Chatbot Generation for Open Data Accessibility

Julie Marie Schnell Heldal

Kathrine Hermansen

Master's thesis in Software Engineering at

Department of Computer science, Electrical engineering and Mathematical sciences,
Western Norway University of Applied Sciences

Department of Informatics,
University of Bergen

November 2023

Abstract

Open data, despite its availability, often remains inaccessible to the average person due to complex data formats and technical barriers. This challenge hinders the realization of open data's transformative potential. Simultaneously, there is an ongoing evolution in human-to-machine interaction, with a notable emphasis on advancements in chatbot technology. In response to this issue, our research explores the integration of chatbots as tools to democratize access to open data. We present an approach that involves developing a natural language interface through the use of chatbots, enabling user-friendly access to open data, by allowing users to make meaningful queries, receive relevant information, and explore their selected JSON-formatted datasets. Our chatbot accepts user queries as input, which serves to extract specific information from the API and return it as output. Additionally, we have investigated the use and awareness surrounding open data by conducting a preliminary survey aimed at individuals with varying levels of technical expertise. With this research, we aspire to encourage the average person to discover information available through chatbots.

Acknowledgments

First and foremost, we would like to express our gratitude to our thesis advisor, Adrian Rutle, for his great support, guidance, and insights throughout the research process.

We extend our appreciation to Jordi Cabot, we would like to express our gratitude for his assistance in shaping and refining our thesis through his insightful research proposal and constructive feedback.

Julie
I would like to express my gratitude to my family and friends for their unwavering encouragement, love, and support throughout this research endeavor. I would like to extend a special word of appreciation to my co-writer and dear friend, Kathrine. Her companionship has made this journey enjoyable and has been a constant source of emotional support.

Kathrine
I want to give a big heartfelt appreciation to my family and friends for their encouragement, love, and consistent support throughout this thesis. Lastly, a special mention to my close friend and co-author, Julie. Her commitment and our collaborative effort have made this journey efficient and enjoyable.

Contents

List of Figures

Acronyms

AI Artificial Intelligence.

API Application Programming Interface.

BERT Bidirectional Encoder Representations from Transformers.

CFG Context-Free Grammar.

CSV Comma-Separated Values.

CUI Conversational User Interface.

DOM Document Object Model.

DSL Domain Specific Language.

GPT Generative Pre-trained Transformer.

HMI Human-Machine-Interaction.

IDE Integrated Development Environment.

JSON JavaScript Object Notation.

LLM Large Language Model.

LM Language Model.

ML Machine Learning.

NLG Natural Language Generation.

NLP Natural Language Processing.

NLU Natural Language Understanding.

SUS System Usability Scale.

UI User Interface.

Chapter 1

Introduction

The concept of human-to-machine interaction was an idea first theorized by Alan Turing in 1950 with the intent to determine if a machine could give off the impression of human-to-human communication. Building on this idea, the first "bots" simulating human conversation emerged around 1960 [5] [58]. To this day, chatbots are often used to enhance the user experience across a range of platforms by automating tasks. They are proven useful for various contexts such as in the customer service market, e-commerce, and for educational purposes. Powered by intricate software applications, chatbots utilize machine learning not only to provide fitting responses to user inputs but also to comprehend user intents and make predictions, whether in text or speech. As the landscape of human-to-machine interaction continues to evolve, chatbots emerge as valuable tools and therefore can hold numerous use cases [92]. In this research endeavor, we seek to explore the use of chatbots to make open data more accessible.

1.1 Open Data Accessibility

It is estimated that hundreds of files of open data get published online every day both from the private and public sectors. The publication of data is driven by the belief that it brings immense benefits to businesses, public administrations, and the general public in that it allows all humans to use, reuse, and redistribute data exactly as they would like. For instance, the European Union Open Data portal unifies and releases massive amounts of data with regard to health, education, and industry among other fields. This portal enables stronger collaborations across European nations by promoting data sharing and cooperation [35] [18]. Nevertheless, several barriers hinder open data usage. These obstacles stem from many potential providers being hesitant to open these datasets due to the practical difficulties they encounter [9]. Open data is generally available in various file types, including XML, JSON, CSV, and HTML. This can be challenging for the average person with no prior technical background to interpret, making them dependent on other third-party applications to interpret the data. As indicated by the study conducted by the EU Data Portal, 73% of open data users describe the process of locating data as either difficult or very difficult [8]. As Cabot (2019) expresses it *"Open data may be open but has not opened to* *the average person"*[14]. To address this, a potential solution lies in creating conversational interfaces using chatbots, enabling user-friendly access to open data.

1.2 Creating Accessible Open Data: The Role of Chatbots

The increasing availability of open data presents both opportunities and challenges. While there are potential benefits in increasing the availability of open data, there also persist hurdles related to data sharing, technical complexity, and user accessibility [8]. Our work seeks to harness the power of chatbots to provide an accessible interface for open data consumption. Through our research, we aim to create the foundational groundwork that addresses the current limitations and also establish scalable and adaptable solutions that empower users to tap into the potential of open data. Among our objectives is the expansion of chatbots and open data utilization to a wider array of contexts and domains.

1.3 Motivation

The concept of employing conversational interfaces through chatbots emerges as a potential solution to the above mentioned challenges with open data. We believe these automated dialog systems can offer a way to make open data more accessible, transcend technical barriers, and enable the data to be utilized by individuals regardless of their technical expertise. Several studies imply that the dialogue systems are both well-known and favored when interacting with them on the web. According to a study performed by *UserLike* in 2022 [99], at least 80% of respondents had interacted with a chatbot before. Another study conducted by *Tidio* in 2023 [40] found that 96% of customers had either heard or knew about chatbots and that 62% of their customers would rather use an online chatbot to see if they could help them out, instead of waiting for a human agent. However, studies have also shown the necessity of a working and functional bot. For instance, in a survey conducted in 2021 by *Drift* [2], 59% of their consumers expected the bot to answer within 5 seconds, and *Outgrow's* study from 2023 [97] asserted that 64% of the customers state that the best thing about the bot is the 24/7 availability.

Therefore, in the path to combine chatbots with open data, it is important to take note of all surrounding factors as well; which are not only the complex structural aspects of data formats but also the technical processes behind building these intelligent dialogue systems providing fast and understandable responses. Our research is driven by the urge to overcome the hurdles that interfere with the realization of open data's transformative potential and to render open data genuinely open, not only in terms of availability. Thus, through an in-depth exploration of chatbot creation and the intricate domain of open-source data, the motivation lies in finding whether chatbots have the potential to bridge the gap between complex data formats and human comprehension, and if so, how this objective can be achieved.

1.4 Research Questions

In light of the challenges surrounding open data accessibility and usability, as detailed in the previous sections, our research wishes to investigate the transformative potential of chatbots as facilitators of open data utilization. To define the scope of this thesis, we formulate the following research questions:

1. How can we develop a natural language interface that empowers users to access and utilize open data sources via chatbots, while automating the bot generation process to ensure adaptability across multiple APIs?
2. In investigating the use of chatbots as a mechanism for the general public to utilize data, how can it assist individuals to utilize and benefit from open data sources?

The research questions we have selected are expected to yield insights and outcomes that contribute to the development and understanding of chatbot creation and utilization of open data sources. The anticipated results encompass various aspects of chatbot generation and knowledge, data utilization, and automation.

1.5 Contributions

In this section, we highlight the key contributions adhered to this project. We will give an overview of the thesis objectives, the proposed solution, and the approach to gather insights from the general public regarding bots and open data utilization.

1.5.1 Chatbot Generation and Data Utilization

The objective is to develop a natural language interface, enabling users - whether technically inclined or not - to access and utilize open data sources with the help of chatbots. This involves creating a web application that consists of a user interface and a chatbot widget integrated into the application. The user interface includes thorough guidance on utilizing the chatbot, including a list that encompasses the multitude of queries the user can ask the bot. Furthermore, the interface features a JSON visualizer capable of retrieving example responses from the user's preferred open-data source, therefore serving as a valuable tool for the users to gain a more comprehensive overview and understanding of the data source. The chatbot widget is an application capable of fetching real-time open data, comprehending user queries, and providing suitable responses by extracting information from the data source. The bot is able to present examples from the different queries directly in the bot for user-friendliness, provide an overview of the different entities available in the data, as well as to change the API if desired. Additionally, the bot incorporates a natural language model to enhance a wider query comprehension, thus accommodating potential type errors from the user making it more versatile and improving overall usability. To scope this project, the open-source data type must adhere to the JSON standard. It was also deemed necessary to automate the bot generation process in order to ensure adaptability across multiple APIs. The functions developed gain automation benefits to the bot in that they can be utilized uniformly across the various intents and entities across different APIs from the user queries, thereby simplifying and accelerating the chatbot development process.

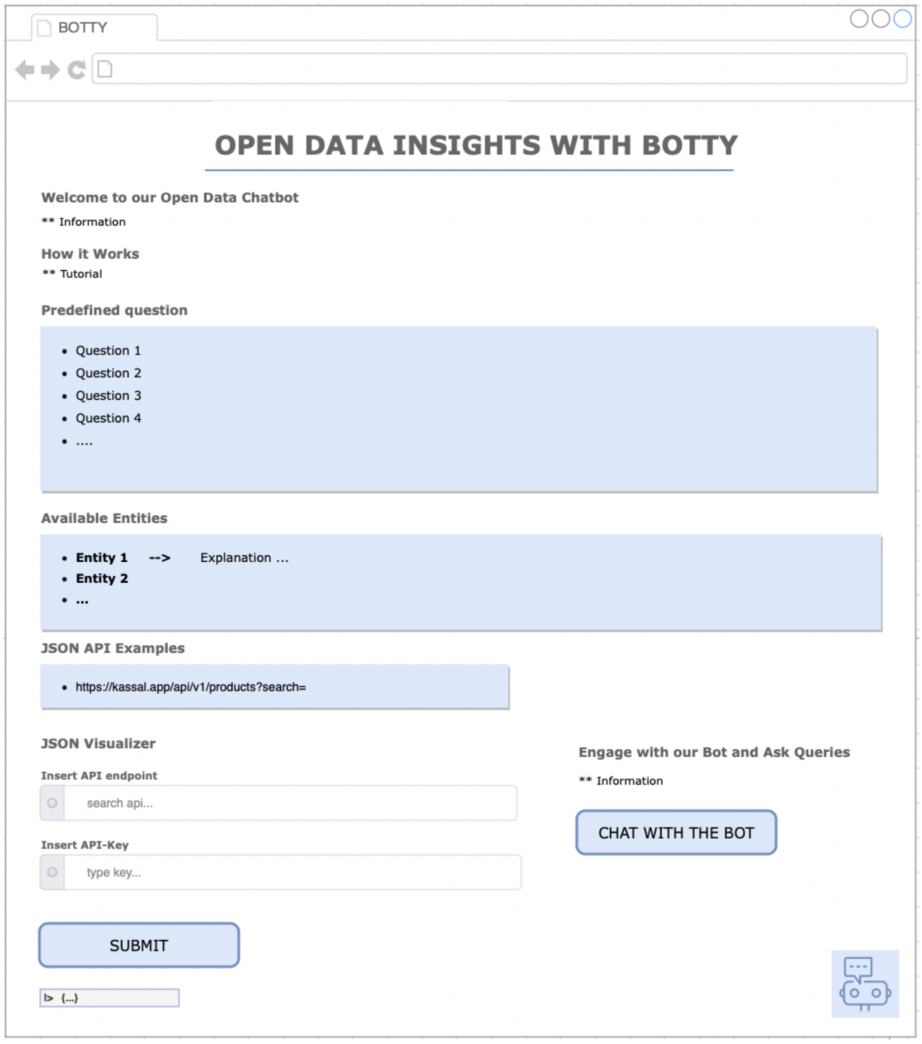


Figure 1.1: A visual sketch representation of the anticipated user interface

1.5.2 Open Data Utilization and Accessibility

By employing chatbots as a means for the general public to access open data sources, we anticipate facilitating a more user-friendly interaction with data, as well as increased user engagement with the data. Users will be able to query the bot for information from open data sources and acquire the specific insights they desire. The chatbot's natural language interface is anticipated to make data more accessible and comprehensible, encouraging users to explore and interact with information they might not have otherwise. Additionally, we investigate the use and awareness surrounding open data by conducting a preliminary survey aimed at individuals with varying levels of technical expertise. Furthermore, the survey delves into people's knowledge and use of chatbots, as well as their perception regarding the use of bots as a mechanism for accessing open data. This survey provides valuable insights into the potential of chatbots and open data.

In summary, the expected results of this research endeavor encompass the creation of automated chatbots capable of fetching real-time information from open data sources, the development of an intuitive interface for understanding chatbots- and data properties, and the enhancement of open data utilization for the general public. These outcomes collectively contribute to the field of chatbot development and facilitate data-driven decision-making processes.

1.6 Methodology

This section explores the methodologies employed in our project to seek answers to our research questions and address the challenges at hand.

1.6.1 Research Approach

Our research approach draws inspiration from the design science paradigm, an approach that emphasizes creating and validating practical solutions to real-world problems. Its research often has a focus on the development and assessment of designed artifacts with the clear intention of enhancing their functional capabilities [27]. As stated by Hevner et al (2004) *"The design-science paradigm seeks to extend the boundaries of human and organizational capabilities by creating new and innovative artifacts"*[47]. The paradigm prioritizes the process of selecting what is achievable and valuable for shaping potential futures, rather than on what is currently existing. Hevner presents a set of seven guidelines for design science research within his article on design science research. We present these guidelines with an explanation of how we consider them in our research:

1. **Design as an artifact:** The creation of something practical, such as a model or method, to solve real-world problems. For our thesis, this involves building the chatbot and related tools.

2. **Problem relevance:** Focus on using technology to solve important and relevant issues. Our research addresses the pressing challenges in open data accessibility.

3. **Design evaluation:** Assessing how well our design works. This involves measuring things like user satisfaction and task success rates to ensure our

chatbot is effective.

4. **Research contributions:** Aiming to make contributions in various areas. This includes improving the chatbot itself, the methods used, and the overall approach to chatbot technology and open data.
5. **Research Rigor:** Ensuring the reliability of our research.
6. **Design as a Search Process:** Exploration of the different ways to reach our objectives.
7. **Communication of Research:** Presentation of our findings and solutions in an understandable manner for both technical and non-technical users.

To briefly explain, this methodology involves the creation and evaluation of an artifact that solves a particular problem. This involves a cyclical process of building, evaluating, and refining the artifact until it meets the desired goals and objectives. It is important to acknowledge that we have taken inspiration from this approach and loosely implemented it. Within this context, the core objective revolves around developing a natural language interface, with the chatbot serving as our artifact. The problem at hand is how to create a chatbot that can empower users to access and utilize open data sources effectively. Furthermore, as our objective is not to deliver a fully developed system as a final product, but rather to explore the feasibility of such a system and construct a prototype to lay the groundwork for this problem description, the sequential steps and iterative process outlined in this methodology are regarded as future research endeavors.

1.6.2 Research Method

Brent and Leedy (2019) [11] expressed that research can be misinterpreted as merely gathering information, documenting facts and extensively searching for a subject matter. Research encompasses more than that; it also involves the process of collecting, analyzing, and interpreting data to gain a thorough understanding of a phenomenon. Research involves a creative endeavor carried out systematically in order to improve the stock of knowledge, including knowledge of humans, culture, and society. Thereafter use this stock of knowledge to develop innovative applications [71]. There exist three common methods to conduct research, them being quantitative-, qualitative- and mixed methods. According to Creswell (2013) [19], quantitative research involves the collection, analysis, interpretation, and presentation of study findings, whereas qualitative research offers an alternative method with a numerical or statistical approach. Therefore, in deciding between those, it prompts the consideration of which data types are needed, whether it is numerical, textural, or a combination of both [105]. Based on this assessment, we found it necessary to employ a mixed-methods approach by conducting both qualitative and quantitative research paradigms, in order to ensure accurate responses to the phenomena in question. As stated by Steckler et al. (1992) *"both paradigms have weaknesses which, to a certain extent, are compensated for by the strengths of the other"* [91].

Mixed-Methods Approach: Combining Numbers and Narratives

The mixed-methods approach took shape in the form of a survey, which is represented in detail in chapter 5.2, Chatbots and Open Data: Survey Findings. The quantitative research paradigm was used to assess the demand and reception of chatbots with the purpose of making open data more easily available. We gathered quantitative data to measure factors such as knowledge levels, interest in such services, types and frequency of open data usage and chatbot interactions, and user preferences related to chatbot functionality. Furthermore, the qualitative research paradigm was used to gain a deeper understanding of the needs and preferences of the potential end-users of our chatbot, particularly concerning the utilization of open data sources. We aim to understand how bots can assist users to access and benefit from open data sources while assessing whether this service fits the preference of the general public, and whether it is a service the public is inclined to make use of. Our approach was to first initiate the process by conducting a self-assessment, thereby reflecting on ourselves as situated in the same context. This included being open to adopting a new perspective and being creative. Thereafter, we decided on questions that seemed relevant to the survey and allowed the respondents to provide detailed explanations and insights.

System Usability Scale

In addition to the survey, metrics related to user satisfaction, completion rates, and task success rates will also be collected to provide a comprehensive understanding of the entire web applications including the user interface, the JSON visualizer, and the chatbot widget. These metrics will be gathered numerically, and the form gathering them will draw inspiration from the System Usability Scale (SUS). SUS provides a reliable and low-cost scale that can be used to measure the usability of a system [12]. This will be further explained in section 5.1.3. Finally, we will also be seeking feedback from competent reviewers and our target group to ensure that our application is as optimal and user-friendly as possible. This feedback will be used to refine and improve the user interface and chatbot until it meets the desired goals and objectives.

1.7 Outline

This section consists of an overview of the outline of the master thesis. The outline is as follows

- **Chapter 1: Introduction** - An overview of the core idea and sets the stage for the entire thesis.
- **Chapter 2: Background** - A discussion of relevant background information, context, terminologies, and concepts related to the thesis topic.
- **Chapter 3: Related Work** - An exploration of prior research and literature closely tied to the themes addressed in the thesis, highlighting the academic landscape.
- **Chapter 4: Design and Implementation** - A thorough explanation of the methodologies and technologies used, as well as illustrations and descriptions of the implementation of the solution approach.
- **Chapter 5: Analysis and Assessment** - An analysis and evaluation of the data gathered.
- **Chapter 6: Discussion** - An in-depth analysis and interpretation of the different findings, including the data gathered and the discoveries made during the development process.
- **Chapter 7: Conclusion and Further Work** - A summarization of the key findings during this research endeavor, a discussion, and a final conclusion for this thesis. Additionally, an exploration of potential future directions for the application.

Chapter 2

Background

This chapter will concern terminologies and concepts that are important to have knowledge of throughout this thesis, thereby establishing a common theoretical foundation. Concepts and their related content in need of clarification include; Open Source Data, Chatbots, Human-to-Machine interactions, Formal Languages, Chatbot platforms, and NLU platforms. This overview provides readers with necessary background knowledge and reveals the relevance of each concept within the context of our research.

2.1 Open Source Data

Open data is widely used in our everyday lives, such as when people check the weather forecast or use GPS apps on their smartphones. The term Open data is described such that; *"Data are considered to be "open" if anyone can freely access, use, re-use, and redistribute them, for any purpose, without restrictions"*[10]. While there is a large amount of data available online, only data that can be reused for other purposes, downloaded in open formats, and read by software can be considered truly open.

The openness of data can be assessed along two dimensions. The first dimension pertains to the legal openness of the data, which means that the data must be located in a public domain with minimal restrictions. Data that are intended to be read as stand-alone documents but cannot be reused for other purposes may not be considered open data. The second dimension relates to the technical openness of the data. For data to be considered technically open, they must be published in electronic formats that are easily readable by machines and not restricted by proprietary software. This allows anyone to access and use the data with commonly available software tools. Furthermore, the data must be accessible on a public server without any restrictions like passwords or firewalls.

To help people find open data more easily, many organizations create and manage open data catalogs. By making data publicly available in a standardized and accessible format, open data can help drive innovation and create new opportunities in various fields [10]. To better understand the principles of open data, let's explore some practical examples of open data sources that have left an impact in various domains. Government agencies worldwide contribute significantly to the open data movement, offering access to a wealth of information ranging from demographics and economics to public services. For instance, the European Union provides extensive data through the European Data Portal[35]. Similarly, meteorological agencies participate in making weather data, forecasts, and climate information accessible as open data. The European Centre for Medium-Range Weather Forecasts (ECMWF) in Europe, for instance, exemplifies this commitment to open data [33].

Open data commonly adapts formats such as CSV, JSON, XML, and HTML, among others, making it accessible to be interpreted by both machines and humans. In scoping our project we have specifically focused on open data presented in JSON format. The following section elaborates on the characteristics and specifications of this chosen file format.

2.1.1 JSON

JSON, an abbreviation for JavaScript Object Notation, is derived from JavaScript literals and acts as a subset of the JavaScript programming language, meaning it does not introduce any additional functionalities beyond what the JavaScript language already offers. While JSON is derived from a programming language, it is not a programming language itself, but rather a data interchange format, that organizes data using a carefully defined set of rules. JSON is widely recognized as the standard for data interchange, implying its suitability for data formatting in diverse exchange scenarios. JSON enables data structuring through two main formats: a collection of key/value pairs and an ordered list of values. JSON has various types, such as strings, numbers, arrays, and booleans. This hierarchical structure of JSON facilitates easy readability and seamless interoperability across differing programming languages and systems [87].

2.2 Chatbots

Chatbots are automated conversational agents capable of interacting with users through the use of natural languages in real-time. Conversational agents are able to understand human language, process it, and interact with people while carrying out specific tasks [5]. The evolution of chatbots has been rapid across various domains in recent years, despite their origins dating back to the 1950s. ELIZA, one of the first chatbots ever created, was created in 1966 by Joseph Weizenbaum at MIT. It was designed to mimic the conversational patterns of various roles, including that of a Rogerian psychotherapist. It explored the connection between humans and machines, thereby being one of the first programs to attempt the Turing test [62]. Today, these interactive systems are found widespread across the web, as well as used in applications like email, live chats, and SMS. They have undergone significant advancements since their introduction on platforms like Facebook, Skype, and others. Chatbots are utilized across a range of fields, including marketing, support systems, education, healthcare, and entertainment, among others. Their increasing popularity can be attributed to the numerous advantages they offer both users and developers. Chatbot implementation is typically platform-independent, offering users seamless access without the requirement for installations [4].

When creating chatbots, there are specific features that contribute to their recognition as valuable tools. These characteristics encompass Natural Language-Processing and Understanding, user-friendliness with task-oriented functionality, emotional intelligence, deep learning capabilities, unlimited scalability, and multilingual support, among others. Among these attributes, the foremost importance lies in the ability of a chatbot to effectively mimic human conversation, a topic elaborated upon in the following sections [96].

2.2.1 Categorization of Chatbots

For simplicity, chatbots can essentially be categorized into two types; Pattern-based and Artificial Intelligence-based bots. *Pattern-based chatbots*, such as the already mentioned pioneering ELIZA, rely on pattern-matching techniques for text classification and generating responses. They operate by following predefined conversation paths related to a decision tree, resulting in predictable and repetitive responses that lack human feeling. Typically, these chatbots initiate conversations by posing questions to users, with responses limited to predefined alternatives or specific keywords. They can only provide accurate responses when the input precisely matches their training data, and they cannot retain or recall earlier parts of the conversation, often leading to conversational deadlocks.

```
  Welcome to
                    EEEEEE  LL      IIII   ZZZZZZ   AAAAA
                    EE      LL       II       ZZ   AA   AA
                    EEEEE   LL       II     ZZZ    AAAAAAA
                    EE      LL       II    ZZ      AA   AA
                    EEEEEE  LLLLLL  IIII ZZZZZZ    AA   AA

  Eliza is a mock Rogerian psychotherapist.
  The original program was described by Joseph Weizenbaum in 1966.
  This implementation by Norbert Landsteiner 2005.


ELIZA: Is something troubling you ?
YOU:   Men are all alike.
ELIZA: What is the connection, do you suppose ?
YOU:   They're always bugging us about something or other.
ELIZA: Can you think of a specific example ?
YOU:   Well, my boyfriend made me come here.
ELIZA: Is it important to you that your boyfriend made you come here ?
YOU:   He says I'm depressed much of the time.
ELIZA: I am sorry to hear that you are depressed.
YOU:   It's true. I am unhappy.
ELIZA: Can you explain what made you unhappy ?
YOU:
```

Figure 2.1: Conversing with ELIZA - The most famous script "DOCTOR" simulating a psychotherapist of the Rogerian school [34].

Artificial Intelligence-based chatbots have gained significant traction due to their machine learning capabilities, enabling them to not only match user input with appropriate responses but also to comprehend context and make predictions. These chatbots are widely employed in optimizing sales consultations, customer support, personal assistance, and more. AI-based chatbots leverage algorithms trained on historical data derived from real user interactions. Their capacity to grasp the context of a message allows them to engage more naturally in conversations without requiring explicit training for specific tasks. Consequently, they can continually adapt and improve based on real-time user feedback [96]. Additionally, *Hybrid chatbots* are also present. They combine the advantages of both AI and rule-based systems such that they are created to provide specific responses to user queries, while also utilizing NLP to comprehend user intent. Although hybrid bots address some limitations of rule-based counterparts, the maintenance of rules becomes challenging as the bot's complexity increases. It is important to note that categorizing bots strictly as either hybrid or AI is challenging since most bots incorporate some rules. For simplicity, hybrid chatbots are often viewed the same as AI chatbots [29].

2.2.2 Chatbot Architecture

As chatbots are software applications there exists naturally no universal recipe all developers must adhere to. However, the general chatbot architecture seems to consist of these five main components; a User Interface (UI) component, a Natural Language Understanding (NLU) component, a Dialog Manager component, a Back-end component, and lastly a Natural Language Generation (NLG) component [5]. The architecture is illustrated in Figure 2.2 below.

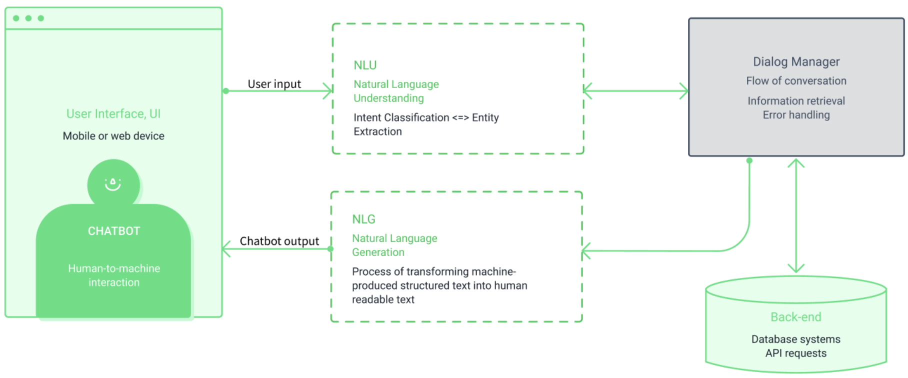


Figure 2.2: Architecture of chatbots [5].

User Interface

The UI component is the front-end of the application and is what enables the user to interact with the bot. It is commonly displayed in the shape of a chatbot widget on websites or integrated into various messaging platforms like Facebook, Telegram, and WhatsApp. For speech-based assistants, it can take shape into the form of an object, such as Amazon's Alexa, or on the phone itself, for example, Apple's Siri. The UI receives all of the user input and thereafter passes it onto the next step, the NLU component.

Natural Language Understanding

Natural Language Understanding is a sub-field of Natural Language Processing (NLP) - this term is further explained in section 2.5 - which focuses on understanding the context and meaning behind human speech by recognizing various patterns in unstructured user input. After receiving the input, the NLU component produces a semantic representation of the user utterance, by identifying the **intent** and then extracting the **entities** from that intent [42] [5].

- Can I order a *pepperoni* pizza?
- Translate *hello* to *Norwegian*
- I would like a flight ticket to *India*

The list above visualizes different intents and the entities extracted from that intent shown in italicized text. More specifically, the intent represents the intention between the user request and the action the bot should take. So for the first example in the list above, the user wants to order a pizza, and the bot should then take action in ordering the pizza or ask follow-up questions regarding delivery, time, etc. An entity, however, is domain-specific parameters extracted from the intent. For the same sentence, the entity will be what type of pizza the user wants to order, in this case, a pepperoni pizza.

Dialog Manager

The Dialog Manager component is a vital component that is responsible for the flow of the conversation between the bot and the user. It has the responsibility of communicating with other components as well as determining the next necessary action or response a system should take in response to the semantic interpretation of the user utterance sent from the NLU component. There exists no agreed definition of a dialog manager's tasks. However, Traum & Larsson (2003) [94] defines these four tasks to be fundamental to a dialog manager system:

1. Updating the dialog context on the basis of the interpreted text. This includes the details about the current state of the conversation; the intents and entities.
2. Provide a context-dependent exception for the observed signals in a conversation. For example, the dialog manager is expected to consider whether the input is a question or a statement, and then respond accordingly based on that context.
3. Coordinating with other components, this can be a database, an API request, execution modules, and other back-end systems.
4. Deciding what information to convey and when to express it.

Thus, the dialog manager component has a wide area of responsibility and must be able to handle diverse sources of information, such as the result of database queries and API requests, as well as keeping track of past dialog history. The complexity of the dialog manager is based on the tasks and the initiator of the conversation, whether it be the user, the bot, or both. Furthermore, it has the ability to request missing information, ask for clarification from users, and pose follow-up questions. In conclusion, it has the main responsibility for governing the actions of the chatbot [63] [42] [94].

