|  |
| --- |
| COM3001 Final Year Project |
| Developing a Chatbot to Answer Wikipedia Queries |
| University of Surrey |

|  |
| --- |
| Alex Turner  4 March 2020 |

# Declaration of Originality

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identified and fully acknowledged all material that is entitled to be attributed to others (whether published or unpublished) using the referencing system set out in the programme handbook.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.

© Copyright Alex Turner, 2020

# Abstract

Abstract here

# Acknowledgments

My acknowledgments here

Table of Contents

[Declaration of Originality i](#_Toc34213334)

[Abstract iii](#_Toc34213335)

[Acknowledgments iv](#_Toc34213336)

[List of Figures vii](#_Toc34213337)

[List of Tables viii](#_Toc34213338)

[Abbreviations ix](#_Toc34213339)

[1 Introduction 1](#_Toc34213340)

[1.1 Overview 1](#_Toc34213341)

[1.2 Objectives 1](#_Toc34213342)

[1.3 Report Structure 1](#_Toc34213343)

[2 Literature Review 2](#_Toc34213344)

[2.1 Chatbots 2](#_Toc34213345)

[2.2 Chatbot Models 3](#_Toc34213346)

[2.2.1 Pattern Matching 3](#_Toc34213347)

[2.2.2 Neural Networks 3](#_Toc34213348)

[2.3 Datasets 3](#_Toc34213349)

[2.3.1 Ubuntu Dialogue Corpus 3](#_Toc34213350)

[2.3.2 DBPedia 4](#_Toc34213351)

[2.3.3 Other Candidates 5](#_Toc34213352)

[2.4 Programming Languages 6](#_Toc34213353)

[2.4.1 Python 6](#_Toc34213354)

[2.4.2 Java 6](#_Toc34213355)

[2.4.3 Clojure? 6](#_Toc34213356)

[2.4.4 Analysis 6](#_Toc34213357)

[2.5 User Interface 6](#_Toc34213358)

[2.6 Existing Solutions 6](#_Toc34213359)

[2.7 Related Works 7](#_Toc34213360)

[2.8 Conclusion 7](#_Toc34213361)

[3 System Analysis 8](#_Toc34213362)

[3.1 Feasibility Study 8](#_Toc34213363)

[3.2 System Requirements 8](#_Toc34213364)

[3.2.1 Functional Requirements 8](#_Toc34213365)

[3.2.2 Performance Requirements 10](#_Toc34213366)

[4 System Design 11](#_Toc34213367)

[4.1 Methodology 11](#_Toc34213368)

[4.2 Design Experimentation 12](#_Toc34213369)

[5 Implementation 13](#_Toc34213370)

[6 Testing 14](#_Toc34213371)

[7 Project Evaluation 15](#_Toc34213372)

[8 Conclusion 16](#_Toc34213373)

[8.1 Overview 16](#_Toc34213374)

[8.2 Conclusions 16](#_Toc34213375)

[8.3 Future Work 16](#_Toc34213376)

[8.4 Final Statement 16](#_Toc34213377)

[Ethics 17](#_Toc34213378)

[References 19](#_Toc34213379)

[Appendix 22](#_Toc34213380)

# List of Figures

[Figure 1: DBpedia Ontology instances per class [24] 14](#_Toc33542301)

[Figure 2: DBpedia extraction framework [23] 15](#_Toc33542302)

# List of Tables

[Table 1: Common DBPedia classes with the number of their instances and example properties, as described in [25] 15](#_Toc33542311)

# Abbreviations

HCI

IoT

ML

RDF

ALICE

IVA

NLP

RG

AIML

XML

# Introduction

## Overview

## Objectives

## Report Structure

# Literature Review

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.

