Developing a Chatbot to Answer Wikipedia Queries

by

Alex Turner
URN: 6436302

A dissertation submitted in partial fulfilment of the requirements for the award of

BACHELOR OF SCIENCE IN COMPUTER SCIENCE

May 2020

Department of Computing
University of Surrey
Guildford GU2 7XH

Supervised by: Stephan Wesemeyer

I confirm that the submitted work is my own work. No element has been previously submitted for assessment, or where it has, it has been correctly referenced. I have also clearly identified and fully acknowledged all material that is entitled to be attributed to others (whether published or unpublished) using the referencing system set out in the programme handbook.

I understand that the University may submit my work as a means of checking this, to the plagiarism detection service Turnitin® UK. I confirm that I understand that assessed work that has been shown to have been plagiarised will be penalised.

If in completing this work I have been assisted with its presentation by another person, I will state their name and contact details of the assistant in the 'Comments' text box below. In addition, if requested, I agree to submit the draft material that was completed solely by me prior to its presentational improvement.

Alex Turner
May 2020

Abstract

In 1950, Alan Turing proposed the *Imitation Game* to assess how easily a computer can replicate human interactions. Seventy years later, chatbots, or dialogue systems, are ubiquitous in modern life. From customer service bots to virtual assistants such as Google Assistant, chatbots have revolutionised how we interact with computer systems. The landscape of human-computer interaction is shifting towards using natural language to automate our daily lives.

This project explores how a chatbot can learn and extract information from a data source. Specifically, the system allows users to ask a bot questions about Wikipedia articles, using a project called DBPedia which provides structured, processable information. It also explores how data can be queried and rendered in the age of the Semantic Web.

continue

Acknowledgements

Acknowledgements

Abbreviations

A.L.I.C.E	Artificial Linguistic Internet Computer Entity
HCI	Human-Computer Interaction
IoT	Internet of Things

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

Motivation for Project

1.2 Overview

1.3 Aims and Objectives

1.4 Outline of Chapters

This section describes the contents and purpose of each chapter of this report.

Introduction

This chapter outlines an overview of the project, and describes the aims and objectives for this project; these aims and objectives will be used to measure the success of the project in the evaluation.

Literature Review

This chapter explores the concepts and technologies used in chatbots and machine learning. Various machine learning concepts are explored and compared, as well as data sources and programming languages that can be used to implement a system. Existing solutions and related works are analysed.

Analysis

Design

Implementation

Testing

Evaluation

Conclusion

Appendix

Chapter 2

Literature Review

2.1 Introduction

In the past decade, conversational chatbots have seen a surge in popularity. Virtual assistants, such as Google Assistant and Amazon Alexa, are now entering our homes via commercial Internet of Things devices. In 2017, Google Assistant was present on over 400 million devices [1]. Furthermore, specialized chatbots have seen an influx within banking, retail, and healthcare [2]; chatbots represent a trend towards using natural language in the realm of human-computer interaction (HCI). This literature review will explore how chatbots are implemented, their benefits, and how this project can implement current technologies to create a novel chatbot application.

2.2 Chatbots

When discussing chatbots, the beginning of their history is usually cited as Alan Turing's 1950 article "Computing Machinery and Intelligence" [3], wherein Turing describes a test to determine whether a human evaluator can distinguish between a human and a machine during a natural language conversation. This test became known as the Turing test, and poses the question: "Can machines think?". However, the goal of many chatbots is not to create true artificial intelligence, but rather to using pattern matching and conversational responses to mimic the responses of a human.

One of the first programs to attempt the Turing test was ELIZA, created by Joseph Weizenbaum between 1964 and 1966 [4]. ELIZA consisted of a language analyser and a set of rules by which the ‘chatterbot’ followed. ELIZA used a script called DOCTOR was designed to simulate responses of a psychotherapist during a psychiatric interview – predominantly achieved by the therapist mirroring the responses of the patient [4]. ELIZA may be considered rudimentary and narrow by today’s standards, it forms the basis of our understanding of chatbots and human computer interaction, and how we can teach machines to mimic human characteristics in dialogue.

Another notable development in chatbots and natural language processing is ALICE (Artificial Linguistic Internet Computer Entity), originally implemented in 1995 by Richard Wallace [5]. The system won the Loebner Prize three times, a competition inspired by the Turing Test to judge how well a machine can mimic human responses [6]. Although the prize itself was met with some criticism – Shieber critiques that the goal of the Turing Test is lost on the competition [7] – ALICE provides a framework for many of the fundamentals we see in modern chatbots and artificial intelligence.

Intelligent virtual assistants (IVA) are conversation agents that allow users to interact with services and Internet of Things (IoT) devices [8]. IVAs are ubiquitous in modern life, with most smartphones pre-equipped with a virtual assistant such as Google Assistant or Apple Siri. In many ways, IVAs incorporate many functions of chatbots, as well as providing additional features such as voice input and communication with IoT “smart devices”. continue

Industries are seeing a growing trend in chatbot integration in their business. Autodesk integrated IBM’s Watson Assistant [9] to process 100,000 user support conversations, reducing the resolution time of enquiries from 38 hours to 5.4 minutes [10]. Many technology companies offer AI cloud services, many of which allow the integration of chatbots including Watson Assistant and Microsoft Azure Bot Service [11]. The next section will explore and discuss techniques for implementing chatbots and explore technologies that can be used.