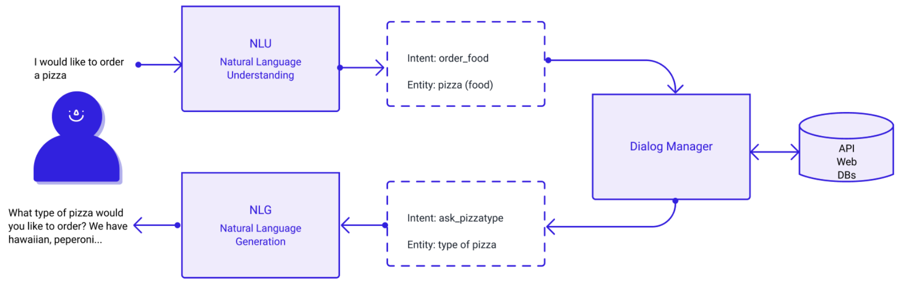


Figure 2.3: An emphasis on the actions performed by the dialog manager, exemplified through a modular-architecture visualization of chatbot [43].

As visualized in the modular architecture above in figure 2.3, the dialog manager is able to produce the appropriate response to user utterances [43].

Back-end

Once the user request is understood, the system proceeds with action execution and information retrieval. This is the responsibility of the back-end component, commonly referred to as the knowledge base. It serves as the selected repository of information the chatbot needs to extract essential data in response to user queries. The knowledge base takes form in different data sources, which can be a database or external sources that can be accessed through an API call [5]. As illustrated in figure 2.3, the bot responds with a list of available pizzas the user can order. The types of pizza are either extracted from a database or an API call.

Natural Language Generation

Natural Language Generation (NLG) is the process of transforming machine-produced structured data into comprehensible human text. The NLG process unfolds through six steps to produce a response:

- Content Determination: Filtering through existing data in the knowledge base to decide what to incorporate into the response.
- Data interpretation: Focus on understanding and grasping the patterns and answers present in the knowledge base.
- Document planning: Organizing the response in a narrative fashion.
- Sentence aggregation: Involves compiling the expression and words for each sentence within the response.
- Grammaticalization: Applying grammatical rules, for instance, punctuation and spell check.
- Language implementations: Entering the data into language templates to ensure a response that is naturally represented.

The NLG techniques above offer insights into constructing symbiotic systems that take advantage of the knowledge and capabilities of both humans and machines [29].

2.3 Human-to-Machine Interaction

Human-Machine Interaction (HMI) establishes the fundamental interaction and communication between human users and machines through a human-machine interface, a connection represented in figure 2.4. In professional settings, the effectiveness of HMI directly influences user experience, system performance, and overall productivity.

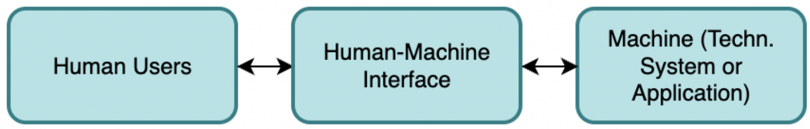


Figure 2.4: Human Machine System with human users, human-machine interface, and the machine

Efficient HMI design involves the creation of interfaces that seamlessly narrow the distance between human cognition and machine functionality. A machine can be defined as *"any mechanical or electrical device that transmits or modifies energy to perform or assist in the performance of human tasks"* [61]. This process requires a thorough understanding of user needs, cognitive abilities, and ergonomic considerations. Designers must prioritize clarity, simplicity, and intuitiveness to ensure that users can interact with machines in a direct manner. An important part of HMI is user interfaces, the selection of the appropriate interface depends on the specific context, user requirements, and the nature of the tasks at hand. In professional environments, HMI design often focuses on minimizing cognitive load and optimizing task efficiency. Clear and concise feedback, error messages, and prompts contribute to a smoother user experience [15].

Advancements in Natural Language Processing (2.5) and Machine Learning have opened new opportunities for more complex human-machine interactions. Conversational interfaces, where users can interact with machines using natural language, have gained importance in various applications. Enhancing HMI necessitates a comprehensive understanding of the unique attributes and capabilities inherent in each entity. Humans have distinct qualities that machines do not, such as creativity, intuition, empathy, and common sense. Humans quickly adapt to new situations, while machines require training and reprogramming. However, capabilities such as processing speed, accuracy, and consistency, are something machines have in advantage. Correspondingly, large amounts of data and the performance of complex calculations are executed far more efficiently by machines, rather than humans. Another important notice is that machines operate without being the subject of human limitations such as fatigue, boredom, or emotional bias [65].

These interfaces require careful design to ensure an accurate understanding of user inputs and appropriate system responses. Professional human-to-machine interactions demand a thorough approach to design, taking into account user needs, cognitive abilities, and ergonomic considerations. The HMI design approach's objective is to create interfaces that facilitate efficient communication, enhance user experience, and contribute to the seamless integration of humans and machines in various contexts [15].

2.4 Formal and Natural Languages

Formal languages refer to a precise and structured way of representing information using symbols and rules. These languages are commonly used in several fields, including computer science, mathematics, and linguistics. They serve as a method to describe and communicate complex concepts and processes in a systematic and explicit manner. Formal languages are characterized by some principal components including the alphabet, syntax, semantics, and grammar. Programming languages, formal logic, automata theory, and natural language processing, among others, all apply formal languages. They provide an exact and definite way to represent and manipulate information, making them essential in multiple areas of computer science and mathematics[64] [83].

Examples of formal languages can include regular expressions. Regular expressions are often used for text matching along with extraction and are a powerful tool for defining patterns in strings. They are widely used for text processing and searching. Some common regex patterns include matching email addresses, and phone numbers for specific formats and extracting URLs from text [82]. Another example is context-free grammars (CFGs), they are used to describe the syntax of many programming languages and are fundamental for parsing. It is a set of recursive rules utilized to generate patterns of strings. CFGs serve as a means to define programming languages and parsers for compilers can be automatically derived from these grammars [17]. An example CFG for a simple arithmetic expression language might look like this:

```
<expression>  → <expression> + <term> | <expression> - <term> | <term>
<term>        → <term> * <factor> | <term> / <factor> | <factor>
<factor>      → ( <expression> ) | <number>
<number>      → 0 | 1 | 2 | ... | 9
```

Unlike formal languages, natural languages were not designed by people but evolved naturally over time. These languages are the ones people speak, for instance, English, Spanish, or French. While formal and natural languages share commonalities like tokens, structure, syntax, and semantics, there are still several differences. Natural languages abound with ambiguity, an aspect dealt with by individuals through contextual clues and supplementary information. Formal languages, on the other hand, are designed to be nearly unambiguous, with each statement holding precisely one meaning independent of context. To compensate for inherent ambiguity, natural languages rely on redundancy, resulting in verbosity. Formal languages indicate exactly what they express. Furthermore, natural languages are filled with phrases and metaphors [1].

2.5 Natural Language Processing

Natural Language Processing (NLP) is an aspect of Artificial Intelligence (AI) that enables computers to understand, interpret, and effectively utilize human languages. It facilitates human-like communication for computers, attempting to close the gap between human and machine communication. This interdisciplinary field draws from computational linguistics and computer science to enhance the interaction between human and computer conversations by also providing computers with the ability to read text, listen to and understand speech, and interpret it. NLP breaks down language into smaller, fundamental units known as tokens, which include words, punctuation marks, etc. It goes beyond mere linguistic analysis to comprehend the relations and nuances between these tokens, striving to make human-machine communication more intuitive and context-aware. By converting the unstructured conversational human language into structured data for the machine to interpret, NLP allows chatbots to recognize the context and meaning behind users' text input and capture their intents. The process of Natural Language Processing involves two primary phases, including preprocessing and algorithm development [41] [60].

2.5.1 Preprocessing

In order for the machines to be able to analyze the unstructured data from human conversations, the input text needs to be prepared and cleansed. Preprocessing transforms data into a workable form and focuses on features from the input that the algorithm is able to work with. There are multiple ways this can be done, including the ones listed below [60].

Tokenization

Tokenization, also called lexical analysis is a process where one divides a string of words into lesser parts known as *tokens*. Tokens can be words, characters, or subwords based on the intention and relation to the entire sentence. Consider a string of text, such as "What restaurants are nearby?" To make this sentence machine-readable, tokenization is performed to divide it into separate components. Through tokenization, the original sentence is transformed into a series of individual words: 'what', 'restaurants', 'are', and 'nearby' [73].

Part-of-Speech Tagging

For each token in a sentence, one predicts whether the word is a noun, verb, adjective, adverb, pronoun, etc. In doing this, it will help in understanding the meaning and context behind the sentence. One can achieve this by feeding the word and the words around it to a pre-trained part-of-speech classification. For instance, consider the sentence "I like to read books". It can be represented in a structured list like this; [("I", "Preposition"), ("like", "Verb"), ("to", "To"), ("read", "Verb"), ("books", "Noun")] [53].

Lemmatization and Stemming

Both lemmatization and stemming have a common goal, which is transforming the word back into its root form. Lemmatization does this by conversing a word to a root by removing suffixes [66]. Stemming achieves its goal by cutting off the ends of a word, and it often involves removing derivational affixes. Lemmatization is considered to be more complex, considering it needs a high degree of knowledge of a language, and performs both a vocabulary and morphological analysis of words. For instance, stemming the word 'Caring' would return 'Car', and lemmatizing the word would return 'Care' [88] [95].

Stop Words Removal

Frequently used words such as 'a', 'and', and 'the' can create a lot of interference in statistical analysis. Such words are called "stop words". By removing these words, we are left with the ones that provide the most relevant information about the text [53].

2.5.2 Algorithm Development

After the input data has been preprocessed, an algorithm is created to handle it. There exist many different NLP algorithms, but these two main types are commonly used:

Ruled-based systems

This is a system that uses carefully designed linguistic rules. This approach was used early on in the development of NLP and is still used [60]. The rule-based approach involves the application of a predefined set of rules or patterns to identify specific structures, extract information, and perform tasks like text classification. Common rule-based techniques include regular expressions and pattern matching. The first step in this approach is the creation of rules, tailored to the desired tasks. These rules can encompass domain-specific linguistic patterns such as grammar, syntax, semantics, or regular expressions. The established rules are then applied to incoming data to identify matching patterns. Following this, the text data is processed based on the outcomes of the matched rules to extract information, make decisions, or perform other tasks. The created rules undergo iterative refinement through repeated processing to enhance accuracy and performance. Feedback from previous iterations informs adjustments and updates to the rules when necessary [84].

Machine learning-based system

This is a system that utilizes statistical methods through the agency of machine learning algorithms. These algorithms learn to perform tasks based on training data they are fed, thereafter adjusting their methods as more data is processed. By using a combination of machine learning, deep learning and neural networks, natural language processing algorithms hone their own rules through repeated processing and learning [60].

2.6 Language Models

Language Modeling (LM), involves employing statistical and probabilistic techniques to estimate the likelihood of a specific sequence of words occurring within a sentence. These models examine extensive text datasets to establish the foundation for predicting words in a given context. The field of language modeling finds applications in artificial intelligence, natural language processing, natural language understanding, and natural language generation systems. This is particularly relevant for systems engaged in tasks such as text generation, machine translation, and question answering.

Among the many current language models, one prominent example is BERT, which stands for Bidirectional Encoder Representations from Transformers. BERT is an open-source machine learning framework specifically tailored for NLP tasks. The language model is renowned for its ability to grasp the contextual meaning of ambiguous language in text considering the surrounding text to establish context. With pre-training from textual sources like Wikipedia, BERT can be fine-tuned with question-and-answer datasets [102].

2.7 Large Language Models

A deep learning algorithm known as a large language model, or LLM for short, is capable of handling a wide range of NLP tasks. These models are based on neural networks, computational systems functioning through layered nodes, resembling neurons in the human brain. LLMs employ transformer models and are trained on extensive datasets, resulting in their ability to recognize, translate, predict, or generate text and other content. Apart from instructing AI applications in human languages, large language models can also be trained for various tasks, such as comprehending protein structures and coding software. Similar to the human brain, these models undergo pre-training and fine-tuning processes to excel in text classification, question answering, document summarizing, and text generation tasks. Their problem-solving capabilities find applications across diverse fields like healthcare, finance, and entertainment, where they power numerous NLP applications like translation, chatbots, AI assistants, and more. These models substantial number of parameters, which serve as their knowledge repository, related to memories acquired during the training process [100].

2.7.1 GPT-3 and GPT-4

Founded in 2015 by Sam Altman, Elon Musk, and others, OpenAI is an AI research and deployment company with the overarching mission of promoting and developing AI that can benefit all of humanity [46]. One of its most known accomplishments is the announcement of language model GPT-3 in 2020, now being widely used in the chatbot ChatGPT, further detailed in the following section 2.7.1. GPT-3, an abbreviation for Generative Pre-trained Transformer, is a third-generation, autoregressive language model that employs deep learning to produce text that resembles the human language. *Generative* meaning it is capable of generating coherent and contextually relevant text. *Pre-trained* defining how GPT models are trained on massive compilations of internet data.

This involves predicting the next word in sentences, given the previous words. GPT models acquire knowledge about grammar, syntax, semantics, and even somewhat factual information by learning from billions of sentences. *Transformer* describing the neural network architecture that is shortly mentioned. The transformer architecture is a type of neural network that exceeds at handling sequences of data. A mechanism of self-attention allows it to weigh the importance of different words in a sentence based on context. By the model being autoregressive, it means it is able to predict future values based on past values.

To put it simply, as outlined by Floridi & Chiriatti (2020) [39], it is essentially a computational system engineered to produce sequences of words, computer code, or other data starting from an initial input, the prompt. The large language model undergoes training on an extensive, unlabelled dataset that is made up of text, for instance, web texts, books, and Wikipedia. By being trained on these massive data sources the language model is able to produce relevant results. As of today, GPT-3 uses 175 billion parameters and is trained on the Microsofts Azure's AI supercomputer [85], with an estimated cost of $12 million. This large language model works for a broad spectrum of use cases, including translation, grammar correction, chatbots, email composition, and much more.

On the 14th of March 2023, OpenAI released the much-anticipated successor to GPT-3, the GPT-4. While its architecture and training methods appear similar to its predecessor, there have been notable advancements in fine-tuning techniques, as well as a significant expansion of the training dataset. A key difference lies in its size; GPT-4 proves to be one of the largest language models ever created, boasting a staggering 1.76 trillion parameters. Furthermore, the language model proves to be multi-modal in the sense it can not only process text, but also images as input [77]. However, the output format remains the same as in being text-based.

ChatGPT

Chat GPT, or Chatbot GPT is an artificial intelligence conversational agent powered by the large language models *GPT* developed by OpenAI[72]. The GPT models are constructed using deep learning techniques, specifically a neural network architecture, which results in the GPT-3 model closely resembling human language. The further advanced GPT-4 model is a great deal more reliable, creative, and nuanced and is available through a paid subscription to the service. Its architecture enables GPT to capture long-range dependencies and understand context effectively. The ChatGPT chatbot is designed to interact with and assist users in great measures, including answering questions, generating content, providing explanations, offering recommendations, and more. Since its official launch in November 2022, the conversational agent has been widely adopted in customer service, content generation, education, and other various domains where natural language understanding and generating are crucial [38][50].

2.7.2 PaLM 2

On the 10th of May 2023, Google released PaLM (Pathways Language Model) 2, a language model that is a worthy rival to OpenAI's GPT-3 and GPT-4. This model resulted from a large, collaborative endeavor involving multiple teams within Google Research. The language model encompasses 540 billion parameters and is capable of handling a large variety of tasks, including complex learning, reasoning, translations, code completion, and common sense reasoning to name a few. PaLM 2 operates as a decoder-only Transformer and is finely trained by Google's own Pathway system; the Pathway system being an advanced AI architecture designed to handle many tasks at once [25]. The model is also trained on parallel multilingual texts and a substantially larger corpus of various languages, thereby making it excel on multilingual tasks and able to serve a global audience. Moreover, PaLM's pre-training phase also encompassed a large number of web pages, source codes, and diverse datasets. This made it perform well in well-known programming languages like JavaScript and Python [68] [79].

Figure 2.5: PaLM illustrating natural language understanding by identifying the movie 'Wall-E' from emoji prompts [68].

PaLM is able to demonstrate impressive natural language understanding and generation capabilities. To exemplify this, take the snapshot in figure 2.5 above [68]. The prompt gives a list of five movies and asks the model to identify the movie by describing it with emojis. The emojis are a robot, a cockroach, a plant, and the globe. The model correctly identifies the movie Wall-E from the received emojis!

2.7.3 Gorilla: Advancements in LLM for API usage

Large Language Models have experienced notable progressions, showcasing their proficiency across a range of tasks, including mathematical reasoning and program synthesis. However, the effective utilization of tools via API calls has remained a challenge for even the most advanced LLMs, such as GPT-4. This challenge primarily arises from their inherent limitations in generating precise input arguments and their occasional tendency to produce incorrect API call usages [51].

In response to these challenges, the fine-tuned Language Model for Mathematical Reasoning (LLaMa) - based model, *Gorilla* has been introduced. The model aimed at surpassing the performance of GPT-4 in creating API calls. When combined with a document retriever, Gorilla exhibits an ability to adapt to changes in test-time documents. This adaptability facilitates flexible user updates and accommodates version changes with ease. Furthermore, Gorilla eases the issue of *hallucination*, a common problem when directly prompting LLMs. Hallucination refers to a situation where the model generates content that it believes to be coherent and contextually relevant, but it is entirely fabricated and does not correspond to reality.

To assess the capabilities of Gorilla, the research introduces "API Bench", a comprehensive dataset encompassing HuggingFace, TorchHub, and TensorHub APIs. The integration of the retrieval system with Gorilla demonstrates the potential for LLMs to employ tools more accurately. Additionally, this approach enables them to keep pace with frequently updated documentation, ultimately enhancing the reliability and applicability of their outputs. Gorilla marks a step forward in harnessing LLMs for effective API usage and addresses the associated challenges in an objective manner [74].

2.8 Chatbot Platforms

In the domain of chatbot platforms, we conducted research to identify and understand the platforms' capabilities to help discover which ones align with our project's requirements and objectives. Our examination focused on platforms that contribute to efficient bot development, considering user-friendliness, flexibility, and integration capabilities. This section provides insights into prominent chatbot platforms, each emerging as suitable contenders for this project.

2.8.1 XatKit

XatKit started as a collaborative research project between Jordi Cabot, an ICREA research professor and a researcher from the SOM Research Lab, and Gwendal Daniel, also a researcher from the SOM Research Lab. Its project's goal was to help others in creating bots better and faster. Today, it is a spin-off of ICREA and the Universitat Oberta de Catalunya and has offices both in Barcelona and Nantes and operates as a fully remote company with contributors and projects from all around the world [13].

XatKit has evolved into an innovative platform that permits users to develop a wide range of digital assistants, including bots, chatbots, and voicebots. Built on an open-source engine, XatKit offers a user-friendly experience with its out-of-box functionality [106]. One of the key advantages of XatKit is its ability to simplify bot development by reducing boilerplate code, complex API understanding, and technical complexities. XatKit allows users to focus on crafting effective conversation logic for their chatbots. The platform introduces a bot-specific definition language, enabling users to specify user intentions, receive events, and define actions using state machine semantics. This language, implemented as a Java Fluent Interface, combines the benefits of low-code development with the flexibility of Java for more advanced bot behaviors. The XatKit Runtime Engine handles the deployment and execution of chatbot specifications, while also providing the flexibility to integrate with existing platforms such as Slack, GitHub, and Telegram [24].

2.8.2 Rasa Open Source

In Berlin in 2016 Alan Nichols and Alex Weidauer co-founded Rasa, an open-source chatbot framework [59]. Today, the framework has over 25 million downloads and proves to be the most popular open-source platform for building chat- and voice-based AI assistants [52]. Rasa is a tool utilized to build custom AI chatbots with the use of Python programming language and natural language understanding. It provides different NLU training models, bot prototypes, as well as tutorials and guidance. The architecture of Rasa consists of two components:

- **Rasa NLU** – An open-source natural language processing tool designed to comprehend the meaning behind the user input. This is the library for intent classification and entity extraction from the query text.
- **Rasa CORE** – The dialogue engine for the framework. Its job is to determine the next best action to perform.

The two components are independent of each other and can be used separately. Rasa also contains multiple deployment channels and integration, and bots built using Rasa today can be deployed on multiple platforms i.e. Facebook and Slack [16].

2.9 Natural Language Understanding Platforms

In our research to develop chatbot technology, we examined various NLU platforms to determine the most suitable ones for our application. Providing an NLU platform enables natural language understanding and effective communication with users. The following sections will provide a brief introduction to DialogFlow and XatKit NLU Server, the platforms we saw as the most fitting options for our application.

2.9.1 DialogFlow - A Natural Language Understanding Platform

DialogFlow is a cloud-based NLU platform developed by Google. It allows developers to create conversational interfaces for diverse applications and services. DialogFlow is used to build chatbot agents, voice-activated applications, and other natural language interfaces. The platform is designed to understand and process user input in the form of text or voice and generate appropriate responses, making it a valuable tool for developing conversational AI applications and chatbots. DialogFlow provides tools for intent recognition, entity recognition, and context management, allowing developers to create dynamic and context-aware conversational experiences. They provide two different virtual agent services for chatbots, DialogFlow CX and DialogFlow ES, the first provides an advanced agent type suitable for large and complex agents. The second provides the standard agent type suitable for small and simple agents. [28].

2.9.2 XatKit NLU Server

XatKit introduces a flexible and pragmatic approach to NLU in the realm of chatbot development. In the context of chatbots, NLU involves understanding the intention or intent behind a user's input. This intent recognition is central for chatbots to effectively respond to user queries, making it a central component for chatbot functionality. In the world of NLU, intent classifiers are typically implemented as neural networks. These classifiers assess user utterances and determine the probability that the user's input corresponds to a specific intent. For instance, in an FAQ-style chatbot, the user's question needs to be correctly matched with the intent to provide an accurate response.

XatKit's NLU Engine stands out from other NLU chatbot engines by offering flexibility to tailor the NLU model to specific semantics of the chatbot, ensuring that it delivers the desired results. While customizing existing platforms can be challenging, XatKit's engine simplifies the process, by providing a valuable resource for developers aiming to fine-tune their chatbot's performance [108].

Chapter 3

Related Work

This chapter explores prior research closely tied to the themes addressed in this thesis. Adopting a top-down approach, we aim to provide context and clarify the foundational work that laid the groundwork for our own contributions.

3.1 Chatbots Enabling Open Data Accessibility

In the domain of open data accessibility and utilization, studies such as *Talking Open Data* from 2017 [69] and *Open Data Chatbot* from 2019 [56] offer insights that are closely related to the objectives of this thesis. These researches, conducted in the context of open data, explores the potential of employing conversational interfaces, specifically chatbots, as a means to democratize access to open data sources.

3.1.1 Talking Open Data

The study delves into the intersection of open data and chatbot technology, shedding light on the advantages and challenges of using chatbots to make open data more accessible for a broader target group, including people with less technical experience. Their solution involves a designed open data search interface supporting natural language interactions via platforms such as Facebook. The chatbot they provide answers search requests and suggests relevant open datasets to the users. The prototype incorporates 18,000 datasets sourced from seven distinct Open Data portals. These datasets encompass descriptions in seven different languages. When a user provides an entity within their search query, the prototype fetches all dataset descriptions annotated with the specified entities. It then aggregates information on the most frequently co-occurring entities. The interaction can occur in two modes. Firstly, there's a free-text search query option where datasets are ranked based on the number of matching entities. The chatbot then returns the API with the highest matching dataset. For instance, a user might ask a question like "How many dogs live in Vienna?". In response, the chatbot would provide the API where such information can be found. Secondly, the user can refine the search results by selecting one of the top co-occurring concepts and entities to further filter the result [69]. This example is visualized in figure 3.1 below.

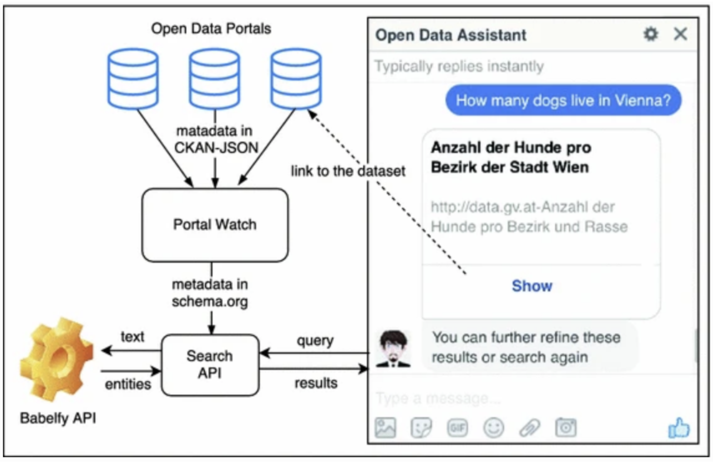


Figure 3.1: The Talking Open Data Chatbot [70].

3.1.2 Open Data Chatbot

Similar to the *Talking Open Data* research, this study investigates an Open Data chatbot prototype. It integrates cutting-edge parsing and semantic technologies to create a conversational search application for a repository of publicly available datasets utilizing geo-entity annotations. This research offers users adaptable interaction modes within the chatbot's interface, as illustrated in figure 3.2. Users can opt for either a free-text search query mode or an exploratory mode. In the former, datasets are ranked based on entity matches, with the chatbot returning the API associated with the most relevant dataset. In the latter, users engage in a guided dialogue, refining their queries and accessing specific datasets based on their preferences [56].

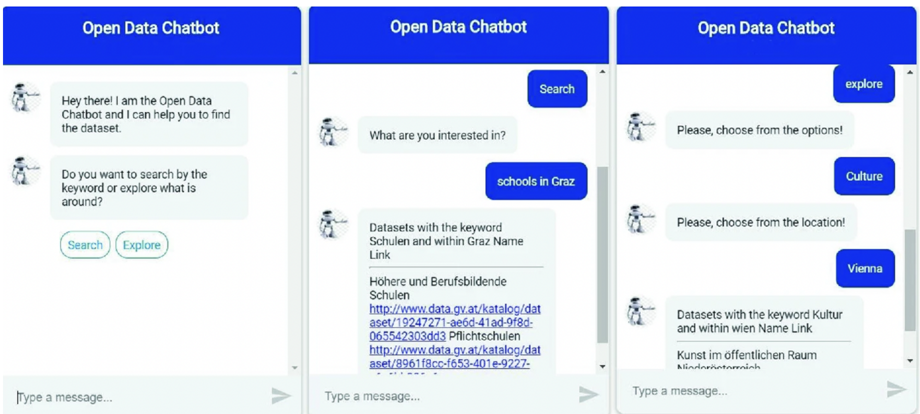


Figure 3.2: The Open Data Chatbot [57].

3.1.3 Relation to Our Work

Much like our thesis, these studies share the overarching objective of enhancing open data accessibility for individuals with limited technical expertise. While our thesis concentrates on empowering users to provide and discover open datasets for personal interpretation and inquiry, these studies pursue a similar goal through a distinct approach.

3.2 Automated Chatbots and Structured Data Exploration

In addition to chatbot-driven solutions for open data accessibility, several projects have emerged with a similar goal, although with a focus on structured data sources. These initiatives leverage chatbots to provide conversational interfaces that simplify the exploration of structured data tables, making them more accessible to a wider demographic. In this section, we delve into two such projects *Accessing Government Open Data Chatbots*[76], and the *BODI* project[44]. While these projects share common ground in utilizing chatbots, they offer unique approaches and insights into facilitating data interaction, particularly with tabular datasets. Let's explore how these endeavors are contributing to the evolution of data accessibility and how they compare with our thesis project

3.2.1 Government Open Data Access

In the domain of government open data accessibility, the study *Accessing Government Open Data Through Chatbots* by Porreca et al. presents an innovative approach [76]. The authors propose the utilization of chatbots as an interface to interact with open data, with a specific focus on data published by public administrations. The motivation for this work arises from the recognition that while open data holds significant potential for e-government initiatives, it remains underutilized due to the lack of user-friendly access methods. Their system, built on the Open Cantieri dataset, illustrates the process of making structured data more available through conversational interfaces.

The chatbot interface is implemented through Facebook Messenger, while the core back-end infrastructure operates on the IBM Bluemix Cloud Platform. A Node.js application coordinates two Bluemix service instances: Watson Conversation for query processing and Compose for MySQL for database interaction. User interactions in the chatbot interface follow a structured flow: users input queries, which are processed by the Node.js application, and then passed to Watson Conversation for response generation. Based on the response, the application constructs an SQL query for Compose for MySQL, which retrieves data. The user's response is generated and delivered through the Facebook Messenger chat [76].

3.2.2 Bots for Open Data Interactions (BODI) - Chatbot Generator for Open Data Sources

Another notable approach in the realm of open data accessibility is the Bot-Driven Interaction (BODI) method, which seeks to democratize access to tabular data through conversational user interfaces (CUIs). This approach is also built through XatKit and has been a source of inspiration for our project's approach. BODI recognizes that while tabular data, often found on open data portals, is a rich source of information, its utility has historically been restricted to individuals with technical expertise. The work presented in the BODI article [44] proposes an innovative solution to address this limitation.