## Chatbots

When discussing chatbots, the beginning of their history is usually cited as Alan Turing’s 1950 article “Computing Machinery and Intelligence” [2], 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 asks 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 [3]. 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 [3]. 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 [4]. 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 [5]. Although the prize itself was met with some criticism - Shieber critiques that the goal of the Turing Test is lost on the competition [6] - 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 [7]. 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”. ---

Industries are seeing a growing trend in chatbot integration in their business. Autodesk integrated IBM’s Watson Assistant [8] to process 100,000 user support conversations, reducing the resolution time of enquiries from 38 hours to 5.4 minutes [9]. Many technology companies offer AI cloud services, many of which allow the integration of chatbots including Watson Assistant [8] and Microsoft Azure Bot Service [10]. The next section will explore and discuss techniques for implementing chatbots and explore technologies that can be used.

## Chatbot Models

A chatbot usually consists of three key components – natural language processing (NLP), response generation (RG) and the knowledge base [11]. 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 [12], and each has its advantages and challenges which will be explored in this section.

### 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 [4]. 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* [4], 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].

### 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 [13]. This can be seen in [13], which uses the *seq2seq* framework [14], 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 [13].

----------

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.

## Datasets

Typically, chatbots are divided into two groups, open-domain and closed-domain [15]. 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 [15]. 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.

### Ubuntu Dialogue Corpus

The Ubuntu Dialogue Corpus (UDC), is one of the largest public dialogue datasets available [16], consisting of 1 million multi-turn dialogues from users receiving technical support for Ubuntu-related problems [17]. This dataset has been used in several dialogue system implementations successfully, as seen in [18] where Lowe et al. compare learning architectures for multi-turn dialogue systems.

* Excerpts

The corpus is widely used in research experiments [21], and has been used to train neural network models for more general use [22]. 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.

### 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 [19]. In terms of size, it eclipses the size of the Encyclopaedia Britannica, its nearest rival, by a factor of ten [20] – 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 [21]. However, Wikipedia’s content is only fit for human reading [19] and is hard to process computationally. Many attempts have been made to formalise and structure this data, as seen in [19], [20], [22], but this review will focus one of these being DBpedia.

DBpedia is a crowd-sourced effort to extract structured content from various Wikimedia projects [23], 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 [24]. This structure enables programs to process this data effectively, including a chatbot application. The size of the DBpedia Ontology is shown in Figure 1, which demonstrates the scale of the project, and how this might be effective for the project.

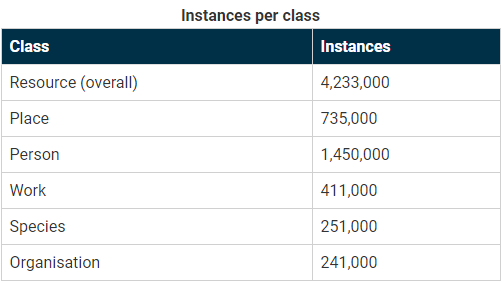


Figure : DBpedia Ontology instances per class [25]

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. This extraction is structured into four phases, as described in [24]:

**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.

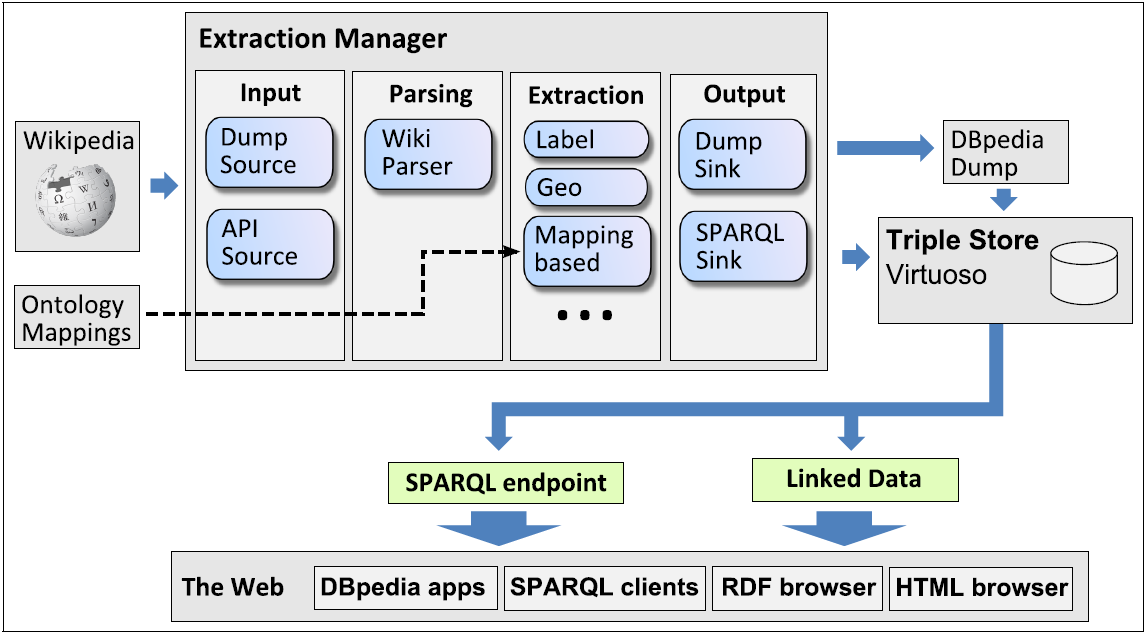


Figure : DBpedia extraction framework [24]

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 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 |
| Athlete | 74,832 | current Team, currentPosition, currentNumber |
| Politician | 12,874 | predecessor, successor, party |

Table : Common DBPedia classes with the number of their instances and example properties, as described in [26]

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 ----

Code example -------

DBPedia provides a number of

### Other Candidates

----

## 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.

### Python

Python has been widely adopted in the scientific industry [27], touted for its extensive collection of scientific libraries [28]; it recently eclipsed Java to become the most popular language, according to IEEE Spectrum [29]. Natural language processing can be quickly implemented with libraries such as *Natural Language Toolkit (NLTK)* [29]; artificial neural networks (ANNs) can be leveraged with many libraries such as *Scikit-learn* [30]. Newer developments in machine learning include TensorFlow [31], which provides a novel learning framework utilised for large scale ML implementations such as healthcare [32] and Google’s own search engine [33]. In the chatbot realm, Python allows for interpretation of AIML with the *python-aiml* library [34]; more comprehensive chatbot libraries can also be utilised, such as *ChatterBot* [35]. Using Python would therefore give 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 [36]. However, this is mainly a concern for enterprise solutions, as Python’s DB-API specification can connect to most databases [37]. Furthermore, connecting to a SPARQL endpoint and parsing RDF graphs is possible through RDFLib [38]. There are also many Python web frameworks that can handle user interaction, routing, and security. Some of the popular web frameworks include Django, TurboGears, and Flask [40]. 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 a number of Machine Learning as a Service (MLaas) services; the Microsoft Azure Machine Learning service uses Python for training and modelling [41]. -------

### Java

### Clojure?

* Lisp - Pandorabots

### Analysis

## User Interface

* Vs standalone/command line
* Web technologies
  + JSP
  + Spring

## Existing Solutions

Chatbots are ubiquitous in business and consumer use, ranging from

* Google Search/Assistant
* DialogFlow
* DBPedia chatbot
  + Analysis
* Mitsuku

## Related Works

* AIML and SPARQL papers

## Conclusion

# System Analysis

1. Feasibility Study
2. Requirements
   1. Requirements gathering
   2. Functional requirements
   3. Non-functional requirements
   4. Requirements prioritisation
3. Planning

## Feasibility Study

* feasibility

## System Requirements

### Functional Requirements

* Intro

#### 1.0.0 User Interaction

* 1.1.0 The application should allow the user to interact with the chat bot
* 1.1.1 The user should be able to access the chatbot in a browser
* 1.1.2 The user should have a text box to type their query
* 1.1.3 The user should clearly see the response of the chatbot in the webpage
* 1.1.4 The user should be able to clearly see their conversation with the chatbot in the webpage

#### 2.0.0 Queries

* 2.1.0 The chatbot can answer basic questions about people:
  + 2.1.1 The application should take a user query about a person - ‘who is X’ - and respond with a description of that person.
  + 2.1.2 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.
  + 2.1.3 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.
  + 2.1.4 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.
  + 2.1.5 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.
  + 2.1.6 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.
  + 2.1.7 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
  + 2.1.8 The application should take a user query about linking to the Wikipedia page of a person, and return a link to that page.
* 2.2.0 The chatbot can answer questions about countries:
  + 2.2.1 The application should take a query about a country, and return the description of that given country.
  + 2.2.2 The application should take a query about the population of a country, and return the population of that given country.
  + 2.2.3 The application should take a query about the capital of a country, and return the capital of the country.
* 2.2.3 The application should take a query about the description of a country, and return the description of that given country.
* 2.2.4 The application should take a query about the flag of a country, and return the flag image of that given country.
* 2.2.5 The application should take a query about where a country is on a map, and return a google maps location of that country.

