 13: User Feedback - Q2	32
Figure 14: User Feedback - Q4	33
Figure 15: User Feedback - Q5	33
Figure 16: User Feedback - Q3	34
Figure 17: User Feedback - Voice Mode - Q1.....	35
Figure 18: User Feedback - Voice Mode - Q3.....	35
Figure 19: User Feedback - Voice Mode - Q5.....	36
Figure 20: User Feedback - Song Playback - Q1.....	37
Figure 21: User Feedback - Song Playback - Q3.....	37
Figure 22: User Feedback - Overall System - Q1	38
Figure 23: User Feedback - Overall System - Q2	38

Contents

1. Introduction.....	8
1.1 Project Aim and Objectives	8
1.2 Motivation for Undertaking this Project	8
1.3 Relevance to Degree	9
1.4 Outline of Report Structure.....	9
2. Professional and Ethical Considerations	9
3. Background and Related Work	10
4. Requirements Analysis and Specification.....	12
4.1 Needs of Intended Users	12
4.2 Ideal System in line with Needs of Target Users.....	12
4.3 Problem Area	13
4.4 System Requirements.....	13
5. System Design and Development Plan	13
5.1 The Method and Design Choices	13
5.2 An Overview of the System	15
6. Implementation Process	15
6.1 Content Acquisition	15
6.2 Question Analysis	17
6.2.1 Text Pre-processing	18
6.2.2 Intent Classification	19
6.2.3 Named Entity Recognition (Extraction of target ‘named entities’ and target ‘other meaningful info’)	20
6.2.4 Entity Linking.....	21
6.3 Answer Hypothesis Generation	23
6.3.1 Search Query and Response Generation.....	23
6.3.2 Improvements to the Existing Functionality of the Spotify Service	25
6.3.3 Handling of Ambiguities and Limitations of Solutions	26
6.4 Extensions to the System	27
6.5 Experimentation with Paraphrase Detection.....	28
7. System Testing.....	29
8. Underlying Principles of the Program	29
9. System Evaluation	31
9.1 Data Analysis of User Testing	31
9.2 Resulting Software Improvements.....	39

10. Conclusion.....	40
References	41
Appendices	44
1. Interim Log.....	44
2. Proposal Document	45
3. Ethical Compliance Form.....	46
4. Sample Questions for potential user input	49
5. Sample Questions categorised by intent.....	50
6. Evaluation of TextBlob extractors	51
7. Comparison of Naïve Bayes and Semantic Similarity	54
8. User Evaluation Questionnaire	56

1. Introduction

1.1 Project Aim and Objectives

This project aimed to develop a question-answering bot capable of answering simple natural language questions pertaining to the contents of a database. The database being that of the Spotify music library, which has been made available to developers through the Spotify Web API (1).

A primary objective of this project was to utilise named entity recognition and linking / disambiguation to identify relevant parts of a natural language text-based input in order to retrieve accurate data as a response. For example, when asked *“Who sings Bodak Yellow?”* the bot would be able to identify *Bodak Yellow* as the name of a song, as well as understand *Who* as a request for artist name information, therefore successfully retrieving ‘Cardi B’ as the answer. Another primary objective was to involve paraphrase detection to further improve the bot’s ability to comprehend the natural language input and retrieve accurate responses, e.g. *“Bodak Yellow singer”* would be identified as having the same meaning / intent as *“Who sings Bodak Yellow?”*.

The potential extension for the project was to include an additional feature of the bot in which a user could ask *“What should I listen to?”* and receive a song recommendation as a response. The bot would first follow up this initial question with context clarification questions such as *“What activity are you currently doing?”* and *“What is your current mood?”* in order to gain better understanding of the context in which the user was requesting a song suggestion. Then, utilising NLP techniques such as NERD again, the bot would identify useful information in the user’s responses in order to search the database and decide on a reasonable suggestion to provide. The decision would be made by considering the user’s current context in relation to the audio features data which Spotify provides for the tracks contained within their database (e.g. tempo, energy, danceability, etc). The bot would also use identification of songs in relevant public playlists (e.g. ‘Cooking’ playlist) in attempt to better select a suggestion appropriate for the context of the user at that particular moment in time. Users would rate the accuracy of the suggestions provided anonymously during user testing.

Due to time limitations, the originally proposed extension was not implemented for this project. However, two different extensions which were able to be finalised within the allocated project time frame were implemented instead. The first of these was the addition of an extra mode by which users could interact with the bot: voice mode. The second was the option for playback of retrieved song suggestions in a Spotify Web Player or Spotify Desktop Player (for users with a Spotify Premium account only). These extensions are discussed in more detail later on in the report.

1.2 Motivation for Undertaking this Project

The undertaking of this project was inspired by an interest in natural language engineering, and the desire to further develop understanding and knowledge of the field. Whilst the complexities of NLE projects like building a chatbot are often not obvious from the onset, there are many challenges which NLP techniques such as paraphrase detection and NERD present.

In terms of named entity recognition and disambiguation for example, a challenge presented is ambiguity. Even within ambiguity, there are various types such as: lexical, semantic, syntactic, and metonymy. A user could ask the bot, “*Who sings Your Song?*”, which although refers to a song titled ‘*Your Song*’ by Elton John (and is therefore a proper noun), could be interpreted as a personal pronoun addressing the bot itself. This is an example of syntactic ambiguity. If in the case discussed later in Section 5.1 *The Method and Design Choices*, the bot was to simply return the pre-defined message “*I prefer not to talk about myself*” whenever it processed user input containing a personal pronoun assumed to be addressing itself (i.e. you, your), this would not produce the correct result. The question “*Who sings Your Song?*” would return the pre-defined message instead of the expected answer, ‘*Elton John*’.

However, this problem was somewhat constrained for this project as the Spotify music library database provided a list of candidate named entities in the form of artist names, album names, and song titles. These could be utilised by the bot in order to better classify the type of entity the user was likely referring to (entity linking / disambiguation) once other NLP techniques had been used to first recognise it as a named entity.

The challenge to tackle such natural language issues in the implementation of an effective chatbot was the motivation behind the undertaking of this project.

1.3 Relevance to Degree

This project was relevant to the degree course as it delved into an area which has become an increasingly growing topic of research and interest in the Computing sector. It was also informed by modules that have been undertaken during the course of the degree, such as Software Engineering - which tackled project management stages in detail, including requirements engineering and system modelling, as well as Human Computer Interaction - which taught the importance of user-focused design when creating interfaces and applications for user involvement / input.

1.4 Outline of Report Structure

Following a discussion on professional and ethical considerations, the report begins by providing a background into the research topic and common techniques used in the field. The discussion then details the key parts of the development of Ava, from the initial definition of system requirements, through to system evaluation, unpacking the natural language engineering techniques behind the implementation of the finished bot.

2. Professional and Ethical Considerations

Concerning professional and ethical issues, this project was developed taking into account the BCS Code of Conduct, particularly Sections 1b and 2d. As the project has been developed utilising data from within the Spotify music library, it has made sure to ‘*have due regard for the ‘legitimate rights of [Spotify]’*’, and to ‘*comply with Legislation in carrying out professional responsibilities*’ (2), i.e. through crediting the open source Spotify API which was used in the development of the bot. Other open source tools used in creating the bot have also been credited.

As the project required data to be gathered from human participants during the user testing stage, it required ethical approval from a supervisor. The project complied with all of the points listed in the Ethical Compliance Form and therefore no ethical review was needed. To name a few, the project adhered to points 3, 6 and 12 of the Ethical Compliance Form as '*all participants [were required to have] explicitly stated that they agreed to take part, and that their data could be used in the project*', '*no participant [was] under the age of 18*', and '*all the data collected from the participants [has been] stored securely, and in an anonymous form.*' A signed copy of the form can be seen in Appendix 3.

3. Background and Related Work

There are many chatbots currently existing in the public domain in various forms. These are often categorised into three most common types: support bots, skill bots, and assistant bots. A question-answering bot like Ava would likely be considered a support bot. A support bot is one which is built with mastery of a single domain, e.g. knowledge about a particular company. (3) However, many bots have been developed with functionalities which merge these categories, and also ones which expand beyond these.

An example of a recent project which produced a Q&A bot with ground-breaking natural language processing capabilities was that of IBM's 'DeepQA' project. The project begot the world-famous bot, Watson, who won first place against human competitors in the television quiz show *Jeopardy!* in 2011. A document published by the team who led the development of the bot, *Building Watson: An Overview of the DeepQA Project* (4), elaborated upon various aspects of the bot's creation, including techniques used.

The report outlined the following as the necessary steps that were required in producing Watson:

- Content Acquisition – identifying and gathering the content to use for the question and evidence sources. This meant manually analysing example questions to get an idea of the types of questions to be answered by the bot, and a characterisation of its domain. Whilst for Watson this consisted of encyclopaedias and literary works, for this project example questions were obtained from a small group of potential users of the bot. Another part of this process for Watson was the collection of resources, including databases such as WordNet (5). This project utilised the TextBlob Python library API (6) which provided an interface for using NLTK and Pattern tools, as well as integration for WordNet data, for use in processing textual input. Additionally, this project also used the Spotify Web API (1) which allowed access to information contained within the Spotify music library database. This was utilised through *Spotipy* (7) which is a Python library for the Spotify Web API.
- Question Analysis – understanding what the user's question is asking and determining how the question will be processed within the bot. At this stage the bot's system would carry out tasks such as parsing, named entity recognition, and relation detection / extraction. A technique which the DeepQA project also employed during this stage was 'Decomposition'. This used rule-based deep parsing (8) and statistical classifications methods to recognise whether a

question should be decomposed into subquestions, and to determine how best to break it down. (4) For this project, a dataset was created using the example questions gathered during the Content Acquisition stage. Each question was given an 'intent' type by which it, and alike questions, could be classified. TextBlob methods were used for the processing of textual data.

- Hypothesis Generation – using the results of question analysis in order to produce potential answers through searching the system's sources. For this process, the Watson system used a mixture of different approaches for searching for hypothetical answers and gathering evidence to prove each correct to varying degrees of confidence. Due to the difference in domain, the depth of this technique was not necessary for the scope of this project. This project attempted to provide a simpler solution for finding and returning appropriate information from the Spotify music database in response to a user's input question.
- Soft Filtering – a strategy for managing resource versus precision trade-off, which was also not required for this project as it did not utilise such a breadth of resources as the Watson bot.
- Hypothesis and Evidence Scoring – scoring algorithms are used to evaluate the evidence gathered from the sources to support each potential answer. The DeepQA project incorporated deep scoring analytics which was beyond the scope of this project, and not necessary to fulfil the requirements of question-answering for its intended domain. The next two stages for Watson follow on from this.
- Final Merging and Ranking – evaluation of all hypothetical answers and their scores to identify a single, most accurate answer.
- Answer Merging – merging of answer scores using algorithms for matching as well as other techniques. Identification of equivalent and related hypothetical answers is also carried out at this stage, and this enables custom merging per feature to combine scores. (4)
- Ranking and Confidence Estimation – ranking potential answers and estimating confidence in each based on their merged scores. For Watson, the DeepQA team adopted a machine-learning approach which involved running the system over a set of training data with known answers, and training a model based on the scores. (4) For this project, machine learning was not used, and instead the accuracy of the bot's responses was evaluated through a combination of unit testing and scoring by human participants during the user testing stage of the development process.

Research reports and case-studies on Q&A bots such as Watson, and the like, were beneficial in providing useful information and guidance during the implementation of natural language processing for the creation of Ava.

In addition, there was a breadth of resources and information available pertaining to natural language processing techniques and chatbot building practises. A paper which inspired ideas for a potential parsing style to implement was *Semantic Parsing via Paraphrasing* by Jonathan Berant and Percy Liang of Stanford University.

The paper discussed the semantic parsing problem of mapping natural language utterances into logical forms to be executed on a knowledge base, and presented an approach for semantic parsing based on paraphrasing. (9) Given an input utterance, this worked by using a deterministic procedure to construct a manageable set of candidate logical forms, then heuristically generating canonical utterances for each logical form based on the text descriptions of predicates from the knowledge base, and finally choosing the canonical utterance that best paraphrased the input utterance - and thereby the logical form that generated it. The technique uses an *association model* and a *vector space model*, and the system is trained on question-answer pairs only.

As detailed in Section 1.1 *Project Aim and Objectives*, a primary objective of this project was to include paraphrase detection to further improve the bot's ability to comprehend natural language input and retrieve accurate responses for the user. Although the technique discussed within the research paper was not used for the project, the method of implementing parsing through paraphrasing was an interesting approach to consider during the development of the bot.

4. Requirements Analysis and Specification

4.1 Needs of Intended Users

The bot was intended to be used by music listeners with at least basic technological competence by which they could interact with the bot effectively. The development of the bot primarily intended to fulfil the user's need for quick information retrieval. In the current climate of shorter, more fast-paced interactions and transactions as a result of technological innovations (e.g. self-checkout, online banking support bots, etc), many do not have the time or patience to actively seek out information whilst having to filter out redundant sources which come as a by-product of access to information in excess in things like search engine results. This project intended to provide a solution which would contribute effectively to the needs of such users. For example, a user could use the bot to easily and quickly find out information about an artist's latest single through the ease and comfort of natural language communication.

4.2 Ideal System in line with Needs of Target Users

These target users required the bot to be capable of quickly comprehending their input question pertaining to music information or a music request, and then quickly produce an appropriate response with a reasonable level of accuracy. The users also required a simple interface with which to interact with the bot. The ideal system would be able to not only process user input effectively to retrieve accurate information about music and artists contained within the Spotify music library, but would also be equipped to handle non-domain questions and comments (e.g. questions about itself, general conversation, etc) appropriately.