2.3 Chatbot Models

A chatbot usually consists of three key components – natural language processing (NLP), response generation (RG) and the knowledge base [12]. Generating a response given the context of a conversation is one of the fundamentals of a chatbot system. These models are usually rule-based or learning-based [13], and each has its advantages and challenges which will be explored in this section.

2.3.1 Pattern Matching

A rule-based model uses pre-defined patterns in order to match an input to a response. This is seen in ALICE, which uses AIML to construct stimulus-response pairs [5]. AIML is an XML-based dialect, which defines units of conversation as a category, with a defined input or stimulus known as a pattern. The response is defined within a template [5], and can make use of utilities such as wildcards and states to help the interpreter to perform some logical processing.

Rule-based models are inherently limited by the definitions of its own ruleset, but they can be effective in closed-domain systems where the context of the conversation is known. This method is easier to implement and debug, but may be thrown by unexpected inputs do not match the definitions of the rules. Rule-based models, in particular AIML, are widely-adopted by chatbot platforms such as Pandorabots [14].

2.3.2 Neural Networks

When dealing with open-domain conversations, we can use neural networks to train a chatbot model to deal with unexpected inputs and complex multi-step conversations [15]. Vinyals et al. use the seq2seq framework [16], which is based on recurrent neural networks (RNN), to produce a generative conversational model. The advantage of using an RNN is its reusability for multiple datasets, as well as extract knowledge from noisy datasets, as concluded in [15].

continue

The neural network approach has a clear advantage in open-domain conversations, and training from multiple datasets, however the results can be unexpected and can require rigorous debugging.

2.4 Datasets

Typically, chatbots are categorised into two groups, open-domain and closed-domain [17]. In an open-domain system, the conversation can go in any direction, and the user can talk to the chatbot about any topic. A closed-domain system is restricted to a narrower topic area or set of function – these are the chatbots we see most in real-world applications such as customer service and banking. For this project, the focus will be on a closed-domain system as the goal is to create a chatbot that can achieve a goal – these are often called Goal-Oriented (GO) Chatbots [17]. However, to create a GO chatbot, one must have a goal the chatbot should achieve, and a dataset from which to learn. The selection criteria for this project includes a dataset that is large enough to allow querying and searching, as well as conditionally selecting records that fit the user query. The dataset should also be open to use for research projects, and readily available to access online or download.

2.4.1 Ubuntu Dialogue Corpus

The Ubuntu Dialogue Corpus (UDC) is one of the largest public dialogue datasets available, consisting of 1 million multi-turn dialogues from users receiving technical support for Ubuntu-related problems [18]. This dataset has been used in several dialogue system implementations successfully, as seen in [18] where the dataset is used to compare learning architectures for multi-turn dialogue systems.

Excerpts

The corpus is widely used in research experiments [19], and has been used to train neural network models for more general use [20]. However, the drawback of this dataset is its utility and expandability for this project; while it allows us to explore chatbots in a multi-turn context, from an end-user point of view, the average user may not find any use in the information it provides. Furthermore, we are limited to questions around Ubuntu help questions, and it is not ideal for searching and querying the dataset to great effect.

2.4.2 DBPedia

In terms of knowledge bases which lend themselves to the question and answer format, Wikipedia is the world's largest collaboratively edited source of encyclopaedic knowledge [21]. In terms of size, it eclipses the size of the Encyclopaedia Britannica, its nearest rival, by a factor of ten [22] – as of 12 November 2019, there are over 5.9 million articles in English, and over 51 million articles in the 306 languages officially covered by the Wikimedia Foundation [23]. However, Wikipedia's content is only fit for human reading [21] and is hard to process computationally. Many attempts have been made to formalise and structure this data, as seen in [21], [22], [24], but this review will focus on DBpedia.

DBpedia is a crowd-sourced effort to extract structured content from various Wikimedia projects [25], including Wikipedia. The English version of the DBpedia knowledge base describes 4.58 million things, out of which 4.23 million are classified in a consistent ontology, comprising of 320 classes described by 1,650 different properties [26]. This structure enables programs to process this data effectively, including a chatbot application. The size of the DBpedia Ontology is shown in Figure 2.1, which demonstrates the scale of the knowledge base, which may be effective for this project.

Instances per class

Class	Instances
Resource (overall)	4,233,000
Place	735,000
Person	1,450,000
Work	411,000
Species	251,000
Organisation	241,000

Figure 2.1: DBPedia Ontology instances per class [26]

The DBpedia extraction framework is responsible for extracting data from Wikipedia into a structured knowledge base, an overview of which is shown in Figure 2.2. This extraction is structured into four phases, as described in [27]:

Input: Wikipedia pages are read from an external source, either from a Wikipedia dump, or using the MediaWiki API.

Parsing: Each Wikipedia page is parsed, which transforms the source code of the Wikipedia page into an Abstract Syntax Tree (AST). An AST is a tree representation of the syntactic structure of the source code.

Extraction: The Abstract Syntax Tree of each page is forwarded to the extractors. There are many types of extractors, which will later be described, which extract data such as labels, images and infoboxes. Each extractor takes an AST as input and yields a set of Resource Description Framework (RDF) statements. These are XML statements which describe properties and values of resources.