The core idea behind BODI is to employ chatbots as conversational interfaces to facilitate the exploration of tabular data sources. Unlike manually created chatbots, BODI's chatbots are automatically generated from the data source itself. In contrast, our thesis project involves the design and development of a chatbot tailored to a more specific use case and user interaction. BODI primarily targets tabular data sources from CSV files, focusing on structured rows and columns, whereas our approach primarily deals with diverse open sources, including JSON files accessed through APIs, encompassing a wider range of formats and types.

3.2.3 Relation to Our Work

In summary, our thesis study, *Accessing Government Open Data Through Chatbots*, and the *BODI* project all share a common ambition of making data more accessible. *AGODTC* focuses on utilizing chatbots to interact with structured government data, primarily through guided dialogues, while *BODI* primarily targets tabular data sources with an emphasis on automated chatbot generation.

In comparison, our project extends the accessibility scope beyond structured data to encompass open data sources such as JSON files through API's. Our approach emphasizes empowering users to initiate queries and interactions, allowing for more user-driven exploration of the data. Whereas *AGODTC* excels in providing guided dialogues for specific data retrieval. In essence, while sharing the goal of open data accessibility, and using the potential of chatbots as a valuable tool to reach it, each project adopts a unique approach to address specific data types and user interaction styles.

Chapter 4

Design and Implementation

In this chapter, we delve into the design and implementation phase of our project. We will explore the development methods, technologies, and solution approach employed in the creation of our chatbot *Botty*, and related tools. This chapter provides an in-depth look at the strategies and techniques used to bring our project to life, outlining the methods and technologies that contributed to its execution.

4.1 Development Method

In this section, the emphasis is on the practices and methodologies that shaped this project as a whole. Given that this master thesis is a collaborative effort in a group of two, it is essential to maintain a structured and manually beneficial approach to teamwork. The following sections will provide insight into the methodologies and tools employed to ensure efficient cooperation.

4.1.1 Agile Methodology

Agile software development is a software development approach and philosophy that values flexibility, cooperation, and end-user satisfaction. It is built upon the Agile manifesto which comprises a set of principles that emphasizes the importance of individuals and their interactions, the production of functional software, collaborating closely with end-users, and being able to adapt to changes. The agile methodology is an iterative and incremental approach to software development and one of its main importance is delivering a working product fast and frequently [67]. Therefore, the adaptation of Agile practices seemed suitable during the creation of our bot software. It was decided to build the software incrementally from the start of the project and extract the coding phase throughout the entire lifespan of the project. Thus, allowing for frequent feedback and opportunities for changes whenever required. However, agile serves mainly as a philosophy and can be implemented through the use of a framework. In the context of this project, it was adopted the Scrum framework.

Utilizing Scrum for Iterative Development

The definition of Scrum is founded on empiricism and lean thinking. Empiricism asserts that knowledge comes from experience and underscores the importance of making decisions based on what is observed. Lean thinking centers around focusing on the essentials and only delivering values of necessity while conserving resources and minimizing waste. Scrum, as a framework, builds on the agile philosophy, thus employing an iterative and incremental approach, characterized by frequent small releases [31]. This framework operates heuristically, meaning it's based on continuous learning and adapting to changing factors. In utilizing Scrum, it highlights that we, as a team, may lack complete knowledge at the project's outset and instead rely on experiential growth to shifting circumstances and user needs [86].

Within this thesis, the Scrum methodology was not strictly followed. Instead, it served as a guiding principle and a source of inspiration for optimizing teamwork. We followed certain Scrum principles, particularly by working in sprints, where each sprint lasted for two weeks. At the beginning of each sprint, a sprint review was conducted, involving an examination of the outcome of the previous sprint. We inspected what was accomplished during the sprint and an analysis of changes or adaptations required. Based on this review, we collaborated on determining the next steps, thereby constituting the sprint planning phase. This planning phase consists of laying out the necessary work for the upcoming sprint, the methodologies for the work, and the rationale underlying the chosen tasks for the sprint [86]. These discussions were held in conjunction with our supervisor. The sprints included either mainly code tasks or thesis writing. This led us to have a full focus on one thing, as well as it allowed for fresh perspectives. Furthermore, ongoing feedback from supervisors benefited us and *Botty* in that it aided in discarding unnecessary solutions for the bot and identifying and prioritizing better, user-friendly solutions.

Trello

In achieving an overview of the progression of each sprint and its corresponding tasks, the application Trello was utilized. Trello, an online collaborative work management developed by Atlassian, stands as an effective solution for tracking team projects. The app is designed to visualize ongoing tasks, task assignees, and provides a comprehensive overview of the project itself, visualizing the workflow of a project spanning from start to finish. The fundamental components are boards, lists, and cards. The board is a visualization of the project itself. Within a board, there are lists that can be created to signify the various stages of project progression. For this thesis, the lists were named *Ideas*, *To Do*, *In Progress*, *PR Review*, and *Completed*. Each list has individual cards that contain information on a specific task and can be relocated between lists when required, such as when a task reaches completion. Cards offer a broad range of information, such as textual descriptions, attachments, comments from other users, and more. For this project, cards were aptly named to correspond with the current sprint, plus a suitable name describing the task – which could range from coding assignments to drafting chapters. Trello proved to be an excellent choice in that it provided a clear overview of the status of the current sprint and the project overall [37].

4.2 Technologies

Before discussing our approach for our contribution to narrowing the gap between people and open-source data through the development of a chatbot, we will first provide an overview of the technologies we deemed necessary to use during development. In order to create scalable and manageable software, it was necessary with a carefully arranged technology stack, with each component seamlessly layered atop another to craft a unified and functional program. The choice of the tech stack affects the type of application one can build, the extent of customization available, as well as the resources required for the development of the application [100]. Figure 4.1 presents the technology stack of the developed chatbot *Botty* in this project.

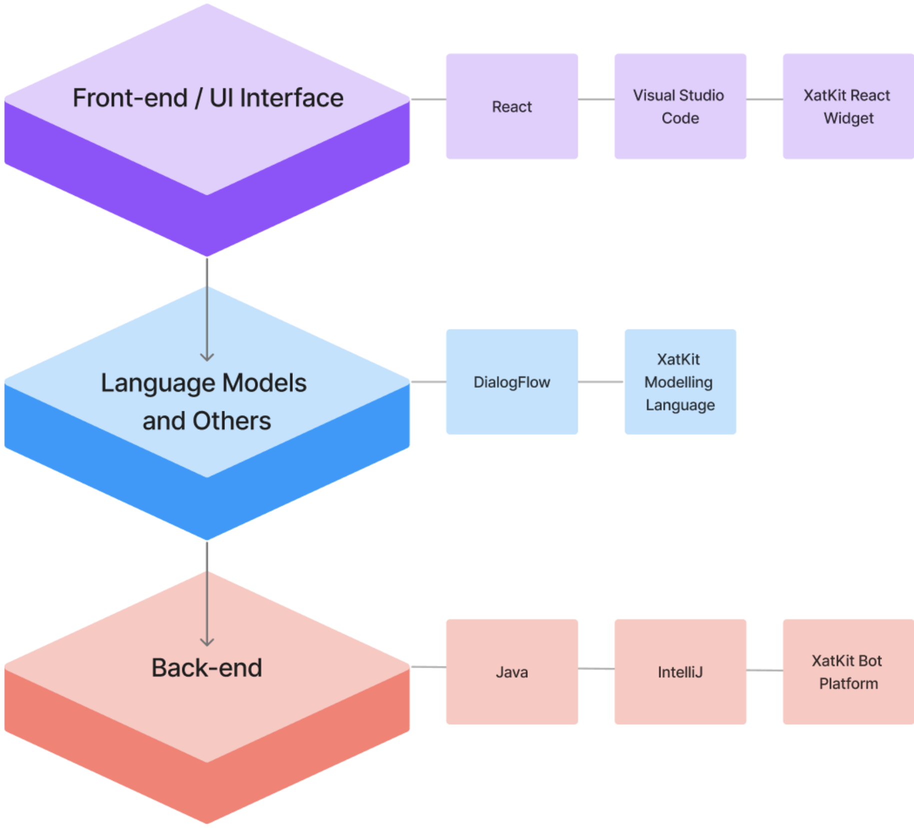


Figure 4.1: The technology stack of the bot application *Botty*.

The following sections will therefore provide an overview of the technologies used in this application.

4.2.1 React

The application's user interface was built using React. React is a JavaScript library and is renowned as one of the most commonly used front-end libraries for web development. It has a declarative and dynamic nature which facilitates the debugging process. The building blocks of any React application are the components and usually, a single application consists of multiple components. These components can be reused throughout the application and have their own logic. Thanks to its virtual DOM, React will update and render only the components affected by changes, thereby enhancing the application's speed and efficiency. XatKit's own bot component is a React widget, which made React an intuitive choice for the application's user interface [26] [80].

Visual Studio Code

Visual Studio Code, commonly referred to as VS code, is a lightweight but robust source code editor created by Microsoft and is compatible with operating systems Windows, MacOS, and Linux. It provides a rich feature set, including debugging tools, syntax highlighting, and an integrated source control system. VS Code consists of built-in support for JavaScript and Node.js, and it also offers a rich ecosystem of extensions catering to other languages and run-time environments. The embedded source control feature highlights a dedicated tab where the users can access version control settings, which in our case was GitHub - further detailed in the section 4.2.3 -, and view relevant changes made to the current project [90] [30].

4.2.2 Java

The programming language used for the development of the bot was Java. It is a widely used object-oriented, high-level, and class-based language that is able to run on many devices [104]. In order to make the bot function together with XatKit it was necessary to install and use the Java 8 version, as well as to install the JDK (Java Development Kit) version 1.8.

IntelliJ IDEA - The IDE of Choice

IntelliJ IDEA, a tool developed by JetBrains and written in Java, was the chosen IDE for this project. An IDE - short for Integrated Development Environment - is a software application designed to enable programmers to write code more efficiently and consolidate the different aspects of writing a computer program. It combines key activities of writing software, such as code editing, building executables, and debugging, all within a single application [101].

4.2.3 GitHub

In this project, we leverage GitHub for efficient code management and collaborative development. GitHub serves as a website and cloud-based service designed to aid developers in storing, managing, and overseeing changes to their code. It revolves around two core principles, version control and Git [103].

Version Control

Version control is central as software projects evolve. It is a system that enables developers to track and manage changes to a software project's code over time. Version control provides a structured method for multiple contributors to collaborate on a project by keeping a historical record of modifications. This allows developers to work on different parts of the code simultaneously, merge their changes seamlessly, and revert to previous states if necessary. Version control enhances collaboration, maintains code integrity, and facilitates efficient project management [7].

Git

Git, an open-source version control system, operates as a distributed version control system. This means that the entire codebase and history are available on every developer's computer, facilitating seamless branching and merging [103]. A substantial majority of developers, as per a Stack Overflow survey, employ Git [89].

GitHub

GitHub, a for-profit company, offers a cloud-based Git repository hosting service, simplifying Git usage for version control and collaboration. Its user-friendly interface accommodates learning coders, making Git more accessible. GitHub is widely adopted, with over 87% of developers utilizing Git according to the survey [89]. The platform allows anyone to host a public code repository for free, promoting popularity among open-source projects [103].

4.2.4 Domain-Specific Languages

Domain-specific languages (DSLs) are specialized computer languages designed to address specific, well-defined problems within a particular domain or context. Unlike general-purpose languages like Java or Python, which are versatile and applicable to a wide range of tasks, DSLs are tailored to handle specific tasks or challenges within a particular field or industry. They are very common in computing, some examples of this include CSS, regular expressions, and SQL among others. DSLs are created to make it easier for developers and domain experts to work together effectively. They offer a way to express concepts, processes, and rules in a manner that is intuitive and closely aligned with the specific domain's requirements. This simplifies communication between domain experts and developers, as DSLs use terminology and structures that are familiar to experts within that field. In the world of computing DSLs have a rich history and are widely used in various applications. They can be categorized into two main types: internal and external DSLs [32].

Internal DSLs

Internal DSLs, also known as embedded DSLs or fluent interfaces, are embedded within a general-purpose programming language. They leverage the host language's syntax and structures to create a more domain-specific, user-friendly interface. Internal DSLs are typically easier to implement and are often used in languages like Ruby and Lisp. While it is often easier to create internal DSLs in languages with low ceremony, like Ruby, it is also possible to implement effective internal DSLs in mainstream languages such as Java and C# [32].

External DSLs

External DSLs have their unique custom syntax, and you need to create a full parser to process them. External DSLs are employed when a highly specialized language is necessary for a specific domain, and the creation of a custom syntax is justified. This approach is common in the Unix community, and variations include encoding the DSL using data structure representation like XML or YAML [32].

Implementation methods

DSLs can be implemented using two main methods: interpretation and code generation. Interpretation involves reading the DSL script and executing it at run-time. This approach is usually more straightforward. In some cases, code generation is necessary, producing high-level code in languages like Java or C# [32].

4.2.5 Bot Platform - XatKit

As explained in 2.2 Chatbots, chatbots have evolved into software artifacts that cover various aspects, including from natural language processing to API integration with instant messaging platforms and third-party services. However, existing chatbot development platforms may struggle in the API department, and can be heavily coupled to external intent providers for NLP frameworks. This may hinder their maintainability and reusability to a great degree [106]. Therefore, we prioritized researching different bot platforms, as mentioned in 2.8 Chatbot Platforms.

At last, we chose XatKit to be the bot platform for the application, a development framework whose aim is to tackle a bigger aspect of these challenges stated. XatKit provides a range of DSLs that not only define the conversational logic of the chatbot but also enable the integration of third-party actions. These chatbots are designed to be platform-independent, allowing deployment across diverse messaging platforms and NLP engines [23]. Another firm advantage of XatKit is that it uses multiple programming languages we already were familiar with, which will be further discussed in section 6.1.3. Figure 4.2 showcases the architecture behind the XatKit Framework. The figure defines three core packages:

- **Intent Package** is used in defining the user's intentions through the utilization of training sentences, extraction of contextual information extraction, and matching conditions.
- **Execution Package** is responsible for associating the user intentions with actions that are part of the chatbot's behavior definition.
- **Platform Package** can specify the possible actions that can be performed in the intended target platform.

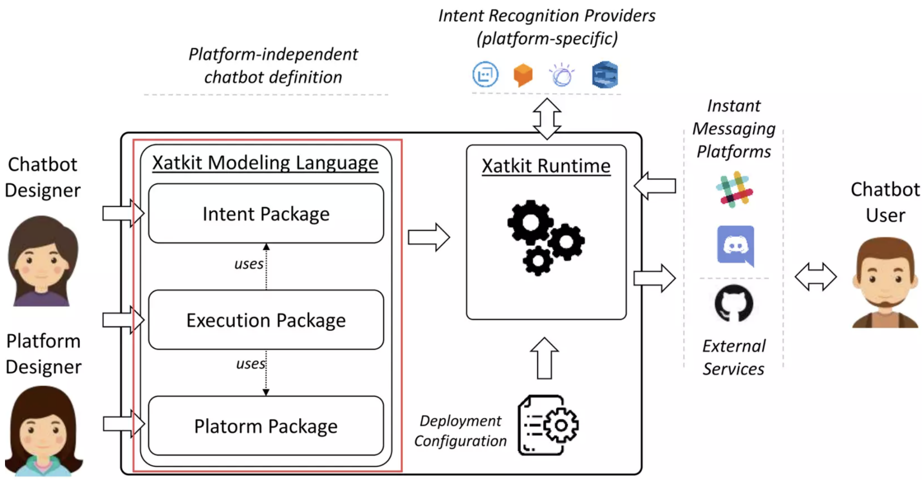


Figure 4.2: Overview of the XatKit Framework [23].

These packages form a model that is supplemented with a Deployment Configuration file responsible for specifying the Intent Recognition Provider platform to use and the platform-specific configuration, if any. Additionally, the file also specifies specific custom execution properties. All of these elements make up the input for the XatKit run-time component which is initiated during deployment of the developed chatbot. This initiation process involves registering the intents with the chosen Intent Recognition Provider, establishing connections with the Instant Messaging Platforms, and launching the External Services described in the execution model. Thus, when the chatbot receives a user input, the run-time components forward it to the Intent Recognition provider, acquire the recognized intent, and proceed to execute the required actions as defined in the chatbot execution model [23].

The decoupled and modular design of the infrastructure brings several benefits. It eases the maintenance and evolution of the created chatbots, in that by using the framework one is not highly dependent on other technology platforms. The Intent Recognition Provider platforms and Instant Messaging Platforms are up to the developer itself which they want to use. However, it is important to take into notice that we did not utilize every component of the XatKit infrastructure; for instance, there was no deployment of the created bot on external services, due to it not being necessary for our thesis and project. For that reason, we used almost none of the services granted by the Platform Package, except for the React platform methods - further elaborated in section The React Widget. Therefore, the upcoming sections will exclusively focus on the components within the XatKit framework that was employed in this project, and delve deeper into their usage and functionalities.

XatKit Modelling Language

The XatKit Modelling Language is a DSL for chatbot development and offers primitives to design the user intentions, execution logic, and the deployment platform of the constructed chatbot. These tools are divided into three core packages, as previously explained in figure 4.2. The DSL is constructed through a metamodel, defining the concepts of the language and their interconnections. In essence, a metamodel in information systems functions as an abstract syntax, a model containing statements about the constructs employed in the model. Just as models in programming systems serve as an abstraction of some reality, metamodels, in turn, serve as an abstraction of those models [54] [23].

Intent Package

The intent package defines the language used to represent the construction of the intent and entity design of XatKit chatbot models. Figure 4.3 provides an illustration of the Intent Package's metamodel. This metamodel defines a high-level class named IntentLibrary which consists of a collection of IntentDefinition. Each IntentDefinition is a named entity that represents user intentions and has a collection of training sentences. These training sentences are the input examples used to detect the underlying user intention hidden within a textual message from the user. They are divided into TrainingSentenceParts, representing fragments of input text for matching. Furthermore, each IntentDefiniton defines a set of outContexts. These are named containers used to store information throughout the conversation between the user and the bot, as well as customizing intent recognition. Each Context embeds a set of ContextParameters, which establish mappings from TrainingSentenceParts to specific EntityTypes, in order to specify which fragments of the Training Sentence to extract and store. Additionally, a Context can define the lifespan of the conversation, allowing for the specification of information from the user to retain and discard, as well as the customization of the conversation experience. The DSL of the Intent Package has been widely used during the construction of our chatbot, this is further detailed in Intent and Entity Design, section 4.3.2.

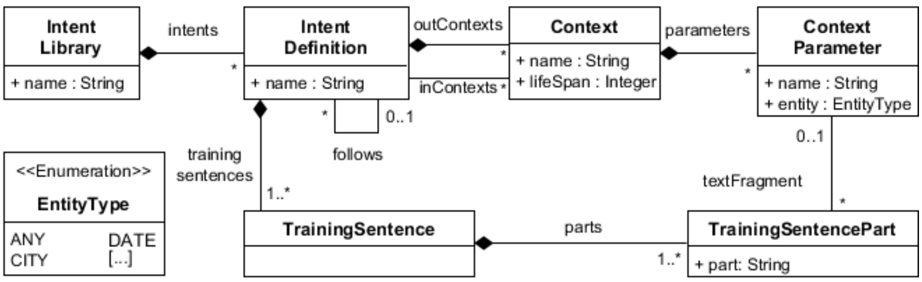


Figure 4.3: Figure of the Intent Package metamodel from XatKit [23].

The Chatbot Logic: Java Fluent API

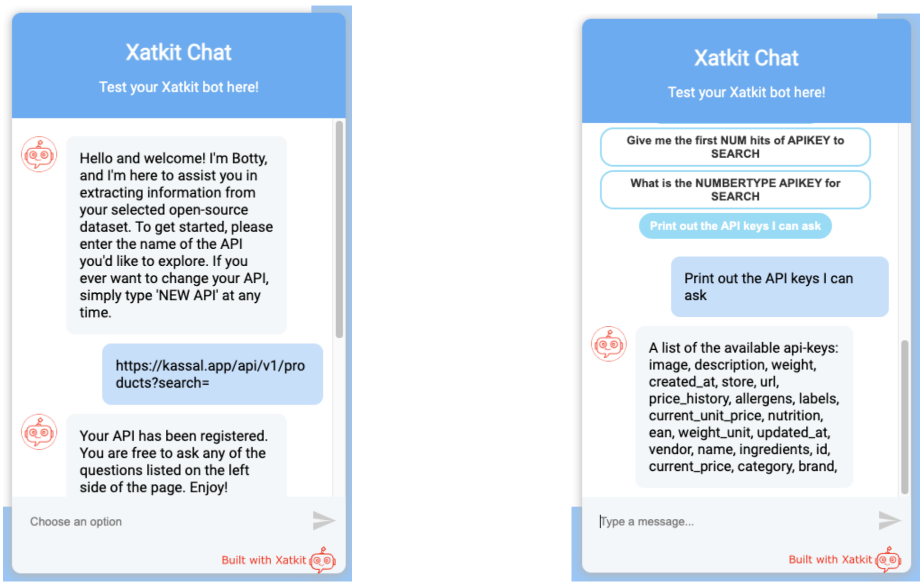
Within the XatKit Bot Platform, the fundamental components of the chatbot's logic are the intents, the training sentences, and the associated business logic executed upon every matched intent. All of this is implemented through their *Java fluent API*, which is further enhanced by *Lombok*, a Java library designed to reduce writing repetitive and boilerplate code [110]. The logic is mostly articulated through XatKit's own internal DSL, which is based on state machine semantics, and also embedded through the Java Fluent API. The API offers several benefits. Firstly, due to it being a fluent interface, it eliminates the need to learn a new programming language and we could use the skills we have already acquired in Java. Secondly, the development of XatKit bots does not depend on any specialized tools, thus any Java IDE would suffice. Therefore, one can benefit from all the existing Java tools when developing and debugging the bots, as well as utilizing the full power of the language and libraries. All of this while retaining the well-defined, high-level semantics provided by the internal DSL. Thus, XatKit combines the strength of a dedicated chatbot DSL with the power of a general-purpose language whom we are already familiar with [109] [20].

The React Widget

XatKit has created its own React component that is able to embed XatKit chatbots in websites by installment through npm or yarn. The component facilitates communication with the Java server via a WebSocket, enabling real-time interactions. The documentation on GitHub provides installment instructions, as well as guidance on how to customize the chatbot's appearance, ranging from colors to overall aesthetics.

4.3 Solution Approach

In section 1.5.1 Chatbot Generation and Data Utilization, we gave an overview of the core objective; the development of a natural language interface, with the aim of facilitating the interaction between users and open data sources through the use of chatbots. The thought-of approach to solve this involved creating a web application, consisting of a conversational user interface (CUI) and a user guide on utilizing the chatbot. In creating this chatbot, we used the technologies provided by the XatKit platform (4.2.5), paired with the programming language Java where we developed the foundation of the chatbot through the implementation of classes, functions, and a multitude of features. Depicted in figure 4.4 below are two visual representations of the chatbot, showcasing its appearance and overall aesthetic. The conversation displayed shows the initial messages delivered by the chatbot during interaction.



(a) The first welcome messages from Botty.

(b) Following the successful registration of an API, Botty provides clickable sample questions.

Figure 4.4: Snapshot of a conversation with Botty, illustrating the messages first delivered by the bot.

In creating this CUI, numerous steps required thorough research and development. This included API investigation, selection, and identification; the design and process of intent and entity recognition and extraction; the solution strategies of the challenges encountered within the XatKit Modelling Language; and the establishment of the flow of the conversation between the user and the bot. This chapter will provide a detailed description of our solutions to these challenges and our development approach.

4.3.1 API Investigation, Selection and Identification

Gaining a substantial understanding of the structure behind various JSON APIs and -datasets proved a pivotal role in the creation of an automated chatbot system. This understanding laid the groundwork for the different intents, entities, and functions the bot originated from. Therefore, API selection and identification were crucial first steps in the development of the system. By examining multiple datasets, the aim was to uncover both the commonalities and distinctions within these APIs. This resulted in a more comprehensive view of the JSON-API landscape and paved the way for a better understanding of generalizing their usage.

The first chatbot generated was named *GroceryBot* which was an example bot created from following XatKit's own beginner tutorial [45]. The API from *kassal.app* [78] was chosen as the primary data source, with several considerations driving this selection. Firstly, the kassal.app API documentation is highly informative and user-friendly. It not only provides a clear explanation of the API endpoints, request parameters, and JSON response format but also offers examples that demonstrate how to effectively interact with the API. The dataset also allows for the inclusion of either none or multiple parameters in the API call. The advantages and disadvantages of this choice are detailed in 4.3.1, Commonalities and Distinctions Found in APIs. Furthermore, the utilization of the API was a gratifying experience. It delivers comprehensive and current data on grocery prices within the Norwegian market [6]. This rich and specific dataset allows the chatbot to deliver personalized and localized responses. This focus on grocery data is highly advantageous for the bot's functionality as it enables the bot to offer users relevant and accurate information. Lastly, the API demonstrates great performance, with reliable up-time and fast response times, ensuring smooth and seamless user experiences.

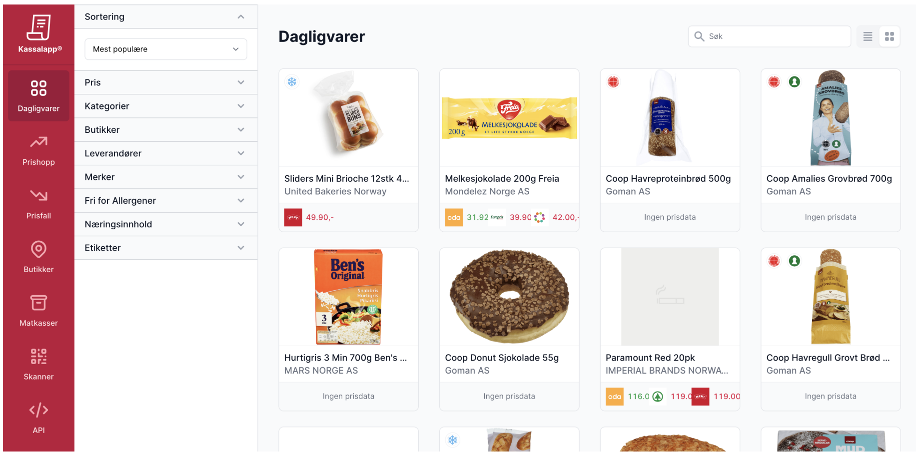


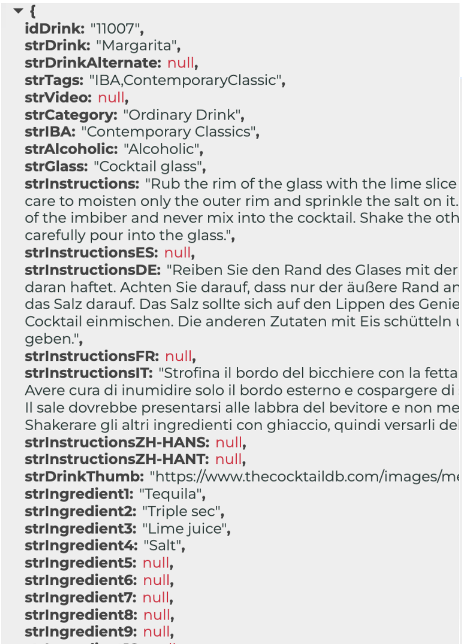
Figure 4.5: Snapshot of the kassal.app homepage. Displayed are some of the various objects the API inhabits and the information of each can be extracted through the API.

After being content with the result of the first bot, the development of a greater and more generalized bot began. With the new bot *Botty* under development, we continued to utilize *kassal.app* as a foundational reference, while also incorporating two additional APIs. The first API is called *TheCocktailDB* and is an open-sourced database consisting of drinks and cocktails from all around the world [93]. This resource empowers a free JSON API which was taken in use. It allows for searches within the database, permitting users to narrow down their queries with various parameters, such as searching cocktails by name, ingredient searches, category filters, and more. Notably, it also enables both none or several parameters. The second API chosen is from Stortinget's - the Norwegian Parliament - own data service and offers extracts from databases used in the Storting's parliamentary proceedings. This API is a subset and fetches an overview of all members of the current government [81]. This API was chosen because it is different from the other two APIs considering it does not allow for any parameters, as well as the domains of the APIs are vastly different.

As Botty continued to evolve, its architecture became even more adaptable. It can currently interact with a wide range of open API endpoints, including those not limited to kassal.app, TheCocktailDB, and Stortinget's API. This evolution was driven by the need to create a more versatile solution capable of working with multiple APIs that share similar JSON structures. This enhancement further extends our chatbot's capacity to interact with diverse data sources, providing a more general and flexible approach to handling open data.

Commonalities and Distinctions Found in APIs

From the start, a key distinction emerged among the selected APIs, particularly concerning the key-value pairs. It was noted that the keys must always be of the data type string, but the values however can encompass a variety of data types, including but not limited to string, float, arrays, and objects. Figure 4.6 represents two segments extracted from two different JSON responses. The bold text on the left of each line represents an API key, while the remainder of the line represents the value result. Looking at the bottom lines of figure 4.6b, we observe that the majority of keys are structured as arrays, such as *price_history* and *allergens*. The *store* key however encompasses an object within itself, indicating that it is a nested object. This observation is important to take into consideration due to Java's categorization as a strongly typed language. In computer programming, a programming language is considered to be strongly typed if it necessitates the declaration of each variable with a data type and that declaration will be known by the machine either at compile-time or run-time. Certain operations may only be permitted with specific data types [111].