4.3 Problem Area

The use of the Spotify API for the development of the bot meant that all information retrieval was restricted to data contained within the Spotify music database, i.e. if a question is asked pertaining to a song, artist, etc. that was not contained within the Spotify music library, the bot would be unable to provide the correct information as a response. The solution for these cases were pre-defined messages which the bot would return as an answer if unable to retrieve relevant information from the Spotify database following an input question, e.g. *"Sorry, I am unable to find this information for you."*

In this respect, the matching methods used aimed to favour high recall and low precision in attempt to find suitable data to return to the user rather than one of the aforementioned pre-defined messages. For example, in an instance in which a user asked, *"What album is 'Love Song' in?"*, the bot would aim to retrieve a selection of albums which contained a song matching the named title. The one with the highest 'popularity' value in the Spotify database would be returned to the user. This meant that even in the case that, amongst the variety of songs contained in the Spotify database with this same title, the one which the user was specifically referring to was not there, the bot would still return an appropriate / reasonable result.

4.4 System Requirements

Taking into consideration the ideal system based on the needs of the intended target users, as well as the potential limitations of the system, the functional and non-functional requirements were defined. These are detailed below.

- Functional Requirements:
 - The bot shall provide a response to all questions / comments given to it by a user.
 - The bot shall parse user input and use natural language techniques to process the question being asked or comment being made, and will determine the user's intent.
 - The bot should retrieve reasonable information from the Spotify music database to form the basis of its response to a user. 'Reasonable' is evaluated by users during the user testing stage.
- Non-functional Requirements:
 - The bot shall take on a neutral persona when providing a response to a user. 'Neutral' is evaluated by users during the user testing stage.
 - The bot shall be interacted with through an easily-usable interface. 'Ease of usability' is evaluated by users during the user testing stage.

5. System Design and Development Plan

5.1 The Method and Design Choices

During the research stage, inspiration for potential system design was gathered from various articles which discussed programming conversational user interfaces. One article

in particular discussed how before beginning the development of a Q&A bot, a developer must first define its domain and personality. The domain knowledge (what a user expects the bot to understand) must be defined due to the notion that *“True artificial intelligence does not exist”* (10), and all bots are restricted to a particular subset of topics - the domain for this project bot being Spotify music data. Often bots are also given a particular type of persona, as with Alexa - the female personal assistant. However, ‘no personality’ is also a type of personality which a bot can be defined as having. Whilst some experimentation was done with tailoring the responses generated by the bot to adhere to a particular type of persona, the design choice to focus on the retrieval of accurate responses without any particular personality stylings was made. Nevertheless, the name Ava was chosen for the bot in order to assist in reinforcing a more realistic sense of communication and interaction between the bot and its users, and to further establish a more comfortable tone of conversation. The name Ava was decided upon in line with the current naming patterns of popular bots like Siri and Alexa: it was kept short with inclusion of vowels to aid ease of pronunciation.

Whilst considering the design of the system further, research was done with regards to exploring the capacity to which the bot would respond to user input about itself, or non-domain related topics. Two ideas were considered. The first one was to utilise simple keyword matching to recognise mentions of personal pronouns like “you”, or mentions of the bot’s name “Ava” in input such as *“How are you doing Ava?”*. The bot would subsequently produce a pre-defined statement such as *“I prefer not to talk about myself”*. The second was composition of more sophisticated responses by adding special case routines for when the bot detected that it had been addressed by the user. For example, a routine for manipulating the user’s input to produce an appropriate output in response: *“You are so helpful Ava”* would become something like *“I have been told I am quite helpful in the past”*. This would further promote the bot’s competence as an artificial intelligence. However, as the intention of the system was to focus on the processing of natural language input pertaining to the bot’s domain, the decision was made to delve less deeply into the responses produced for handling user input outside of this domain. Therefore, it was decided that for a small selection of pre-defined ‘general conversation’ inputs such as *“Hello”* and *“How are you?”*, the bot would retrieve the relevant pre-defined messages, *“Hi”* and *“I’m fine, thanks.”*, while for inputs detected as both outside of this selection and outside of the bot’s domain knowledge, the bot would respond, *“Sorry, I cannot answer that. Maybe try re-phrasing and ask again.”*

The types of questions which the bot would be able to answer related to its domain was also considered further. The decision was made that the bot would only be able to respond to open questions requiring actions such as information retrieval, e.g. *“What Kendrick album is Loyalty on?”*, *“Can you give me some pop song recommendations?”*, etc. The bot would not be able to handle closed questions requiring a ‘yes / no’ answer, or requiring the bot to choose from a set of options, e.g. *“Is Loyalty on Kendrick’s DAMN. album or To Pimp A Butterfly album?”*.

Although many NLP technologies were explored such as LUIS.AI, Google’s Dialogflow, Stanford CoreNLP, and spaCY, the main resource chosen to develop the bot was TextBlob. TextBlob (6) is an open-source library which is actively developed on GitHub, and is used for the processing of textual data. It utilises Natural Language Toolkit (11) and the Pattern.en (12) web mining module to provide a simple API for performing NLP tasks such as POS tagging, sentiment analysis, classification, tokenisation, and parsing, etc. Whilst TextBlob was used for the processing of the natural language input, the

Spotipy Python library (7) was used for carrying out searches on the Spotify database utilising the processed data.

The PyCharm IDE was the chosen coding environment for the development of the bot using the Python programming language, specifically Python version 3.7.

5.2 An Overview of the System

The development of the bot aimed to achieve efficient processing of user input data through simple but effective solutions which utilised and combined natural language processing techniques such as named entity recognition and linking, paraphrase detection, and intent classification. As aforementioned, the project did not attempt to achieve a bot which would compose sophisticated responses to questions or comments about itself using techniques such as a templating engine or Context-Free Grammar. Instead, this problem area was addressed through the solution of pre-defined messages for such cases.

A brief overview of the proposed system design is demonstrated through the pipeline below:

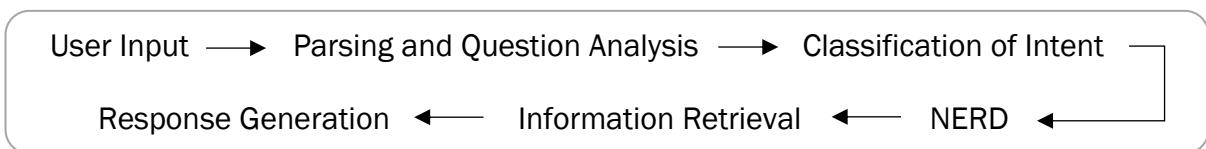


Figure 1: System Pipeline

6. Implementation Process

6.1 Content Acquisition

Upon concluding the design of the system, along with the development plan, the implementation process was initiated. Taking inspiration from the development process of IBM's Watson (as discussed in Section 3 *Background and Related Work*), the first stage was the acquisition of content.

A sample group of 5 users within the intended target audience were individually communicated the concept of the project and the bot it intended to develop. These users were then asked to come up with a list of at least 5 questions they would want to be able to ask the bot, or would expect the bot to be able to answer. This collection of sample questions (Appendix 4) allowed for a better idea of the types of potential user inputs the bot would face, and therefore gave a better characterisation of its domain in terms of what aspects of Spotify music it should be made artificially intelligent concerning. Specific details which users included in their sample questions (e.g. specific artist names) were substituted for placeholders in order to retain only the general overall essence of the question.

These questions were then grouped into categories based on the general gist of their intent (Appendix 5). The 6 categories were:

- time period (e.g. *Can you give me some albums of [genre X] from [A period to B period]?*)
- popularity (e.g. *What's the top-rated song by [artist X]?*)
- genre-specific (e.g. *Give me playlists for [genre X]*)
- mood-specific (e.g. *Suggest songs for [mood X]*)
- similar to (e.g. *Do you know any songs like [song X]?*)
- extras (e.g. *Can you tell me all the covers of [song X]?*)

The bot did not attempt to handle the types of questions that were categorised as 'extras'. This was mainly due to the type of questions it contained either being less frequently repeated in the set of user sample questions, outside of the intended scope of the bot's capability, or due to the complexities which the question could present in implementation. As many challenges relating to the processing of natural language input would have already been presented within the questions of other categories, a decision was made on what particular natural language ambiguities the project would attempt to tackle during the time constraints. In this vein, the decision was also made to forgo the implementation for handling questions within the 'time periods' category.

Additionally, keywords / phrases which were reoccurring in questions grouped into the same category were noted (Appendix 5). This gave the idea for defining a "vocabulary" for the bot that was associated with its domain of knowledge. This vocabulary was a collection of the words which were the most reoccurring across all of the sample user inputs. If a user input did not contain at least one of these words, it was highly likely that it was not related to the bot's domain of music, or an input that the bot would not be able to generate an appropriate response for. Therefore, upon detection of a user input outside of the context of this vocabulary, the bot returned the template message, *"Sorry, I cannot answer that. Maybe try re-phrasing and ask again."*

Also a part of content acquisition, was the collection of datasets which would be used for classification of entities during entity linking. Kaggle (13) is an online community for data science and machine learning on which users can find and publish datasets. This was one of the main resources through which datasets were acquired for the project.

The datasets gathered contained entries of information which could each be classified as one of the following:

- artist name
- track name
- album name
- genre
- mood
- activity / interest

The table below provides information on each of the Kaggle datasets used, as well as which classifications were associated with the data it contained:

Name of dataset	Size of dataset	Classifications pertinent to dataset
Spotify Tracks DB	228,159 tracks	artist_name, track_name, album_name
19,000 Spotify Songs	19,000 songs	artist_name, track_name, album_name
Hobbies	742 items	genre, mood, activity_interest

Figure 2: Kaggle Datasets

All datasets were transferred to excel sheets, filtered to remove any duplicates and redundant information, and then labelled accordingly. These were then converted into csv files for use in conjunction with a classifier to perform disambiguation of extracted named entities at a later stage in the process.

6.2 Question Analysis

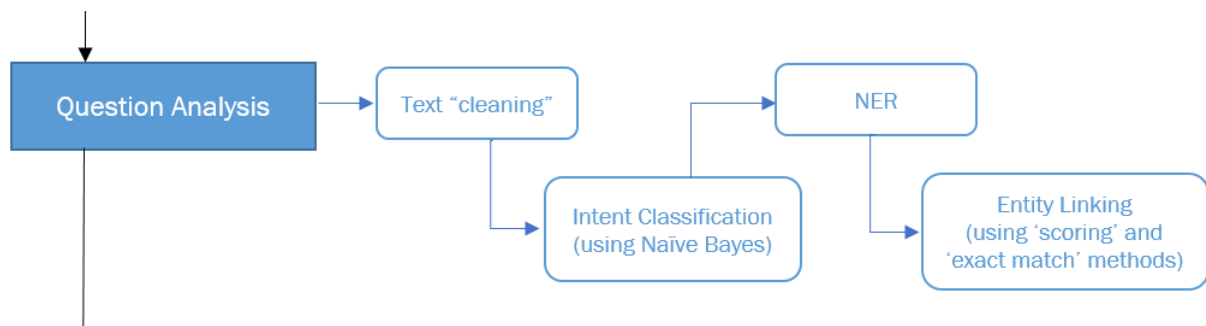
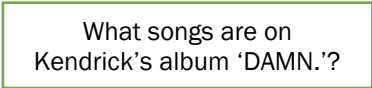


Figure 3: Question Analysis pipeline

The next stage in IBM’s ‘DeepQA’ project was the analysis of questions in order to gain understanding of what a user’s question was asking and determine how the question would be processed within the bot. This was the stage in which the bot’s system would carry out tasks such as parsing, named entity recognition, and relation detection / extraction. For this project, relation detection / extraction was not performed; however, named entity recognition and entity linking (Sections 6.2.3 and 6.2.4) were carried out in line with the primary objectives of the project earlier established. As relation detection / extraction was not implemented, this meant that each user input was parsed as new. For example, if a user input was “Give me Kendrick songs”, and then the user asked, “By the way, what is the name of his album?”, the bot would not relate ‘his’ to ‘Kendrick’ and would instead parse the second input as new.

For each of the following steps, the method of the process is detailed, and using the sample input below, the resulting output of each stage is illustrated.

Sample input:



What songs are on
Kendrick's album 'DAMN.'?

Figure 4: Sample input - original

6.2.1 Text Pre-processing

Prior to parsing and NERD, the implementation method of this project involved passing the input text through a step most commonly referred to as 'text pre-processing' or "cleaning". Text pre-processing involves the normalisation of data so that it can be more efficiently processed when using the other NLP techniques. In the case of this project, normalisation included the removal of unnecessary whitespaces and punctuation: whitespaces at the very start and / or end of a user's input would be stripped as they would provide no meaningful information, and all punctuation was removed except for a specific pre-define group. Python RegEx was the method used to define the punctuation symbols "&", "+", "-", "." and " ' " as the group to ignore when removing all punctuation marks in a user's input text. However, for all of the symbols except " ' ", this was made only to apply to cases when these characters were found within words (for example, R&B would remain as R&B but R & B would become R B). The single quote was allowed outside of the in-word boundaries as users were able to place parts of their input within single quotes for the bot to specifically target in order to improve accuracy of the response retrieved, e.g. if a user's input was "Play My lady's on fire by Ty Segall" and the bot was not parsing this accurately enough, the user could try inputting "Play 'My lady's on fire' by Ty Segall", which would guarantee extraction of the string within single quotes as a named entity. The in-word punctuation group was chosen with respect to the types of text input that would be expected within the domain for which the bot provides its service: typically entities such as artist names and genre categories can include symbols such as these; the dataset acquired from Kaggle that contained Spotify artist names had entries such as 'Will.i.am'.