#### 3.0.0 Advanced Queries

* + 1. The chatbot can perform advanced query searches and comparisons:
* 3.1.0 Advanced People queries
  + 3.1.1 The user can ask for a list of people born in a given year, e.g. ‘who was born in 1995’ and the chatbot returns a list of notable people born in that year
  + 3.1.2 The user can ask for a list of people born in a given place, e.g. ‘who was born in London’ and the chatbot returns a list of notable people born in that place
  + 3.1.3 The user can ask for winners of a given prize, e.g. ‘who won the Nobel Peace Prize [in 2009]’ and the chatbot returns a list of winners of that prize, or the winner of the prize in a given year.
* 3.2.0 Advanced Country queries
  + 3.2.1 The user can ask the chatbot for a list of countries with a given official language, e.g. ‘which countries speak Italian’, and the chatbot will return a list of countries which identify that language as their official language.
* 3.3.0 Combining queries with AND operator:
  + 3.3.1 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’
* 3.4.0 Context-aware queries:
  + 3.4.1 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’.

#### 4.0 Conversation

* 4.1 The user should greet the chat bot and be returned with a similar greeting – e.g. Hello.
* 4.2 The user should be able to ask for example queries and the chat bot returns a number of working example queries
* 4.3 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.

### Performance Requirements

#### 5.1 Performance

* 5.1.1 The web page should load fully in less than 5 seconds
* 5.1.2 The chat bot should respond to each query within 5 seconds