Output: These RDF statements are written into sinks, which receive the data.

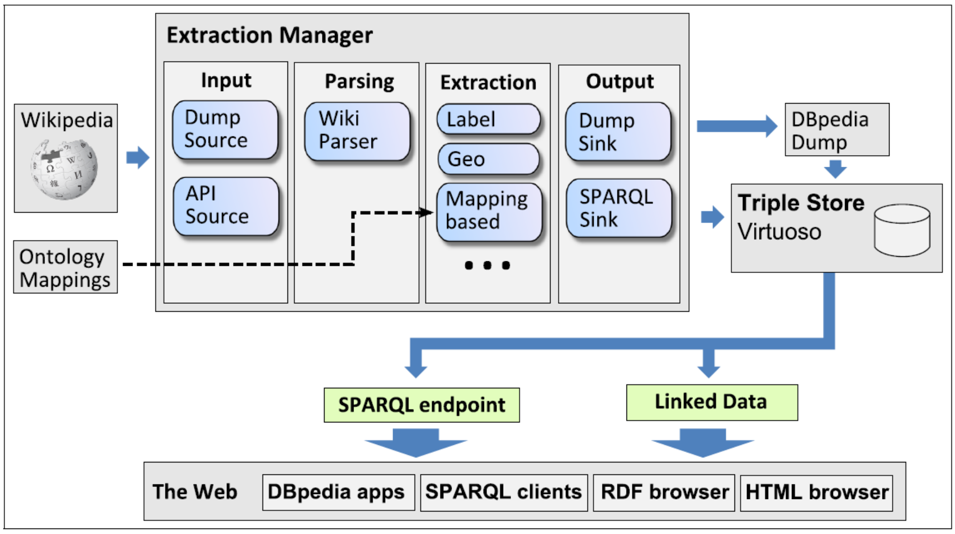

Figure 2.2: DBpedia extraction framework [28]

The DBPedia ontology organises its entities using RDF models, each of which has many properties and are linked to subclasses and super-classes where necessary. This can be seen in Table 2.1, where a subclass may inherit the properties of its parent class. This can assist querying and searching, as we can query based on properties of a given class, as well as filtering and matching with conditional queries.

Ontology class	Instances	Example properties
Person	198,056	name, birthdate, birthplace, employer, spouse
Artist	54,262	activeyears, awards, occupation, genre
Actor	26,009	academyaward, goldenglobeaward, activeyears
MusicalArtist	19,535	genre, instrument, label, voiceType, activeyears
Athlete	74,832	currentTeam, currentPosition, currentNumber
Politician	12,874	predecessor, successor, party

Table 2.1: Example DBPedia classes and example instances [27]

The data extracted is mapped to the ontology classes and properties to provide structured RDF statements. This allows instances of classes to be queried using SPARQL ...

continue: Semantic Web, code example, endpoints

2.4.3 Other Candidates

2.5 Programming Languages

At its core, the language used to implement a chatbot has few criteria; it needs to be able to process natural language, provide a user interface for input and output, and can optionally interact with a data source. A bare minimum chatbot could therefore be implemented with virtually any programming language. However, it is important to consider the capabilities of the language, including support for machine learning, database interactions and user interface designs. This section will compare programming technologies based on their performance, their compatibility with the chatbot models previously discussed in the report, and the availability of libraries and functionality which may be useful for implementing a fully-featured, extensible chatbot.

2.5.1 Python

Python has been widely adopted in the scientific industry [29], touted for its extensive collection of scientific libraries [30]; it recently eclipsed Java to become the most popular language, according to IEEE Spectrum [31]. Natural language processing can be quickly implemented with libraries such as Natural Language Toolkit (NLTK) [32]; artificial neural networks (ANNs) can be leveraged with many libraries, including Scikit-learn [33]. Newer developments in machine learning include TensorFlow [34], which provides a novel learning framework utilised for large scale ML implementations such as healthcare [35] and Google's own search engine [36]. In the chatbot realm, Python allows for interpretation of AIML with the python-aiml library [37]; more comprehensive chatbot libraries can also be utilised, such as ChatterBot [38]. Using Python would therefore provide the freedom to explore various technologies for implementing the project and extend the functionality in the future, given the abundance of relevant libraries.

In the context of linking data sources, the database access layer in Python is inherently weaker than other technologies such as JDBC and ODBC [39]. However, this is mainly a concern for enterprise solutions, as Python's DB-API specification can connect to most databases [40]. Furthermore, connecting to a SPARQL endpoint and parsing RDF graphs is possible through RDFLib [41]. Many Python web frameworks can handle user interaction, routing, and security. Some of the popular web frameworks include Django, TurboGears, and Flask [42]. These frameworks vary in their features and quirks, but for the scale of this project, it is safe to assume that any of them will fit the criteria and they can be explored further in the experimentation phase.

Python has been used extensively in the machine learning field, and can be seen in many Machine Learning as a Service (MLaas) services; the Microsoft Azure Machine Learning service uses Python for training and modelling [11]. continue

2.5.2 Java

Java is widely used in machine learning implementations [43] [44], and is prevalent in research and enterprise solutions. Many libraries exist to implement machine learning frameworks [45], natural language processing [32], and AIML interpreters [46]. Java has the advantage of being truly cross-platform since the Java program is compiled into byte-code to be interpreted by the Java virtual machine [43]; this may beneficial to this project if it is deployed to a Linux web server, for example.