(a) Segment of response from *TheCocktailDb*

(b) Segment of response from *kassal.app*

Figure 4.6: An analysis of the difference between two different JSON responses

Another observation that occurs is the variety of the names of the keys in an API. Initially, there was a belief that the handling of the keys had somewhat of a pattern, but we quickly discovered there was little to no way to generalize. As seen in figure 4.6, the key names in both figures differ, as the key names in 4.6a are prefixed with *'str'*, but key names in 4.6b are not. Another instance of differences is the data-set provided by Stortinget which is documented in Norwegian rather than English. Furthermore, the majority of APIs provide users the ability to specify parameters for the purpose of narrowing their search. These parameters are specified in the API-URL itself and can be changed according to the user's wants and needs. For instance, we can specify queries for kassal.app's API endpoint by appending *'?search=cookie'* at the end of the URL. This action leads to the dataset being refined to return objects that match the *'cookie'* criteria or any other filter that may be set. However, some APIs do not offer this functionality and return the entire dataset instead, such as the Stortinget API.

Finally, we observe a common structural pattern within the selected API endpoints. Figure 4.7 below exemplify a typical response from the outermost layer of the received data to two of the APIs mentioned. These responses all exhibit a recurring pattern – in each case, the entire response is structured as a JSONObject, the most common data structure in JSON-APIs. A JSONObject is recognizable in that the content is enclosed in curly brackets. Inside the object, the desired data is encapsulated in the form of a JSONArray. A JSONArray is detectable in that the content is enclosed in square brackets. The JSONArray houses one or multiple JSONObjects. As exemplified in the figure, the data we want to extract from Storting's API response is in the array *'regjeringsmedlemmer_liste'*. The desired data from kassal.app API however, is in the array *'data'*. We also detected the same pattern in TheCocktailDB API, with the array name being *'drinks'*. However, by researching other various APIs, we found that if the data only contains one data source, then the expected JSONArray transforms into a single JSONObject. In rare instances, the API might just respond with one data type overall. For instance, a String is typically returned with exceptionally small datasets or responses.


```
{
  "respons_dato_tid": "/Date(1695643060172+0200)/",
  "versjon": "1.6",
  "regjeringsmedlemmer_liste": [
    {
      "respons_dato_tid": "/Date(1695643060187+0200)/'
      "versjon": "1.6",
      "doedsdato": null,
      "etternavn": "St\u00F8re",
      "foedselsdato": "/Date(-295149600000+0200)/",
      "fornavn": "Jonas Gahr",
      "id": "JGS",
      "kjoenn": 2,
      "departement": "Statsministerens kontor",
      "parti": {
        "respons_dato_tid": "/Date(1695609965946+0200)
        "versjon": "1.6",
        "id": "A",
        "navn": "Arbeiderpartiet",
        "representert_parti": true
      },
      "sortering": 1,
      "tittel": "Statsminister",
      "verv": "Statsminister"
    },
```

```
{
    "data": [
        {
            "id": 1,
            "name": "Pepsi Max 1,5lx4 fl
            "brand": "Pepsi",
            "vendor": "Ringnes as",
            "ean": "7044618874687",
            "url": "https:\/\/spar.no\/n
            "image": "https:\/\/bilder.n
            "description": "Pepsi MAX er
overlegen p\u00e5 smak, og for mange er
            "ingredients": "Kullsyrehold
syrer (fosforsyre, sitronsyre), surhetsr
fenylalaninkilde",
            "current_price": 89.9,
            "current_unit_price": 14.98,
```


Figure 4.7: Snapshot of the structure of Storting's API and Kassal.app API.

To summarize, APIs come in diverse structures, and by identifying distinctions and commonalities we got better equipped to face the challenges given by the complexities surrounding them. The next sections will revolve around the code solutions we developed for API registration and data retrieval, integrating the insights we gained from this research.

Handling Differences During Registration of JSON-Structured APIs

There were several functions implemented for the sole purpose of registering and comprehending the incoming APIs from the user. The similarities and differences within the APIs were important to take into consideration during the development. The distinctions highlighted in this section are the differences among data types, variables in parameter usage within APIs, and lastly, the registration of varying response structures within datasets.

As shown in both figure 4.7 and figure 4.6, the data types associated with values in key-value pairs can encompass a range of possibilities, including, but not limited to string, numeric values, arrays, and objects. Ergo, in the nature of APIs it is important to not impose strict requirements on the data type; instead, remain receptive to whatever data is provided. However, this can prove faulty in our programming language of choice, Java. As earlier explained, Java is a strongly typed language, thereby it demands the specification of the data type when declaring a variable. If the provided data type is incorrect it can result in an immediate program crash. As a result, it was necessary to implement functions capable of parsing the incoming data types before initialization of the response, subsequently delivering an appropriate response for each encountered data type. This was done by the use of JSONTokener class; the class is a valuable tool as it can distinguish whether the object is a JSONObject, JSONArray, or any other data type without causing disruptions to the program's functionality. Another important aspect is the use of try-catch exception handling. In the event of an error, Java typically halts its execution and generates an error message. In enclosing the method in a try-catch block, if an error occurs it will run the code in the catch block instead of shutting down the program instantly. Algorithm 1 is a pseudo-code for one of the parse functions created, in this case, the parse function of JSONObject, showcasing an example of how the code works.

Algorithm 1 Pseudo-code for JSON parsing

```
Parse the JSON response using a JSONTokener
Retrieve the next value of the response
if parsed value is JSONObject then
    Try:
    Extract value from JSONObject which key corresponds to the desired key
from user input
    Pass the key and value to a function that handles the value
    Catch:
    Error handling
end if
```

Another important factor to consider is the difference in parameter usage across APIs. While the majority of APIs grant users the capability to define parameters to fine-tune their queries, it is worth noting that users may not provide parameters and that certain APIs may not support this feature and instead return the complete dataset. Hence, we developed a function that examines whether the URL allows for parameter usage. The function's return type is a Boolean and the result is permanently stored so that the intent and entity extraction is different based on parameter usage or not. Further elaboration on this topic is explained in the next chapter, Intent and Entity Design. The code snippet for the function is illustrated below.

```java
public static boolean allowsParameters(String apiUrl) {
    return apiUrl.endsWith("=");
}
```

In figure 4.7, we observed both the common patterns and the distinctions that emerged across the various APIs we encountered. In order to accommodate these varying response structures, we created a function that is able to efficiently interpret and extract data from API responses, regardless of their format. The logic of the function mainly revolves around the function's ability to iterate through every key in the response body. Initially, the function checks whether the data type of the key is a JSONArray or a JSONObject. If true, it immediately terminates the while-loop and returns the key as a string. In the absence of a match, the function returns the entire response body as a string. The functioning of this process is visually represented in the accompanying flowchart in figure 4.8. This figure illustrates the design and logic behind the management of the varied response scenarios encountered across different APIs.

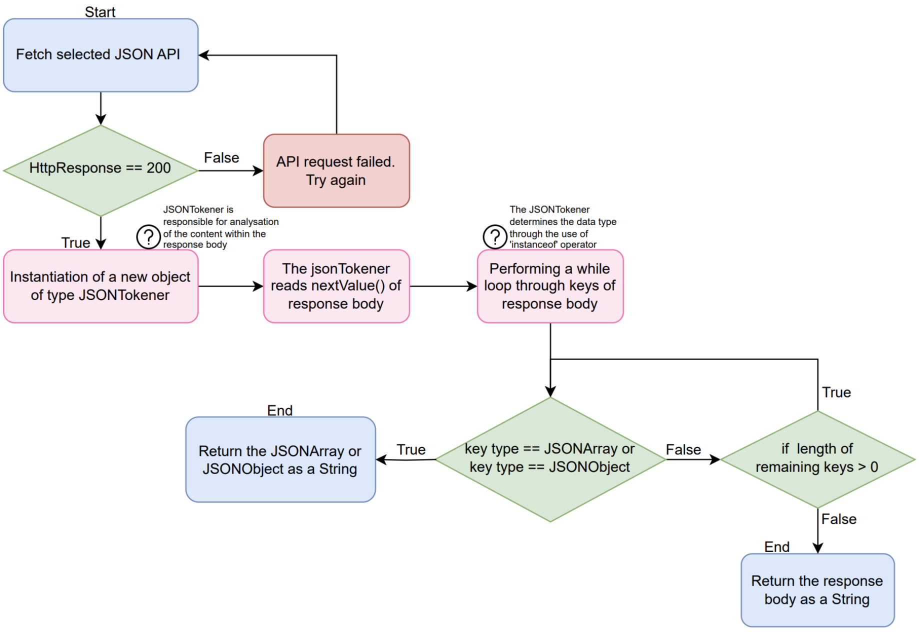


Figure 4.8: Flowchart: The logic behind handling diverse API response scenarios.

In conclusion, exploring multiple open-source APIs was essential in acquiring the knowledge needed for automatic chatbot creation. It was quickly discovered that data manipulation was a significant aspect of bot generalization. Gaining insight into which specific data segments to manipulate, led to a better comprehension of the nuances of intent and entity extraction, as well as the structure of the bot's questionnaires, thereby enhancing the overall performance of the bot.

4.3.2 Intent and Entity Design

Intent and entity extraction is one of the fundamental bricks behind chatbot creation; the core principles are that the intent is to be identified from the user utterance, and the entity is then extracted from that intent. In our bot application for our solution approach, the intents and entities are defined within the Java application on the back-end. To specify the intents we deemed necessary we utilized XatKit's DSL - previously detailed in section 4.2.5 - and the intent definitions along with their corresponding training sentences are stored in properties files available in both English and Norwegian languages. Table 4.1 display an overview of the intents defined in the semantics of the bot, together with their corresponding training sentences, and their associated entities highlighted in bold text. Provided at the bottom of each intent is an example of a given input and output, demonstrating the utilization of the API endpoints discussed earlier (4.3.1).

Intent Definition	Training Sentences with Entities
GiveMeXHits	Give me the first **NUM** hits of **APIKEY** to **SEARCH** I want the first **NUM** hits of **APIKEY** for **SEARCH** Do you have the first **NUM** hits of **APIKEY** to **SEARCH**
Example	Input: I want the first **3** hits of **strName** for **tequila** Output: The first **3** hits of **strName** for **tequila** are: Tequila Fizz, Tequila Sour, Tequila Sunrise
GenericQuestionMatchAPI	What is **APIKEY** to **SEARCH** What is **APIKEY** Give me **APIKEY** to **SEARCH** Give me **APIKEY**
Example	Input: What is the **departement** to **Vedum**? Output: The departement to **Vedum** is Finansdepartementet
HighestOrLowest	What is the **NUMBERTYPE APIKEY** for **SEARCH** Which **SEARCH** has the **NUMBERTYPE APIKEY** Give me the **NUMBERTYPE APIKEY** for **SEARCH** Retrieve the **NUMBERTYPE APIKEY** for **SEARCH**
Example	Input: Retrieve the **maximum current_price** for **Turtleneck Zip Pierre Robert** Output: The **maximum current_price** for **Turtleneck Zip Pierre Robert** is 1699 kr
PrintOutAPIKeys	Print out the keys I can ask Print out entities I can ask Which entities can I ask
WriteInAPI	**APIURL**
ChangeAPI	NEW API New API

Table 4.1: A tabular representation of Botty's intents together with their training sentences and entities is highlighted in bold text. These examples are drawn from the utilization of the three specific API endpoints earlier discussed (4.3.1).

In determining the definitions of these intents, it was vital to consider their appropriateness for addressing generic queries related to the open-source data given by the user, while also ensuring they were user-friendly and easy to understand. It is essential to note the table offers a general perspective of the training sentences; there exist multiple versions of each training sentence, which include modifications like the removal or alteration of words like 'for', 'to', 'of' etc, for grammatical purposes.

Intent Recognition Provider

The selected intent recognition provider for our application is DialogFlow (2.9.1). The integration of NLU platform has elevated the level of automation and bot generation within the system. DialogFlow's natural language capabilities are able to accommodate spelling errors within the intent. Figure 4.9 offers a visual representation of DialogFlow in action on their web page. Upon integrating DialogFlow into our application the intents are incorporated into the platform, giving us the flexibility to make adjustments as needed.

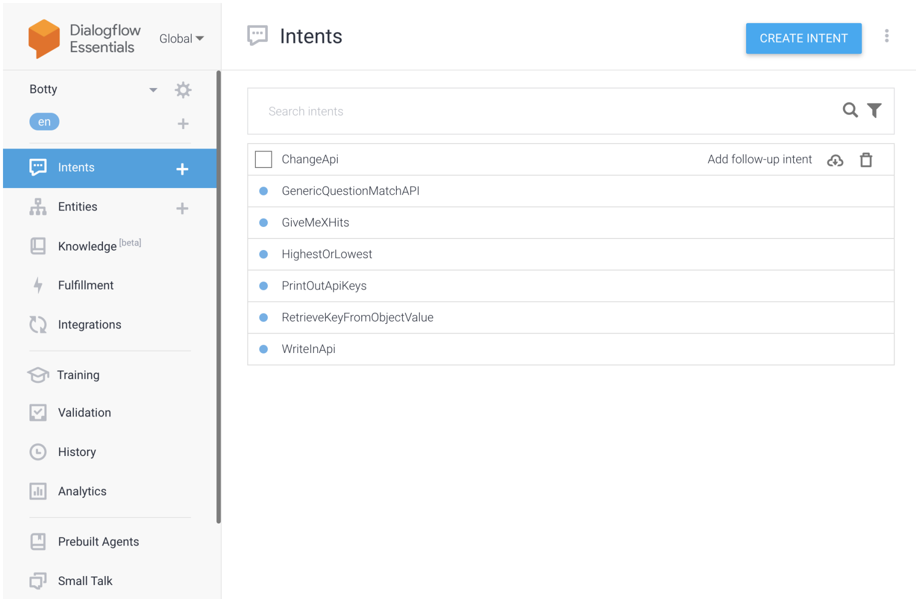


Figure 4.9: Snapshot of the DialogFlow web page.

Entities

Similar to the determination of the intents, the entities were defined in a highly generic manner. Table 4.2 provides an overview of each entity, including their name, data type, a brief description of their intended purpose and usage, and lastly examples of potential entity values. These entities are designed to be populated with keys derived from an API.

Entity	Data-type	Description	Examples
APIKEY	String	A key from the user inserted API	name current_price brand
SEARCH	String	Refining the search scope	object value parameter
NUMBERTYPE	String	Used for intents which finds the highest or lowest value in a data-set	maximum minimum highest lowest
NUM	Integer	A number	1,2,3,4...
APIURL	String	A valid API	https://kassal.app/api/v1/products?search=

Table 4.2: A tabular list of entities defined in the semantics of the bot.

To gain an understanding of the logic of the intent and entity extraction, we will take a look at the intent *Give Me X Hits*. This particular intent is designed to retrieve and display the first X number of hits of a given key from the user-specified API. The code snippet below shows the outline structure of the intent and registers the training sentences and the entities (parameter). The code structure is replicated across other intents, the differences being the training sentences, the intent itself, and the entities.

```
giveMeXHits = intent("GiveMeXHits")
    .trainingSentences(BUNDLE.getStringArray("GiveMeXHits"))
    .parameter("apikey").fromFragment("APIKEY").entity(any())
    .parameter("search").fromFragment("SEARCH").entity(any())
    .parameter("num").fromFragment("NUM").entity(any())
    .getIntentDefinition();
```

Once the intent has been identified from the user input, the next step is to extract and manage the entities associated with that intent; in this case, the entities 'apikey', 'search', and 'num'. The following sections will provide a comprehensive breakdown of this process.

4.3.3 Entity Extraction and Recognition

The procedure of extracting the recognized entities from the intents is a complex process, due to the various and compounded structures of APIs, as detailed in section 4.3.1 API Investigation, Selection and Identification. The process consists of different functions, each one being carefully constructed for its use case. In developing these functions, we prioritized creating them in a more decoupled fashion, thus allowing for a more modular design that enhances the reusability of the functions, as well as making them easier to debug. Figure 4.10 displayed below provides an overview of the functions we have developed for the purpose of entity extraction and recognition, followed by an explanation of the main function *ProcessUserInput()*, and an example extracted from one of the intents.

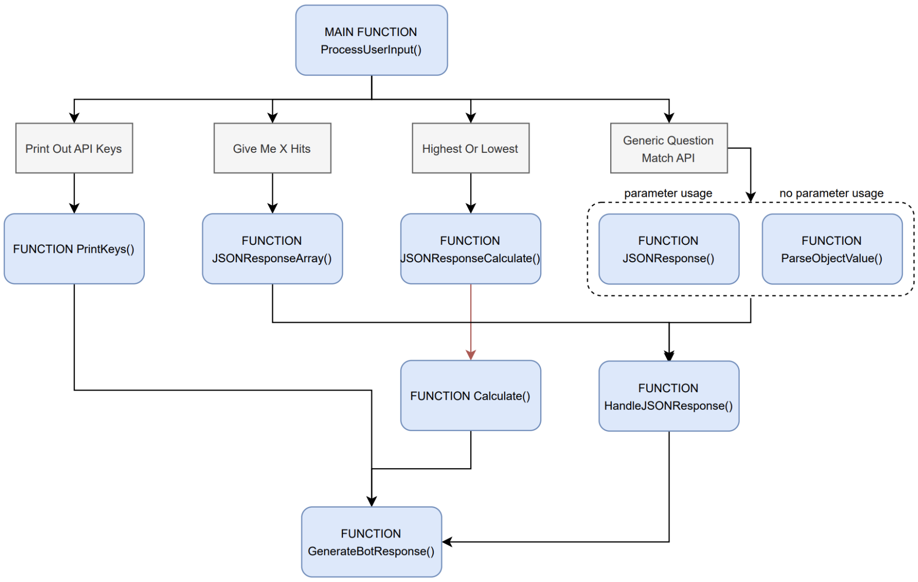


Figure 4.10: Overview of the functions responsible for the entity extraction process.

Figure 4.10 showcases an overview of all the functions used to interpret and unwrap the entities provided by the user. In this visual representation, the functions are depicted in blue boxes while intents are represented by grey boxes. The arrows illustrate the decision-making process guiding the selection of functions. The workflow starts with the main function, *ProcessUserInput()*, being triggered once the user input has been recognized as a valid and existing intent. The function proceeds to extract the intent definition using the Intent Definition class from the Intent Library, as detailed in XatKit Modelling Language. Additionally, it extracts and retrieves all the defined entities. Subsequently, the function verifies the presence of each entity by checking for non-null values using the getValue() method from the Intent Library. Depending on the API type, as previously outlined in section 4.3.1, the function conducts an API call, potentially with or without an entity as a parameter. Following that, the function forwards the intents and entities to another auxiliary function for the extraction of data, the choice of the function being dependent on the intent expressed by the user. To illustrate this, we will furnish an example below involving the *Highest Or Lowest* intent used in conjunction with the kassal.app API:

User Input What is the maximum current_price for Norvegia Ost

Intent Definition Highest Or Lowest

Entities maximum, current_price, Norvegia Ost

The function being called upon by the recognition of this exact intent is *JSONResponseCalculate()*, which in turn triggers the execution of *Calculate()*. Provided below are pseudo-codes for these. The first function accepts three parameters; 'jsonResponse', representing the API response with the entity value 'Norvegia Ost'; apiKey, set as 'current_price'; and numberType, specified as 'maximum'. It thereby proceeds to initialize the API response as an array and a parsedData variable with the intention to store the extracted data and eventually return it to the primary function. Synonym lists, catering to accommodate alternative terms for minimum and maximum values, are initialized to facilitate flexible input. The result variable initializes at zero and adjusts accordingly when searching throughout the API response. The main objective of the function is to iterate through the array, inspecting each JSON object within to identify the presence of the specified key. If found, the code extracts the numeric value associated with the key and passes the value forward to the *Calculate()* function to compute the result. After processing all of the elements in the array, it converts the result into a string and stores it in the parsedData, mapping the apiKey and the result. Finally, the function returns the mapping to the primary function.

```
function JSONResponseCalculate(jsonResponse, apiKey, numberType)
    initialize jsonArray as new JSONArray(jsonResponse)
    initialize parsedData as new HashMap()
    initialize minimumSynonyms as ["lowest", "smallest", "minimum"]
    initialize maximumSynonyms as ["highest", "maximum", "greatest"]
    initialize result as 0.0

    if numberType is not part of minimumSynonyms or highestSynonyms
        return null
    end if

    loop through jsonArray
        if jsonArray[i] contains apiKey then
            initialize valueObject as jsonArray[i][apiKey]
            if valueObject is a numeric value then
                result = calculate(minimumSynonyms,
                maximumSynonyms, valueObject, numberType, result)
            end if
        end if
    end loop

    return parsedData(apiKey, result)
end function
```

The *Calculate()* function is designed to calculate either the minimum or maximum value based on the provided synonyms and the input value. The result is accumulated and returned back to the function that calls upon it. It works in that it takes five parameters; the two synonym lists, maximumSynonyms and minimumSynonyms; valueObject, the current value retrieved from the key 'current_price'; numberType, a string indicating whether the calculation is for finding the maximum or minimum result; and result, which is the current result of the calculation. Firstly, the value object is converted into the data type double to ensure consistency and enable the comparison of data types more consistently. The code then proceeds to validate whether the provided numberType falls within the maximum- or minimum-synonyms. If maximum, it updates the result by selecting the maximum value between the current result and the provided value. Alternatively, if minimum, it firstly verifies whether the current results equals zero; if so, the result is set to the value encountered, as this represents the first value in the calculation. If not, it updates the results by choosing the minimum value between the current result and the provided value. Finally, the function returns the updated result.

```
function calculate(maximumSynonyms, minimumSynonyms, valueObject,
    numberType, result)
    convert valueObject to double

    if maximumSynonyms contains numberType then
        update result to the maximum of current result and
            valueObject
    else if minimumSynonyms contains numberType then
        if result is 0.0
            update result to valueObject
        end if
        update result to minimum of current result and valueObject
    end if

    return result
end function
```

To summarize, the process of extracting recognized entities is intricate and necessitates the development of designed functions capable of accommodating a wide range of data types. The values may include not only various numerical values but also textual information, dates, and more. Furthermore, it entails several error-handling mechanisms due to the diverse and often intricate structures of APIs. Therefore, with careful function design and robust error-handling strategies, it becomes possible to navigate the intricate landscape and extract entities accurately.

4.3.4 Solutions to Challenges Encountered Within the XatKit Modelling Language

In navigating the complexities surrounding the XatKit DSL landscape, we encountered challenges that required innovative solutions. The lack of specific documentation, as well as the scarcity of examples from the developers of the framework, led to an extended trial-and-error phase during the chatbot development process. This topic is further discussed in section 6.1.3 Challenges Surrounding the XatKit Bot Platform. One of the most prominent challenges stemmed from the inherent design of the DSL itself, particularly within the context of Intent Definition. The DSL is designed to accommodate only one single word per entity, which proved problematic when we wanted to enhance the precision of our search queries. To exemplify this issue, consider a scenario where we apply the kassal.app API alongside the Give Me X Hits intent:

Give me the first 5 hits of current_price to Freia Melkesjokolade

The DSL will only register the word 'Freia' as an entity, thus neglecting the subsequent text following the whitespace. This also hinders the intent recognition process, as those words are not associated with the intent definition, nor the training sentences. We were not able to implement language models working for this particular problem. Consequently, we implemented a two-pronged approach for the solution. Firstly, we extended the entity definitions in the intent training sentences by applying more words in the regular expression. Below is an exemplification of the regular expressions the training sentences originate from, this intent being the Generic Question Match API. We set a constraint with the maximum words per entity being 5.

```
GenericQuestionMatchAPI=\
  What is APIKEY to SEARCH\n\
  What is APIKEY to SEARCH SARCH SRCH SCH SC\n\
  What is APIKEY to SEARCH SARCH SRCH SCH\n\
  What is APIKEY to SEARCH SARCH SRCH\n\
  What is APIKEY to SEARCH SARCH\n\
  What is APIKEY
```

The second part of the two-pronged approach involved creating a mechanism capable of registering and measuring the word length of the entity, thus allowing for a more accurate parsing of intent queries. To achieve this, we developed a helper function with the ability to process the entire user utterance, identify the entity, and determine the exact number of words within the entity. Below is a pseudocode demonstrating the work of the function, followed by a subsequent code explanation for clarity.

```
function getEntities(StateContext context)
    initialize entity as context.getIntent().getValue("search")
    initialize entity1 as context.getIntent().getValue("search1")
        ...repeat with entity2,entity3,entity4
    initialize stringBuilder as new StringBuilder()

    if entity is not null then
        append entity to stringBuilder
    end if
    if entity1 is not null then
        append "+" or append " " and entity1 to stringBuilder
    end if
    ..repeat with entity2,entity3,entity4

    return stringBuilder.toString()
end function
```

The function accepts the user input in the form of a StateContext object as a parameter. It then attempts to extract a maximum of five entity strings from the object by employing the getValue() method from the Intent Library. Both the StateContext object and the Intent Library are from the XatKit Modelling Language, previously explained in section 4.2.5. Next, we introduce a StringBuilder, a Java class designed for managing changeable sequences of characters. The StringBuilder maintains a buffer array and stores the character sequences, as well as adjusting dynamically as characters are added or modified [3]. This choice allows us to seamlessly append new words (entities) to the existing entity string. Following that, the function examines each of the entity strings to determine whether they hold a non-null value. If true, the function appends it onto the StringBuilder followed by a separator. The choice of the separator, whether a plus sign '+' or whitespace, depends on the nature of the API type and its parameter usage, as earlier put into detail in chapter 4.3.1 Commonalities and Distinctions Found in APIs. This ensures that the entities are concatenated with either a plus sign or a whitespace between them. Finally, the function converts the contents of the StringBuilder into a unified textual string and returns this to the main parse function. This enhanced entity string can now be effectively utilized for conducting searches and other relevant operations.

4.3.5 Conversation Flow

The majority of chatbot solutions incorporate an event-driven execution mechanism. In essence, a bot application is defined as a set of intent/reaction pairs, as stated by Daniel (2021) [21]. When the bot receives an intent, it triggers the execution of a specific code segment. This area we have decided to refer to as the conversation flow between the user and the bot. It encompasses the decisions made regarding the bot's messages, its reactions and responses to user inputs, and the overall progression of the conversations.

As mentioned in section 4.2.5, we constructed the chatbot application with the use of the XatKit Bot platform, thereby embedding their Java DSL (4.2.4, 4.2.5) to define and specify the chatbots behavior. This behavior resembles that of a state machine, capable of comprehending the inputs and triggers received from the user before reacting accordingly [22]. These responses can be articulated as states, thus we can represent the bot as a graph of states that is interconnected via user intentions. Each state encompasses a body with arbitrary code to execute. When the bot receives an intent, it transitions from one state to another, and in doing so, triggers the execution of the corresponding state code.

In the preceding section 4.3.2 Intent and Entity Design, we presented the various intents we have devised for our approach. Given the nature of these intents, it is considered faulty to make them universally accessible throughout the entire dialog between the user and the bot. With the use of the state machine semantics provided by XatKit, we can control what intents are available at any given point during the bot execution. For instance, it would be illogical to allow a user to request the printing of API keys before they have successfully provided a valid API. To visually represent the designed conversation flow of the chatbot, we have created a diagram that encompasses all of the various states along with some illustrative responses. The diagram is in figure 4.11.

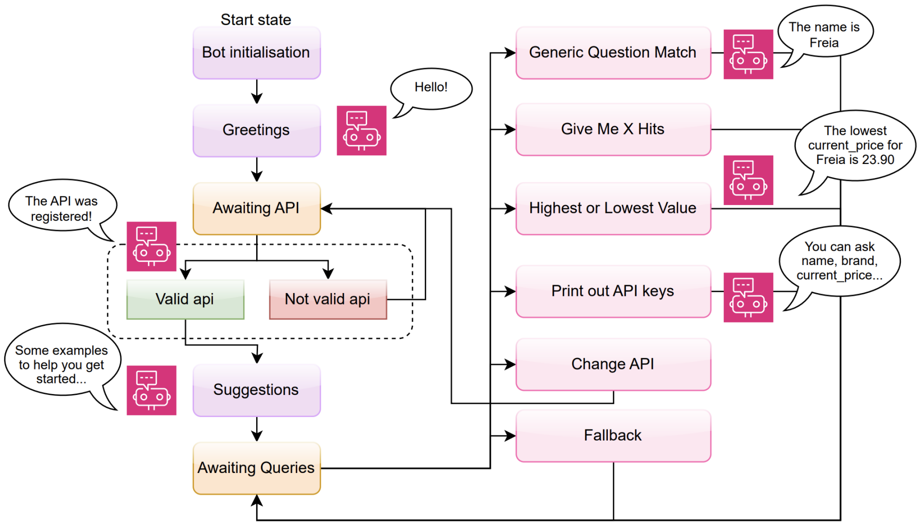


Figure 4.11: The conversation flow of the chatbot.

Illustrated in figure 4.11 are rounded boxes visualizing the various states of the chatbot, and the background color of each box signifies the states' unique functionality. Lilac boxes indicate states that are capable of carrying out actions, such as initializing the bot, greeting the user, or printing out query suggestions to help the user get started with conversing. Meanwhile, the orange boxes represent states that await user input within the console for interpretation. Furthermore, the pink boxes signify various recognized intents the bot is capable of handling, except for the Fallback box, which is a state managing user input that does not align with recognized intents. All of these states are interconnected by arrows, representing the dialog flow between the chatbot and the user. Additionally, the figure showcases simplified examples from the bot's response scattered throughout. It is also important to take into account the error handling which is not included in the diagram; for instance, if the user input is matched with a recognized intent, but the entities provided are faulty, there will come an error and the chatbot will handle it accordingly.