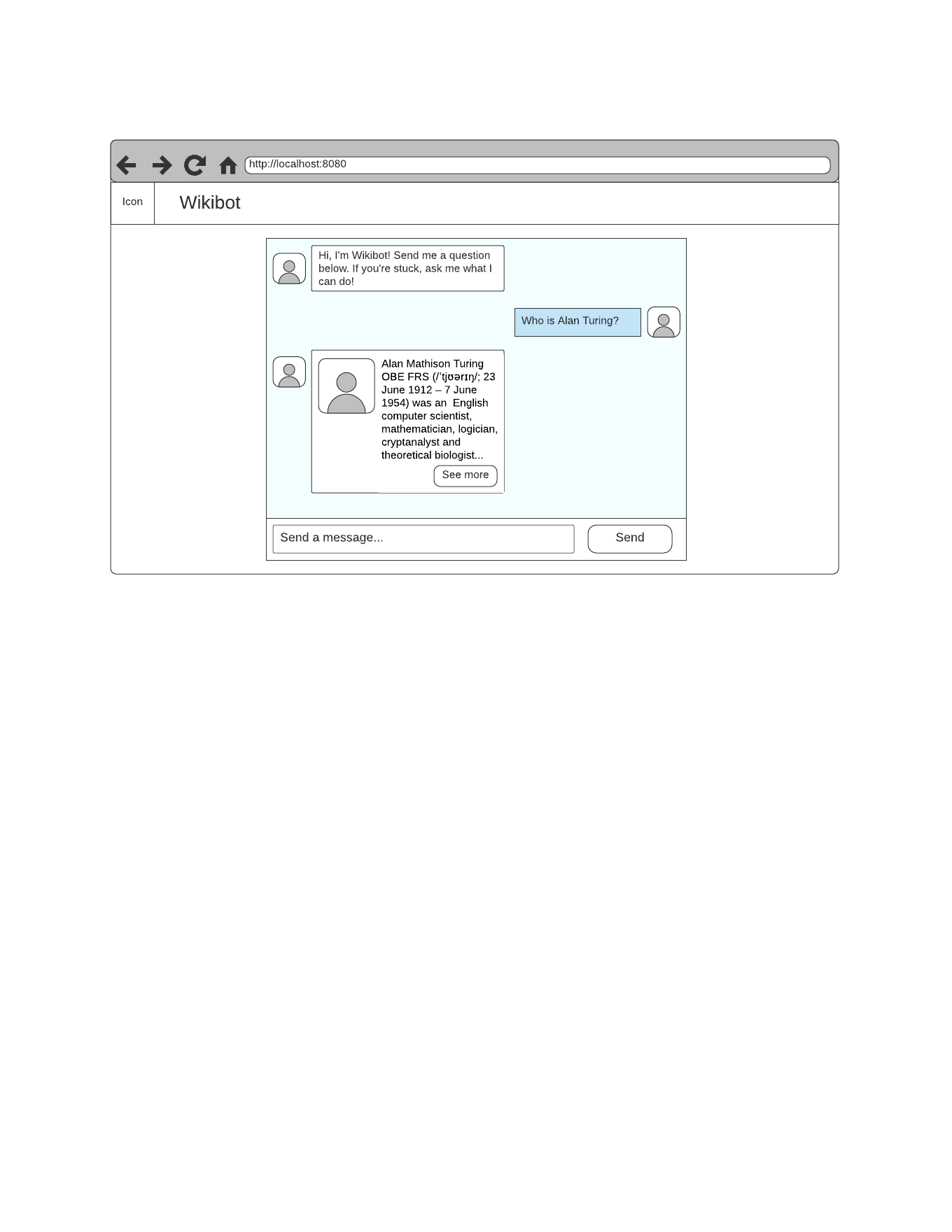
#### 5.2 Reliability

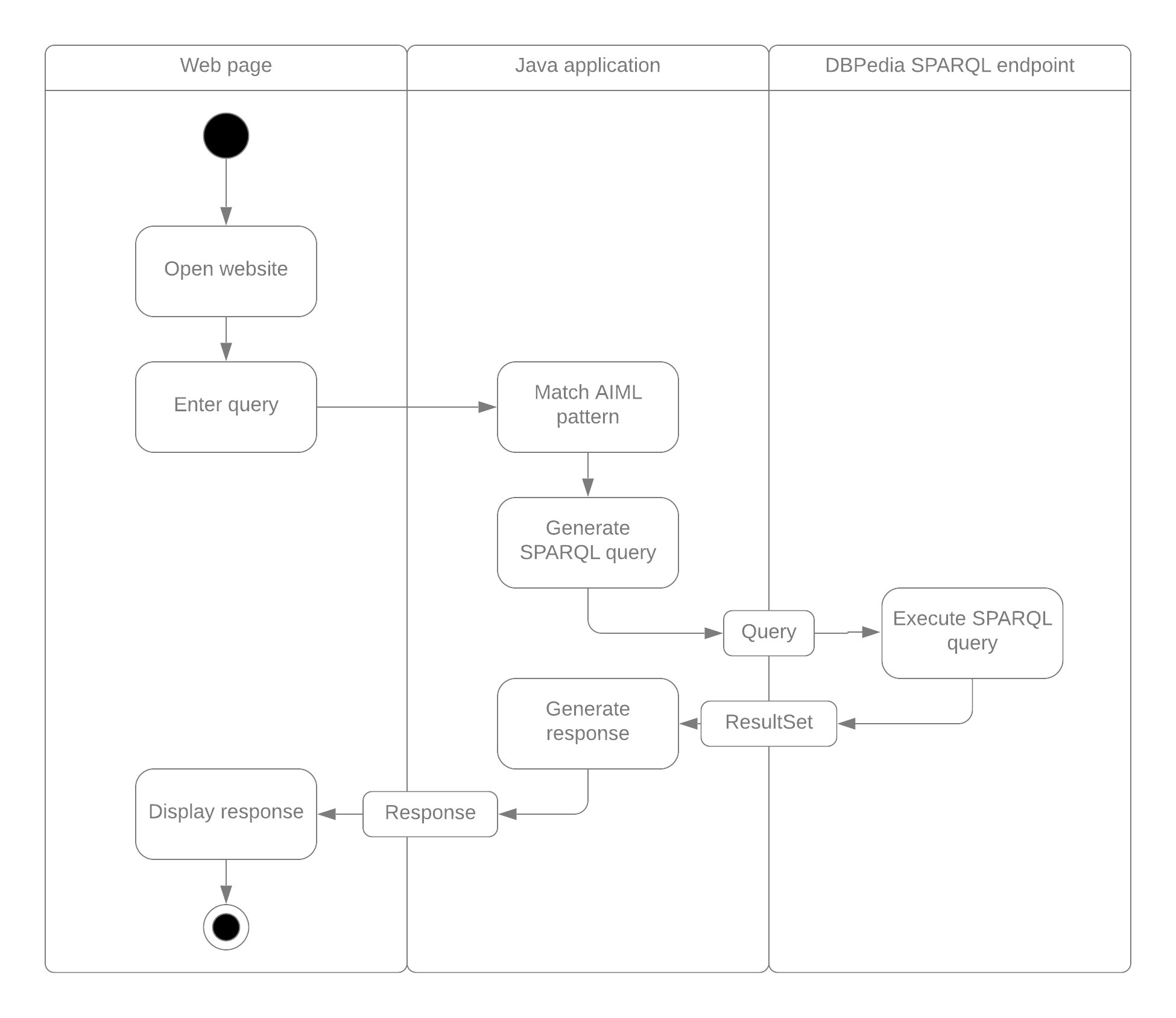
* 5.2.1 The application should function without failure
* 5.2.2 Any errors that do occur during normal operation should be logged, and the user should be clearly informed that an error has occurred.