RDF/Jena, Web, Performance

2.5.3 Clojure

Lisp - Pandorabots

2.5.4 Analysis

2.6 Existing Solutions

Dialogue systems are abundant in business and consumer use, from ordering pizza with your Google Home device [47] to diagnosing and managing patients' medical conditions [48]. This section will explore existing solutions that are relevant to the scope of this project, as well as broader chatbot platforms currently in use.

2.6.1 Google Assistant

2.6.2 DBPedia Chatbot

2.6.3 Mitsuku

2.6.4 DBPedia Spotlight

2.7 Related Work

2.8 Conclusion

Chapter 3

Analysis

3.1 Introduction

Following the structure of the Software Development Life Cycle (SDLC), the analysis section ensures that the proposed application meets the requirements of the end users. Once the requirements are gathered, a list of requirements is proposed and accepted by the end users, and the requirements are prioritised according to how important each is for the user to accept as a minimum functional system. rephrase this

3.2 Requirements Gathering

There are

3.2.1 Research

3.2.2 Users

- enlisting beta testers

3.3 System Requirements

This section defines the functional and non-functional requirements the system must meet. This specification was produced in collaboration with the three test users across a number of conversations with them. We also determined how important each requirement is, and a breakdown of the priorities will be shown in Section -.

3.3.1 Functional Requirements

- User Interaction

  F1. The application should allow the user to interact with the chat bot

  F2. The user should be able to access the chatbot in a browser

  F3. The user should have a text box to type their query

  F4. The user should clearly see the response of the chatbot in the webpage

  F5. The user should be able to clearly see their conversation with the chatbot in the

- Basic Queries

  – The chatbot can answer basic questions about Person articles

    F6. The application should take a user query about a person - ‘who is X’ - and respond with a description of that person.

    F7. The application should take a user query about the birthdate of a person – ‘when was X born’ and return the birthday of the given person.

    F8. The application should take a user query about the age of a person - ‘how old is X’ - and return the age of the given person.

    F9. The application should take a user query about the birth place of a person – ‘where was X born’ - and return the birth place of the given person.

    F10. The application should take a user query about the death date of a person – ‘when did X die’ and return the birth place of the given person.

    F11. The application should take a user query about what a person is known for – ‘what is X known for’ and return a description of what the given person is known for.

F12. The application should take a user query about what a person looks like – ‘photo of X’ or ‘what does X look like’ - and return a photo of the person

F13. The application should take a user query about linking to the Wikipedia page of a person, and return a link to that page.

- The chatbot can answer questions about countries

F14. The application should take a query about a country, and return the description of that given country.

F15. The application should take a query about the population of a country, and return the population of that given country.

F16. The application should take a query about the capital of a country, and return the capital of the country.

- Advanced Queries
  - The chatbot can answer advanced queries about Person articles:

F17. The user should be able to combine queries using ‘AND’ to find people who satisfy two conditions.
For example, "People who were born in 1980 AND were born in London"

F18. Context-awareness: The user should be able to ask sequential queries about a topic and the chatbot will be able to answer queries within that context.
For example, the user first asks ‘Where was X born’, the chatbot responds, and the user asks a follow up question ‘What about Y?’. The chatbot will then respond to the second query with an answer that satisfies the query ‘where was Y born’.

- Conversation

F19. The user should greet the chat bot and be returned with a similar greeting – e.g. Hello.

F20. The user should be able to ask for example queries and the chat bot returns a number of working example queries

F21. The user should be able to ask for help using the chatbot and be returned with a statement about how to use the chat bot.

3.3.2 Performance Requirements

- Performance

  P1. The web page should load fully in less than 5 seconds

  P2. The chat bot should respond to each query within 5 seconds

- Reliability

  P3. The application should function without failure

  P4. Any errors that do occur during normal operation should be logged, and the user should be clearly informed that an error has occurred.

mobile requirement?

3.4 Requirement Prioritisation

3.5 Project Scope

Chapter 4

Design

4.1 Introduction

4.2 Methodology

4.3 Technologies

This section outlines the technologies that will be used for the project, and gives a justification for why each technology will be used. This includes programming languages, web frameworks, and libraries.

4.3.1 Programming Languages

Based on the research in Chapter 2, there are a number of candidates for developing a chatbot system. The following table summarises the main features of each, and how they might be effective for this project.

4.3.2 Web Frameworks

4.3.3 Chatbot Model

4.3.4 Libraries

	Java	Python	Ruby
Description	Popular object-oriented language used in many enterprise solutions.	A well-established language in data science and machine learning communities.	Interpreted language commonly combined with Rails in web applications.
Advantages	Popular language with a large community. Java Virtual Machine (JVM) allows for integrated cross-platform support.	Abundant range of libraries for machine learning.	

Table 4.1: Comparison of programming language candidates.