4.3.6 Code Generation for Chatbot States and Transitions

The code generation for the conversation flow of the chatbot is programmed in various states and transitions and is written in XatKit Modelling Language (4.2.5) and Java (4.2.2). Figure 4.12 illustrates the pattern between the chatbot and the user, encountered in each state.

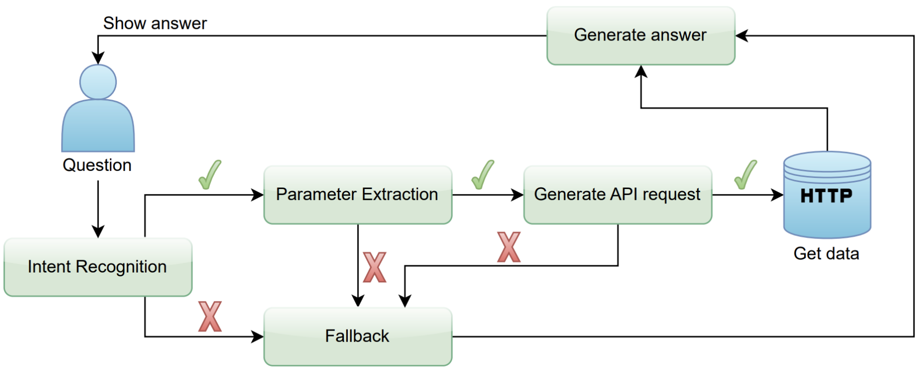


Figure 4.12: Diagram illustrating the flow of each state in the chatbot [44].

Each state usually comprises five main methods: body(), next(), when(), moveTo() and fallBack(). The body() method contains the code to be executed, which generally revolves around interpreting and processing user input, as well as publishing the bot response on the chosen platform. In figure 4.12 above, this method embraces Parameter Extraction, Generate API requests, and Generate answers. The next() method simply determines the next action of the bot, often relying on both the when() and moveTo() methods. When() is a conditional function that guides the bot's behavior based on the context, while moveTo() states which intent the bot should transition to next. In regards to the figure, both when() and moveTo() correspond to Intent Recognition. The fallBack() method handles the bot's response when it encounters an unexpected intent. Understandably so, this method corresponds to Fallback in the figure.

Greetings, Awaiting Queries and Awaiting API States

To visualize the code generation behind the states further, we will first provide three examples; the first one being the Greetings State, the second one being the Awaiting Query state, and the third one being Awaiting API state. Provided below is the code underlying the Greetings state. This state encompasses all five essential methods and is characterized by its ease of comprehension. Despite its concise nature, it encapsulates the core functionalities required for the states.

```
1  val greetings = state("Greetings");
2
3  greetings
4  .body(context -> {
5      reactPlatform.reply(context, messages.getString("Greetings"));
6  })
7  .next()
8  .when(intentIs(intents.writeInApi)).moveTo(awaitingApi)
9  .fallback(context -> {
10     reactPlatform.reply(context, messages.getString("ErrorTypeInApi
           "));
11 });
```

The state's task is to display a welcome message to the user, previously visualized in figure 4.4a. Subsequently, it awaits input from the user. Lines 4 to 6 represent the main content of the state, wherein the context lies. In this case, the context is limited to the bot's response, which has been stored in a properties file containing a diverse collection of messages. Line 8 presents a conditional statement; if the next input is recognized as the intent 'writeInApi' (4.1), the bot will transition to the next state, awaitingApi. However, if the input does not align with this intent, a fallback error mechanism will be triggered. This safeguard is in place because the user's progress in the conversation depends on the submission of an API.

```
awaitingQueryState
    .next()
    .when(intentIs(bot.intents.genericQuestionMatchAPI)).moveTo(
        genericQuestionMatch)
    .when(intentIs(bot.intents.giveMeXHits)).moveTo(giveMeXHits)
    .when(intentIs(bot.intents.highestOrLowest)).moveTo(
        highestOrLowestValue)
    .when(intentIs(bot.intents.retrieveKeyFromObjectValue)).moveTo(
        retrieveKeyFromObjectValue)
    .when(intentIs(bot.intents.changeApi)).moveTo(changeApi)
    .when(intentIs(bot.intents.printOutApiKeys)).moveTo(printKeys);
```

The code presented above showcases the Awaiting Query state, which, in simple terms, is designed to wait for user queries. When an intent has been identified from the user input, it guides the conversation flow toward that specific intent. It is essentially the conversation flowchart figure 4.8 transformed into code format, however, this state comes into play after a successful validation and registration of an API.

The developed code for the state Awaiting API exhibits a higher degree of complexity. Simply put, the state's purpose is to register and validate the API endpoint from the user, respond with an appropriate message, and transition to another state based on the validation. To facilitate understanding, we will present a simplified code representation of the Awaiting API state below, accompanied by two key values used within the state on top. It is worth mentioning that certain sentences are presented in pseudo-code format.

```
1  initialize String api;
2  initialize boolean allowsParameter;
3  initialize boolean moveToStartState = false;
4  val awaitingApi = state("AwaitingApi");
5
6  awaitingApi
7  .body(context -> {
8      initialize botResponse;
9      intialize boolean validInput = fetchApi(context);
10     try {
11         if validInput is true {
12             declare api as validInput and store in main bot-class
13             declare allowsParameter by function call
                   allowsParameter(api)
14             change moveToStartState to true
15             declare botResponse = "ApiSelected"
16         } else {
17             declare botResponse = "ApiFailed"
18         }
19     } catch (IOException e) {}
20     reactPlatform.reply(context, messages.getString(botResponse));
21 })
22 .next()
23 .when(context -> moveToStartState is true).moveTo(fetchIntents)
24 .when(context -> moveToStartState is false).moveTo(tryApiAgain)
25 .fallback(context -> {
26     reactPlatform.reply(context, messages.getString("BrokenApi"));
27 });
```

The 'Awaiting API' state accomplishes its objectives through a series of four key steps. Firstly, it handles API registration and validation by fetching the API endpoint provided by the user (as observed in line 9). This is achieved by invoking the function fetchApi(), which is part of the API client class. This class contains multiple methods for parsing and registering API endpoints, and you can find an overview of it in figure 4.13 in section 4.3.7. The function utilizes a GET-HTTP request to fetch the endpoint, and depending on the response ('200' indicates successful request) will return a boolean value. Secondly, the state proceeds to make decisions based on the response, as demonstrated in lines 11 to 18. If valid, the input will be assigned to the String variable api (line 1) making it accessible to all functions. Additionally, line 32 features a function call to allowsParameter(), which response - based on the parameter usage of the endpoint - will be stored in the accessible variable shown in line 2. Thirdly, the chatbot generates an appropriate response to the user, contingent upon the validity of the API endpoint (exemplified in lines 15 and 17). Lastly, the transition stage is defined in lines 22 to 24. Based on the boolean value of moveToStartState (lines 3 and 14), it will determine whether to transition to state Fetch Intents, or state Try API Again.

Common Structure of Intent States

The code generation for the states created for the various intents shares a commonality, as they predominantly rely on the functions developed for entity extraction and recognition, as previously elaborated in 4.3.3 and depicted in figure 4.10. While the provided code snippet pertains to the state for intent Generic Question Match, it essentially serves as a template for the majority of intent states. This code relies on the variables stored earlier during the Awaiting API state and leverages the functions created for entity extraction to provide accurate responses.

```
genericQuestionMatch
    .body(context -> {
        initialize botResponse;
        try {
            if (allowsParameter is true) {
                botResponse = ParamSearchUtils.processUserInput(
                    context, api);
            } else {
                botResponse = ObjectSearchUtils.processUserInput(
                    context, api);
            }
        } catch (IOException e) {
            botResponse = messages.getString("
                DefaultFallbackMessage");
        }
        reactPlatform.reply(context, botResponse);
    })
    .next().moveTo(awaitingQueryState);
```

Default and Local Fallbacks

A fallback serves as the bot's response when faced with an unexpected intent. In our application, we categorize fallbacks into two types; default and local. The default fallbacks represent a generic error reaction embedded throughout the application and are triggered whenever an error occurs. For instance, if the received textual message fails to be recognized as any existing intent, the default fallback message will be returned to the user. Its primary purpose is to ensure the user is informed that the bot was not able to comprehend their query, instead of silently failing. On the other hand, a local fallback is specifically defined at a failure point in the bot's conversation flow and possesses access to contextual information. Utilizing this additional information, a local fallback can offer more precise feedback to the user [21]. To demonstrate these fallbacks, we present examples below illustrating the bot's responses to different queries.

User Input Hello chatbot, I want to know the department to Jonas Gahr Støre.

Default Fallback Oops! It seems I didn't quite catch that. Please try rephrasing your request, or explore some of the available queries you can ask to the far left.

User Input What is the maximum department for Jonas Gahr Støre?

Local Fallback Sorry, we couldn't find the information you requested. Please double-check if the key delivers numeric values.

User Input What is the department for Jonas Gahr Støre?

Successful Query The department for Jonas Gahr Støre is Statsministerens kontor.

In the first query, no intent is recognized, thus triggering the default fallback. Moving to the second query, it is identified as the Highest Or Lowest intent (table 4.1). However, the specified entity key 'department' lacks numeric values. The conversation flow of the bot is able to catch this issue through the entity extraction and recognition functions (figure 4.10), inspecting the entity's key value before attempting to extract values for determining the maximum value. As a result, the bot response is altered, offering a more precise error message about potential issues with the query. The final query stands as an example of a successful query, successfully retrieving the correct answer for the user.

4.3.7 Overview of the Back-end of the Application

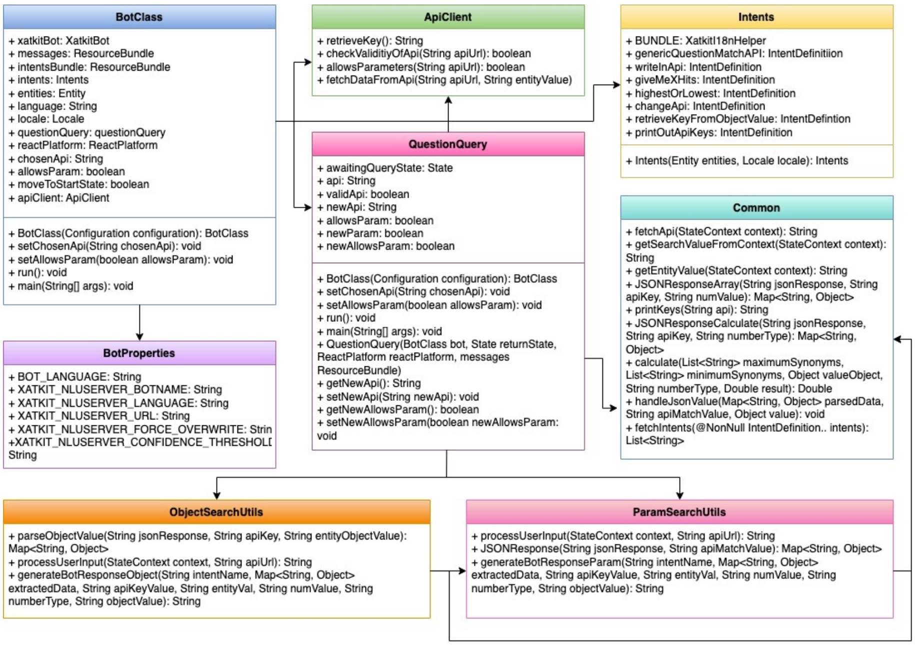


Figure 4.13: UML Class Diagram of the Application.

Figure 4.13 above provides a comprehensive representation of the structure of the Java classes responsible for the bot's functionality. The figure is a UML class diagram and outlines a structural overview of the system by modeling its core classes, their attributes and their methods, and the relationship between each class.

Chapter 5

Analysis and Assessment

This chapter presents a comprehensive analysis and assessment of the outcomes derived from our research, which was conducted using a mixed-methods research approach combining both quantitative and qualitative methodologies (1.6.2). This chapter demonstrates the performance of our technology, while also evaluating the impact chatbot technology has on data utilization and accessibility. This includes the results generated by our web application and chatbot as tools for engaging in open data, and the representation of our survey results to understand how and if, the use of chatbots can help people to utilize and benefit from open data sources. It represents an essential phase of our research, where we analyze the potential of these technologies to empower individuals to leverage open data resources effectively.

5.1 Web Application and Chatbot Results

This section presents an analysis and assessment of the performance, functionality, and user interaction aspects of our web application. The results are derived from overall usability evaluations to provide insights into the effectiveness and user-friendliness of our implemented solution.

5.1.1 User Interface

In section 1.5.1 Chatbot Generation and Data Utilization, we gave an overview of the core objective; the development of a natural language interface, with the aim of facilitating the interaction between users, with varying degrees of technical experience, and open data sources through the use of chatbots. The thought-of approach to solve this involved creating a web application, consisting of a CUI and a user guide on utilizing the chatbot. The overall user experience when interacting with the application is illustrated in figure 5.1.

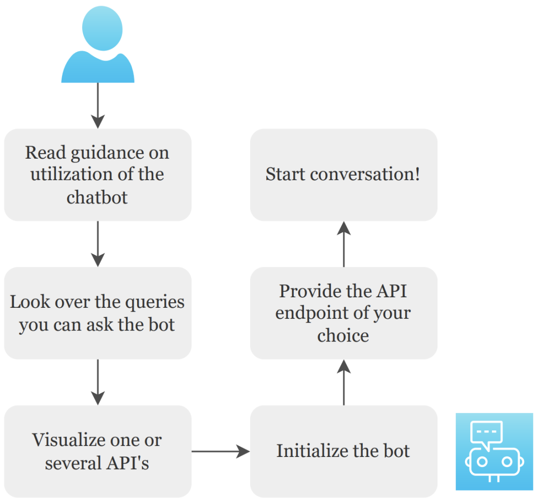


Figure 5.1: A depiction of the workflow when interacting with the web application.

In creating this natural language interface, numerous steps required thorough research and development. These steps are covered in detail in our Solution Approach. The results are presented through a web application developed with React and Java. Our initial concept was introduced in 1.5.1, as shown in figure 1.1. The final results are illustrated as follows.

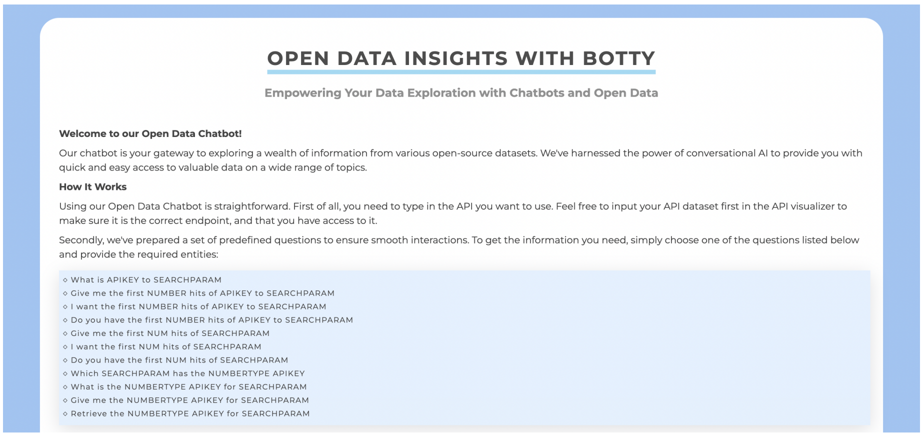


Figure 5.2: Welcoming state of the user interface.

In the first figure 5.2 depicting our user interface, users are provided with a brief introduction to the functionalities of our chatbot. The following section elaborates on how to operate and use it, offering insight into its functionalities. This includes a list of predefined questions designed to facilitate general and seamless interactions, ensuring users" awareness of the type of queries the chatbot can respond to.

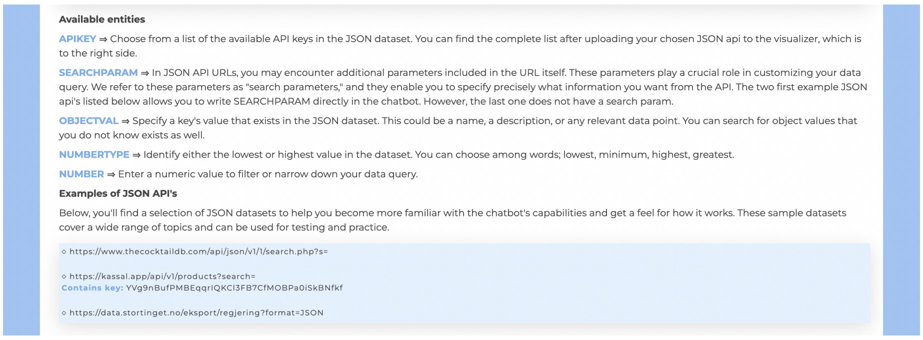


Figure 5.3: The user interface presents the available entities and provides examples of JSON APIs.

Next, depicted in figure 5.3, is the illustration of how we present the available entities to users. Within the list of predefined questions, certain words are written in uppercase letters, these words represent our entities. To facilitate user comprehension and guide them in matching these entities with those from their provided APIs, we include an explanation of all currently available entities. Additionally, we offer a selection of sample open APIs to familiarize users with the chatbot's capabilities, providing them with practical examples to explore.

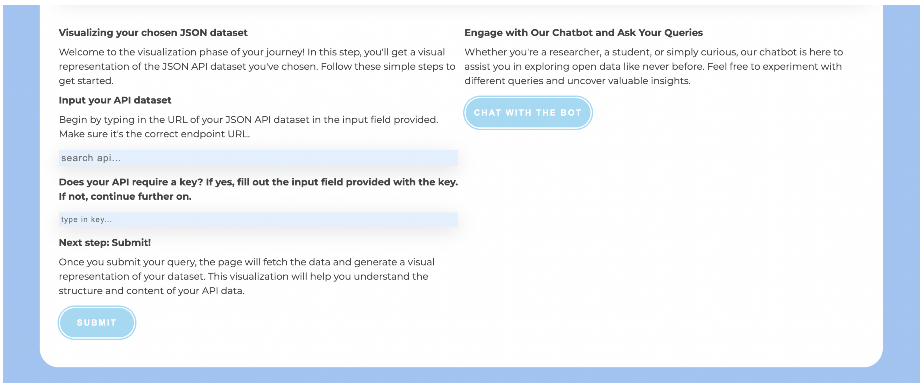


Figure 5.4: The user interface displays a JSON visualizer and a button to connect with the chatbot.

In figure 5.4, users can observe the feature allowing them to visualize an example response from their selected API endpoint using a JSON visualizer. A descriptive guide on how to proceed is provided, along with an input field for the API endpoint and, if required, the API key. Upon submission, users will have the ability to view a structured JSON response from their API in the bottom left of their screen, highlighting entities in bold for clarity. Simultaneously, users can initiate a conversation with our chatbot *Botty*, and the chatbot widget will appear in the bottom right corner of the screen. Both the JSON visualizer and the chatbot widget are depicted in figure 5.5 below.

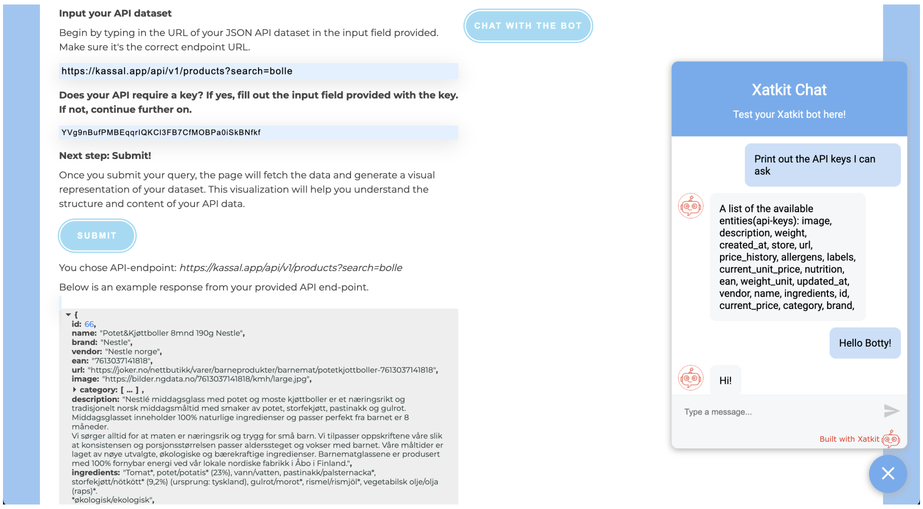
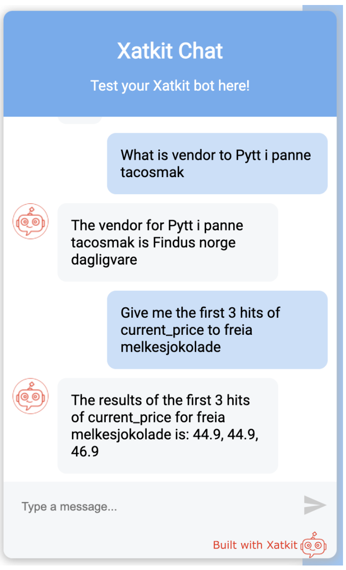


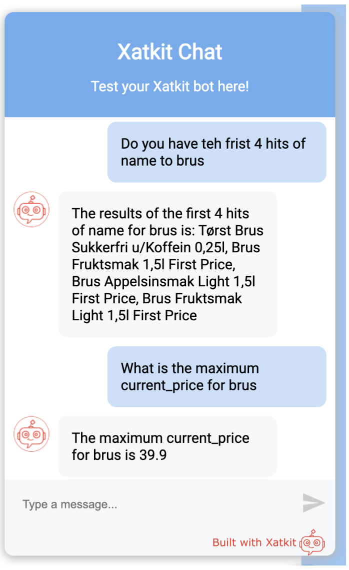
Figure 5.5: Utilization of the JSON visualizer and the opened chatbot widget.

5.1.2 Results of the Chatbot

This section provides visual representations of the chatbot's results, encapsulated in a series of figures that offer a representation of some of its capabilities and interactions. These figures are snapshots extracted from the demo video representing our UI accessible in Appendix A. The figures encompass a diverse range of scenarios, including intents, fallbacks, and error-handling mechanisms. The bot currently exclusively operates in English.



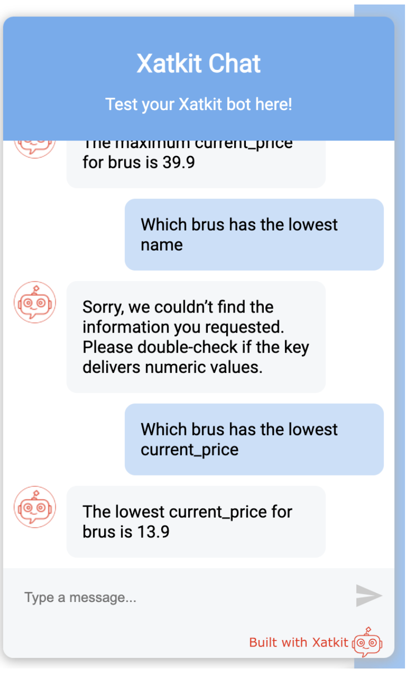
(a) Intents: Generic Question Match and Give Me X Hits.



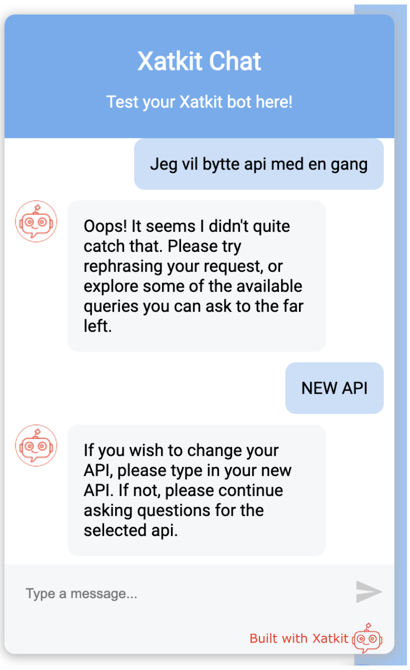
(b) Intents: Give Me X Hits (with error) and Highest Or Lowest.

Figure 5.6: Screenshot of three of the different intents in action (table 4.1).

In figure 5.6a the presented queries exemplify the bot recognizing the intents *Generic Question Match* and *Give Me X Hits*. In this example, we use the kassal.app API (4.3.1). In figure 5.6b, the first query reiterates the Give Me X Hits intent, this time with spelling errors. Botty is able to navigate and overlook this and continues on with the intended query. The second query demonstrates the *Highest to Lowest* intent, finding the maximum current price for soda.



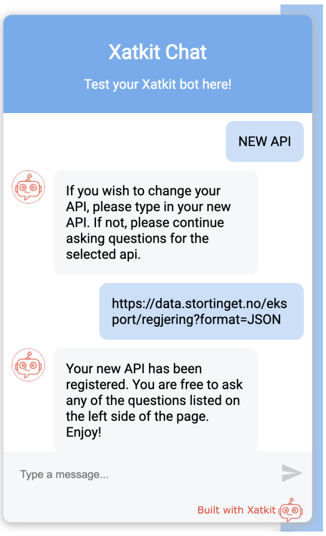
(a) Intents: Highest Or Lowest, local Fallback.



(b) Intents: NEW API, error-handling mechanisms.

Figure 5.7: Showcasing some of the various fallback and error mechanisms.

Figure 5.7a illustrates a local fallback in action. In the first query, we prompt the bot to retrieve the lowest name, an illogical request since 'name' is a string value, not a numeric one. The bot discerns this by identifying the data type of the entity before attempting to perform calculations. Consequently, it triggers a local fallback, prompting us to review the provided entities. In the second query, we pose a valid request, and the bot successfully retrieves the soda with the minimum price. Moving on to figure 5.7b, it depicts a default fallback scenario where the bot fails to recognize any uttered intent (due to it being in Norwegian), followed by a successful request to change APIs.



(a) Successfully registered a new API.

(b) Intents: Generic Question Match API.

Figure 5.8: Additional examples of conversations with Botty.

In figure 5.8a the registration of a new API is showcased, and figure 5.8b presents an additional conversation example with a new API. We aimed to illustrate some variety in our examples by including conversations from diverse APIs, ranging from inquiries about groceries to questions related to politicians. This was intended to demonstrate the incorporation of generic training sentences capable of adapting to a diverse range of APIs, regardless of their specific content. As described in section 4.3.2, the entities are intended to be filled with keys derived from the user's chosen API. Therefore, they must be named exactly the same as the key's name to match it with the API. When combining our bot, which currently supports only English, with an API containing key names in Norwegian, it can appear odd.

5.1.3 Evaluation of our User Interface

To ensure a reliable evaluation of our user interface and chatbot, we intended to employ the System Usability Scale (SUS), a reliable metric designed to assess system usability and user satisfaction. This endeavor would have gathered user feedback from proficient reviewers and our target demographic, enabling us to refine and optimize the user interface and chatbot functionality. The scheduled questionnaire, as seen in Appendix C, starts with gathering participants' names and occupations. It then contains 15 questions assessing the user's experience with our application, along with two questions measuring their prior experience with both chatbots and open data. Lastly, the participants have the ability to specify potential improvements/additional features and give additional feedback. This iterative feedback mechanism would have facilitated continual improvement to align our application with the predetermined goals and objectives. Regrettably, due to delays in the chatbot development process, this specific evaluation phase had to be postponed for additional refinement.

5.2 Chatbots and Open Data: Survey Findings

In the analysis of our survey, we prioritized a comprehension of the diverse needs and preferences of potential chatbot users. Through the qualitative research paradigm, we explored user expectations, preferences, and experiences, enlightening how chatbots can effectively facilitate access to, and utilize open data sources. Simultaneously, we incorporated the quantitative research paradigm to assess the demand and reception of chatbots as a means of enhancing open data accessibility. The collection of quantitative data allowed us to measure factors such as user knowledge levels, engagement with the chatbot, frequency of open data sources, and specific preferences related to chatbot functionalities.

The survey was shared through multiple platforms such as Facebook, LinkedIn, and Instagram and also forwarded to and by family and friends. The data is gathered through Google Forms and the responses remain anonymous to ensure reliability and trust for our respondents. It was written both in English and Norwegian to ensure the opportunity of a broad specter of demographics. In total, we collected 49 respondents, 38 in Norwegian and 11 in English, these responses can be found in Appendix A.

5.2.1 Survey Structure

Our survey starts off with an introduction including some theory of open data as this may be a term not everyone is familiar with. Following, the first seven questions consider some demographic characteristics of the respondents. Understanding this helps us analyze the survey results in the context of different backgrounds and experiences. This information is important for ensuring that our findings are representative of a diverse range of perspectives. Next, we explore a series of eight questions designed to gauge respondents' familiarity with open data, their current usage patterns, and the obstacles they face when working with it. This initial set of inquiries provides valuable insights into their existing knowledge and background related to open data. It serves as a foundational step, preparing for the subsequent set of questions centered around the potential role of chatbots in addressing open data accessibility.

Prior to examining the specific use of chatbots in this context, we include two questions that review respondents' general knowledge and experiences with chatbots. This initial insight helps us to better comprehend the respondents' existing awareness and interactions with chatbot technology. Subsequently, we inquire whether chatbots can serve as a solution to enhance open data accessibility and, if so, what specific features and functionalities would contribute to achieving this objective. These questions shift the focus from the respondents' general experiences to their perspectives on the application of chatbots as a means to facilitate open data utilization. By exploring this transition, we aim to gain a comprehensive understanding of the respondents' perspectives and expectations. The complete survey is presented in Appendix B.