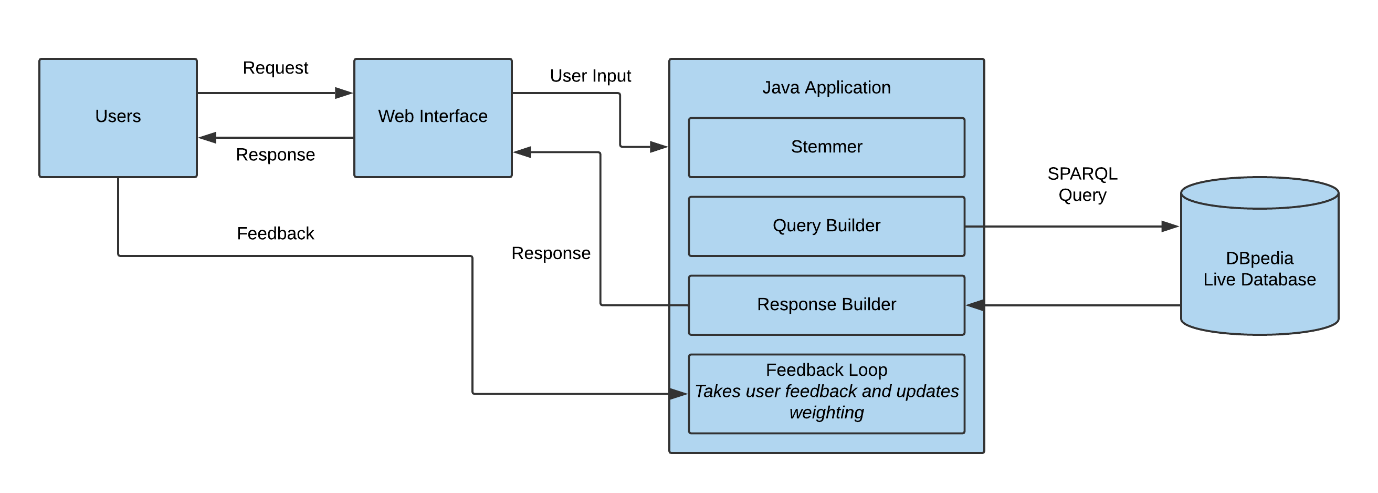
# System Design

## Methodology

* Software development methodology with justification







## Design Experimentation

# Implementation

# Testing

# Project Evaluation

# Conclusion

## Overview

## Conclusions

## Future Work

## Final Statement

# Ethics

# References

|  |  |
| --- | --- |
| [1] | R. Chandra, “How Google Home and the Google Assistant helped you get more done in 2017,” 5 Jan 2018. [Online]. Available: https://blog.google/products/assistant/how-google-home-and-google-assistant-helped-you-get-more-done-in-2017/. [Accessed 5 Nov 2019]. |
| [2] | A. M. Turing, “Computing machinery and intelligence,” *Mind,* vol. 59, no. 236, pp. 433-60, 1950. |
| [3] | J. Weizenbaum, Computer Power and Human Reason: From Judgment to Calculation, New York: W.H. Freeman and Company, 1976. |
| [4] | R. S. Wallace, “The Anatomy of ALICE,” in *Parsing the Turing Test*, Dordrecht, Springer, 2009, pp. 181-210. |
| [5] | E. Keedwell, “The Loebner Prize, a Turing Test competition at Bletchley Park,” University of Exeter, 8 Dec 2014. [Online]. Available: https://blogs.exeter.ac.uk/exeterblog/blog/2014/12/08/the-loebner-prize-a-turing-test-competition-at-bletchley-park/. [Accessed 25 Feb 2020]. |
| [6] | S. M. Shieber, “Lessons from a Restricted Turing Test,” *Communications of the Association for Computing Machinery,* no. 37, pp. 70-78, 4 Apr 1994. |
| [7] | H. Chung and S. Lee, “Intelligent virtual assistant knows your life,” *arXiv preprint,* vol. arXiv:1803.00466, 2018. |
| [8] | IBM, “Watson Assistant,” 15 Oct 2017. [Online]. Available: https://www.ibm.com/cloud/watson-assistant/. [Accessed 13 Feb 2019]. |
| [9] | C. Schneider, “How Autodesk sped up customer response times by 99% with Watson,” IBM, 12 Oct 2017. [Online]. Available: https://www.ibm.com/blogs/watson/2017/10/how-autodesk-sped-up-customer-service-times-with-watson/. [Accessed 13 Feb 2020]. |
| [10] | Microsoft Azure, “Azure Bot Service,” [Online]. Available: https://azure.microsoft.com/en-gb/services/bot-service/. [Accessed 13 Feb 2020]. |
| [11] | J. Cahn, “CHATBOT: Architecture, design, & development,” *University of Pennsylvania School of Engineering and Applied Science Department of Computer and Information Science,* 26 Apr 2017. |
| [12] | 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. |
| [13] | O. Vinyals and Q. Le, “A Neural Conversational Model,” 19 Jun 2015. [Online]. Available: https://arxiv.org/pdf/1506.05869.pdf. [Accessed 25 Feb 2020]. |
| [14] | I. Sutskever, O. Vinyals and Q. Le, “Sequence to Sequence Learning with Neural Networks,” 14 Dec 2014. [Online]. Available: https://arxiv.org/abs/1409.3215v3. [Accessed 20 Feb 2020]. |
| [15] | V. Ilievski, “Building Advanced Dialogue Managers for Goal-Oriented Dialogue Systems,” 16 Mar 2018. [Online]. Available: https://arxiv.org/pdf/1806.00780.pdf. [Accessed 13 Nov 2019]. |
| [16] | S. Kojouharov, “Ultimate Guide to Leveraging NLP & Machine Learning for your Chatbot,” Chatbots Life, 18 Sep 2016. [Online]. Available: https://chatbotslife.com/ultimate-guide-to-leveraging-nlp-machine-learning-for-you-chatbot-531ff2dd870c. [Accessed 13 Nov 2019]. |
| [17] | R. Lowe, N. Pow, J. Pineau and I. V. Serban, “The Ubuntu Dialogue Corpus: A Large Dataset for Research in Unstructed Multi-Turn Dialog Systems,” 4 Feb 2016. [Online]. Available: https://arxiv.org/pdf/1506.08909.pdf. [Accessed 13 Nov 2019]. |