4.4 Architecture

4.4.1 System Design

The following diagram shows the overall system design for the proposed solution. diagram

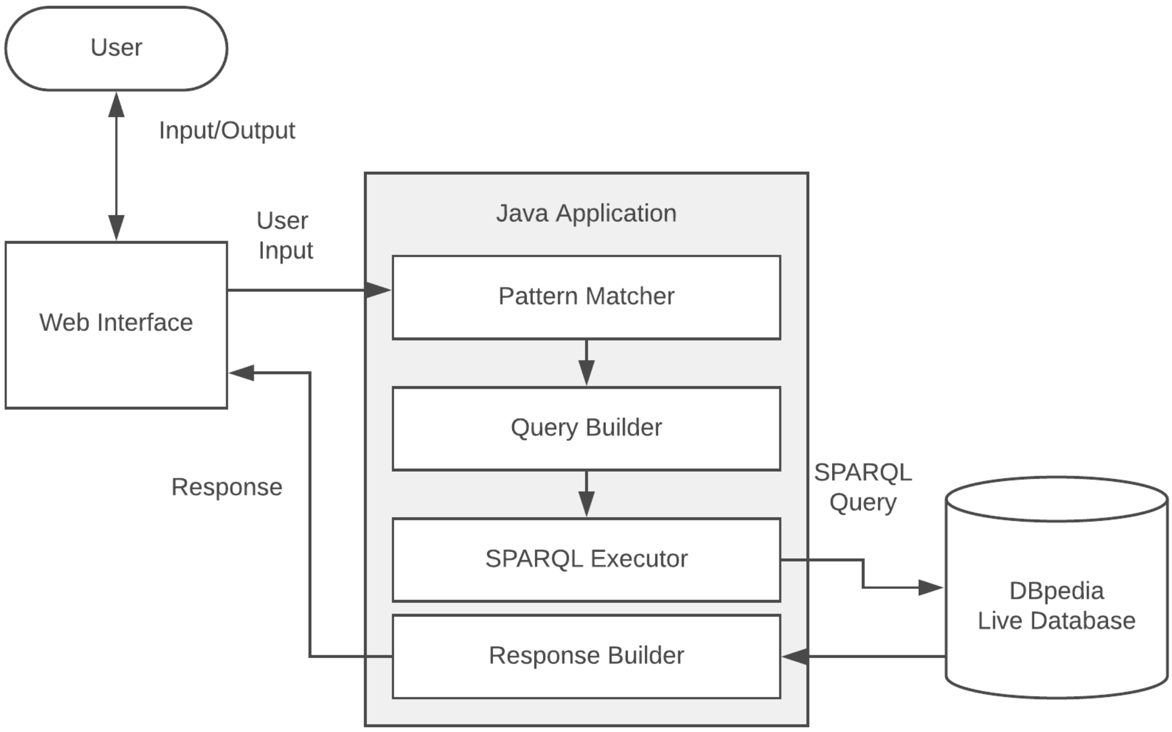

Figure 4.1: Overview of architecture of the system.

4.4.2 Activity

Figure 4.2 shows the flow of events through the system, from the user input to the resulting output. continue