5.2.2 Research results

In the context of our research on open data utilization and the role of chatbots, quantitative data offers a structured and numerical perspective on the domain of investigation. Although the dataset is relatively small for drawing definitive conclusions, it still enables us to quantify trends, patterns, and relationships, providing a basis for certain observations. This quantitative data provides us with a measurable foundation, while our qualitative data complements this by offering deeper insights into the respondents' perceptions, experiences, and preferences. Together, these two data types provide a comprehensive understanding of the subject matter, ensuring a well-rounded evaluation of our research questions.

Demographic Insights

Firstly, we present an overview of the demographic data gathered from our survey. Understanding the demographics of our respondents is essential as it allows us to understand the diversity of perspectives and experiences within our sample. This information forms the foundation for analyzing how various demographic factors may influence responses to our survey questions.

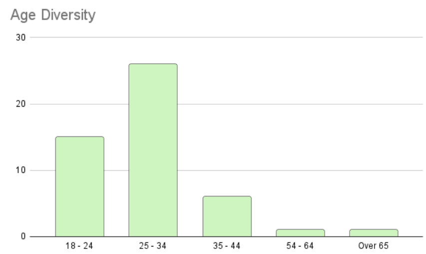


Figure 5.9: Representation of age diversity.

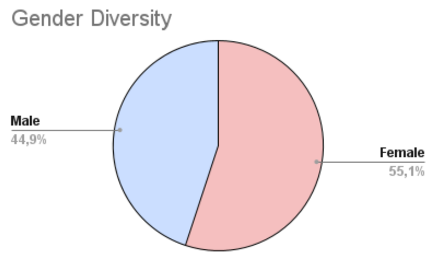


Figure 5.10: Representation of gender diversity.

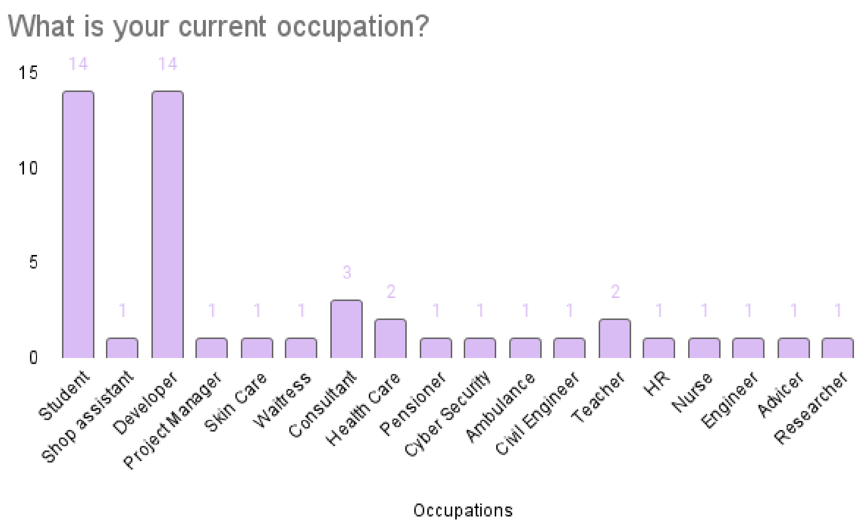


Figure 5.11: Representation of respondents occupations.

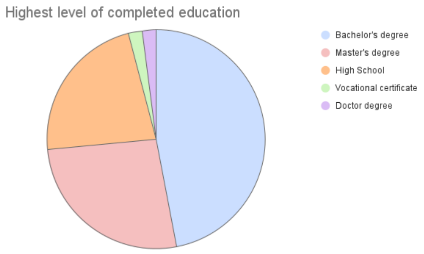


Figure 5.12: Representation of level of education.

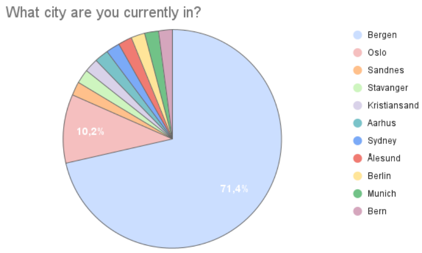


Figure 5.13: Representation of respondents current locations.

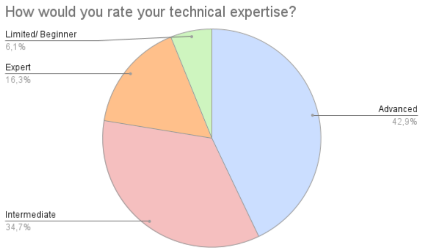


Figure 5.14: Representation of respondents' self-assessment of their technical expertise.

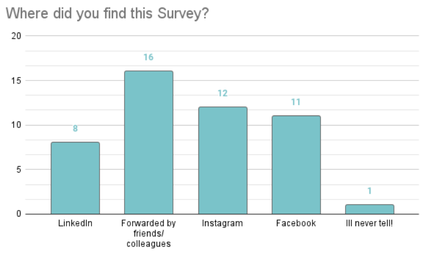


Figure 5.15: Where the respondents found the survey.

As mentioned, the demographics of the respondents can provide valuable insights into the potential of chatbots as a solution to encourage the general public to use more open data. Figure 5.9 represents the age diversity of the group of respondents. Age can indicate the generational divide in technology adoption and usage. Younger respondents might be more familiar with chatbots and open data, making them early adopters. Older participants may have different preferences or needs, which could inform how chatbots should supply to a wider age range. Among the respondents the majority is in the range of 18-34 of age, however, we still managed to get some range of variety of every age group except the category for under 18. Gender, as illustrated in figure 5.10, might reveal gender-specific trends or preferences in the use of chatbots and open data. For instance, if one gender shows more interest or engagement with the technology, this information could be used to tailor chatbot interfaces of services. Education level can be an indicator of technical literacy. Respondents with higher education may be more comfortable with technology and open data. Over 75% of the respondents have a degree of bachelors or higher, and this group could potentially benefit more from advanced chatbot functionalities. Furthermore, the type of occupation a respondent has can influence the need for open data and chatbots. People in data-driven professions, which a great deal of the respondents did have (figure 5.11), may see more value and demand in these technologies. For instance, researchers might require different features compared to a healthcare worker. Geographical trends can be uncovered from the question of current location. For example, urban and rural respondents may have distinct use cases for open data and chatbots. This information can help design region-specific features. The self-assessment of technical expertise as demonstrated in figure 5.14, can guide the design of chatbot interactions. Users with low technical expertise might prefer simplified and user-friendly interfaces, while experts require more advanced options.

Open Data Insights

In the following questions, we aim to gain a deeper understanding of our respondents' knowledge, usage, and the challenges they face when dealing with open data sources. Open data serves as a cornerstone in our research, and evaluating our respondent's familiarity and experiences with it helps us place their views on chatbots as a potential solution into context.

Open Data Awareness

Initially, we asked how familiar our respondents are with the term open data and if they have accessed or used it for any purpose, the results are displayed in figures 5.16 and 5.17 below. The findings reveal a balanced distribution in range, with slightly favoring the side associated with greater familiarity and usage of the open data. A higher level of familiarity among most respondents can be beneficial because it indicates that they might have some prior knowledge or experience related to open data. This can lead to more informed and nuanced responses during the rest of the survey, providing us with more insights and suggestions with open data. The diversity in responses also enriches our understanding by providing multiple perspectives.

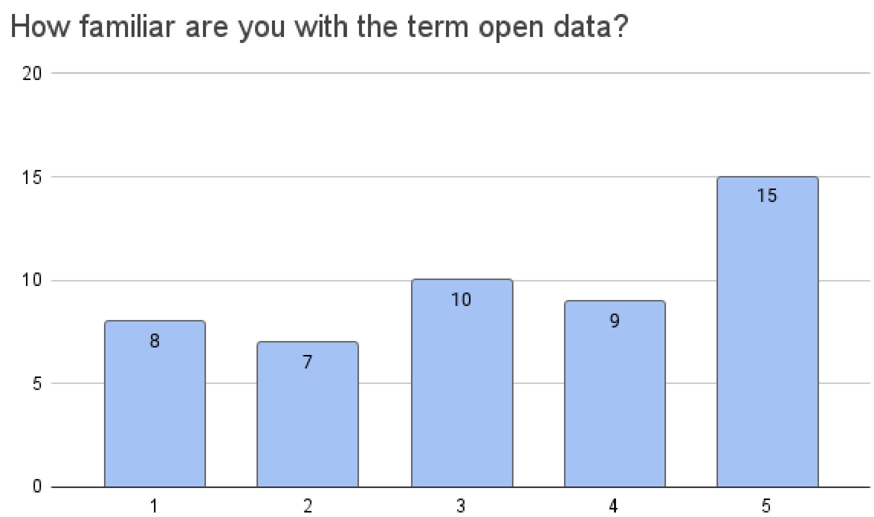


Figure 5.16: Representation of respondents' familiarity with the term open data, measured on a scale from 1 (not at all) to 5 (very familiar).

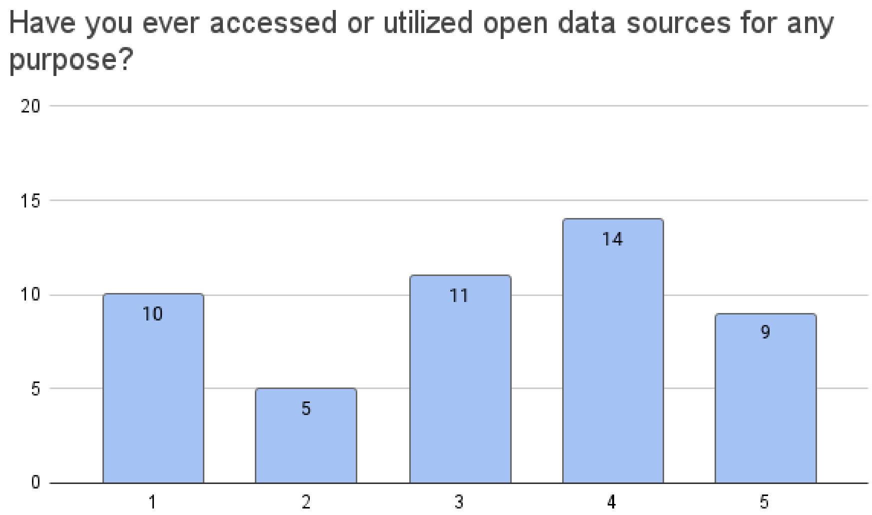


Figure 5.17: Representation of respondents' frequency of accessing or utilizing open data sources, measured on a scale from 1 (never) to 5 (frequently).

Furthermore, we inquired whether respondents who answered yes to some extent to the lateral question, *'"Have you ever accessed or utilized open data sources for any purpose?"* could provide details about the contexts in which they used open data. Our analysis reveals that many respondents have indeed utilized open data in various contexts such as coding, school projects, and work-related tasks. Notably, these respondents consistently rate their familiarity and utilization score between 4 and 5, while also tending to have higher levels of education. However, some respondents have high familiarity and utilization scores despite demonstrating a misconception of the term open data in their responses. This group displays a lack of clarity in their understanding of open data. Interestingly, they do not share a common pattern in terms of education or occupation but are distributed across various groups. These respondents commonly hold the misconception that they are using open data, but in reality, their usage is mediated through third-party services rather than direct access to open data sources. Recurring responses mention activities like checking bus schedules, using weather services, relying on Google, and utilizing GPS descriptions. This indicates a potential misunderstanding or lack of awareness about the nature of open data.

The group of respondents who have little to no experience in working with open data expressed their reasons for why displayed in figure 5.18.

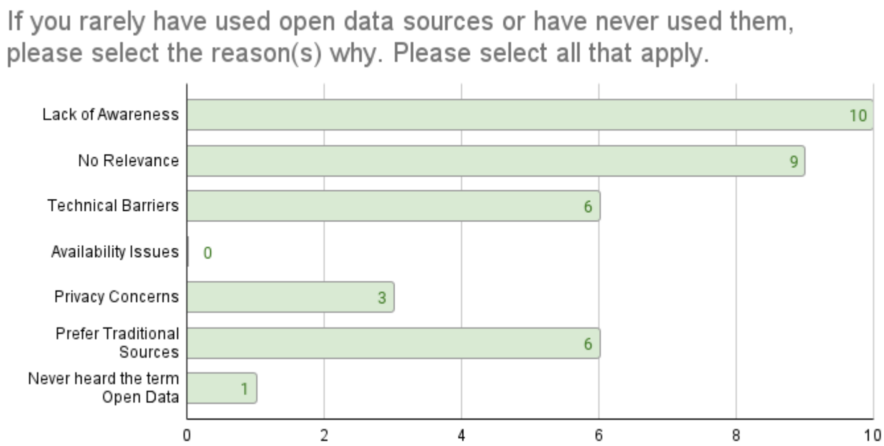


Figure 5.18: Representation of respondents' reasons for rarely or never utilizing open data sources. Respondents were asked to select all applicable reasons.

Respondents answering this question differed from answering between 1-3 on the scales of utilization and familiarity. Analysis of their responses reveals these themes;

Lack of Awareness: Ten respondents express that they rarely or never use open data due to a lack of awareness. This suggests that they might not be familiar with what open data is or how to access and utilize it. Addressing this theme implies the need for educational efforts to increase awareness about open data.

No relevance: Another common theme is the perception of no relevance. Nine respondents feel that open data sources are not directly applicable to their needs or interests. They may not see the practical uses of open data in their everyday lives or work contexts. Addressing this concern involves demonstrating the value of relevance of open data to various domains that they might not be aware of.

Technical Barriers: Six respondents cite technical barriers as a reason for the absence of open data usage. This suggests that they may encounter challenges in accessing or using open data sources, such as difficulties in finding, understanding, or working with open data. Overcoming these technical barriers through user-friendly interfaces and guidance could encourage greater adoption.

Privacy Concerns: Privacy concerns are expressed by three respondents. They may be hesitant to use open data due to concerns about their personal information exposure. Addressing these concerns can involve ensuring data privacy and security in open data applications.

Prefer Traditional Sources: Six respondents indicate a preference for traditional sources over open data. They may be more comfortable with sources they already are familiar with and see no compelling reason to switch to open data sources. Demonstrating the advantages of open data over traditional sources could be a solution to influence their preferences.

Never heard of Open Data: One respondent admitted to having never heard of the term open data. This highlights the issue of awareness regarding the concept of open data.

These responses underline the significance of awareness, relevance, and technical support as primary factors influencing open data utilization. By addressing these aspects through educational initiatives, demonstrating the relevance of open data, and simplifying its accessibility, it may be possible to encourage more individuals to embrace open data as a valuable resource.

Utilization of Open Data

On the contrary, we also asked the respondents which open data sources they are currently using, if any, the results are listed in figure 5.19. The options available also come with a short explanation of the content, so that it should be easier for the respondents to recognize. It includes the following clarifications; *Government Datasets* - (Data provided by government agencies, including census data, public records, and official reports.), *Websites* - (Open data accessible directly from websites and online platforms.), *Specialized platforms* - (Data from dedicated platforms designed for open data sharing and access.), *Data repositories* - (Data hosted in repositories or databases designed for open data storage and distribution. Example: GitHub.), and *Datasets* - (such as Weather, Geospatial, and Transportation Data, etc.).

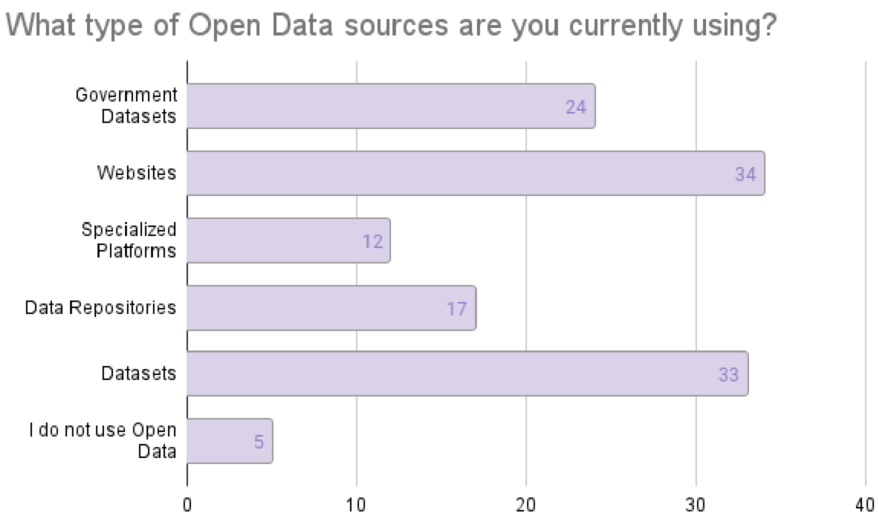


Figure 5.19: Representation of what open data sources respondents are currently using. Respondents were asked to select all applicable.

Firstly it is important to take note of the relatively low number of respondents who explicitly state that they do not use open data (5 responses). This implies that the majority of participants have some level of interaction with open data. However, the high number of responses related to "Websites" and "Datasets" compared to the other categories suggests an interesting trend. The abundance of responses in these two categories might indicate that some respondents might have a misconception about their use of open data. This relates to the suggested theory raised in the previous section, where some respondents appear to believe they use open data because they accessed information from websites or similar, but in reality, they might not have directly engaged with open data in pure form. This highlights the importance of distinguishing between direct usage of open data and accessing data via intermediary platforms or websites. This finding demonstrates the relevance of suggesting a mechanism such as chatbots to bridge the gap between people and open data sources.

Interest in Open Data

There is little use for solutions, if there does not exist an interest in exploring them. Therefore, it is interesting to address the next question in our survey, shown in figure 5.20.

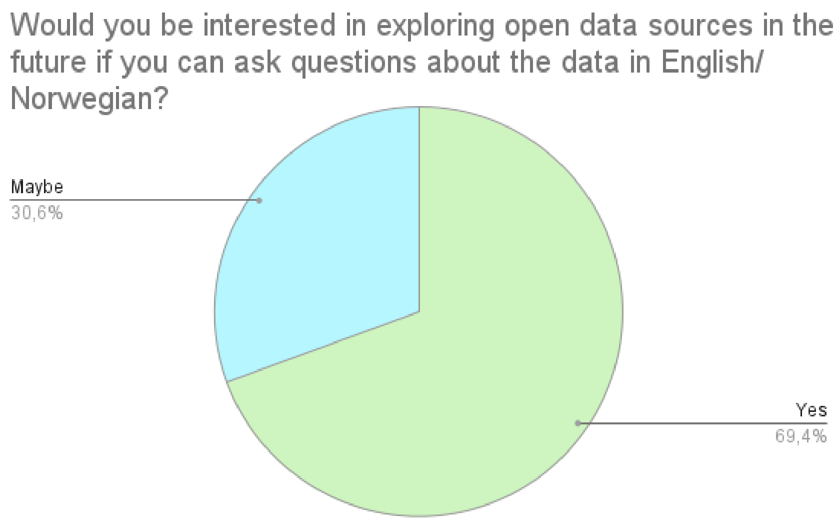


Figure 5.20: Representation of respondents' interest in exploring open data sources in the future, contingent on the ability to ask questions about the data in both English and/or Norwegian.

The majority of respondents expressed a clear interest in exploring open data sources when equipped with the capability to ask questions in English or Norwegian. This response indicates a high level of receptivity to the idea and suggests a potential willingness among the respondents to engage with open data if some existing barriers are resolved. When examining the responses from participants who answered the question about their reasons for limited open data usage and comparing these responses to the current question about their interest in exploration, a fairly balanced distribution becomes evident. This outcome indicates a certain diversity in the attitudes and motivations of the respondents. While a significant portion of respondents chose "maybe," this group still represents a considerable level of openness to exploring open data sources in the future. It's important to recognize that many respondents in this group might have practical concerns that could be eased through more information or a smoother user experience. An intriguing finding from this question is that no respondents answered with a direct "no." This suggests a general absence of strong resistance to the idea of exploring open data when related barriers are removed.

When comparing this question to previous responses, particularly those regarding familiarity and utilization of open data, the overwhelmingly positive and open responses to this question signal that suggesting a language-related conversational interface could significantly boost interest and engagement in open data sources. It demonstrates that when made more accessible and user-friendly, open data has the potential to appeal to a broad range of individuals, including those who may not have been extensively exposed to it before. This information provides valuable insights for shaping strategies to promote open data utilization.

Challenges In Open Data Usage

Lastly, we asked our participants to address the challenges they encounter when wanting to utilize open data. From these responses, we notice these common trends;

- Data quality and accuracy are fundamental concerns
- Accessibility issues often revolve around outdated standards or platforms.
- Documentation, formatting, and the suitability of data for user needs are key challenges.
- The public sector's role in making data accessible and structured is a factor.
- Lack of knowledge is also a barrier, indicating the need for improved user education and awareness.

These findings supply to our research by highlighting the real-world issues respondents face when engaging with open data. It emphasizes the potential role of chatbots in addressing some of these concerns, such as guiding users on data format or providing explanations about data quality and relevance.

Chatbot Insights

Initially assessing participants' familiarity with open data provides a foundation for an exploration of how chatbots can improve open data access. Subsequently, we move into the domain of chatbots, investigating our respondents' perceptions, past experiences, and preferences regarding their use.

Chatbot Interactions

The frequency and types of chatbot interactions among our respondents are also important for understanding patterns and insights in user behavior. As shown by the statistics in figure 5.21, we can observe varying levels of frequency with chatbots, unraveling the relevance and impact chatbots have in respondents' lives. By examining these different levels of frequency, we can also gain insights into the factors that influence their usage patterns. This analysis can help us understand the appeal and utility of chatbots across different user segments.

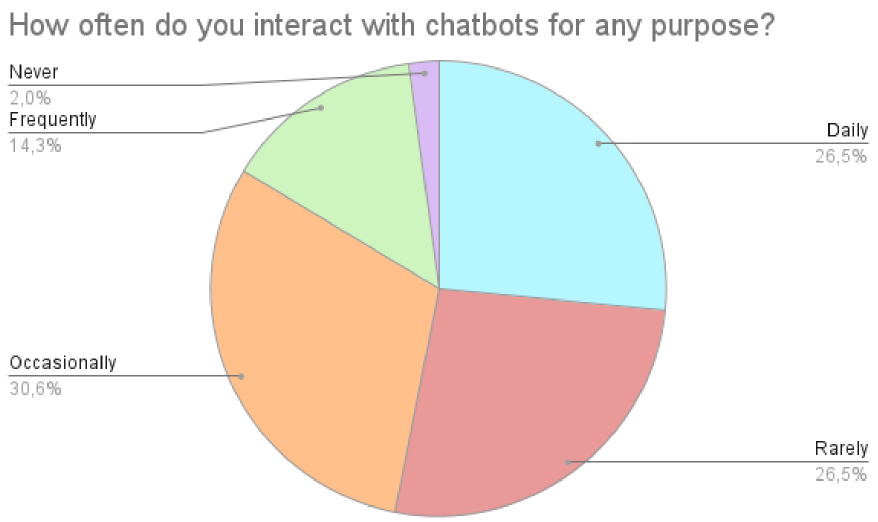


Figure 5.21: Representation of respondents' frequency of chatbot interactions.

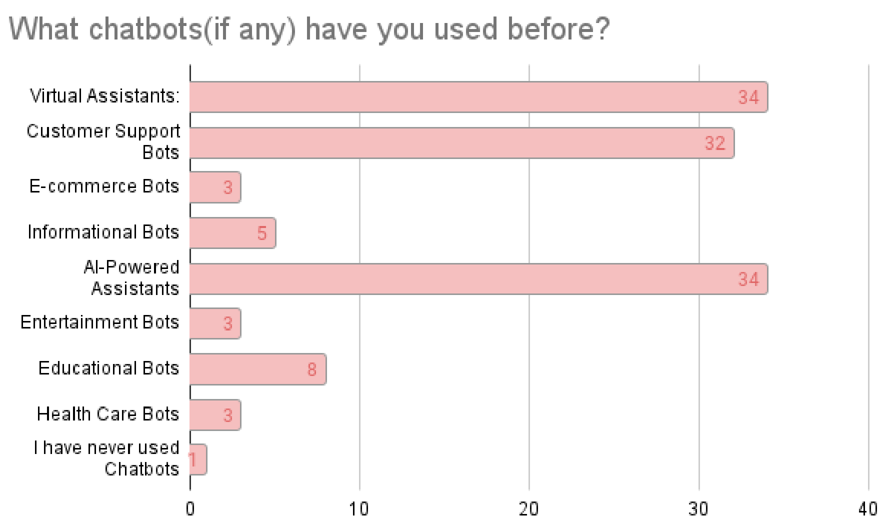


Figure 5.22: Representation of which chatbot(s) the respondents use. Respondents were asked to select all applicable.

The representation of statistics regarding chatbot categories is illustrated in figure 5.22. Starting with the observation of AI-powered assistants and virtual assistants being the most common in our group of participants with 34 respondents each claiming to use them. AI-powered Assistants such as Chat GPT, and Virtual Assistants such as Siri, Alexa, or Google, tend to be among the most commonly used chatbots. This popularity may be due to their NLP capabilities and advanced Language Model technologies. These features enable them to simulate human language and interactions effectively. Users find this human-like communication appealing as it offers a more conversational and intuitive experience. Whether it is seeking information, setting reminders, or controlling smart devices, these AI-powered chatbots excel in understanding and responding to a wide range of queries, making them a preferable choice.

Following closely are the Customer Support Bots with 32 respondents. This may be a well-liked choice as it often offers the alternative of connecting to a human customer support representative when complex issues arise. These bots aim to address common inquiries and provide initial troubleshooting assistance. If the chatbot cannot resolve the issue, it typically offers a seamless transition to a human support agent. This makes them an attractive component of customer service operations.

The other categories show a modest interest level, compared to the favored options. The usage of educational, e-commerce, informational, entertainment, and healthcare bots is relatively limited, this can be due to several factors. AI-powered and virtual assistants can perform a wider range of tasks, from answering general queries to controlling smart devices. In contrast to general-purpose AI models like Chat GPT (2.7.1), each of the less favored bots can also have implemented language models, but still provide customized solutions for particular use cases within their domain. Therefore, their interest level will depend largely on their interest level in each domain. They are not designed for everyday tasks and may be used sparingly or only in specific situations. Another reason for the low usage of these bots may be due to limited awareness of these bot categories. If users are not aware of the benefits these bots offer, they are less likely to seek them out. Overall, the popularity of different types of chatbots is influenced by their versatility, practicality, and awareness among users. The limited interest in specific bot categories might reflect the fact that these bots serve more specialized or less common use cases compared to the broader functions of AI-powered assistants and Customer Support Bots.

Exploring Perceptions of Chatbots for Open Data Access

The following questions reveal the qualitative aspects of how respondents view the potential of chatbots for accessing open data and it aims to understand what specific features users consider important when utilizing chatbots in the context of open data.

Upon analyzing the responses to the question, "*Do you think chatbots could make it easier to access and use open data? Why or why not?*", reveal several points and patterns. Many respondents see chatbots as a potential tool to make accessing and utilizing open data easier, and they are perceived as helpful for explaining complex information and providing quick answers. The convenience of asking chatbots questions, especially during moments of confusion or when specific data is needed, is appreciated among multiple participants. Another assessment is that several respondents emphasize the importance of chatbots having knowledge of the data they are seeking. If chatbots can effectively locate and explain the required data, users are more likely to find them valuable. Some participants express uncertainty or mixed feelings about chatbots, suggesting that the effectiveness of chatbots in assisting open data access may depend on their quality and knowledge. A preference for finding data themselves was also stated by a few, indicating that chatbots may not be suitable for all users' situations. The significance of chatbots such as Chat GPT was mentioned, which have advanced NLP capabilities. These advanced chatbots are expected to provide more detailed and context-aware answers. The ability of LLMs (2.7) to structure data makes it more understandable and appreciated by some respondents. One of the perceived benefits of using chatbots to access open data is that it would make the users find the right data sources faster, saving them time and effort. Also, the potential chatbots have to offer recommendations, create examples, and provide in-depth explanations was seen as valuable for open data users. Overall the users are looking for more user-friendly and intuitive ways to interact with open data, and they believe chatbots could fulfill this role. While most of the respondents have positive views, some express concerns about chatbots providing biased or untrustworthy information. They believe that critical data analysis is still required to validate chatbot-provided information. They consider chatbots valuable if they facilitate interactions with APIs or offer clear "how-to" instructions.

The collected responses indicate that there is potential interest in chatbots for open data access and utilization. Users perceive them as beneficial tools to simplify the process, especially when these chatbots seem intelligent and capable of providing accurate and structured information. However, some users also acknowledge that chatbots may not be the perfect solution for all scenarios and may need to improve in terms of quality and reliability. These insights can help guide the development of chatbots for open data-related tasks.