In addition, normalisation for the text pre-processing involved in this project also included removal of any diacritic marks such as accents on letters (i.e. Beyoncé was transformed into Beyonce). Although some artist / track names did in fact contain accents, this was not valuable or relevant information to retain for the other textual processing stages as the Spotify API search methods accounted for potential matches containing accents, even if a search was made without (e.g. a search for Beyonce would still return Beyoncé as a match). Aside from such cases, text input of users was incapable of having any valuable meaning behind accented lettering as the bot was developed to only support English (UK / US) language input.

Although the TextBlob library included a 'correct' method which would autocorrect any spellings considered erroneous in the user's input text, this was not used. This decision was based on the domain of the system being music: song and artist names can be highly unique, and can often be intentional variations of already existing words in terms of their spelling; therefore, automatic correction of input was likely to have been more disadvantageous than beneficial in providing the appropriate "cleaning" of user input.

Output for sample input:

What songs are on
Kendrick's album 'DAMN.'

Figure 5: Sample input - "cleaned"

6.2.2 Intent Classification

Using the sample questions obtained during the Content Acquisition stage, a dataset was created to be used for classifying the intent of a user's input question / comment. Each of the questions was added and labelled with their intent type. Some additional questions were added for each existing intent type, and some new questions were added for new intent type categories. Whilst similar to the initial categories used to divide the user sample questions, the labels for 'intent type' within the Intent Classifier dataset were slightly altered. The intents defined were:

- type of music
- popularity
- similar to
- artist info
- album info
- track info

From the original categories of division, 'mood-specific' and 'genre-specific' were grouped into 'type of music'. The new labels added were: artist info, album info, track info. These were added as further potential user questions were realised during expert testing which was carried out at regular intervals whilst implementing the processes for handling input related to the original intent categories.

With the labels of intent types more clearly defined, and a wider set of questions for each category, the dataset became more reliable to be used in conjunction with the Naïve Bayes Classifier method available within TextBlob.

Naïve Bayes is a classic algorithm for text classification and natural language processing in which each word in a sentence being classified is treated as having no connection with other words in the sentence (14). There are many different implementations of Naïve Bayes. In the case of the NLTK Naïve Bayes Classifier used in TextBlob, the Boolean Multinomial Naïve Bayes implementation was used. The algorithm uses training data to create feature sets of words ('features') which occur for each particular class contained within the dataset (e.g. class: intent). Words are reduced to their stems during this process (e.g. 'having' and 'have' are reduced to the stem 'hav', and would both match this feature). The Multinomial method uses the count of the occurrence of each word in each class (word frequency), and an account for the commonality of the word, in order to assign each class a probability score. The class with the highest probability is decided as the most likely classification of the input text.

Factoring for word commonality is important in the process (15). For example, matching the word "it" would likely be considered less meaningful than a match for the word

“cheese” for a class such as ‘Food Request’. However, in the implementation of the Boolean (“binarized”) version of Multinomial Naïve Bayes, the measurement of word frequency is clipped at 1, thus allowing only binary features. This approach carries the sentiment that the occurrence of a particular word (e.g. “cheese”) may matter more than the frequency of the word (16).

Once an input text was “cleaned” by the bot, it was passed through the Naïve Bayes classifier which utilised the Intent Classifier dataset, and returned the classification of the question’s intent.

Output for sample input:

What songs are on Kendrick’s album ‘DAMN.’	Intent: track info
--	--------------------

Figure 6: Sample input - intent classified

6.2.3 Named Entity Recognition (Extraction of target ‘named entities’ and target ‘other meaningful info’)

Following the classification of intent, the input was then parsed to extract named entities as well as other entities holding information to be considered meaningful and useful for better understanding of the user’s query and a more effective search. Named entities to be extracted were considered one of the following: artist, album, track. Other meaningful information entities to be extracted were divided into the following: genre, activity / interest, mood, or other.

The process for the recognition of named entities involved parsing the input to obtain noun phrases using a combination of the two noun phrase extractors available within TextBlob. Noun phrase extractors generally use chunk parsing to match particular POS patterns in a text in order to retrieve likely ‘noun phrases’. In English language, a noun usually ends the phrase. (17) The two extractors offered were the Conll Extractor and the Fast NP extractor (which is the TextBlob default). Initially, each was individually tested with a set of 37 sample questions to evaluate its own accuracy in extracting the named entity types intended as targets for extraction (e.g. artist, album track). For each question, a list of the noun phrases extracted by each extractor method was noted (Appendix 6). The Fast NP extractor scored slightly higher (Fast NP: 19/37, Conll: 13/37) in its extraction of entities deemed relevant or useful (either as ‘named entities’ or ‘other meaningful info’ entities).

As aforementioned in Section 4.3 *Problem Area*, the implementation of the bot was intended to favour high recall, low precision. In that vein, so that no useful data that was extracted by the Conll extractor and not by the Fast NP extractor would go amiss, the approach utilised both extractors. The text input was parsed once with each extractor and so both sets of extracted noun phrases were retained.

During parsing, each word in the text input is part-of-speech tagged. TextBlob’s default POS tagger is that of the Pattern library (12), which utilises the Penn Treebank II tag set for English language (18). The POS tagging of the words in the sample questions used

during the comparison of extractors was retrieved in order to see which tags were returning named entities data correctly in the question's list of extracted noun phrases. There were a few tags which were always true for named entities and a few which were only sometimes true for the group. The tags which were always true for named entities in the extracted noun phrases list were: NNP (singular proper noun) and NNPS (plural proper noun). Therefore, after both sets of noun phrases were extracted from the text input using both extractors, the list containing all of these was reduced to create a new list containing only singular and plural proper nouns - a list of extracted target 'named entities'. Any duplicates produced as a result of beginning with the use of both extractors were removed.

A similar approach was used to produce the list of target 'other meaningful info' entities. After both sets of noun phrases were produced for each question, the list containing all of them was reduced to create a new one using the tags which were almost always true of meaningful information data (JJ, VBP, NN, CD, RBS, JJS, VBG, NNS, RB, and VB). An element of implementation was added to detect particular sequences of specific tags (e.g. JJ and RBS) in order to group phrases such as "most popular" and "most recent". Once again, any duplicates in the final list were discarded.

One limitation of using the POS tags for the filtering of the extracted noun phrases was that incorrect tags were sometimes given to words ('tokens') within input text if not grammatically correct. For example, if an artist name was not a common name like 'Michael' (e.g. an artist name like 'Cardi B'), and was inputted by a user without proper capitalisation of the first letter, it was consequently considered "ambiguous" by the TextBlob parser and this would often lead to incorrect tagging. In order to combat this, users were advised through 'Helpful Usage Tips' displayed upon initialisation of the program, that single quotes could be used to specify particular parts of their input for extraction in the event that the bot was not achieving an accurate parse. Any information within single quotes was guaranteed to be extracted by the bot, e.g. for the user input "What songs are on Ariana's album 'Sweetener'?", 'Sweetener' would definitely be extracted as a named entity. The TextBlob parser uses the implementation of the parser in the Pattern (12) library.

All extracted entities were made lowercase at this stage in order to prevent any complications when comparing data strings in subsequent processes resulting from Python comparative operators such as 'in' being case-sensitive.

Output for sample input:

What songs are on Kendrick's album 'DAMN.'	Named entities: ['kendrick', 'damn.']}	Other info: ['songs', 'album']
--	---	-----------------------------------

Figure 7: Sample input - NER performed

6.2.4 Entity Linking

Once both lists of 'named entities' and 'other meaningful info' were produced, entities contained in the lists were then linked to what type of entity they referred to. A separate

method was created for the linking of each type of extracted entities list; however, a similar method was used for both.

In order to be able to ascertain the most likely type of each extracted entity in the lists, similarly to the process for the classification of intent, datasets had to be created with example data entries for each type of entity (i.e. artist name, genre, album name, etc). Two datasets were created: Spotify Features Classifier and Other Features Classifier. The first dataset held entries for artist names, album names, and song names. The data used to form this dataset was gathered from the Kaggle datasets, 'Spotify Tracks DB' (19) and '19,000 Spotify Songs' (20). The second dataset contained entries for the labels genre, activity / interest and mood. The data for genres was obtained from the Spotify Web API list of genre seeds and playlist categories (21) (22), and the 'mood' data was collected from lists of 'core' human emotions based on various psychology theories such as the Robert Plutchik theory (23). The data for entries marked as 'activity / interest' were obtained from another Kaggle dataset, 'Hobbies' (24).

Initially, the TextBlob Naïve Bayes Classifier method was intended to be re-used to classify the entities in each list, but this proved inefficient due to significant volume of data in the datasets, as well as the type of data contained within them. Whilst the Naïve Bayes approach proved useful for the intent classification stage, it was not the most suitable approach for the classification of the 'named entities' and 'other meaningful info' data. One of the reasons for this was due to the nature of the entries in i.e. the Spotify Features Classifier dataset – containing entry names in one column, and their related labels (artist, album, track) in another. Data like song names can be very short and seemingly "random", which does not allow much room for the learning of particular features and commonalities within the data to aid classification via the Naïve Bayes method. With data in the form of questions, such as in the Intent Classifier dataset, the Naïve Bayes method immediately has a richer feature set to draw from based on things such as reoccurrences of specific words used in questions of a particular intent, and the syntactic structure of the questions labelled as a particular intent, etc.

Hence a different approach was taken for the linking of entities placed within either of the two lists. For 'other meaningful info' entities, the linking method was made simple by the fact that no item had the potential to be found in multiple categories. Therefore, the dataset was scanned for the particular entity and if found, it would be linked to the associated label; otherwise, it would be labelled as 'other info'. This was the 'exact match' method. For 'named entities', the approach used had to incorporate somewhat of a scoring system as particular entities could potentially be found within multiple categories (e.g. an artist name could also be the name of an album or a track in the database). The method used scanned the dataset for the entry and kept a count of each time it was found with a particular label. The conclusion being that the label with the highest count was the most probable label with which to link the entity to. This was the 'scoring method'.

As this approach utilised the Python 'max' method, if a tie was encountered, the first maximum would be returned as the label. If all counts were 0, the method would favour the label 'artist' – this label was chosen instead of 'other' as entities within the 'named entities' list would have had to have been POS tagged as either NNP or NNPS. This made it unlikely that the entity would not associate with any of the label types as for it to be a pronoun it would have to be the name of something. As the domain of the bot is limited to music related queries, this would mean it would most definitely refer to at least one of

the labels (artist name, track name, song name). A likely reason for the entity not being found would be as a result of the artist, song or album name simply not being contained within the Spotify music database (as discussed in Section 4.3 *Problem Area*). Consequently, the label 'artist' was decided as a default in order to increase potential of returning an answer to the user.

Another drawback of using Spotify data as the basis for linking was the potential for data to have been labelled incorrectly within the Spotify database (e.g. an artist labelled as a track). This was not something the bot attempted to handle; if a response generated by the bot was correct as per any data misinformation, it was still treated as accurate.

Output for sample input:

What songs are on Kendrick's album 'DAMN.'	Named entities: ['kendrick': artist, 'damn.': album]	Other info: ['songs': other, 'album': other]
--	---	---

Figure 8: Sample input - Entities linked

6.3 Answer Hypothesis Generation

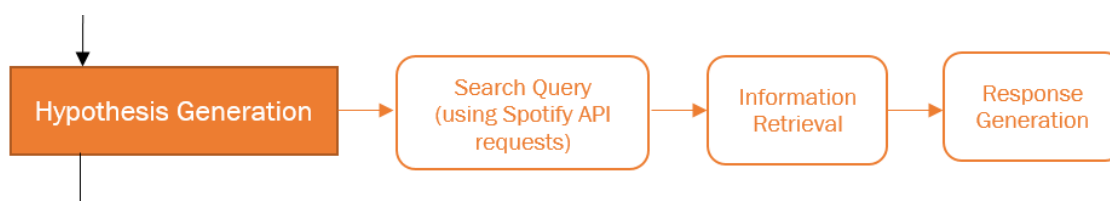


Figure 9: Hypothesis Generation pipeline

6.3.1 Search Query and Response Generation

The final stage of relevance to this project, maintaining inspiration from IBM's 'DeepQA' project, was the use of the question analysis results for the generation of reasonable answers through search of the system's sources. Once all the process of NERD was completed, the dictionary data of the names of the entities for both lists, and their associated links, was passed into a 'search by intent' method. This method took the data of both dictionaries, as well as the intent classification of the input question, and attempted to use the most relevant method from the Spotify library (of those contained within the 'search by intent' method) to perform a search query with which to return an appropriate / reasonable response to the user.

For example, considering the case of the sample input in which the NERD process resulted in the following data for the question, "What songs are on Kendrick's 'DAMN.' album?":

- Cleaned text: What songs are on Kendrick's 'DAMN.' album
- Intent classification: track info
- Named entities: 'kendrick' – artist, 'damn.' - album
- Other meaningful info: 'songs' – other info, 'album' – other info

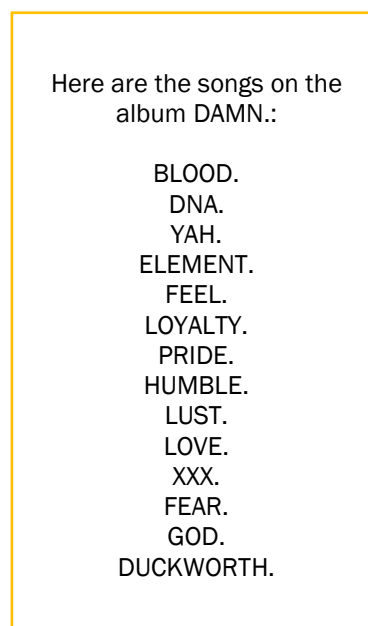
This would result in the ‘search by intent’ method performing a search query using the ‘track info’ method. The process would first search the named entities list for an entity labelled as an ‘album’, and then would search for this name within the ‘albums’ portion of the Spotify database. If matching albums were found, in the event that an entity labelled ‘artist’ was also in the named entities list, the method would search for this item name within the ‘artists’ list for each album. If a match was found, the songs on that album would be retrieved. In the event that no match was found, the most popular album in the Spotify database containing the entity name labelled as ‘album’ would be used.