| [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,” 4 Feb 2016. [Online]. Available: https://arxiv.org/abs/1506.08909. [Accessed 25 Feb 2020]. |
| [19] | M. Völkel, M. Krötzch, D. Vrandecic, H. Haller and R. Studer, “Semantic Wikipedia,” in *Proceedings of the 15th international conference on World Wide Web*, Edinburgh, 2006. |
| [20] | 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. |
| [21] | Wikimedia Foundation, “List of Wikipedias,” 17 Oct 2019. [Online]. Available: https://meta.wikimedia.org/wiki/List\_of\_Wikipedias. [Accessed 13 Nov 2019]. |
| [22] | F. Wu and D. S. Weld, “Autonomously semantifying Wikipedia,” in *CIKM '07 Proceedings of the sixteenth ACM conference on Conference on information and knowledge management*, Lisbon, 2007. |
| [23] | DBpedia, “About | DBpedia,” 2019. [Online]. Available: https://wiki.dbpedia.org/about. [Accessed 13 Nov 2019]. |
| [24] | J. Lehmann and et al., “DBpedia – A Large-scale, Multilingual Knowledge Base Extracted from Wikipedia,” *Semantic Web,* vol. 6, no. 2, pp. 167-195, 2015. |
| [25] | DBpedia, “Ontology | DBpedia,” 2019. [Online]. Available: https://wiki.dbpedia.org/services-resources/ontology. [Accessed 13 Nov 2019]. |
| [26] | C. Bizer, J. Lehmann, G. Kobilarow, S. Auer, C. Becker, R. Cygniak and S. Hellman, “DBPedia - A crystallization point for the Web of Data,” *Journal of web semantics,* vol. 7, no. 3, pp. 154-165, 2998. |
| [27] | S. Bird, E. Klein and E. Loper, Natural Language Processing with Python, Sebastopol, CA: O'Reilly Media, Inc., 2009. |
| [28] | H. Koepke, “Why Python rocks for research,” *Hacker Monthly,* vol. 8, 2011. |
| [29] | NLTK Project, “Natural Language Toolkit - NLTK,” 20 Aug 2019. [Online]. Available: http://www.nltk.org/. [Accessed 3 Mar 2020]. |
| [30] | P. Fabian, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel and M. Blondel, “Scikit-learn: Machine learning in Python,” *Journal of machine learning research,* vol. 12, no. Oct, pp. 2825-2830, 2011. |
| [31] | M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis and J. Dean, “TensorFlow: A System for Large-Scale Machine Learning,” in *12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16)*, Savannah, 2016. |
| [32] | J. Polzin, “Intelligent Scanning Using Deep Learning for MRI,” GE Healthcare Global Magnetic Resonance Imaging, 1 Mar 2019. [Online]. Available: https://blog.tensorflow.org/2019/03/intelligent-scanning-using-deep-learning.html. [Accessed 3 Feb 2020]. |
| [33] | S. Pichai, “TensorFlow: smarter machine learning, for everyone,” Google, 9 Nov 2015. [Online]. Available: https://googleblog.blogspot.com/2015/11/tensorflow-smarter-machine-learning-for.html. [Accessed 3 Feb 2020]. |
| [34] | P. Villegas, “python-aiml,” GitHub repository, 2019. [Online]. Available: https://github.com/paulovn/python-aiml. [Accessed 3 Feb 2020]. |
| [35] | G. Cox, “About ChatterBot,” ChatterBot, 12 Jan 2019. [Online]. Available: https://chatterbot.readthedocs.io/en/stable/. [Accessed 3 Feb 2020]. |
| [36] | A. F. Vermeulen, Industrial Machine Learning, Berkeley, CA: Apress, 2020. |
| [37] | M.-A. Lemburg, “PEP 249 -- Python Database API Specification v2.0,” 11 Nov 2017. [Online]. Available: https://www.python.org/dev/peps/pep-0249/. [Accessed 3 Feb 2020]. |
| [38] | RDFLib Team, “rdflib 4.2.2,” 7 May 2019. [Online]. Available: https://rdflib.readthedocs.io/en/stable/. [Accessed 3 Mar 2020]. |

# Appendix