While several of the responses to this next question, *"Are there any specific features or functionalities you would like to see in chatbots to better assist in open data utilization?"*, align with responses already given in the previous question, these additional aspects are noteworthy features mentioned in this set of responses;

- *Explanation and Guidance*: Respondents emphasize the need for chatbots to provide explanations about the data retrieved, instructions on how to retrieve it, and frequently asked questions.
- *Study-related content*: The desire for study-related content suggests that participants see chatbots as valuable tools for academic and research purposes.
- *Support for various input and output formats:* Users express a need for chatbots to support a wide range of input and output formats, not just text. They would find it useful if they could handle requests for data between specific dates and deliver results in formats like ZIP files.
- *Custom Data Analysis:* The need for a function that allows users to upload large datasets for chatbots to analyze based on their requirements is mentioned. This suggests an interest in advanced data processing capabilities.
- *Improved language models:* The implementation of LLMs in chatbots is frequently mentioned to make responses more coherent and meaningful.
- *Live data fetching:* The ability of chatbots to fetch live data from the internet is advantageous.
- *Enhanced language understanding:* Users would appreciate chatbots with the ability to understand various languages and dialects.
- *Machine-Readable formats:* Some users highlight the importance of chatbots providing data in machine-readable formats.
- *Simpler and smarter interactions:* Respondents appreciate the idea of chatbots offering smart suggestions with minimal interaction.
- *Suggestions and categorization:* The ability to suggest data categories based on user prompts and characteristics of the available data. This can help users who are unsure of what they are looking for.

These perspectives emphasize the diverse range of features that participants find important in chatbots for open data access. It underlines the need for chatbots to offer comprehensive support, including data explanations, a variety of data formats, and advanced data analysis capabilities, to meet user expectations.

Chatbot Use Cases for Open Data Access

The respondents express a diverse set of use cases they find important when asked *"Can you describe any specific scenarios or use cases where you believe chatbots can be particularly helpful in open data access?"*. The use cases specified were;

- *General Data Retrieval:* by simplifying the process and saving time for obtaining specific data.
- *Cooking/ Planning*
- *Academically Related Tasks:* seen as valuable for academic purposes, particularly scenarios related to studies, research, or thesis writing to help

with data analysis and access to information.

- *Database Searches:* in situations where chatbots act as user interfaces for searching databases or similar systems.
- *Public (National) Data:* a potential tool for accessing public national data, including maps and other related resources.
- *Data Visualization:* by being employed to generate graphs and charts directly from open data, as well as to export datasets tailored to specific needs.
- *Statistics and news retrieval:* by assisting in extracting statistics and new sources quickly and conveniently.
- *Research and analysis:* by analyzing malware and vulnerabilities, chatbots can facilitate data access and processing.
- *Everyday Information needs:* for daily queries or quick informational needs that do not involve sifting through articles and web pages, chatbots are seen as a potential solution.
- *Data Structuring and Cleaning:* can be used to structure and clean data, making it more usable for specific projects.
- *Guidance and Data import:* can provide guidance and support in finding the right data sources and assist with the import and formatting of data for individual projects.
- *Document and Visa Applications:* Assist with document preparation, such as visa applications and checklists.

These responses cover a broad range of potential use cases for chatbots in accessing open data, there are a few use cases that might not be well-suited for this context or could benefit from further clarification. The response of *Public National Data*, mentioning maps as a use case is somewhat unclear. While open data often includes geospatial information and maps, the response does not provide specific details on how chatbots would be used in this context. The use of chatbots for *Cooking/ Planning* purposes is rather vague. While chatbots can provide recipes or assist with planning, it is not typically used for open data access. Further, the response in *Document and Visa Application* are tasks that may require personal or confidential information. As such, these use cases would need to address privacy and security considerations. It is possible some respondents had different interpretations or contexts in mind when suggesting the use cases. Further exploration and clarifications could help uncover these specific scenarios in which they meant chatbots could be useful. To summarize these use cases emphasize the versatility of chatbots in different scenarios, from academic and research contexts to everyday informational needs and data processing tasks. Users see chatbots as a means to simplify processes, enhance data accessibility, and improve efficiency across various domains.

Communication Channels

In the following question, we inquired about the communication channels preferred by the respondents for interactions with chatbots. These preferences are helpful in evaluating user expectations and can be used for measuring our own solutions. The results reveal a distribution of preferences as shown in Figure 5.23 below.

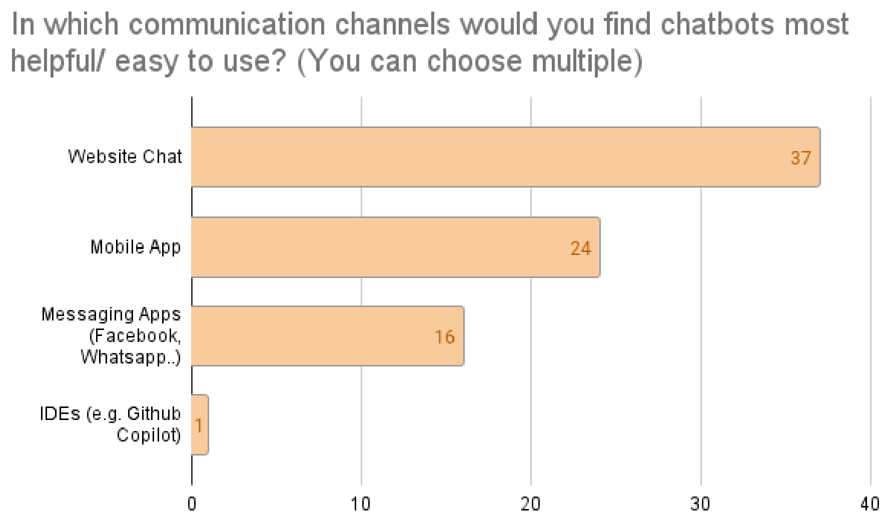


Figure 5.23: Representation of which communication channels respondents prefer for chatbot usage. Respondents were asked to select all applicable.

From these results, we observe that a majority of respondents favor using chats on their own website as a preferable chatbot interaction method. This may indicate the convenience and accessibility of chatbots integrated into websites, Mobile apps also appear to be a popular choice, suggesting the importance of mobile-friendly chatbot interfaces. Messaging apps like Facebook and WhatsApp are another relevant choice, though less favored compared to website chats and mobile apps. The single response indicating a preference for IDEs (e.g., GitHub Copilot) for chatbot interaction suggests a niche interest in integrating chatbots directly into development environments.

Challenges in Chatbots for Open Data access

It is also important to evaluate if respondents see any challenges with using chatbots for open data access. Understanding the challenges users may expect with chatbot utilization for open data access is valuable for refining chatbot design, ensuring a user-centric approach, enhancing the user experience, and proactively addressing potential issues in the development process.

The responses to this question offer some valuable insights, such as that some respondents expressed concerns regarding the difficulty in formulating correct questions or prompts for chatbots, especially those with less technical expertise.

It was addressed that chatbots often require users to be concise and clear in their queries, which some users might consider an obstacle. Additionally, concerns about chatbots delivering incorrect or biased information are raised. Respondents highlight the need for chatbots to improve their ability to understand user intent and provide accurate responses. Remarks regarding a potential lack of trust among users who are less technically inclined are made, as they may not fully comprehend how chatbots or open data function.

On the other hand, a few respondents believe that chatbots can make it easier for users with less technical knowledge to access open data if the chatbots offer user-friendly interfaces and predefined prompts. Overall, these responses are helpful in understanding various challenges and areas of improvement for chatbots in the context of open data access.

Additional Comments

The last two questions let the respondents add additional comments about the potential of chatbots being used for open data access, or just an additional comment to the subject of the survey in general. Not too many respondents gave additional comments, but those who did offer a variety of perspectives. There is one suggestion for using multiple-choice options to make chatbots more accessible to a broader group of people. The opinion is based on simplifying the interaction for users with different levels of technical expertise. Concerns are raised regarding the use of personal information conflicts with the General Data Protection Regulation (GDPR). This comment highlights that the creators of chatbots may need to be mindful of privacy regulations and data usage, especially when the user's intent and provided information can indirectly reveal personal details. Another comment enables the idea of users to develop their applications and customize solutions is suggested. They mean this can empower users to tailor open data access to their specific needs and potentially improve current offerings provided by manufacturers. One respondent expressed uncertainty about the topic, indicating a lack of sufficient knowledge to provide detailed comments.

The comments highlight some potential benefits and concerns about the role of chatbots in open data access, with a focus on user-friendliness, privacy, customization, and regulatory compliance.

5.2.3 Validity Considerations in Survey Analysis

In this section, we will discuss the threats to validity and then detail the strategies and actions taken to mitigate them. Our objective in this section is to ensure the accuracy and reliability of the findings from our survey.

Internal Validity concerns whether the observed effects within the experiment can be attributed to the manipulated variables [36]. In the context of our survey analysis, the internal validity threat is related to the potential of irrelevant variables or unconsidered factors influencing the observed outcomes or relationships between variables. This threat could emerge if, for example, we failed to account for variables like participants' prior experience with chatbot and open data technology, when we were indeed studying the impact of chatbot use on open data access. If variables like this were not considered or controlled for, it could have influenced the results differently. To mitigate this threat to internal validity, we maintained a careful survey design process, including three iterations involving ourselves and project supervisors to ensure the survey's precision.

External Validity relates to how well the survey findings can be generalized to a broader population or other settings [36]. The external validity threat is the potential limitation in generalizing our survey findings to a larger population, possibly due to the characteristics of our sample or the specific condition in which the survey was conducted. This threat might arise if our sample is not representative of the broader population we aim to draw conclusions about, or if the survey conditions are significantly different from the situations we would want to apply the findings to. Considering the small sample size, we acknowledge the existence of this threat. Furthermore, our survey was shared predominantly within our network, which is composed of a substantial number of developers. This selection of participants is a consequence of our survey distribution method. To mitigate this threat, we ensure to be transparent about the limited size of our dataset and describe the results as observations rather than drawing definite conclusions. We provide detailed information about our sample and the survey's context, acknowledging that it may not fully represent the entire population. A variety of people from the general population is still represented, thereby improving our understanding of different groups of interest.

Construct Validity focuses on whether measurement methods accurately reflect the underlying theoretical constructs [36]. This threat can arise if the questions or metrics used in the survey do not align well with the theoretical concepts one intended to investigate. For example, as we intend to investigate the interest in chatbot usage with open data, but our survey question did not effectively capture this construct, construct validity is threatened. To mitigate this threat, we ensured that our survey questions and response scales are based on established measurement tools and relevant theory such as our mixed-method research approach. During the analysis, we ensure to make efforts to demonstrate how the responses to our survey items relate to the theoretical concepts we are investigating.

Conclusion Validity refers to the extent to which the conclusions made from the data collected are accurate and well-supported. It involves assessing whether the data collected genuinely supports the conclusions drawn [36]. In the context of this survey analysis, conclusion validity is threatened if there are concerns about the accuracy of the conclusions we draw based on the survey responses. This threat can arise if our data analysis techniques or interpretations are flawed or if we overgeneralize our findings. To mitigate this threat, we aimed to avoid making broad claims and conclusions beyond what our data can support. We tried to be transparent about the limitations of our research and discuss potential alternative explanations for our results. By providing a balanced and thorough assessment of conclusions, we can enhance the validity of our survey analysis.

5.3 Comparison and Integration

This section draws parallels between the insights gained from our comprehensive survey on open data access and the functionalities incorporated into our developed web application. The survey highlighted key considerations, including user interest, challenges, and perceptions of chatbots. Our web application, designed with these insights in mind, reflects a commitment to addressing user demands and facilitating seamless interactions with open data.

In summarizing the findings from our survey, we note the following key observations. The survey responses emphasize a considerable interest among users in exploring open data, particularly when equipped with the ability to ask questions in English or Norwegian. The identified challenges include data quality concerns, accessibility issues, and the need for improved user education and awareness. For chatbot interaction and perception, survey respondents demonstrate varying levels of engagement with chatbots, with AI-powered and virtual assistants being the most commonly used. Concerns about biases and the importance of accurate responses were noted. Diverse use cases were outlined, covering tasks from general data retrieval to academic purposes. Preferences for communication channels varied, with website chats being a favorable interaction method.

Our web application is designed with features aimed at enhancing the user experience and addressing potential challenges identified in the survey. The application strives to offer a user-friendly conversational interface, also for less technical users. The visual representation of the user experience 5.1 emphasizes a commitment to clarity and simplicity. From the initial sketch in figure 1.1 to the final/current result in figures 5.2, 5.3, 5.4, the design aims to ensure a smooth interaction flow, complemented by a user guide for effective utilization. The incorporation of a JSON visualizer aims to enable users to interactively explore responses from API endpoints, which can be an advantage while users are interacting with the chatbot. The integration of the chatbot and its widget is intended to facilitate dynamic conversations.

Our web application, developed with insights from the survey, aspires to be the starting phase of a strategic response to user needs and challenges in open data access. By aligning with perceived user preferences, the application aims to serve as an effective tool for promoting accessibility, user-friendliness, and engagement with open data sources. Continuous refinement will be guided by user feedback, ensuring the solution remains adaptive to evolving user expectations.

Chapter 6

Discussion

This chapter serves as an exploration and discussion surrounding the challenges encountered, including struggles in chatbot implementation, diverse approaches to NLU integration, potential weaknesses in survey responses, and the ethical considerations that shaped our decision-making process.

6.1 Challenges in Automation: Overcoming Struggles in Chatbot Implementation

In the initial phase of developing the bot application, the primary objective was to construct a chatbot from the definition of the JSON structured data source itself. The thought-of approach involved creating an application where the user could input a JSON API endpoint on the front-end. Subsequently, the back-end would fetch the API endpoint, populate the chatbot's entities with keys from the API response, and return this chatbot to the user for interaction. This approach depended on the chatbot to be created as generic as possible. One significant source of inspiration stemmed from the BODI-generator method for open data sources, as previously introduced in 3.2.2. BODI shares multiple similarities with this approach, whereas the primary distinction lies in the data source format, with the BODI-generator specifically targeting CSV files.

The development of the project began with the creation of a sample bot *GroceryBot*, which successfully retrieved grocery store items in Norway (4.3.1). However, as we progressed with the code, we recognized the complexity and challenges associated with our specific objective of creating a chatbot directly from the definition of the data source. Consequently, our overall objective diverged towards creating a versatile bot capable of handling a diverse range of JSON-based APIs. We faced prolonged periods of stagnation due to technical hurdles, challenges which can be categorized into two main areas: the clash between the diversity of JSON structured APIs and generic chatbot creation, and the difficulties encountered within the XatKit Modelling Language. Time constraints further intensified the project's challenges. We also found ourselves repeatedly reflecting on the balance between maintaining the generic nature behind bot creation while ensuring user-friendliness. Within this section, we will discuss the evolution of our project, thereby discussing the various obstacles encountered and the strategies we employed.

6.1.1 Diversity in JSON Structures

The lack of standardization in JSON structures emerged as a difficult hurdle. Unlike CSV, where data is often simple and tabular, JSON's flexibility allows for intricate and nested structures. This diversity, while powerful, introduced complexities that were time-consuming and not easily resolvable. While the outermost structure of a JSON response may be consistent, the internal content exhibits considerable variability. Encountering different scenarios, such as repeated values or data encapsulated in nested objects, demands thorough handling to avoid errors and crashes.

Another evident obstacle is the absence of common rules for JSON API keys. Varying naming rules and structures among different APIs hinder the development of a uniform approach. For instance, one API might use keys like "name" and "brand", while another employs keys like "strName" and "strBrand". This lack of consistency also extends to the data type associated with each key, leading to potential crashes if a value does not align with the expected type. Handling parameters in JSON APIs also comes with challenges. Some APIs allow for optional parameters, while others do not, and some do not support parameters at all. The multiplicity of use cases for parameters, coupled with the unpredictable nature of API responses, added complexity to the development process.

"CSV should be chosen if the data is simple, flat, and tabular. JSON should be chosen if the data is complex, nested, and object-oriented." (Pedamkar, 2018)[75]. Based on this standpoint alone, JSON structured data may not seem to be the most natural fit for a simple, generic chatbot. However, it can be argued that Java appears as a good companion to JSON data, as it is an object-oriented language, supporting complex and nested classes. To mitigate JSON structure challenges in the initial thought approach, facilitating chatbot development directly from API data, we wanted to use Java Converters. These code generators convert your JSON data into Java Objects. The JSON keys are converted to private variables with getter and setter methods, and the inner objects in JSON are converted as inner classes in Java Objects [48]. This would have provided a structured foundation for further processing. However, we encountered some issues with the implementation when using the entity library within XatKit. XatKit enables users to build a custom Named-Entity Recognition system for their chatbot engine [107]. While this feature is promising, we did not figure out how to use it correctly. Though we managed to accurately identify intents, our Entity class did not manage to extract entities necessary for responding to requests, which is further elaborated in 6.1.3 Challenges Surrounding the XatKit Bot Platform. Consequently, among other considerations described in the following sections, we decided to deviate from the initial strategy of directly generating chatbots from APIs. Instead, we shifted to our current approach of dynamically fetching real-time data from APIs.

6.1.2 The Balance Between User-Friendliness and Generic Bot Creation

During the bot creation process, we repeatedly revisited a challenging aspect, which turned out to be one of the most difficult areas to address effectively. This challenge centered around striking the right balance between the bot's generality and automation while maintaining its user-friendly nature. In the context of our survey, it was noted that some users expressed concerns about forming precise prompts, as described in section 5.2.2. It was addressed that chatbots may require users to be concise and clear in their queries, potentially being an obstacle. The emphasis on automation and generalization during our chatbot's development led us to prioritize clear and precise queries with minimal room for error. This may pose a challenge for users with this concern and decrease user-friendliness. However, a few respondents believe that chatbots can enhance accessibility to open data, provided that they offer user-friendly interfaces and predefined prompts. These are aspects our UI incorporates to enhance user-friendliness. Nevertheless, respondents emphasized the importance of chatbots to improve their ability to understand intent and deliver precise responses. Therefore, we will discuss not only the chatbot's responsiveness in terms of precision but also evaluate the user-friendliness inherent in the chosen design.

Precision Challenges in Intent Formation

When interacting with our chatbot, precision is key, particularly when it comes to the formulation of intents. Each intent represents a specific user request and adheres to the predefined intent structure and rules. Sadly, this may be at the expense of the user experience, as evaluated by some of the respondents in the survey who saw this as a potential obstacle. To accommodate a degree of flexibility, we incorporated training sentences, as well as the natural language model DialogFlow. The model allows for error spelling in the user input. However, the entities within the intents that are used for API-key retrieval must adhere to their definition and cannot deviate. This may impact user-friendliness, particularly for users with limited technical expertise who might be unaware of this requirement. One approach to this could be to implement a solution that allows for a more relaxed approach to typing the keys. Specifically, if the API key associated with an entity is not matched within the API response, a function could identify the closest key in terms of characters and return that. However, this requires complex algorithms and functions. This can be a possible add-on in further work.

Degree of Accuracy in Chatbot Responses

We experienced several difficulties regarding the delivery of precise responses. Although the bot effectively handles user queries, it lacks the capability to provide additional context or information beyond the direct answer to the question. This poses challenges especially when dealing with expansive APIs that yield extensive responses with significant variation.

User Input Give me the maximum current_price for Cheese

Bot Response The maximum current_price for Cheese is 357 kr

Above is an example of an intent to showcase the simplicity of the response given by the bot. The first intent uses the kassal.app API and incorporates the entity 'cheese' into the endpoint. This response encompasses groceries from sliced cheese to cheese-flavored desserts. To retrieve the answer, the bot loops through the entire response body, which may consist of 20 or more objects, and ultimately returns the maximum value found for the 'current_price' entity. While it successfully achieves its intended purpose of obtaining the maximum value, it falls short in providing additional details about the specific item. This area left us often pondering, what is the user truly trying to find out and to what degree can it be automatized. Defining the boundaries proved to be challenging.

We explored multiple solutions for the stated problem. One solution was to always return up to three random key-value pairs in the answer to provide context. However, the order of the key-value pairs is randomized, and multiple times we retrieved insignificant values that added no value of significance. One of the values might be an array, causing the bot's response to be extended and potentially adding confusion for the user. Another solution was to elongate the intents with the intention of allowing the user to type in several API keys they wanted to extract. We also thought of creating an intent where the user could store a fixed key variable they wanted to print out. For example, for each intent always return the key "name" as well. Below are examples of intent, illustrating the second solution mentioned.

User Input Solution 2 Give me the maximum current_price and name for Cheese

Bot Response The maximum current_price is 357 kr and name is Lemon Cheese Cake Muffins for Cheese

However, we were concerned that these additional features might create a somewhat confusing experience for the user or elevate the overall difficulty level of the chatbot. We were indecisive about whether the trade-off was worthwhile, and the complexity increased as we looked further into it. In the end, we opted to maintain the consistency of intents and entities, shifting our focus to other aspects like error handling, which is integral to ensuring user-friendliness in a bot. This particular area appears more relevant for future developments when expanding the bot's functionalities and capabilities.

Enabling Simultaneous Adaptability to Multiple APIs

At the outset when conceptualizing the functionalities of the bot, we envisioned an idea of simultaneous adaptability to multiple APIs. This involves allowing the user to input two APIs instead of just one. We perceived this approach not only as a means to enhance user satisfaction but also to add an enjoyable dimension to the overall concept. However, we foresaw several challenges at the outset of this idea after investigating the structure of various APIs, such as those previously discussed in 6.1.2. This implies the need for a considerable number of other functions to be developed before arriving at this step, with these methods advancing the bot's functionality. One of the thought-of solutions revolving around this multifaceted API approach involves a scenario where a user requests one entity from the first API and another from the second API. In such cases, we would need to create methods capable of searching through specified entities in both APIs and returning them upon finding a match. However, this approach poses complications, especially if the value of one or both entities is present in both APIs, risking the retrieval of the entity from the wrong API, which then again could impact the trustworthiness of the bot. Another point to consider is determining the appropriate behavior in scenarios where one of the APIs employs parameters, while the other does not. Overall, this idea would result in elevating the overall complexity of the bot, thus potentially compromising user-friendliness by causing confusion among users, and were therefore not prioritized.

6.1.3 Choice of Chatbot Platform

In the early stages of this project, we held little to no knowledge surrounding chatbot creation. We spent time exploring various chatbot platforms and identified Rasa and XatKit as the most suitable options for our application. XatKit had several compelling features that set it apart. Firstly, one of the project proposal supervisors happens to be the founder of XatKit and suggested it as a potential solution. Additionally, the framework utilizes the Java programming language, a language already familiar to us. Nevertheless, we acknowledged the importance of not hastily choosing XatKit solely based on our familiarity. Rasa is renowned as the most popular open-source platform for building chat and voice-based AI assistants. It utilizes the Python programming language and provides different NLU language models (2.8.2). However, we encountered challenges when using the platform. Rasa by default installs TensorFlow, unfortunately leading to technical complications on Mac computers with M1 chips. Reasonings for these complications are further elaborated in section 6.2.2. Addressing these issues required time-consuming workarounds. We also observed that numerous other bot platforms utilized ML-based chatbots, using Python and TensorFlow as their foundational technologies. Thus, these technological hurdles, combined with our time frame and familiarity with Java, led us to favor the XatKit framework.

Challenges Surrounding the XatKit Bot Platform

Overall, we had a good experience with the XatKit bot platform. Nevertheless, certain aspects of the platform contributed to the stagnation periods during development. One challenge was the lack of specific documentation related to the XatKit DSL. Though we experienced the provided tutorials on the XatKit GitHub to be helpful, they fell short of providing the comprehensive understanding we sought for the platform. For instance, some of the imported packages in the Entity library. The library shares similarities with the Intent library in that it defines entities and offers methods to create and handle them. Sadly this library became a source of prolonged stagnation due to difficulties in understanding its proper utilization, and the packages in turn affected the intents.

However, this challenge forced us to adopt a more inventive approach and to really take a deep dive into API extraction. Initially, our plan was to populate the entities with keys from the received API. Given that we couldn't figure out how to correctly utilize the entity library, this prompted a shift in our objective. With the new approach, entities were fetched in real-time and defined within the intents, a deviation from our original plan, prompting us to think more creatively and enhance our focus on generalization. With this shift, our focus intensified on data manipulation. Fortunately, XatKit's decoupled nature facilitates a smooth integration of custom methods while maintaining compatibility with the platform. The business logic of the bot became loosely coupled with the bot platform, and the functions created therefore are independent of XatKit. This not only facilitates their reusability in other Java environments but also amplifies the automation and generalization of the bot, aligning with one of our primary goals.

Reflections

In examining the survey results and recognizing the significance of a chatbot with human-like conversational abilities, we did consider alternative measures to our solution. During one of the prolonged stagnation periods, we pondered the idea of reconsidering our commitment to XatKit and instead allocating more time to address the technical compatibility issues with Python, TensorFlow, and our MacOS environment. Extensive online research revealed that MacOS might not be the most compatible platform for machine learning-based programs, particularly those essential for human-like language models [55]. One potential solution involved setting up a Virtual Machine, such as Ubuntu, to run a Windows environment, providing a more suitable environment for machine-learning programs. However, given the investment of time and effort we had already invested into the XatKit-based chatbot, we deemed it impractical to shift our entire approach mid-development. Rather than dividing our efforts between multiple platforms, potentially resulting in a partly functional chatbot in XatKit and another in Rasa (or another ML-based bot), we chose to maintain focus on refining and enhancing the already-built application. This alternative might have been more relevant if considered earlier in the development process.

Nevertheless, our choice of adhering to XatKit not only navigated us through the difficulties of data extraction but also deepened our understanding of using APIs to acquire valuable information. It underscored the importance of developing creative, but robust functions capable of parsing API endpoints and extracting essential data. While the appeal of creating a human-like chatbot was present, our commitment still remained towards opening data sources for the average person. We emphasized functionality and usability over the pursuit of a fully human-like interaction.

6.2 Diverse Approaches to NLU Integration

Within this section, we discuss the exploration and various methods for integrating NLU and also identify and evaluate the trade-offs associated with each approach. These trade-offs range from the balance between accuracy and efficiency to navigating the complexities of privacy and scalability.

6.2.1 Natural Language Understanding Platforms: DialogFlow vs. XatKit NLU

While aiming to develop chatbot technology for open data access, we explored some different NLU platforms, as explained in Natural Language Understanding

Platforms. The choice of an NLU platform is an important step of the development process because it enables effective communication between chatbots and their users. Following this section, we discuss our considerations regarding two NLU platforms, DialogFlow and XatKit's own NLU server.

DialogFlow emerges as a robust and versatile candidate in our NLU integration evaluation. This platform empowers developers with the tools and capabilities required for constructing conversational interfaces across applications and services. The platform has built-in tools for intent recognition, entity recognition, and context-aware conversational experiences [28]. XatKit's approach to NLU distinguishes itself by offering flexibility and pragmatism. In the NLU domain, intent classifiers are often implemented using neural networks, assessing user utterances to determine the behind their input. XatKit's NLU Engine stands out by empowering developers to tailor NLU models to their chatbot's unique semantics, ensuring that chatbots consistently deliver the intended results. Unlike many other platforms, XatKit's engine simplifies the process of fine-tuning the chatbot's NLU model, making it accessible to developers aiming to optimize their chatbot's performance [108].

6.2.2 Choice and Considerations

The NLU integration of our project originated with an exploration of XatKit's NLU Server, recognizing its potential, especially its flexibility in fine-tuning NLU models to cater to a chatbot's specific needs. However, during the chatbot development process, practical factors influenced our decision-making process. While the XatKit NLU Server holds great potential, certain implementation challenges arose. We initially focused on establishing the chatbot's logic before adapting NLU integration. Hence, the implementation of NLU was set to the final stages of development. This led to a limited time scope for the implementation process, preventing us from fully harnessing its potential within the time available. In order to install the XatKit NLU engine some key requirements are based on Python libraries such as Tensorflow, and the ASGI web server package, Uvicorn. We both developed this project using our Mac computers with M1 chips. The M1 chip architecture is based on ARM, while most traditional Python packages and libraries were developed for the x86_64 architecture. This transition to ARM architecture required various changes and updates to make Python and its ecosystem compatible with M1 Macs. Some Python packages may have dependencies or code written in a way that is incompatible with ARM-based architectures. This can lead to crashes or errors when running these packages on M1 Macs [55]. A lot of time-consuming workarounds were then necessary to make the key requirements compatible with our Macs. Consequently, we chose to prioritize differently, since DialogFlow was a platform we were already familiar with the implementation of, it therefore emerged as the optimal choice for NLU integration.

Ideally, our vision encompassed implementing a large language model into our chatbot. Large language models bring advanced problem-solving capabilities to chatbot development, and as comprehended from Analysis and Assessment, section 5.2, this is a feature highly requested by users to enhance chatbot interactions. This was however not a priority within this project's time frame, instead, it remained an exploratory feature that we aimed to take advantage of if time permitted.

This decision underlines the consideration of practical factors including development time, platform familiarity, and alignment with project goals. The priority of achieving as advanced chatbot logic as we could within the given time frame took precedence over the integration of an advanced NLU model. The selection of DialogFlow for this project reflects the need for a balance between aspiration and pragmatism in development processes. We anticipate that future endeavors will delve deeper into the integration of a larger language model to unlock the full potential of chatbots in the context of open data access, aligning with the insights collected from our survey.

6.3 Addressing Potential Misinterpreted Survey Responses

In order to obtain meaningful insights and valuable conclusions, a survey was conducted for this study to gather data from participants. However, like any research endeavor, challenges arise and survey responses may occasionally be open to misinterpretation. As mentioned in 1.6.2, Brent and Leedy (2019) [11] expressed that research can be misinterpreted as merely gathering information, documenting facts and extensively searching for a subject matter. Research encompasses more than that; it also involves collecting, analyzing, and interpreting data to gain a thorough understanding of a phenomenon. This section delves deeper into situations where survey responses may not align with their intended meaning or where the complexities of human language create ambiguity. Our goal is to identify and address these misinterpretations to ensure that our analysis and findings accurately reflect the perspectives and experiences of the participants. Through precise examination and discussion, we clarify the intricacies of these responses, emphasizing the need for a nuanced approach when navigating the domain of human communication within the survey context.