For search queries like requests for song recommendations for a particular genre, an offset was added to the data retrieved from the Spotify database in order to ensure unique results if a user were to re-ask the same question. Furthermore, for search queries which required the bot to return multiple dynamic results (i.e. in the case of the input “*What are some good pop songs?*”), a list of 5 results maximum from the retrieved data was returned to the user. However, for queries which required the bot to return multiple results which were static information (i.e. the tracks on a particular album are always the same), the full list of results was retrieved.

For cases in which a result could not be produced, the bot would return the most suitable template response (as addressed in Section 4.3 *Problem Area*). An example of this is: “*Sorry, I could not find any [genre X] songs in the Spotify music library for you.*”

Final output:

*Note: All track names on this album are completely capitalised.



Here are the songs on the
album DAMN.:

BLOOD.
DNA.
YAH.
ELEMENT.
FEEL.
LOYALTY.
PRIDE.
HUMBLE.
LUST.
LOVE.
XXX.
FEAR.
GOD.
DUCKWORTH.

Figure 10: Sample input - final output

6.3.2 Improvements to the Existing Functionality of the Spotify Service

During the implementation process, further research was carried out into the needs of already existing users of the Spotify music service in order to inspire the development of the bot and its functionalities. A particular user forum on the Spotify Community website (25) helped to provide further insight into the current issues users were facing with particular aspects of the already existing Spotify music system. As the ‘Response Generation’ stage of the bot’s implementation process was based upon the utilisation of the Spotify Web API, any issues regarding this system needed to be taken into consideration.

One issue which the forum discussed was the limitations of the current approach used for the ‘Search’ feature of the Spotify music service. For example, one user complained that when they searched for the indie band called ‘Helm’, although there was an exact match for this (and only one) in the Spotify ‘Artists’ list, instead of the band being displayed as the first result on the search results page it was shown in the 16th position. The user commented that they seemed to always have that problem when searching for less commonly known artists using Spotify ‘Search’, which they frequently did. Other users agreed to having similar experiences.

Upon looking into the existing approach of the ‘Search’ feature as a result of reading the forum discussion, it was found that the current approach of the Spotify ‘Search’ feature was simply searching for a specific character string in the names of the artists contained within the database (in the case of search for an artist). For example, in response to the user searching for the indie band ‘Helm’, the Spotify service would search for this string within the names of artists in the database, thus returning artist results including ‘Helmet’, ‘Helmuth Rilling’, and ‘Asleep at the Helm’. These results were then displayed in the results list based on popularity – which explained the 16th position of ‘Helm’ due to being an indie band not commonly known.

This was taken into consideration when implementing the methods for processing search queries within the bot using the ‘search’ method of the Spotify Web API. For a user input case such as, *“Do you know any good Helm songs?”*, the bot would first identify any exact match within the Spotify database and use this for generating a response to the user. If no such match existed, the bot would use the most popular of the results found in the database for its hypothesis generation.

An issue discussed on a Spotify Developers forum (26) on GitHub was concerning using the ‘Search’ method of the API for searching for items of particular genres within the Spotify music database. A person utilising the API for the development of their own project communicated that sometimes ‘Search’ would return results for items with the genre classification ‘acoustic’ and sometimes it would not. They had also noticed this for other genres. A member of the Spotify API team explained that their system used different lists of ‘genres’ for the classification of different data types in their database (i.e. songs, albums, playlists, artists). Therefore, whilst ‘acoustic’ as a genre filter for the search of playlists would return results, it would not be a valid genre name for the search of songs.

Taking this into consideration during implementation, an option was added for users to input the key phrase ‘show genres’ to display a list of possible genre names they would

be able to use during communication with the bot. This list was divided into genre names allowed for each data type (i.e. songs, albums, playlists, artists).

6.3.3 Handling of Ambiguities and Limitations of Solutions

A few example of the ways in which the system was built to deal with particular ambiguities from user input, and the limitations of these solutions, is detailed below using examples:

1. *“What are some cool 80s songs?”* – for questions such as this which included subjective terminology such as “cool”, the search query carried out a search for recommendations based on a degree of popularity. The parameter for popularity given was ‘minimum: 50, maximum: 100’ (with ‘100’ being the highest popularity). A range was given in order to not simply return songs of highest popularity as a user might have already known these – giving a range allowed for increased potential of unfamiliarity of the recommendation given whilst still retaining a reasonable degree of potential likeability for the user.
2. *“Give me some R&B songs to listen to”* – as discussed in the previous section, an option was implemented to give users an idea of the types of genres they would be able to communicate to the bot effectively. For genres such as ‘R&B’, the Spotify ‘Search’ method uses the format ‘r-n-b’ for song items and ‘R&B’ for others. A print message was added to convey such information to a user in the event of a search query being unable to return any results, in case of this being the reason. This would suggest re-phrasing of the input text for a potentially better result.
3. *“Can you play me a cover of Please Me?”* – it was decided that the bot would not handle the processing of variations of same music items (e.g. explicit version, clean version, cover, original, remix, re-make, extended edition, etc).
4. *“Give me a rap song and album”* – for queries involving multiple data types for a same entity type (e.g. genre, artist, etc), the bot would only return one of the requested data types. This was the order of the priority for this choice from highest to lowest: album, playlist, song (or any other general request such as ‘music’). Therefore, in the example case, the bot would return a rap album.
 - Further examples:
 - *“I want to listen to a rap song and a pop album”* – this query would result in a rap album and a pop album being returned to the user.
 - *Can you give me a rap and pop song?”* – the bot would retrieve songs for both genres individually.
5. *“Do you know any sad Ariana Grande songs?”* – for inputs which would result in entities being in both the ‘named entities’ list and ‘other meaningful info’ list, the bot search method would retrieve results for each list respectively. In the case of the example, the bot would return results for both ‘sad songs’ and ‘Ariana Grande songs’ separately. This method of handling could be considered a limitation of the system but was chosen due to the difficulty of being able to ascertain a logical parameter to use as a search filter based on a subjective term such as “sad” where it related to data items of particular ‘named entity’ (e.g. an artist) – i.e. it

would be difficult to define a parameter for what would be considered a ‘sad’ Ariana Grande song.

6. “What is Will’s latest album?” – for inputs which contained a name (i.e. an artist) that could refer to multiple results within the database, the search was carried out using the most popular result either containing the name, or an exact match of the name. The user was notified of this by the bot, *“I have used the most popular artist with the name [search name] to find results for you.”*

6.4 Extensions to the System

Upon finalising the core functionalities of the implementation, a few extensions were added to further enhance the capabilities of the bot as a conversational user interface, and to add further value to the system in terms of meeting the needs of its target users (see Section 4.1 *Needs of Intended Users*).

The first step in doing this was adding gTTS (27) to the system which returned the bot’s communications to users in a vocal format. The Google Text-to-Speech Python library and CLI tool worked by creating a vocal sound file of a Python text string, and saving this to an allocated place within the system directory. This was then retrieved and played back to the user as “speech” via a pygame mixer. Pygame (28) is a Python wrapper module for the Simple DirectMedia Layer (SDL) multimedia library which has support for audio output. This extension allowed for users to hear spoken responses of the answers given by the bot in accompaniment of the printed text in the interface.

Secondly, the SpeechRecognition (29) module was added which utilised the Google Cloud Speech API to listen to a user’s voice input (in this case, using the PC’s microphone as a source point), and then send this to a ‘recognise’ method which would return the input in a text format able to be used as a variable within Python. This extension allowed the addition of a separate ‘voice mode’ as well as the original ‘text mode’ by which users could interact with the bot. Implementation was added to allow users to switch between both modes at any point in their conversation with the bot by simply stating, *“text mode”* (in the case of being in voice mode), or typing ‘voice mode’ (in the case of being in text mode).

The final extension made to the system was the incorporation of playback for songs retrieved as a response to a user’s input question. The playback could be either via a Spotify Web Player or Spotify Desktop Player - which the user would need to have open at the time of making the playback request. Users would also need to be logged into the player with a Spotify Premium account in order to have access to this feature.

The playback extension was added for the playback of song results only (i.e. not for any data retrieved as an album or playlist). For example, for a response in which a list of 5 pop song results was returned to a user, if the user wished to begin playback for the first song, they would simply type or say “1” (depending on which interaction mode they were in). This was made possible through the ‘start_playback’ method of the Spotipy library, which accessed the ‘Start/Resume a User’s Playback’ endpoint of the Spotify Web API.

In accompaniment of this extension, an additional intent (‘play’) was added to the Intent Classifier dataset, along with potential user inputs associated with this intent (e.g. *Play*

[song X]). This was added as the inclusion of a song playback feature meant that it was highly likely that, as a by-product, users would want to be able to directly request the playback of particular songs themselves, instead of only being able to choose from a list of results retrieved by the bot. Therefore, with the addition of the ‘play’ intent, users could directly command “*Play [song X]*”, and the bot would begin playback of the song.

6.5 Experimentation with Paraphrase Detection

During the implementation of intent classification (as discussed in Section 6.2.2 *Intent Classification*), experimentation was carried out with regards to the method which would be used to relate user input text to data within the Intent Classifier dataset in order to determine the most likely intent of the user’s input. Although the approach chosen was Boolean Multinomial Naïve Bayes, an implementation for semantic similarity (“paraphrase detection”) was tested to compare results for the accuracy of intent classification when using each of the two measures of similarity.

The notion of ‘similarity’ can be considered relative to what type of information is desired. For example, whilst the sentences “*This is a tree*” and “*This is not a tree*” would be considered structurally similar, their semantic meanings are in fact opposite. In addition, whilst “*Gorgeous creatures are felines*” and “*Cats are beautiful animals*” would be considered structurally dissimilar, they are in fact highly semantically related. Many different measures for ascertaining semantic similarity exist and each are more relevant for use in a particular context.

For this project, semantic similarity was computed via WordNet (5). The approach used was inspired by an online tutorial by George-Bogdan Ivanov (30), whose method utilised an algorithm proposed by Mihalcea et al. in the paper ‘*Corpus-based and Knowledge-based Measures of Text Semantic Similarity*’ (31). The process built a symmetric similarity measure which produced a score for the semantic similarity between two given sentences.

This was done by retrieving the most common synset for each word in each sentence and placing these into a list for the particular sentence. The synsets of each word described its meaning, part of speeches, and synonyms / antonyms. For each word synset of the ‘focus sentence’, the similarity value of the most similar word in the ‘comparison sentence’ was used for the accumulation of a score which would provide the similarity of both sentences. The final similarity score of both sentences was calculated by obtaining the average of all of the accumulated similarity values. The function was made symmetrical by performing the calculation twice, with each sentence swapping role as the ‘focus’ and the ‘comparison’ sentence. The score values produced from each turn were then added together and the sum of these was divided by 2 in order to produce the final symmetrical similarity score of the two sentences.

The ‘similarity value’ for each pair of synsets can be calculated using different word similarity metrics: the one chosen for the experimental implementation used in this project was the thesaurus-based metric ‘Path Similarity’. This computed the shortest number of edges from one word sense to another word sense, assuming a hierarchical structure like that of WordNet (essentially a graph). Word senses which were found to have a longer path distance were distinguished as less similar than those with a much shorter path distance (32).

The Path Similarity algorithm does not attempt to differentiate between semantic similarity and semantic relatedness. As discussed in *'Speech and Language Processing, 2nd Edition'* (33), 'word relatedness' characterizes a larger set of potential relationships between words: antonyms have high relatedness but low similarity - 'word similarity' is thus a sub case of 'word relatedness'. As a result, when a similarity metric such as Path Similarity is used to rank the similarity of word pairs such as ['love', 'hate'] and ['love', 'romance'], the similarity score of 'love' and 'hate' was found to be considerably higher than 'love' and 'romance' (32). Therefore, the algorithm gave more of a sense of the semantic relatedness of two sentences when used in the experimentation of paraphrase detection within this project.

The experimentation process used 'test' user input sentences and classified these first using the Naïve Bayes approach, and then using semantic relatedness by retrieving the intent label (e.g. type of music) of the sentence in the Intent Classifier dataset which was found to be most similar to the input sentence given. This meant that it found the question / request in the dataset that was closest to being a 'paraphrase' of the input question, and based the decision for intent classification on this. The results were recorded and can be seen in Appendix 7.

The results found the Naïve Bayes approach to be more accurate than the semantic similarity method in the majority of cases. While both struggled more when attempting to accurately classify the intent of a more complex input (e.g. with multiple sentences or higher ambiguity of intent) such as, *"I don't want general Busta Rhymes music. What is his most popular music?"*, for simpler inputs like *"Hmm, do you know any songs by The 1975?"*, Naïve Bayes performed better over the majority of test user inputs, and thus was chosen for the final implementation of the intent classification method for the bot. This could suggest that for the context in which the detection of similarity is required for this project, the occurrence of particular features in a user's input text are a better measure of the notion of 'similarity' than the semantic relatedness of the user's input to a pre-classified sentence.