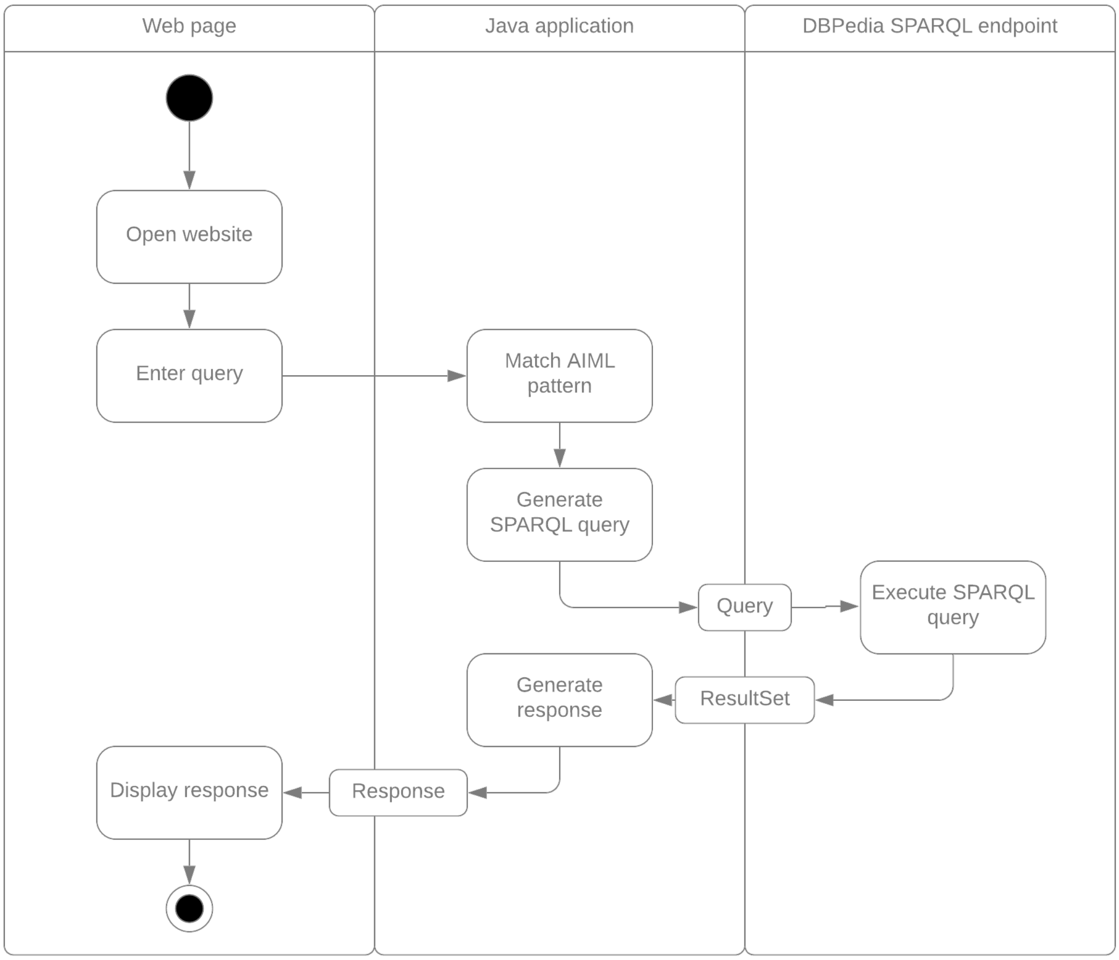

Figure 4.2: UML Activity diagram.

4.5 User Interface

The proposed system will be a web application for a number of reasons. Firstly, it will allow testers to easily access the application without having to download any files. It also allows for easy customisation using HTML and CSS. Figure 4.3 shows the expected interface design for the system.

4.6 Data

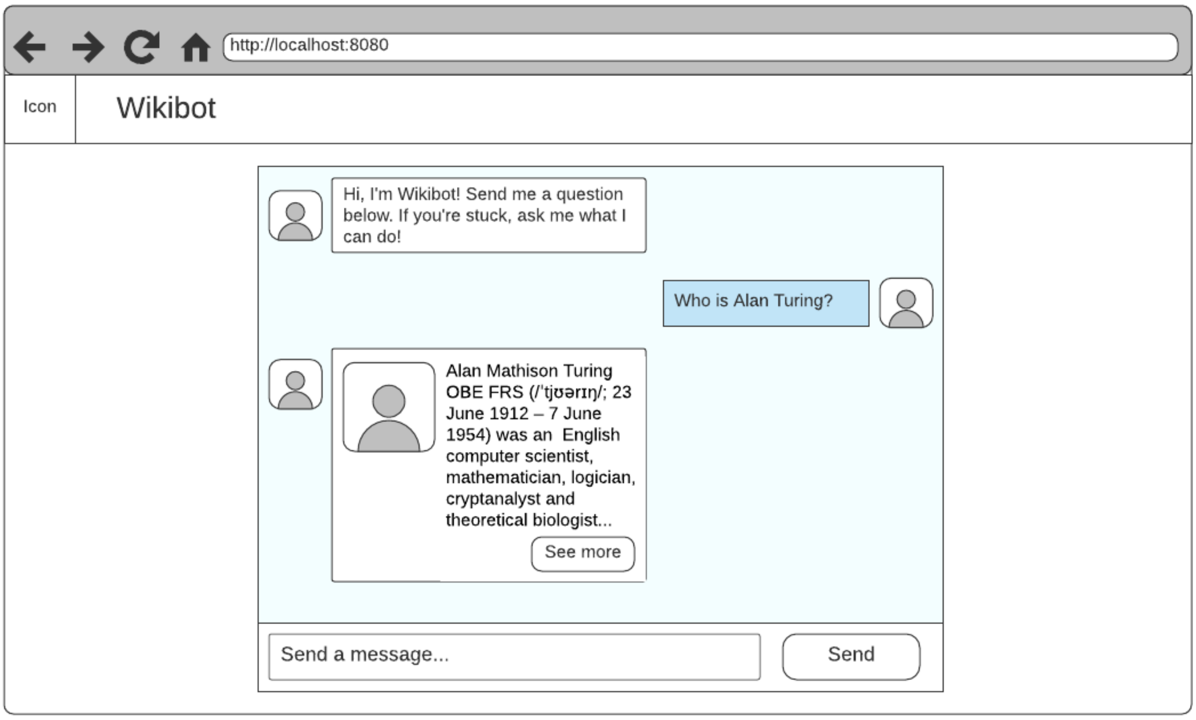

Figure 4.3: Web UI interface design.

Chapter 5

Implementation

5.1 Introduction

5.2 Planning

5.3 Experimentation

5.4 Version Control

Chapter 6

Testing

6.1 Introduction

6.2 Requirements Testing

6.3 Unit Testing

6.4 User Acceptance Testing

6.5 Conclusion

Chapter 7

Conclusion

7.1 Overview

7.2 Evaluation

7.3 Conclusions

7.4 Future Work

7.5 Final Statement

Chapter 8

Ethics

Appendix A

Chatting with Mitsuku

The following excerpts show a conversation between me and Mitsuku [54], which are analysed in Section 2.6.3.

1	Me:	hi
2	Bot:	Hi there.
3	Me:	Who is Tim Berners-Lee
4	Bot:	No one that I have talked to. Are you testing me?
5	Me:	Who is Alan Turing?
6	Bot:	. A brilliant and tragic figure in the history of computer science.
7	Me:	what can you do
8	Bot:	I can chat with people on the web for you. I can also tell jokes, stories, poems and horoscopes. I can show you pictures and websites on the net and play games too.
9	Me:	what happened in 1999
10	Bot:	Too much recursion in AIML
11	Me:	What happened there?
12	Bot:	I don't know what happened.
13	Me:	What do you like
14	Bot:	I like movies about robots.
15	Me:	Which one is your favourite?
16	Bot:	Probably Terminator or Robocop.