7. System Testing

During the development process, unit testing was performed at regular intervals in order to ensure that appropriate results were being returned for each of the completed methods individually. The system was also frequently tested as a whole through interaction with the bot from the point of program initialisation, as a user would. This was carried out through the Terminal interface.

8. Underlying Principles of the Program

In comparison to the brief overview of the proposed system given in Section 5.2 *An Overview of the System*, below is a system flow diagram to give a more detailed idea of the way in which the key procedures and modules within the finished system interact:

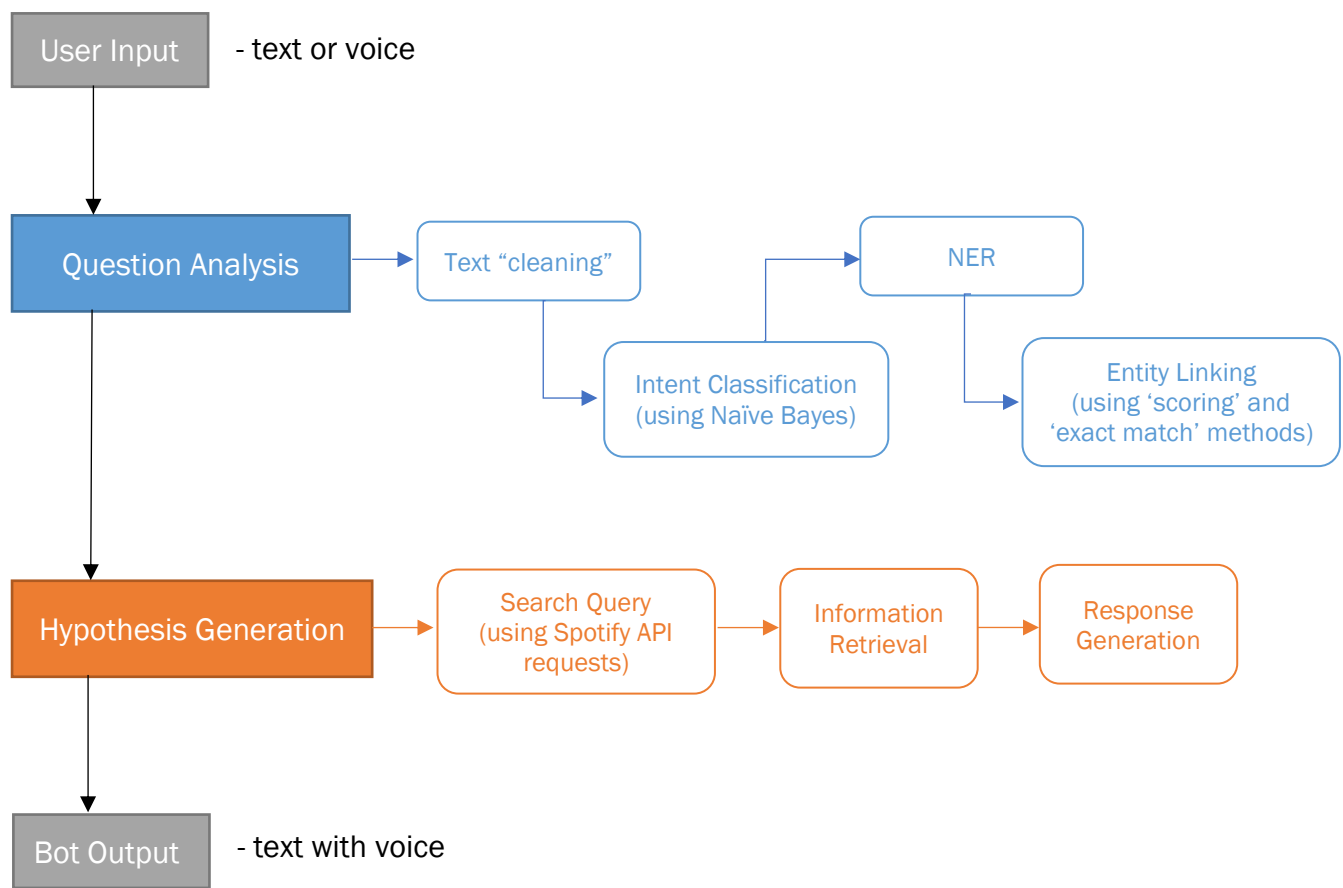


Figure 11: System Flow Diagram

Although the significance of key data structures was not depicted in the diagram above, each of the stages of the implementation process required data to be passed to them in a particular format in order to handle them as intended. Within these transactions of data, the data was either manipulated or utilized in some way by the process, and returned either in the same format, or translated into a new one.

An example of one of the key data structures involved was the Python 'Dictionary' data type which was the output of the 'Entity Linking' stage in the process. The dictionary data structure has significance here as it allowed for the items in both lists generated during the 'NER' stage to be attached with their respective labels. If a data structure such as a Python 'List' were to be used, the labels for the entities would not be able to be indexed directly by the label name (e.g. genre) in order to retrieve the corresponding entity name (e.g. pop), and vice versa. The quick retrieval of specific items of data permitted by the key-value format of the dictionary data type was necessary during the 'Search Query' stage in which these specific items of data were utilized to make API requests to the Spotify database. This type of indexing in a list data type would require linear iteration of the list which would be highly inefficient (especially in the event of searching the entire list), and would not satisfy the need of users concerning quick information retrieval as stated in Section 4.1 *Needs of Intended Users*.

9. System Evaluation

9.1 Data Analysis of User Testing

Once completed, the bot was evaluated by 6 users. Each user tested a ‘basic’ version of the bot which contained none of the extensions, and an ‘extended’ version which contained all extensions. For each version of the bot users tested 5 inputs. The inputs used in testing were different to those used in the training of the bot: users were first communicated the bot’s domain, and then were able to try out any unique input they could come up with within the context of this domain. Once both versions of the bot were tested, each user completed a questionnaire (Appendix 8) which evaluated the performance of the system against the original system requirements, as stated in Section 4.4 *System Requirements*. For parts of the questionnaire in which users were asked to provide a score or to choose from a list of options, an additional comment box was added to encourage users to detail the reasons for their choice. This method was useful as it provided further insight when analysing the data.

The following results are based on user evaluation of the core functionality of the completed system (‘basic’ version) in line with the originally defined requirements. The first functional requirement of the system was to ensure that a response was returned for all user inputs. The results showed that this requirement was fulfilled.

Did the bot always give a response to your input?

6 responses

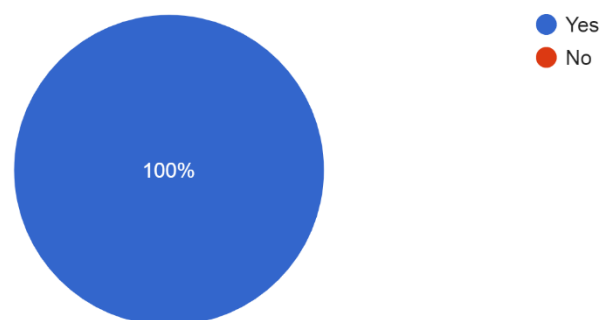


Figure 12: User Feedback - Q1

Another functional requirement was for the bot to be able to retrieve reasonable information from the Spotify music database to form the basis of its response to a user. The majority of users felt that ‘reasonable’ information (as defined by the user), was retrieved ‘sometimes’.

Did you feel that the bot retrieved reasonable responses to your input?

6 responses

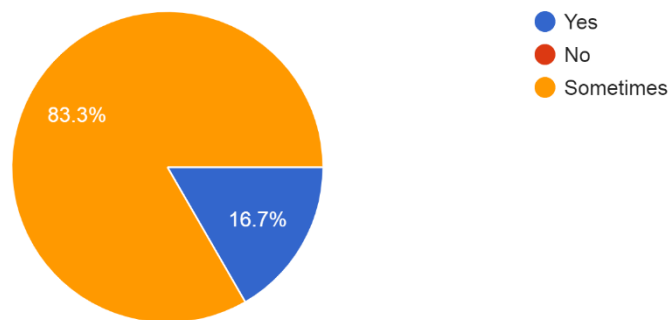


Figure 13: User Feedback - Q2

User comments detailed that for some of the inputs which users tried, although a response was retrieved, the response did not always satisfy the intent of the question. For example, one user asked when a particular song by an artist was released, and the bot retrieved a popular song by the artist as a response. This was due to the bot classifying the intent of the input as 'popularity', whilst an input of this kind would have been better associated with the intent 'time period'. As discussed in Section 6.1 *Content Acquisition*, the implementation for the 'time period' intent category was not carried out. Therefore, the bot chose the most appropriate category of those contained within the Intent Classifier dataset to classify the input: this was 'popularity' - which lead to the retrieval of the aforementioned response.

Other users commented that in a few cases the bot linked entities incorrectly, leading to the retrieval of an incorrect response. In one of these cases, a song name was linked as 'artist'. This meant that the scoring method used for named entities (as discussed in Section 6.2.4 *Entity Linking*) had fallen short in the case of that input. This was likely due to the entity having an equal score for both 'artist' and 'track' when classified using the Spotify Features Classifier dataset which contained album, artist and track names. As in the event of an equal score, the label 'artist' was favoured, this may have lead to an incorrect result in this case.

Concerning the non-functional requirement for the bot to maintain a neutral persona, 100% of users agreed that this requirement was fulfilled.

When communicating with the bot, did you feel that it had a neutral persona or a particular type of personality? Please state your impression if not 'neutral'.

6 responses

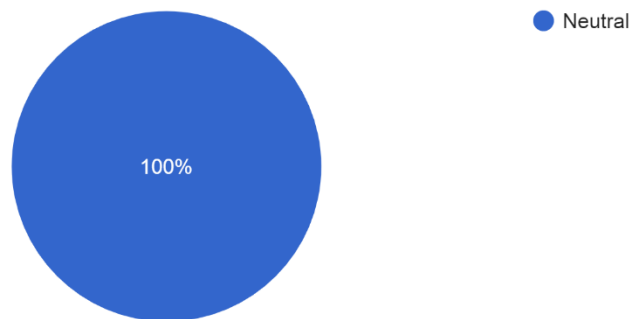


Figure 14: User Feedback - Q4

Another non-functional requirement was for the bot to be interacted with through an easily-usable interface. When asked to score the ease-of-usability of the interface in which the bot was tested, the average score of the users was 4/5. User comments included, “super easy”, “very basic”, “simple...not many things could go wrong”. However, a couple of users commented that the interface could have been more visual and user-friendly if a graphical user interface (GUI) was incorporated instead of using the PC terminal. Whilst the main objectives of this project focussed on the utilisation of natural language engineering techniques in the production of a bot capable of question-answering, in the event of actual release of the program into the public domain, a GUI would definitely be required.

A few users also felt that the introductory ‘Helpful Usage Tips’ text displayed on initialisation of the program should be reduced.

On a scale of 1-5, how easy-to-use did you find the interface through which you interacted with the bot?

6 responses

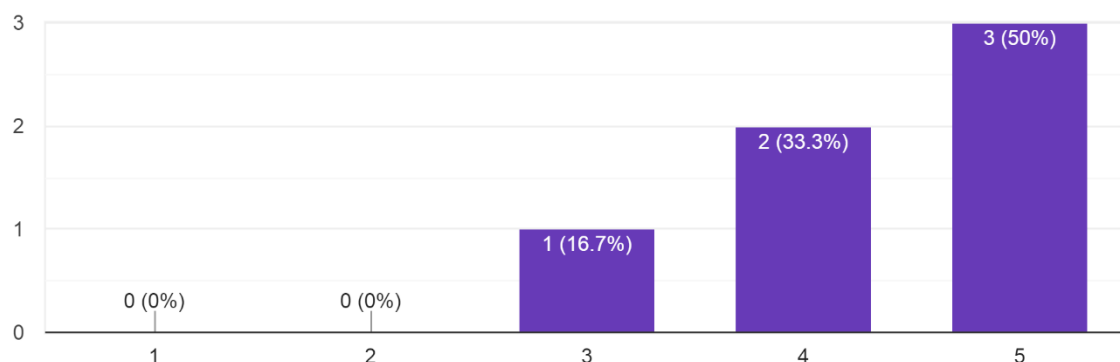


Figure 15: User Feedback - Q5

Users were also asked how easy they found it to communicate with the bot, as per the functional requirement for the bot to be able to parse user input and use NLP techniques to appropriately determine the user's intent.

Did you find it easy to communicate with the bot? (i.e. have it understand your input, and be able to understand its response)

6 responses

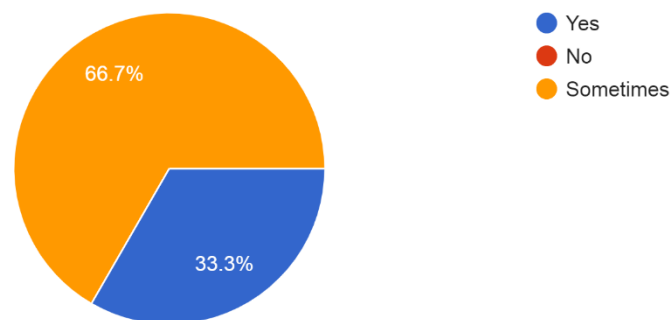


Figure 16: User Feedback - Q3

The next set of data results are based on user evaluation of the extensions to the system ('extended' version). Firstly, users evaluated the 'voice mode' extension. Users were asked the following 5 questions pertaining to 'voice mode':

1. Did you find it easy to use voice mode?
2. Did you prefer to listen to the bot speak its responses, or read the responses as text?
3. On a scale of 1-5, how much do you feel voice mode enhanced your interaction with the bot?
4. Did you feel that the addition of voice mode was of benefit to you as a user?
5. Did you prefer voice mode or text mode?

In response to the first question, the majority of users voted 'sometimes'. A few users felt that the bot struggled to correctly parse their voice input as text on a few occasions. One user attempted to utilise 'voice mode' in an environment with some background noise and found that the bot struggled on almost all inputs. Whilst the microphone quality of the PC device on which user testing was performed could have been a factor, it would be unrealistic to assume that a real user would be utilising the bot in an environment with no background noise at all. Therefore, the capacity to which 'voice mode' can detect voice input whilst distinguishing and filtering background audio would have to be explored and improved in the event of further development.

Did you find it easy to use voice mode?

6 responses

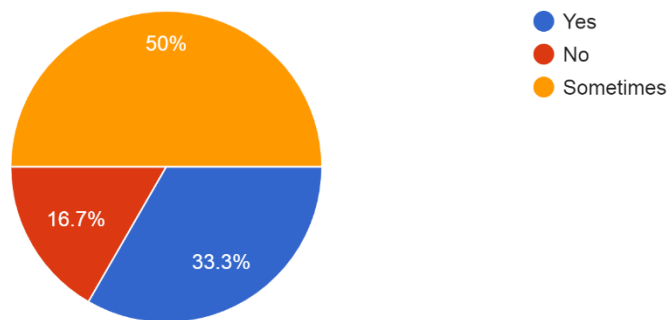


Figure 17: User Feedback - Voice Mode - Q1

In response to the second question, the majority of users preferred to read the text response themselves instead of it being read to them by the bot's text-to-speech function. Almost all users communicated that this was due to the speed of the bot's "voice" being too slow. The Google Text-To-Speech (gTTS) tool allowed only two speeds of audio output: normal and slow. Although the bot utilised the 'normal' setting, most users felt that this was still too slow, and stated that it was faster to read the printed text of the response themselves. Most users mentioned that they would have preferred the voice output if it was a faster speed. This also tied in with the feedback given for the third question which asked users to score how much they felt 'voice mode' enhanced their interaction with the bot.

On a scale of 1-5, how much do you feel voice mode enhanced your interaction with the bot?

6 responses

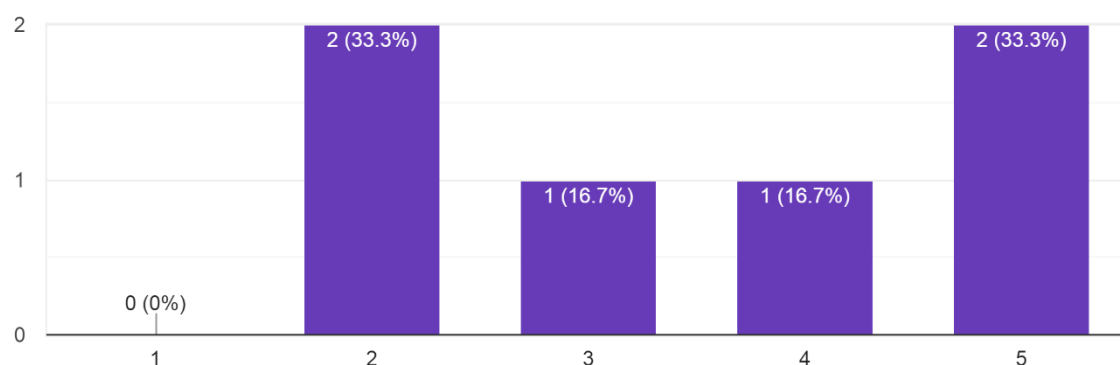


Figure 18: User Feedback - Voice Mode - Q3

The average score of the votes was 3.5/5. Aside from the bot's "voice" being undesirably slow, users also felt that the template response messages returned in such cases as the bot being unable to extract any relevant information from user input, were too long.

In response to the fourth question, the majority of users felt that 'voice mode' was of benefit to them as a user. However, when asked if users preferred 'voice mode' or 'text mode', the results were 50/50.

Did you feel that the addition of voice mode was of benefit to you as a user?

6 responses

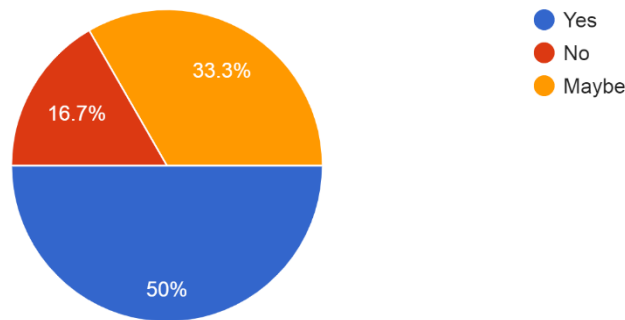


Figure 19: User Feedback - Voice Mode - Q5

A few of the reasons for choosing each mode are detailed below.

Voice mode:

- Easier than typing
- More personal
- Faster to speak than type
- Useful for when not in front of input device

Text mode:

- Better accuracy
- Voice mode speaks too slowly
- Voice mode sometimes struggles to parse voice input correctly

The feedback suggested that with improvements to the accuracy of 'voice mode', it would likely be the preferred mode of users.

Secondly, users evaluated the 'song playback' extension (including addition of the 'play' intent). Users were asked the following 3 questions pertaining to 'song playback':

1. Did you find it easy to use the song playback functionality?
2. On a scale of 1-5, how much do you feel song playback enhanced your interaction with the bot?
3. Did you feel that the addition of song playback was of benefit to you as a user?

For the first question, 5/6 users answered 'yes'. One user answered 'sometimes' as they would have liked to have been able to pause / stop the song playback via the bot, as

opposed to having to do so via the Spotify Web Player through which the song was being played.

Did you find it easy to use the song playback functionality?

6 responses

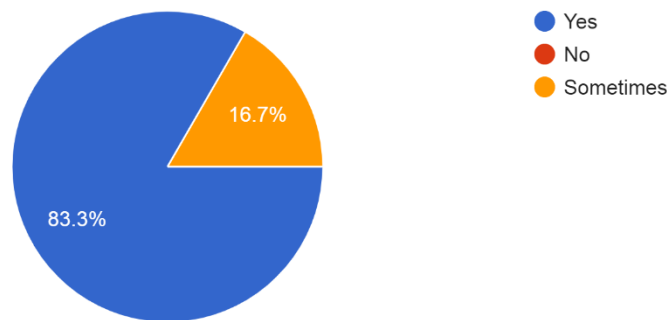


Figure 20: User Feedback - Song Playback - Q1

The average of the scores for the second question was 5/5. This was further supported by the response to the third question: all users felt that the 'song playback' feature was of benefit to them.

Did you feel that the addition of song playback was of benefit to you as a user?

6 responses

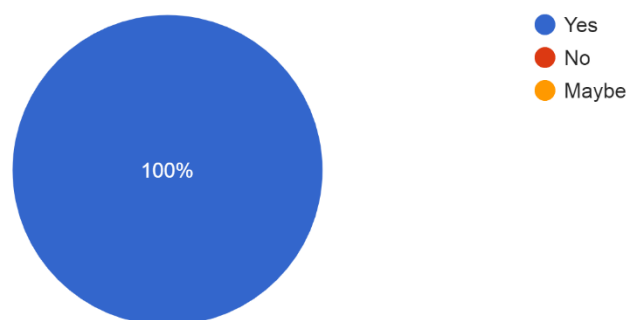


Figure 21: User Feedback - Song Playback - Q3

The feedback on the 'extended' version concluded that, although the needs of users were better satisfied through the implementation of the extensions such as 'play' intent and 'song playback', the performance of the system was not greatly improved through the addition of the 'voice mode' feature.

The final questions that users were asked related to their experience of the system as a whole in terms of its question-answering ability, as well as their general experience interacting with the bot overall.

How well did you feel the system performed as a whole overall (in terms of question-answering ability)?

6 responses

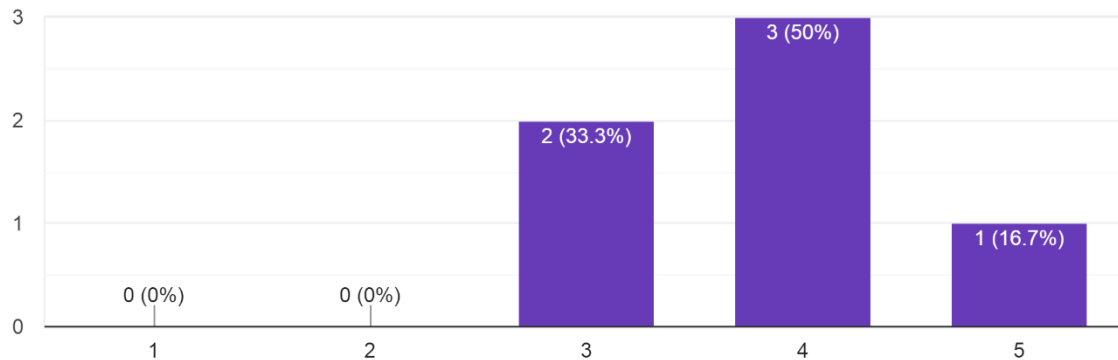


Figure 22: User Feedback - Overall System - Q1

How well would you rate your experience as a user interacting with the bot overall?

6 responses

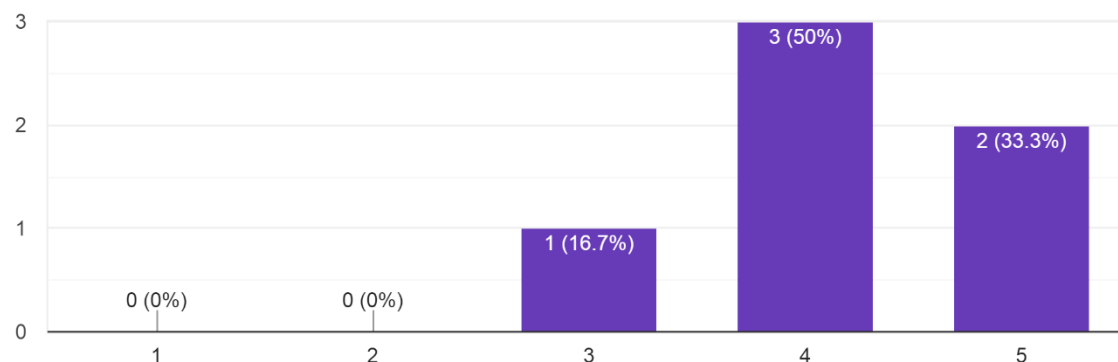


Figure 23: User Feedback - Overall System - Q2

The average score for both questions was 4/5, which suggests that the system produced was a success in terms of both its original requirements, and its primary question-answering objective.

In additional feedback, a few users further reiterated that 'voice mode' needed to be improved in terms of both audio speed, and accuracy of speech recognition. One user also suggested for additional intents to be added for a further increase of the variety of potential inputs which can be handled by the bot. In this respect, many of the inputs which users attempted that could not be appropriately handled by the bot would have been associated with the 'time period' intent category.

9.2 Resulting Software Improvements

From the analysis of the user testing data, the potential improvements to be made to the system were the following:

- Coverage of a larger variety of intents
- Increased speed of “voice” in voice mode
- Better understanding of voice input in voice mode
- Alternative to ‘scoring method’ used for linking of named entities
- User-friendly GUI
- Shortening of introductory ‘Helpful Usage Tips’ text
- Shortening of template responses
- Ability to pause / stop playback via bot

As a result of the feedback, the introductory text and template responses were shortened. The pause / stop playback functionality was also added.

Concerning a GUI and larger coverage of intents, these would be beneficial additions in the event of the bot being developed further, particularly the ‘time period’ intent as this was a reoccurring theme in the inputs tested by users during User Testing. Implementation of this intent would also inspire further exploration into handling ambiguities related to natural language processing, e.g. in the input *“Which songs did The 1975 release in 2010?”* the bot would be required to differentiate between which is the artist name information and which is the time period information. This would benefit from further research into Relationship Extraction / Detection, Machine Learning and Neural Networks.

Regarding the suggestions for improvement to the speed of audio output in ‘voice mode’ and the quality of speech recognition, these could not be enhanced as the free versions of the tools used were limited in their available functionality and scope for alteration.

User testing identified that the ‘Entity Linking’ stage of the system pipeline was proving less reliable than other parts of the pipeline, and thus leading to less accuracy at the ‘Response Generation’ stage. Subsequently, improvements were made to the ‘scoring’ method used for the linking of named entities, in order to achieve better accuracy of entity disambiguation.

Regular expressions were used to identify particular elements of the data items within the Spotify Features Classifier dataset which the named entity items were being classified against. This was in attempt to enable the bot to be able to better differentiate between the labels ‘artist’, ‘album’ and ‘track’. For example, if a named entity item was found within a dataset item labelled as ‘track’, instead of simply adding to the count for this label, RegEx was used to identify if the dataset item contained any brackets, and if the search name was within those brackets. If the search name was found within the brackets, it was likely the name of an artist mentioned in the dataset item and not the name of the track itself, e.g. Please me (feat. Cardi B) – therefore the count for the label ‘artist’ was incremented.

A few sample input tests were carried out following this update and the results suggested a reasonable improvement to the system. However, this aspect of the system would also benefit from further research into alternative methods which could be potentially better suited for carrying out the disambiguation process.