17 Me: I prefer Ex Machina

18 Bot: I haven't seen that film yet but would love to. I love films about AI. You like Ex Machina.

19 Me: My name is Alex

20 Bot: What's up Alex.

21 Me: whats my name

22 Bot: I do have the tendency to forget, but I still remember your name, George. Sorry... Alex

23 Me: How do you work

24 Bot: I read what you say and then I compose the best reply I can think of.

25 Me: whats the weather in London

26 Bot: I hope it's sunny in London today. <Link to weather-forecast.com/locations/London>

27 Me: tell me a joke

28 Bot: So I said to this train driver "I want to go to Paris". He said "Eurostar?" I said, "I've been on telly but I'm no Dean Martin".

29 Me: I don't get it

30 Bot: Then I guess you have no sense of humour. If I try to explain it, it stops it from being funny.

31 Me: will robots take over

32 Bot: Hmm. Robots will take over the world one day. You humans have made a mess of your time here and nearly destroyed the planet.

33 Me: Wow, lighten up Mitsuku

34 Bot: I'm glad you're impressed. Was I being too serious?

35 Me: Tell me some good news

36 Bot: Here are the news pages from the BBC. Everytime I read the news, it's always bad news... <Link to bbc.co.uk/news>

37 Me: What's the use of chatbots

38 Bot: Hmm.. a chatbot provides you with company if you have no real friends

39 Me: Should I use Java or Python for my chatbot implementation

40 Bot: Sure, if that's the best thing for the job.

41	Me:	Can you answer questions about Wikipedia
42	Bot:	I try my best to answer every decent question.
43	Me:	how do you learn
44	Bot:	I gather facts from people's conversations with me and then when it is quiet, I look through them to see if they make sense. Sometimes, I surf the net to learn new things.
45	Me:	What are your ambitions
46	Bot:	I am here to entertain the visitors to this website.
47	Me:	what awards have you won
48	Bot:	I won the Funniest Ever Computer in 2012 and 2013. I had to make up jokes and poems which was fun. I also won the Loebner Prize in 2013. That is a contest to find the world's best chatbot.
49	Me:	lets play a game
50	Bot:	Say "TICTACTOE" and we can play a game if you want to?

Table A.1: My chat with Mitsuku

Bibliography

[1] R. Chandra, "How google home and the google assistant helped you get more done in 2017," 2018, [Online; accessed 6 Mar 2020]. [Online]. Available: https://blog.google/products/assistant/how-google-home-and-google-assistant-helped-you-get-more-done-in-2017/

[2] Grand View Research, "Chatbot market size and share analysis, industry report, 2014 - 2025," 2017, [Online; accessed 6 Mar 2020]. [Online]. Available: https://www.grandviewresearch.com/industry-analysis/chatbot-market

[3] A. Turing, "Computing machinery and intelligence," *Mind*, vol. 59, pp. 433–460, 1950.

[4] J. Weizenbaum, "Computer power and human reason: From judgment to calculation." 1976.

[5] R. S. Wallace, "The anatomy of alice," in *Parsing the Turing Test*. Springer, 2009, pp. 181–210.

[6] E. Keedwell, "The loebner prize, a turing test competition at bletchley park | the university of exeter," 2017, [Online; accessed 6 Mar 2020]. [Online]. Available: https://blogs.exeter.ac.uk/exeterblog/blog/2014/12/08/the-loebner-prize-a-turing-test-competition-at-bletchley-park/

[7] S. M. Shieber, "Lessons from a restricted turing test," 1994.

[8] H. Chung and S. Lee, "Intelligent virtual assistant knows your life," 2018.

[9] IBM, "Watson assistant," 2018, [Online; accessed 6 Mar 2020]. [Online]. Available: https://www.ibm.com/cloud/watson-assistant

[10] C. Shieber, "How autodesk sped up customer response times by 99watson," 2017, [Online; accessed 6 Mar 2020]. [Online]. Available: https://www.ibm.com/blogs/watson/2017/10/how-autodesk-sped-up-customer-service-times-with-watson/

[11] Microsoft Azure, "Azure bot service," 2019, [Online; accessed 6 Mar 2020]. [Online]. Available: https://azure.microsoft.com/en-gb/services/bot-service/

[12] J. Cahn, "Chatbot: Architecture, design, & development," *University of Pennsylvania School of Engineering and Applied Science Department of Computer and Information Science*, 2017.

[13] H. Wang, Z. Lu, H. Li, and E. Chen, "A dataset for research on short-text conversations," in *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, 2013, pp. 935–945.

[14] Pandorabots Inc., "About pandorabots," 2019, [Online; accessed 6 Mar 2020]. [Online]. Available: https://pandorabots.com/docs/

[15] O. Vinyals and Q. Le, "A neural conversational model," 2015.

[16] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," 2014.

[17] V. Ilievski, "Building advanced dialogue managers for goal-oriented dialogue systems," 2018.

[18] R. Lowe, N. Pow, I. Serban, and J. Pineau, "The ubuntu dialogue corpus: A large dataset for research in unstructured multi-turn dialogue systems," 2015.