10. Conclusion

As introduced in Section 1.1 *Project Aims and Objectives*, this project aimed to develop a question-answering bot capable of answering simple natural language questions pertaining to the contents of a database. The database being that of the Spotify music library. The analysis of the User Testing stage concluded that the finished product developed through the project was a success in terms of achieving its objectives and fulfilling its system requirements, as well as the needs of target users.

The project attempted to recreate a bot similar to Apple's 'Siri' in the pursuit of acquiring a better understanding of natural language engineering and the techniques which it encompasses. The project made an effort to incorporate some level of intelligence into the bot for being able to understand the wider range of input which conversational natural language brings, as opposed to simple recognition of command words such as that of Amazon's 'Alexa' (e.g. "Alexa, 'play'..."). As this was achieved without the use of Machine Learning or Neural Networks, there were limitations to the accuracy by which natural language processing could be carried out using the alternative methods used. As aforementioned, if developed further, these routes could be explored to analyze if an implementation of the bot using these techniques would produce significantly better results for its domain.

Though the bot was implemented and tested on a PC device, if utilised with another device such as a smartphone or smart-home device (e.g. Amazon Echo), the bot would hope to be capable of performing similarly to bots currently available for the same domain, such as 'Alexa' (Amazon Music) and 'Siri' (Apple iTunes). Spotify Music currently has no such technology available for their service specifically.

Whilst the system was not completely perfected to handle the ambiguities of all potential user inputs, it attempted to deal with a select few in an appropriate manner, and was reasonably successful in doing so.

References

1. Spotify. Web API. *Spotify for Developers*. [Online] [Cited: 11 01, 2018.] <https://developer.spotify.com/documentation/web-api/>.
2. The British Computer Society. BCS Code of Conduct. *bcs.org*. [Online] 06 03, 2015. [Cited: 11 05, 2018.] <https://www.bcs.org/upload/pdf/conduct.pdf>.
3. Mason, Mitch. 3 types of business chatbots you can build. *IBM.com*. [Online] 12 12, 2017. [Cited: 11 05, 2018.] <https://www.ibm.com/blogs/watson/2017/12/3-types-of-business-chatbots-you-can-build/>.
4. David Ferrucci, Eric Brown, Jennifer Chu-Carroll, James Fan, David Gondek, Aditya A. Kalyanpur, Adam Lally, J. William Murdock, Eric Nyberg, John Prager, Nico Schlaefer, and Chris Welty. Building Watson: An Overview of the DeepQA Project. *AAAI.org*. [Online] 07 28, 2010. [Cited: 11 05, 2018.] <https://www.aaai.org/ojs/index.php/aimagazine/article/view/2303>.
5. WordNet. *Princeton University*. [Online] [Cited: 11 12, 2018.] <https://wordnet.princeton.edu/>.
6. Loria, Steve. TextBlob: Simplified Text Processing. *TextBlob*. [Online] [Cited: 11 01, 2018.] <https://textblob.readthedocs.io/en/dev/>.
7. Lamere, Paul. Welcome to Spotipy! *spotipy*. [Online] [Cited: 12 01, 2018.] <https://spotipy.readthedocs.io/en/latest/>.
8. M. C. McCord, J. W. Murdock, B. K. Boguraev. Deep Parsing in Watson. *brenocon.com*. [Online] 07 2012. [Cited: 11 05, 2018.] http://brenocon.com/watson_special_issue/03%20Deep%20parsing.pdf.
9. Jonathan Berant, Percy Liang. *Semantic Parsing via Paraphrasing*. s.l. : Stanford University, 2014.
10. Chatbot Fundamentals. *World Writable*. [Online] [Cited: 10 07, 2018.] <https://apps.worldwritable.com/tutorials/chatbot/>.
11. NLTK. Natural Language Toolkit. *nltk.org*. [Online] [Cited: 01 12, 2019.] <https://www.nltk.org/>.
12. Center, CLiPS Research. Pattern. *CLiPS*. [Online] [Cited: 01 12, 2019.] <https://www.clips.uantwerpen.be/pattern>.
13. Inc., Kaggle. Datasets. *Kaggle*. [Online] [Cited: 11 10, 2018.] <https://www.kaggle.com/datasets?sortBy=hotness&group=public&page=1&pageSize=20&size=all&filetype=all&license=all>.
14. gk. Text Classification using Algorithms. *Medium*. [Online] 01 11, 2017. [Cited: 11 01, 2018.] <https://chatbotlife.com/text-classification-using-algorithms-e4d50dcba45>.
15. —. Soul of the Machine: How Chatbots Work. *Medium*. [Online] [Cited: 11 01, 2018.] https://medium.com/@gk_/how-chat-bots-work-dfff656a35e2.

16. Jurafsky, Dan. Sentiment Analysis. *web.stanford.edu*. [Online] 08 07, 2015. [Cited: 11 23, 2018.] https://web.stanford.edu/~jurafsky/slp3/slides/7_Sent.pdf.
17. Noun Phrase Extraction. *Lexalytics*. [Online] [Cited: 01 03, 2019.] <https://www.lexalytics.com/noun-phrase-extraction>.
18. Antwerp, University of. Penn Treebank II tag set. *CLIPS*. [Online] [Cited: 12 05, 2018.] <https://www.clips.uantwerpen.be/pages/mb-sp-tags>.
19. Hamidani, Zaheen. Spotify Tracks DB. *Kaggle*. [Online] [Cited: 11 02, 2018.] <https://www.kaggle.com/zaheenhamidani/ultimate-spotify-tracks-db>.
20. Ramirez, Edwin. 19,000 Spotify Songs. *Kaggle*. [Online] [Cited: 11 02, 2018.] https://www.kaggle.com/edalrami/19000-spotify-songs#song_info.csv.
21. Spotify. Available genre seeds. *developer.spotify.com*. [Online] [Cited: 12 02, 2018.] <https://developer.spotify.com/documentation/web-api/reference/browse/get-recommendations/#available-genre-seeds>.
22. —. Categories. *developer.spotify.com*. [Online] [Cited: 11 07, 2018.] <https://developer.spotify.com/documentation/web-api/reference/browse/get-list-categories/>.
23. Wikipedia. List of emotions. *Wikipedia.org*. [Online] [Cited: 12 05, 2018.] https://simple.wikipedia.org/wiki/List_of_emotions.
24. Adel, Mohamed. Kaggle. *Hobbie*. [Online] [Cited: 11 05, 2018.] <https://www.kaggle.com/muhadel/hobbies>.
25. adalynrichards. Spotify's Search Feature is Subpar. *Spotify Community*. [Online] 07 27, 2018. [Cited: 01 05, 2019.] <https://community.spotify.com/t5/Other-Partners-Web-Player-etc/Spotify-s-Search-Feature-is-Subpar/td-p/4535415>.
26. Smith, Daniel. Some genres not recognized by Spotify Search API. *GitHub*. [Online] 01 24, 2017. [Cited: 01 05, 2019.] <https://github.com/spotify/web-api/issues/410>.
27. Durette, Pierre Nicolas. gTTS. *PyPi*. [Online] [Cited: 01 10, 2019.] <https://pypi.org/project/gTTS/>.
28. pygameci. pygame 1.9.5. *PyPi*. [Online] [Cited: 01 09, 2019.] <https://pypi.org/project/pygame/>.
29. (Uberi), Anthony Zhang. SpeechRecognition 3.8.1. *PyPi*. [Online] [Cited: 01 10, 2019.] <https://pypi.org/project/SpeechRecognition/>.
30. Ivanov, George-Bogdan. Compute sentence similarity using Wordnet. *NLP for Hackers*. [Online] 08 25, 2016. [Cited: 12 03, 2018.] <https://nlpforhackers.io/wordnet-sentence-similarity/>.
31. Rada Mihalcea, Courtney Corley, Carlo Strapparava. Corpus-based and Knowledge-based Measures of Text Semantic Similarity. *aaai.org*. [Online] [Cited: 12 03, 2018.] <https://www.aaai.org/Papers/AAAI/2006/AAAI06-123.pdf>.

32. user3898238. What do WordNet Similarity Scores mean? *Linguistics Beta - Stack Exchange*. [Online] 10 06, 2014. [Cited: 12 06, 2018.]
<https://linguistics.stackexchange.com/questions/9084/what-do-wordnetsimilarity-scores-mean>.

33. Dan Jurafsky, James H. Martin. *Speech and Language Processing, 2nd Edition*. s.l. : Prentice Hall, 2008.

Appendices

1. Interim Log

- Thursday 23rd Aug – Received Berant research paper from supervisor as material to inspire ideas for approaching project.
- Monday 17th Sep – Initial face-to-face meeting with supervisor to discuss project ideas and possibilities. Decided on final project idea.
- Monday 24th Sep – Received DBpedia resource link from supervisor for open-source datasets and APIs. Began collecting relevant project resources and trying out a few chatbot creating tutorials online.
- Tuesday 2nd Oct – First fortnightly group meeting with supervisor to present and share project ideas - was unable to attend. Began writing project proposal.
- Tues 16th Oct – 2nd group meeting with supervisor to explain and discuss some underlying technical theory related to project work. Realised that I need to find out what Spotify data is available to me through the open source API in order to better specify the domain for the bot and what will be achievable. Read up on the Spotify API functions and NLP tools available within TextBlob.
- Thursday 1st Nov – 3rd group meeting with supervisor to discuss project plan and schedule, and questions about interim report – was unable to attend. Began writing interim report.
- Tuesday 6th Nov – Meeting with supervisor to discuss ethical considerations and decide if Ethical Review is needed. Only Ethical Compliance Form necessary.
- Tuesday 20th Nov – 4th group meeting to give general progress update and discuss one particular component of our project currently being implemented. Got more ideas for potentially useful resources from peers, and gained inspiration from ideas being used by others in their own projects.
- Friday 1st Feb – Meeting with supervisor to review and sign Ethical Compliance Form.
- Friday 22nd Feb – Individual meeting with supervisor to discuss current progress made. Got advice for dealing with ambiguities of potential user input. Was advised not to try to perfect the system for every possible input as this would be impossible, but rather attempt to tackle a small selection of them in a reasonable way and discuss the rest in report.

2. Proposal Document

Final Year Project Proposal

Candidate no.: 146783

Supervisor: Dr. Julie Weeds

Title: Conversational UI – Q&A bot

Aim: This project aims to develop a question and answering bot which can handle answering simple natural language questions on the contents of a database. I may explore utilising Google's Dialogflow.

Motivations for undertaking this project: I was interested in the Question Answering project as I became interested in natural language processing after completing the NLE module in my 2nd year. Upon reading the description of the project, I believe it would be a good task to carry out in order to further develop my understanding of NLE. I did well in the module in my second year, and so I believe that whilst I have some background knowledge and capability, it would be a good challenge for me to implement such a bot effectively.

Primary objectives:

- answer simple questions on the contents of a database
- named entity recognition and linking
- paraphrase detection

Extensions:

- provide a suitable music/movie recommendation to a reasonable degree of accuracy

Relevance to degree: This project relates to my degree course as it deals with NLE which has become an increasingly growing topic of research and interest within the Computing sector.

Resources required:

- Python IDE
- NLTK
- Database containing dataset with information relating to domain of choice
- Tutorials

Project schedule:

- Gantt chart can be viewed here: <https://www.tomsplanner.com/public/fypscheduleoliviadamume>

Bibliography of background reading thus far:

1. Jonathan Berant, Percy Liang. *Semantic Parsing via Paraphrasing*. s.l. : Stanford University, 2014.
2. Nichol, Alan. Building Chatbots in Python. *DataCamp*. [Online] [Cited: 08 10 2018.] <https://www.datacamp.com/courses/building-chatbots-in-python>.
3. Daly, Liza. Chatbot Fundamentals. *World Writable*. [Online] [Cited: 08 10 2018.] <https://apps.worldwritable.com/tutorials/chatbot/>.
4. Paul, Moses Sam. How to build a chatbot with Dialog flow. *Medium*. [Online] 21 07 2018. [Cited: 08 10 2018.] <https://medium.com/swlh/how-to-build-a-chatbot-with-dialog-flow-chapter-1-introduction-ab880c3428b5>.

Interim Log:

- Thursday 23rd Aug – received Berant research paper from supervisor as material to inspire ideas for approaching project
- Monday 17th Sep – initial face-to-face meeting with supervisor, discussed project ideas and possibilities
- Monday 24th Sep – received DBpedia resource link from supervisor, for open-source datasets and API
- Tuesday 2nd Oct – first fortnightly group meeting with supervisor to present and share potential project ideas - was unable to attend

3. Ethical Compliance Form

Ethical Compliance Form for UG and PGT Projects*

School of Engineering and Informatics

University of Sussex

This form should be used in conjunction with the document entitled “Research Ethics Guidance for UG and PGT Projects”.

Prior to conducting your project, you and your supervisor will have discussed the ethical implications of your research. If it was determined that your proposed project would comply with all of the points in this form, then both you and your supervisor should complete and sign the form on page 3, and submit the signed copy with your final project report/dissertation.

If this is not the case, you should refer back to the “Research Ethics Guidance for UG and PGT Projects” document for further guidance.

1. Participants were not exposed to any risks greater than those encountered in their normal working life.

Investigators have a responsibility to protect participants from physical, mental and emotional harm during the investigation. The risk of harm must be no greater than in ordinary life. Areas of potential risk that require ethical approval include, but are not limited to, investigations that require participant mobility (e.g. walking, running, use of public transport), unusual or repetitive activity or movement, physical hazards or discomfort, emotional distress, use of sensory deprivation (e.g. ear plugs or blindfolds), sensitive topics (e.g. sexual activity, drug use, political behaviour, ethnicity) or those which might induce discomfort, stress or anxiety (e.g. violent video games), bright or flashing lights, loud or disorienting noises, smell, taste, vibration, or force feedback.

2. The study materials were paper-based, or comprised software running on standard hardware.

Participants should not be exposed to any risks associated with the use of non-standard equipment: anything other than pen-and-paper, standard PCs, mobile phones, and tablet computers is considered non-standard.

3. All participants explicitly stated that they agreed to take part, and that their data could be used in the project.

Participants cannot take part in the study without their knowledge or consent (i.e. no covert observation). Covert observation, deception or withholding information

*This checklist was originally developed by Professor Steven Brewster at the University of Glasgow, and modified by Dr Judith Good for use at the University of Sussex with his permission.

are deemed to be high risk and require ethical approval through the relevant C-REC.

If the results of the evaluation are likely to be used beyond the term of the project (for example, the software is to be deployed, the data is to be published or there are future secondary uses of the data), then it will be necessary to obtain signed consent from each participant. Otherwise, verbal consent is sufficient, and should be explicitly requested in the introductory script (see Appendix 1).

4. No incentives were offered to the participants.

The payment of participants must not be used to induce them to risk harm beyond that which they risk without payment in their normal lifestyle. People volunteering to participate in research may be compensated financially e.g. for reasonable travel expenses. Payments made to individuals must not be so large as to induce individuals to risk harm beyond that which they would usually undertake.

5. No information about the evaluation or materials was intentionally withheld from the participants.

Withholding information from participants or misleading them is unacceptable without justifiable reasons for doing so. Any projects requiring deception (for example, only telling participants of the true purpose of the study afterwards so as not to influence their behaviour) are deemed high risk and require approval from the relevant C-REC.

6. No participant was under the age of 18.

Any studies involving children or young people are deemed to be high risk and require ethical approval through the relevant C-REC.

7. No participant had a disability or impairment that may have limited their understanding or communication or capacity to consent.

Projects involving participants with disabilities are deemed to be high risk and require ethical approval from the relevant C-REC.

8. Neither I nor my supervisor are in a position of authority or influence over any of the participants.

A position of authority or influence over any participant must not be allowed to pressurise participants to take part in, or remain in, any study.

9. All participants were informed that they could withdraw at any time.

All participants have the right to withdraw at any time during the investigation. They should be told this in the introductory script (see Appendix 1).

10. All participants have been informed of my contact details, and the contact details of my supervisor.

All participants must be able to contact the investigator and/or the supervisor after the investigation. They should be given contact details for both student and supervisor as part of the debriefing.

11. The evaluation was described in detail with all of the participants at the beginning of the session, and participants were fully debriefed at the end of the session. All

participants were given the opportunity to ask questions at both the beginning and end of the session.

Participants must be provided with sufficient information prior to starting the session, and in the debriefing, to enable them to understand the nature of the investigation.

12. All the data collected from the participants is stored securely, and in an anonymous form.

All participant data (hard-copy and soft-copy) should be stored securely (i.e. locked filing cabinets for hard copy, password protected computer for electronic data), and in an anonymised form.

Project title: _____

Student's Name: _____

Student's Registration Number: _____

Student's Signature: _____

Date: _____

Supervisor's Name: _____

Supervisor's Signature: _____

Date: _____

4. Sample Questions for potential user input

Alessia:

1. What is the first album of artist X?
2. What's the top-rated song by artist X?
3. Can you tell me all the covers of song X?
4. What songs have X in the title?
5. What songs has artist X released between A and B date?

Akin

1. Give me 5 songs like song X
2. Get me a playlist for mood X (e.g. cooking, smoking, working out)
3. Can you give me some albums of genre X from A period to B period?
4. When is artist X's new album scheduled to be released?
5. What are the latest songs in X genre?

Sofia:

1. Can you give me some song suggestions for genre X?
2. What are the most popular songs at the moment?
3. Songs from X period (90s, etc)?
4. What is artist X's most recent song?
5. Do you have any songs like song X?

Govind:

1. Give me playlists with artists X, Y, Z
2. Suggest artists in genre X
3. Give me playlists for genre X
4. Suggest songs for mood X
5. Give me a playlist for mood X

5. Sample Questions categorised by intent

Time period:

What is the first album of artist X?

What songs has artist X released between A and B date?

Can you give me some albums of genre X from A period to B period?

What are the latest songs in X genre?

Songs from X period (90s, etc)?

What is artist X's most recent song?

Keywords: “most recent”, “latest”, “first”, “from...to”, “between...and...”

Similar to:

Give me 5 songs like song X

Do you have any songs like song X?

Keywords: “like”

Genre-specific:

Can you give me some song suggestions for genre X?

Suggest artists in genre X

Give me a playlists for genre X

Keywords: “for X genre”, “suggest...genre”

Mood-specific:

Get me a playlist for mood X (e.g. cooking, smoking, working out)

Suggest songs for mood X

Give me a playlist for mood X

Keywords: “for X mood”, “suggest...mood”

Popularity:

What's the top-rated song by artist X (top-rated at the moment)?

What are the most popular songs at the moment?

Keywords: “top-rated”, “most popular”

Extras (the bot is unlikely to make attempt to handle / address these):

Give me playlists with artists X, Y, Z

What songs have X in the title?

Can you tell me all the covers of song X?

When is artist X's new album scheduled to be released?

What are the most popular artists in the UK right now?

Bot “vocabulary” keywords:

["song", "sing", "by", "style", "artist", "music", "recommend", "suggest", "playlist", "tune", "listen", "album", "track", "play", "songs", "sings", "styles", "singer", "artists", "recommendations", "suggestions", "playlists", "tunes", "albums", "tracks", "something"]

6. Evaluation of TextBlob extractors

Red = Conll extractor 13/37 correct

Blue = Fast NP extractor 19/37 correct

What is Kendrick's first album?

kendrick, first album

kendrick

What songs did Beyonce release between 2016 and 2017?

beyonce release

beyonce

Can you give me some pop albums from 2000-2010?

-

-

What are the latest rock songs?

-

rock songs

Can you suggest some hiphop songs from the 90s?

-

hiphop songs

What is Cardi B's most recent song?

cardi b, most recent song

cardi, b 's, recent song

What are some cool 80s songs?

-

cool 80s songs

What is Solange's newest single?

solange

solange

What is James Morrison's oldest album?

james morrison

james morrison

Give me some songs like Thotiana

thotiana

thotiana

Do you know any songs like Swish by Tyga?

swish, tyga

swish, tyga

What is a song like Yellow Submarine?

yellow submarine

yellow submarine

Give me some soul album recommendations

soul album recommendations

soul album recommendations

Can you suggest some jazz songs to me?

-

jazz songs

Can you give me some rap song suggestions?

rap song suggestions

rap song suggestions

Suggest some rnb artists

-

suggest, rnb artists

Give me a pop playlist

pop playlist

-

Get me a playlist for cooking

-

-

Can you recommend some workout songs to me?

-

workout songs

Suggest a playlist for running

-

suggest

Give me a romance playlist

romance playlist

romance playlist

Recommend some happy songs?

-

recommend, happy songs

Do you have any good songs for jogging?

-

good songs

Do you know any good love songs?

good love songs

-

What's the top-rated song by Solange?

top-rated song, solange

solange

What are the most popular songs at the moment?

-

popular songs

What's Ariana's most popular song?

ariana, most popular song

ariana, popular song

Who sings Blue Jeans?

blue jeans

who, jeans

Whose album is DAMN?

-

whose, damn

Yellow Submarine singer?

yellow submarine singer

yellow submarine

Which artist sings Mad World?

mad world

artist sings, mad

Which Kendrick album is Loyalty on?

kendrick album

kendrick, loyalty

What album is Be Careful on?

-

be careful

What album Yellow Submarine by the Beatles from?

what album yellow submarine

yellow submarine, beatles

Which one of SZA's albums is Supermodel off?

supermodel

sza, 's albums, supermodel

What tracks are on Invasion of Privacy by Cardi B?

invasion, cardi b

invasion, privacy, cardi

Which songs are on Beyonce's B'Day album?

beyonce, ' day album

beyonce, b'day

7. Comparison of Naïve Bayes and Semantic Similarity

**Have put the sentence retrieved from Intent Classifier dataset as “most similar” by SS method in intent classification result line

NB: Naïve Bayes 7/15 correct
simple qs = 5/7, complex qs = 2/7

SS: Semantic Similarity 5/15 correct
simple qs = 4/5, complex qs = 1/5

Q: I don't want general Busta Rhymes music. What is his most popular music?

NB: time period

SS: popularity – “What is Ariana Grande's most popular music?”

Q: What is some generic Ariana music?

NB: time period

SS: time period – “What is Solange's newest single?”

Q: Do you know what song was on Cardi's album 'Invasion of Privacy'?

NB: track info

SS: track info – “What tracks are on Invasion of Privacy by Cardi B?”

Q: I really like love songs. What are the most popular love songs?

NB: popularity

SS: time period – “What is Solange's newest single?”

Q: What album is the newest of Kendrick Lamar?

NB: popularity

SS: time period – “What is Kendrick's first album?”

Q: I'm in the mood to listen to some music. Do you have anything from the 00s?

NB: type of music

SS: type of music – “Do you have any nice rock tunes I can listen to?”

Q: What are the newest Beyonce songs? Wait, actually, what are the most popular Beyonce songs?

NB: popularity

SS: time period – “What is Solange's newest single?”

Q: What are some good r-n-b songs?

NB: type of music

SS: time period – “What are the latest rock songs?”

Q: Do you know what Michael Jackson's first album was?

NB: type of music

SS: time period – “What is Solange's newest single?”

Q: Bang is my favourite Ariana Grande song. Which album is it on?

NB: popularity

SS: time period – “What is Solange's newest single?”

Q: Hmm, do you know any songs by The 1975?

NB: type of music

SS: similar to – “Do you know any songs like Swish by Tyga?”

Q: I want to listen to something but not just general songs, do you have any songs by artists like Bill Withers?

NB: type of music

SS: time period – “I want to listen to a new pop song.”

Q: What is the newest song by James Morrison?

NB: time period

SS: time period – “What is Solange’s newest single?”

Q: It is my desire to listen to songs by the artist Bad Bunny

NB: time period

SS: time period – “What is Solange’s newest single?”

Q: I’ve been listening to a lot of pop songs recently, do you know any from the 80s?

NB: type of music

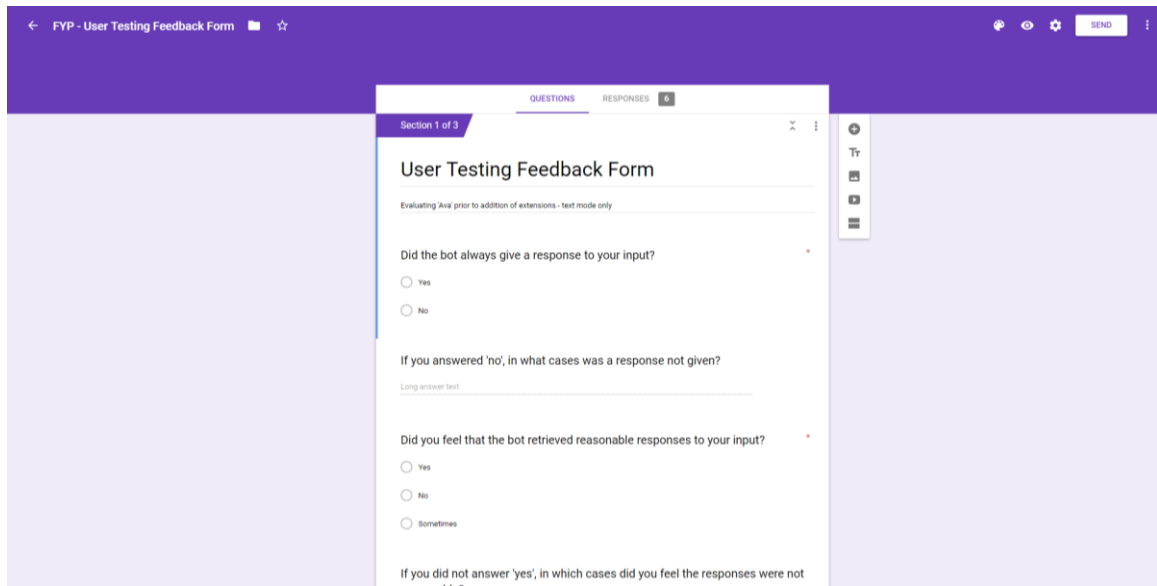
SS: type of music – “I want to listen to a new pop song?”

Q: Do you know any cool dance songs?

NB: type of music

SS: similar to – “Do you know any songs like Swish by Tyga?”

8. User Evaluation Questionnaire



The screenshot shows a Google Forms interface with a purple header. The form is titled "User Testing Feedback Form" and is in "Section 1 of 3". The form is currently in "Evaluating" mode, as indicated by the text "Evaluating Awa! prior to addition of extensions - test mode only". The form contains three questions:

- Question 1: "Did the bot always give a response to your input?" with radio button options "Yes" and "No".
- Question 2: "If you answered 'no', in what cases was a response not given?" with a "Long answer text" input field.
- Question 3: "Did you feel that the bot retrieved reasonable responses to your input?" with radio button options "Yes", "No", and "Sometimes".

Below the third question, there is a partially visible question: "If you did not answer 'yes', in which cases did you feel the responses were not". The form has a "SEND" button in the top right corner and a "QUESTIONS" tab selected in the top navigation bar.

The full feedback form can be viewed at:

https://docs.google.com/forms/d/1fN0HaBXy_EwdNjIAgy41CZouzQYxc4ZYgFwvom8Hgxm/edit?usp=sharing