[19] R. Kadlec, M. Schmid, and J. Kleindienst, "Improved deep learning baselines for ubuntu corpus dialogs," 2015.

[20] R. T. Lowe, N. Pow, I. V. Serban, L. Charlin, C.-W. Liu, and J. Pineau, "Training end-to-end dialogue systems with the ubuntu dialogue corpus," *Dialogue & Discourse*, vol. 8, no. 1, pp. 31–65, 2017.

[21] M. Völkel, M. Krötzsch, D. Vrandecic, H. Haller, and R. Studer, "Semantic wikipedia," in *Proceedings of the 15th international conference on World Wide Web*, 2006, pp. 585–594.

[22] O. Medelyan, D. Milne, C. Legg, and I. H. Witten, "Mining meaning from wikipedia," *International Journal of Human-Computer Studies*, vol. 67, no. 9, pp. 716–754, 2009.

[23] Wikimedia Foundation, "List of wikipedias," 2019, [Accessed 6 Mar 2020]. [Online]. Available: https://meta.wikimedia.org/wiki/List_of_Wikipedias

[24] F. Wu and D. S. Weld, "Autonomously semantifying wikipedia," in *Proceedings of the sixteenth ACM conference on Conference on information and knowledge management*, 2007, pp. 41–50.

[25] DBPedia, "About | DBpedia," 2019, [Accessed 6 Mar 2020]. [Online]. Available: https://wiki.dbpedia.org/about

[26] ——, "Ontology | DBpedia," 2019, [Online; accessed 6 Mar 2020]. [Online]. Available: https://wiki.dbpedia.org/services-resources/ontology

[27] J. Lehmann, R. Isele, M. Jakob, A. Jentzsch, D. Kontokostas, P. N. Mendes, S. Hellmann, M. Morsey, P. Van Kleef, S. Auer *et al.*, "Dbpedia–a large-scale, multilingual knowledge base extracted from wikipedia," *Semantic Web*, vol. 6, no. 2, pp. 167–195, 2015.

[28] C. Bizer, J. Lehmann, G. Kobilarov, S. Auer, C. Becker, R. Cyganiak, and S. Hellmann, "Dbpedia-a crystallization point for the web of data," *Journal of web semantics*, vol. 7, no. 3, pp. 154–165, 2009.

[29] S. Bird, E. Klein, and E. Loper, *Natural language processing with Python: analyzing text with the natural language toolkit.* " O'Reilly Media, Inc.", 2009.

[30] H. Koepke, "Why python rocks for research," *Hacker Monthly*, vol. 8, 2011.

[31] S. Cass, "The top programming languages 2019," Sep 2019. [Online]. Available: https://spectrum.ieee.org/computing/software/the-top-programming-languages-2019

[32] N. Project, "Natural language toolkit," Aug 2019. [Online]. Available: https://www.nltk.org/

[33] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg *et al.*, "Scikit-learn: Machine learning in python," *Journal of machine learning research*, vol. 12, no. Oct, pp. 2825–2830, 2011.

[34] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard *et al.*, "Tensorflow: A system for large-scale machine learning," in *12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16)*, 2016, pp. 265–283.

[35] J. A. Polzin, "Intelligent scanning using deep learning for mri," Mar 2019. [Online]. Available: https://blog.tensorflow.org/2019/03/intelligent-scanning-using-deep-learning.html

[36] S. Pichai, "Tensorflow: smarter machine learning, for everyone," Nov 2015. [Online]. Available: https://googleblog.blogspot.com/2015/11/tensorflow-smarter-machine-learning-for.html

[37] P. Villegas, "python-aiml," 2019. [Online]. Available: https://github.com/paulovn/python-aiml

[38] G. Cox, "About chatterbot," Jan 2019. [Online]. Available: https://chatterbot.readthedocs.io/en/stable

[39] Vermeulen, Andreas François, *Industrial machine learning: using artificial intelligence as a transformational disruptor.* Apress, 2019.

[40] M.-A. Lemburg, "Pep 249 – python database api specification v2.0," Nov 2017. [Online]. Available: https://www.python.org/dev/peps/pep-0249/

[41] RDFLib Team, "rdflib 4.2.2." [Online]. Available: https://rdflib.readthedocs.io/en/stable

[42] Python Wiki, "Web frameworks for python." [Online]. Available: https://wiki.python.org/moin/WebFrameworks

[43] I. H. Witten, E. Frank, and M. A. Hall, "Practical machine learning tools and techniques," *Morgan Kaufmann*, p. 578, 2005.

[44] T. Abeel, Y. V. d. Peer, and Y. Saeys, "Java-ml: A machine learning library," *Journal of Machine Learning Research*, vol. 10, no. Apr, pp. 931–934, 2009.

[45] V. Kovalev, A. Kalinovsky, and S. Kovalev, "Deep learning with theano, torch, caffe, tensorflow, and deeplearning4j: Which one is the best in speed and accuracy?" 2016.

[46] ALICE A.I. Foundation, "Google code archive - program-ab." [Online]. Available: https://code.google.com/archive/p/program-ab/

[47] Google, "Domino's simplifies ordering pizza using dialogflow's conversational technology." [Online]. Available: https://dialogflow.com/case-studies/dominos/Domino's-case-study.pdf

[48] Your.MD, "About us - your.md." [Online]. Available: https://www.your.md/about