As mentioned in section 5.2.2, it is highlighted that some respondents in the survey might have some misconceptions of the term open data. The combination of answers especially regarding familiarity and utilization, current use of open data sources, and chatbot use cases for open data reveals a pattern upon closer examination. It becomes evident that these respondents might have a misconception of the term "open data". Despite their misconception, we observed that these respondents come from diverse education and occupational backgrounds. If our dataset had been larger, there might have been a possibility of detecting a pattern related to these backgrounds. However, in our current sample, we did not gather enough responses to make such observations. Interestingly, the observed pattern was that these respondents believe they are using direct open data sources, but instead, their usage is facilitated through third-party services. The recurring responses mentioned activities such as checking bus schedules, using weather services, relying on Google, and utilizing GPS descriptions. While many of these services utilize or provide direct open data sources, they are not direct open data sources in themselves.

This indicates a potential misunderstanding or lack of awareness about the true nature of open data. The reasons behind these misconceptions can be attributed to various factors. One possible explanation is the ambiguity surrounding the term "open data" itself, as individuals can interpret it differently. The lack of clear communication and education about open data may also contribute to the misunderstanding. The respondents might not have been adequately informed about the principles and practices of open data, leading to their misconceptions. Although we included a section to clarify open data in the survey, we have to take into account that perhaps our explanations led to misunderstandings by being too complex for people with less awareness of the term. Perhaps providing picture examples could have reduced misinterpretations of the true nature of open data.

To address these misconceptions, it is important to provide clear and comprehensive explanations of open data, its definition, and its characteristics. Education and awareness campaigns can play a vital role in helping individuals understand the true meaning and potential of open data. By addressing these misconceptions and enhancing understanding, we can ensure that the analysis and findings accurately reflect the perspectives and experiences of the participants, leading to more meaningful insights and conclusions. We incorporated findings from this study into our project by developing a tutorial-style user interface. Our aim is to clarify the term and its application within chatbots. However, user testing remains to determine whether the interface is helpful and easy to understand, or if it's too complicated and needs improvement.

6.4 Ethical Considerations at the Intersection of Chatbots and Open Data

While researching the development of chatbots, as well as the potential benefits for the public's use of open data, it becomes important to contemplate the ethical concepts surrounding this research. This section analyzes ethical considerations in human-to-machine interactions, aiming to enhance our understanding of developing reliable technologies aligned with societal values. Although many ethical considerations are domain-specific, our research transcends these specialized fields and encompasses themes such as transparency, privacy, and accountability.

Open data, characterized by its transparency and accessibility, also dispose of its own set of ethical challenges and considerations. These concerns revolve around issues such as data privacy, responsibility, accuracy, and consent [98]. With data becoming progressively more accessible, a balance is needed between openness and protecting individuals' privacy, addressing consent for data incorporation. Another important aspect is the accuracy of open data since faulty predictions can impact the judgments and actions of people who depend on the data [49]. Thus, it is essential to include a discussion on the ethical challenges posed by both chatbots and open data.

Under the scope of ethical considerations, we delve into two significant aspects. The first discusses the issue of trust in chatbot interactions and how the responsibility shifts when users choose the APIs that provide information. The second one discusses user worries about possible violations or misuse of their personal information while interacting with chatbots, how this relates to our approach and strategies to mitigate these risks

6.4.1 Responsibility and Trust: Shifting Accountability Focus to Data Sources

The issue of trust in chatbot-driven interactions is a discussed subject, and also a fact reflected in responses from our survey. Users may express concerns about the reliability of information provided by chatbots, questioning the accuracy and credibility of the data they receive. However, in the context of our research approach, where information is directly fetched from APIs chosen by users themselves, the dynamics of responsibility and security take on a different perspective.

One important distinction lies in the source of the data. Traditional chatbots may rely on pre-packaged information or external databases. Additionally, chatbots like ChatGPT are also highly leveraged these days, and many people may use it as a search engine. However, users may not be aware that ChatGPT operates as a language model and not as a flawless information retrieval system. As a language model, ChatGPT may make mistakes or provide information that is not up-to-date, causing potential concerns about reliability [51].

In contrast, our chatbot approach addresses these concerns. Unlike these traditional chatbots or language models, our chatbot extracts real-time information directly from the APIs nominated by users. It is important to acknowledge that while our approach addresses the security risk associated with false information, this method places a significant portion of responsibility on the users themselves. This emphasizes the need for them to ensure trustworthiness and security from the APIs they select. In essence, our approach aims to enhance the collaborative responsibility shared between users and the chatbot, creating a relationship where users actively contribute to the reliability and security of the information they seek through the chatbot interface.

6.4.2 Personal Privacy and Security in Chatbot Interactions

In the survey results from chapter 5, some respondents raised concerns about user privacy and security. Users may question the safety of sharing information under interaction with chatbots, fearing potential breaches or misuse of personal data. Users' apprehensions about privacy and security often stem from uncertainties about how their data is handled and stored. Common worries include the fear of data breaches, unauthorized access to sensitive information, or the misuse of personal details provided during chatbot interaction.

While these concerns are valid, the actual security risks associated with using chatbots can vary based on multiple factors, including chatbot design, the platforms it operates on, and the security measures implemented by its developers. In the case of our approach, where the chatbot primarily interacts with open data APIs, the risk to users' private information is inherently minimized. By prioritizing the extraction of information directly from APIs chosen by the users, this design choice limits the collection and storage of user-specific data. You could think of the chatbot as a conduit, facilitating the flow of information without retaining personal details. Consequently, the security measures primarily lie with external API data sources that the users choose for the chatbot to interact with. While the inherent risks are low in our approach, users are encouraged to follow general guidelines for safe interactions with chatbots. These include being cautious about sharing overly sensitive information and verifying the credibility of data sources.

Chapter 7

Conclusion and Further Work

Throughout this thesis, we aim to address the benefits and challenges of developing a natural language interface to facilitate open data access for the general public. This aligns with the ongoing trend towards heightened transparency and openness in public administration, with hundreds of open data files being released daily.

Our findings based on a preliminary survey reveal a clear interest in the solution approach. Our survey respondents (49) demonstrate varying levels of age, occupation, education, and engagement with chatbots, to represent the general public. The challenges disclosed include concerns about data quality, accessibility, and the need for user education. To address this, our web application features a conversational interface with a tutorial structure and a JSON visualizer, to ensure ease of use. Our solution aligns with user preferences including AI capabilities such as DialogFlow, and using a website chat as the channel of communication. The application's architecture enables the extraction of real-time information directly from APIs chosen by users. The user interface emphasizes clarity and simplicity, promoting accessibility for a broad target group.

In concluding this research, it is important to emphasize that the presented web application and its associated methodologies represent a conceptual leap toward addressing the challenges of open data access through chatbots. Following the design science paradigm, the paradigm prioritizes the process of selecting what is achievable and valuable for shaping potential futures, rather than what is currently existing. While our current implementation answers the objective of our research well, it is essential to acknowledge that this is not a final product, but rather a foundational concept.

Future work will focus on enhancing user satisfaction with the chatbot. Results from the conducted survey emphasize the significance of a well-articulated bot with human-like language capabilities. To address this, exploring language models with enhanced capabilities is prioritized, and Gorilla emerges as a promising candidate. Gorilla demonstrates impressive capabilities in creating precise API calls, addressing in-test-time documents, and mitigating issues like hallucination. This makes it a compelling choice for our chatbot's further evolution, considering that Gorilla aligns with our broader vision of enhancing the reliability and applicability of our chatbot's outputs.

While the current version of the bot delivers accurate responses to user inquiries, its functionality is confined to this aspect; thus, lacking the capability to provide any additional information. This can be at the expense of the overall user experience. Hence, we have considered and discussed a range of functions for potential implementations. For instance, extending the intents by incorporating optional entities the user wishes to obtain, thereby enriching the query response with additional information. Another example involves allowing users to designate a constant entity to appear in the output, alongside the remains of the intent response. Additionally, the concept of enabling simultaneous adaptability to multiple APIs has been entertained. However, these functions encounter challenges due to uncertainties and ambiguity regarding the user's specific desires and the information they intend to extract from the APIs. Hence, an in-depth investigation of user preferences and expectations is warranted, preferably through the use of a standardized method to quantify and evaluate the overall user-friendliness and effectiveness like the System Usability Scale (SUS).

Appendix A

Source code

The source code for the web application can be found here: `https://github.com/kathrinehermansen/BottyTheChatbots`

A video demonstration of the web application can be viewed here: `https://drive.google.com/file/d/1LlNGUN1GcEwD1TvQ12mlmeQWIJNUxStJ/view?usp=drive_link`

Answers to Survey in excel format: `https://drive.google.com/drive/folders/1TUoIMHfYPjOAkUjOQ3XP8opxxjyFNVxb?usp=sharing`

Appendix B

Questions from Survey: Chatbots and Open Data: Your Perspective and Insight

Chatbots and Open Data: Your Perspective and Insights

Welcome to the "Chatbot Survey on Open Data," a crucial contribution to our master's thesis titled "Automatic Chatbot Creation from Open Data Sources." We greatly appreciate your participation in this survey. The purpose of this survey is to gather insights on how chatbots can facilitate access to and the benefits of open data sources for the general public.

What is Open Data?

Open data refers to data that is freely accessible to everyone, without restrictions, allowing for use, reuse, and redistribution. These data are in the public domain with minimal restrictions and are published in electronic formats that are easily machine-readable and not restricted by specific software. This enables anyone to access and use the data with commonly available software tools. For example, this may include weather data such as temperature, humidity, and wind speed for your local area. Open data means that anyone, without special permission, can access and use this weather information to understand the climate. Another example is a dataset containing bus schedules in your local area or city. If this data is open, anyone could use it to create useful apps or websites that display when the next bus is coming.

Your Contribution:

Your participation is invaluable in helping us understand how chatbots can assist people in accessing and benefiting from open data sources. We would also like to hear your thoughts on any challenges or needs you perceive in this context.

Please take the time to answer the following questions. Your responses will help us shape future solutions that can enhance access to open data sources through chatbots.

This survey is **anonymous**, except for the information you provide in the questions below. Your personal identity will not be disclosed through the completion of this form.

Thank you so much for taking the time to participate in the survey. If you have any further questions or comments, please do not hesitate to contact us.

julieheldal@icloud.com Bytt konto

Ikke delt

* indikerer at spørsmålet er obligatorisk

About you

Understanding the demographic characteristics of our respondents helps us analyze survey results in the context of different backgrounds and experiences. This information is important for ensuring that our findings are representative of a diverse range of perspectives

What is your age? *

- ◯ Under 18
- ◯ 18 - 24
- ◯ 25 - 34
- ◯ 35 - 44
- ◯ 45 - 54
- ◯ 54 - 64
- ◯ Over 65

What is your gender? *

- ◯ Female
- ◯ Male
- ◯ Prefer not to say
- ◯ Andre:

What is your highest level of education completed? *

- Primary School
- High School
- Bachelor's degree
- Master's degree
- Doctor's degree
- Andre:

What is your current occupation? *

Svaret ditt

How would you rate your technical expertise? *

- Limited / Beginner
- Intermediate
- Advanced
- Expert

What is your current location? (City) *

Svaret ditt

Where did you find this survey? *

- Facebook
- LinkedIn
- Instagram
- Forwarded by friends/family/colleague
- Andre:

Open Data Awareness

How familiar are you with the term Open Data? *

Not at all familiar

1

2

3

4

5

Very familiar

If familiar with the term, how would you define open data in your own words?

Svaret ditt

Open Data usage

Have you ever accessed or utilized open data sources for any purpose? *

Never

1 ○

2 ○

3 ○

4 ○

5 ○

Frequently

If so, briefly describe context where you used open data

Svaret ditt

If you rarely have used open data sources or have never used them, please select the reason(s) why. Please select all that apply.

- [ ] Lack of Awareness - (I wasn't aware of the existence of open data sources.)
- [ ] No Relevance - (I have not found any open data sources relevant to my needs or interests.)
- [ ] Technical Barriers - (I lack the technical skills or knowledge to access or use open data.)
- [ ] Availability Issues - (Open data sources I need are not available for my location or domain.)
- [ ] Privacy Concerns - (I have concerns about data privacy and security related to open data.)
- [ ] Prefer Traditional Sources - (I prefer to use traditional or familiar sources for information.)
- [ ] Andre:

Would you be interested in exploring open data sources in the future if you can ask questions about the data in English/ Norwegian? *

- Yes
- No
- Maybe
- Andre:

What types of Open Data sources are you currently using? *

- [ ] Government Datasets - (Data provided by government agencies, including census data, public records, and official reports.)
- [ ] Websites - (Open data accessible directly from websites and online platforms.)
- [ ] Specialized platforms - (Data from dedicated platforms designed for open data sharing and access.)
- [ ] Data repositories - (Data hosted in repositories or databases designed for open data storage and distribution. Example: Github)
- [ ] Datasets such as Weather, Geospatial, and Transportation Data, etc.
- [ ] I do not use open data
- [ ] Andre:

Challenges with Open Data

Have you encountered any challenges or difficulties when trying to access or make use of open data sources? If yes, please specify

Svaret ditt

Chatbot Usage

How often do you interact with chatbots for any purpose? *

- Never
- Rarely
- Occasionally
- Frequently
- Daily

What chatbots (if any) have you used before? *

- [ ] Virtual Assistants - (services like Siri, Alexa or Google)
- [ ] Customer Support Bots - (used on websites, these chatbots assist users with inquiries, troubleshoot issues, and provide support for products or services)
- [ ] E-commerce Bots - (enhance the online shopping experience by offering product recommendations, assisting with purchases, and answering product-related questions)
- [ ] Informational Bots - (deliver specific information, such as news updates, weather forecasts, or stock market data, to users)
- [ ] AI-Powered Assistants - (services like ChatGPT)
- [ ] Entertainment Bots - (engage users in fun and enjoyable interactions, including telling jokes, playing games, and providing entertainment-related content)
- [ ] Educational Bots - (facilitate learning and training by delivering lessons, providing explanations, and offering quizzes or exercises to help users acquire new knowledge.)
- [ ] Healtcare Bots - (assist users with health-related questions, provide medication reminders, and offer general health advice or symptom checking.)
- [ ] I have never used Chatbots
- [ ] Andre:

Do you think chatbots could make it easier for you to access and utilize open data? Why, or why not? *

Svaret ditt

Are there any specific features or functionalities you would like to see in chatbots to better assist in open data utilization?

Svaret ditt

Can you describe any specific scenarios or use cases where you believe chatbots can be particularly helpful in open data access?

Svaret ditt

Challenges with chatbots

In which communication channels would you find chatbots most helpful/ easy to use? (You can choose multiple) *

- [ ] Website Chat
- [ ] Mobile App
- [ ] Messaging Apps (Facebook Messenger, Whatsapp, Slack, etc.)
- [ ] Andre:

Do you foresee any challenges in using chatbots to access open data, particularly for people with limited technical skills? *

Svaret ditt

Additional Comments

Do you have any additional comments or insights regarding the potential of chatbots to aid in open data utilization?

Svaret ditt

Do you have any additional comments or insights overall?

Svaret ditt

Send

Tøm skjemaet

Appendix C

System Usability Scale

Name:____

Occupation:__________________________________

	Claim	Strongly disagree 1	2	3	4	Strongly agree 5
1	I found the user interface was clear and easy to use	○	○	○	○	○
2	I felt confident in my ability to retrieve relevant information using the chatbot	○	○	○	○	○
3	I needed to learn a lot before I could start using the system	○	○	○	○	○
4	I thought the system was tricky to use	○	○	○	○	○
5	The design of the web application made it easy to understand the available features	○	○	○	○	○
6	I thought there was too much inconsistency in the system	○	○	○	○	○
7	The tutorial structure of the web application enhanced my understanding of the chatbot	○	○	○	○	○
8	I would imagine that most people would learn to use the system quickly	○	○	○	○	○
9	I found the system unnecessarily complex	○	○	○	○	○
10	I would need the support of a technical person to be able to use the system	○	○	○	○	○
11	I would want to use this system to explore open data	○	○	○	○	○
12	Using the chatbot to access open data was straightforward	○	○	○	○	○

13	I found the integration of the chatbot with the web application to be seamless	○	○	○	○	○
14	I would feel confident in recommending the chatbot and web application to others	○	○	○	○	○
15	I see the potential for further development of this system	○	○	○	○	○

What is your former experience with chatbots? Circle the answer that fits you the best.

A. Never used chatbots
B. I have tried using chatbots
C. I use chatbots occasionally
D. I use chatbots often.

What is your former experience with open data? Circle the answer that fits you the best.

A. Never used open data
B. I have tried using open data
C. I use open data occasionally
D. I use open data often.

Any features you feel are missing or could improve in the web application?

____

____

____

____

Other comments?

____

____

____

Bibliography

[1] *1.11. Formal and Natural Languages — How to Think like a Computer Scientist: Interactive Edition.* URL: https://runestone.academy/ns/books/published/thinkcspy/GeneralIntro/FormalandNaturalLanguages.html (visited on Nov. 17, 2023).

[2] *2021 State of Conversational Marketing.* en-US. Oct. 2021. URL: https://www.drift.com/books-reports/conversational-marketing-trends/ (visited on Oct. 9, 2023).

[3] Aayush. *StringBuilder in Java with Examples, Methods, and Constructors.* en. Jan. 2022. URL: https://www.scaler.com/topics/java/stringbuilder-in-java/ (visited on Nov. 2, 2023).

[4] Eleni Adamopoulou and Lefteris Moussiades. "An Overview of Chatbot Technology." en. In: *Artificial Intelligence Applications and Innovations.* Ed. by Ilias Maglogiannis, Lazaros Iliadis, and Elias Pimenidis. IFIP Advances in Information and Communication Technology. Cham: Springer International Publishing, 2020, pp. 373–383. ISBN: 978-3-030-49186-4. DOI: 10.1007/978-3-030-49186-4_31.

[5] Eleni Adamopoulou and Lefteris Moussiades. "Chatbots: History, technology, and applications." en. In: *Machine Learning with Applications* 2 (Dec. 2020), p. 100006. ISSN: 2666-8270. DOI: 10.1016/j.mlwa.2020.100006. URL: https://www.sciencedirect.com/science/article/pii/S2666827020300062 (visited on Sept. 21, 2022).

[6] *API for Dagligvarer og Priser - Kassalapp®.* URL: https://kassal.app/api/docs (visited on Sept. 22, 2023).

[7] Atlassian. *What is version control — Atlassian Git Tutorial.* en. URL: https://www.atlassian.com/git/tutorials/what-is-version-control (visited on Nov. 15, 2023).

[8] *Barriers in working with Open Data — data.europa.eu.* URL: https://data.europa.eu/en/publications/datastories/barriers-working-open-data (visited on Aug. 31, 2023).

[9] Martin Beno et al. "Open Data Hopes and Fears: Determining the Barriers of Open Data." In: *2017 Conference for E-Democracy and Open Government (CeDEM).* May 2017, pp. 69–81. DOI: 10.1109/CeDEM.2017.22.

[10] Martin Beno et al. "Open Data Hopes and Fears: Determining the Barriers of Open Data." In: *2017 Conference for E-Democracy and Open Government (CeDEM).* May 2017, pp. 69–81. DOI: 10.1109/CeDEM.2017.22.

[11] Edward Brent and P. D. Leedy. "Practical Research: Planning and Design." en. In: *Teaching Sociology* 18.2 (Apr. 1990), p. 248. ISSN: 0092055X.

DOI: 10 . 2307 / 1318509. URL: http : / / www . jstor . org / stable / 1318509?origin=crossref (visited on Oct. 11, 2023).

[12] John Brooke. "SUS: A quick and dirty usability scale." In: *Usability Eval. Ind.* 189 (Nov. 1995).

[13] Jordi Cabot. *About us - Xatkit*. en-US. About XatKit. Jan. 2021. URL: https://xatkit.com/about/ (visited on Oct. 9, 2023).

[14] Jordi Cabot. *Five major roadblocks to (usable) open data - Livable Software*. en-US. Section: research. July 2019. URL: https://livablesoftware.com/five-roadblocks-usable-open-data/ (visited on Sept. 21, 2022).

[15] James Cannan and Huosheng Hu. "Human-Machine Interaction (HMI): A Survey." en. In: ().

[16] *Chatbots Using Python and Rasa*. en-US. Section: Machine Learning. Aug. 2020. URL: https://www.geeksforgeeks.org/chatbots-using-python-and-rasa/ (visited on Oct. 11, 2023).

[17] *Context Free Grammars — Brilliant Math & Science Wiki*. en-us. URL: https://brilliant.org/wiki/context-free-grammars/ (visited on Oct. 13, 2023).

[18] Mike Cooray and Dr. Rikke Duus. *The importance of open data*. en. URL: https://www.weforum.org/agenda/2016/02/the-importance-of-open-data/ (visited on Sept. 21, 2022).

[19] John W Creswell. *Research design: Qualitative, quantitative, and mixed methods approaches*. Sage, 2013.

[20] Gwendal Daniel. *A new Java Fluent API for building chatbots is coming to town! - Xatkit*. en-US. Section: Releases. Aug. 2020. URL: https://xatkit.com/fluent-interface-building-chatbots-bots/ (visited on Aug. 17, 2023).

[21] Gwendal Daniel. *Define meaningful fallbacks for your chatbot - Xatkit*. en-US. Section: Features. Jan. 2021. URL: https://xatkit.com/define-meaningful-fallbacks-for-your-chatbot/ (visited on Nov. 7, 2023).

[22] Gwendal Daniel. *Ease your bot development with Xatkit - Xatkit*. en-US. Section: Features. Dec. 2020. URL: https://xatkit.com/ease-your-bot-development-with-xatkit/ (visited on Nov. 7, 2023).

[23] Gwendal Daniel. *Multi-Platform Chatbot Modeling and Deployment with Xatkit*. en-US. Section: article. Mar. 2019. URL: https://modeling-languages.com/multi-platform-chatbot-modeling-deployment-xatkit/ (visited on Oct. 25, 2023).

[24] Gwendal Daniel et al. "Xatkit: A Multimodal Low-Code Chatbot Development Framework." In: *IEEE Access* 8 (2020). Conference Name: IEEE Access, pp. 15332–15346. ISSN: 2169-3536. DOI: 10.1109/ACCESS.2020.2966919.

[25] Jeff Dean. *Introducing Pathways: A next-generation AI architecture*. en-us. Oct. 2021. URL: https://blog.google/technology/ai/introducing-pathways-next-generation-ai-architecture/ (visited on Oct. 16, 2023).

[26] Chinmayee Deshpande. *What Is React? [Easily Explained]*. en-US. Oct. 2023. URL: https://www.simplilearn.com/tutorials/reactjs-tutorial/what-is-reactjs (visited on Oct. 12, 2023).

[27] *Design science (methodology)*. en. Page Version ID: 1151653838. Apr. 2023. URL: https://en.wikipedia.org/w/index.php?title=Design_science_(methodology)&oldid=1151653838 (visited on Oct. 10, 2023).

[28] *Dialogflow Documentation.* en. URL: https://cloud.google.com/dialogflow/docs (visited on Oct. 16, 2023).

[29] Cem Dilmegani. *How to Build a Chatbot: Components & Architecture in 2023.* en-US. URL: https://research.aimultiple.com/chatbot-architecture/ (visited on Nov. 15, 2023).

[30] *Documentation for Visual Studio Code.* en. URL: https://code.visualstudio.com/docs (visited on Oct. 12, 2023).

[31] Claire Drumond. *Scrum - What is it, how it works, & how to start.* en. URL: https://www.atlassian.com/agile/scrum (visited on Oct. 3, 2023).

[32] *DSL Guide.* Aug. 2019. URL: https://martinfowler.com/dsl.html (visited on Oct. 16, 2023).

[33] *ECMWF.* en. Text. Oct. 2023. URL: https://www.ecmwf.int/ (visited on Oct. 15, 2023).

[34] *ELIZA_conversation.png (751×487).* URL: https://upload.wikimedia.org/wikipedia/commons/7/79/ELIZA_conversation.png (visited on Oct. 11, 2023).

[35] Data Europa. *The benefits and value of open data — data.europa.eu.* URL: https://data.europa.eu/en/datastories/benefits-and-value-open-data (visited on Sept. 21, 2022).

[36] Robert Feldt and Ana Magazinius. "Validity Threats in Empirical Software Engineering Research - An Initial Survey." en. In: ().

[37] Matthew Finnegan. *What is Trello? A guide to Atlassian's collaboration and work management tool.* en. Aug. 2021. URL: https://www.computerworld.com/article/3226447/what-is-trello-a-guide-to-atlassians-collaboration-and-work-management-tool.html (visited on Sept. 29, 2023).

[38] Mehmet Firat. *How Chat GPT Can Transform Autodidactic Experiences and Open Education?* Jan. 2023. DOI: 10.31219/osf.io/9ge8m.

[39] Luciano Floridi and Massimo Chiriatti. "GPT-3: Its Nature, Scope, Limits, and Consequences." en. In: *Minds and Machines* 30.4 (Dec. 2020), pp. 681–694. ISSN: 1572-8641. DOI: 10.1007/s11023-020-09548-1. URL: https://doi.org/10.1007/s11023-020-09548-1 (visited on Oct. 16, 2023).

[40] Maryia Fokina. *The Future of Chatbots: 80+ Chatbot Statistics for 2022.* en. Apr. 2023. URL: https://www.tidio.com/blog/chatbot-statistics/ (visited on Oct. 9, 2023).

[41] Keith D. Foote. *A Brief History of Natural Language Processing (NLP).* en-US. May 2019. URL: https://www.dataversity.net/a-brief-history-of-natural-language-processing-nlp/ (visited on Sept. 21, 2022).

[42] Boris Galitsky. "Chatbot Components and Architectures." In: *Developing Enterprise Chatbots: Learning Linguistic Structures.* Cham: Springer International Publishing, 2019, pp. 13–51. ISBN: 978-3-030-04299-8. DOI: 10.1007/978-3-030-04299-8_2. URL: https://doi.org/10.1007/978-3-030-04299-8_2.

[43] Jianfeng Gao et al. *Neural Approaches to Conversational Information Retrieval.* Jan. 2022.

[44] Marcos Gomez, Jordi Cabot, and Robert Clarisó. *Towards the Automatic Generation of Conversational Interfaces to Facilitate the Exploration of*

Tabular Data. arXiv:2305.11326 [cs]. May 2023. DOI: 10.48550/arXiv.2305.11326. URL: http://arxiv.org/abs/2305.11326 (visited on Sept. 22, 2023).

[45] Daniel Gwendal. *xatkit-examples/GreetingsBots at fd65495246351403107ebc09a9a0eaba4f235781 · xatkit-bot-platform/xatkit-examples*. URL: https://github.com/xatkit-bot-platform/xatkit-examples/tree/fd65495246351403107ebc09a9a0eaba4f235781/GreetingsBots (visited on Nov. 6, 2023).

[46] Cameron Hashemi-Pour. *What is OpenAI? Definition and History from TechTarget*. en. Oct. 2023. URL: https://www.techtarget.com/searchenterpriseai/definition/OpenAI (visited on Oct. 16, 2023).

[47] Alan Hevner et al. "Design Science in Information Systems Research." In: *Management Information Systems Quarterly* 28.1 (Dec. 2008). ISSN: ISSN 0276-7783/ISSN 2162-9730. URL: https://aisel.aisnet.org/misq/vol28/iss1/6.

[48] *How to convert a JSON to Java Object using the Jackson library in Java?* en. URL: https://www.tutorialspoint.com/how-to-convert-a-json-to-java-object-using-the-jackson-library-in-java (visited on Nov. 17, 2023).

[49] E. R. P. Informers. *What is Data Accuracy? - How to do Accurate Data Entry?* en-us. Section: Business Intelligence. Feb. 2015. URL: https://www.erp-information.com/data-accuracy.html (visited on Aug. 29, 2023).

[50] *Introducing ChatGPT*. en-US. URL: https://openai.com/blog/chatgpt (visited on Aug. 28, 2023).

[51] *Introducing ChatGPT*. en-US. URL: https://openai.com/blog/chatgpt (visited on Nov. 16, 2023).

[52] *Introduction to Rasa Open Source & Rasa Pro*. en. Oct. 2023. URL: https://rasa.com/docs/rasa/ (visited on Oct. 11, 2023).

[53] Jaydeep. *Introduction to Natural Language Processing*. en-us. Section: Machine Learning. Jan. 2023. URL: https://www.geeksforgeeks.org/introduction-to-natural-language-processing/ (visited on Feb. 22, 2023).

[54] Manfred A. Jeusfeld. "Metamodel." en. In: *Encyclopedia of Database Systems*. Ed. by LING LIU and M. TAMER ÖZSU. Boston, MA: Springer US, 2009, pp. 1727–1730. ISBN: 978-0-387-39940-9. DOI: 10.1007/978-0-387-39940-9_898. URL: https://doi.org/10.1007/978-0-387-39940-9_898 (visited on Nov. 1, 2023).

[55] Varun Joshi. *Why Your Python Version (or other apps) Don't Work on the Apple MacBook M1*. en. July 2021. URL: https://levelup.gitconnected.com/why-your-python-version-or-other-apps-dont-work-on-the-apple-macbook-m1-416af07de57b (visited on Nov. 10, 2023).

[56] Sophia Keyner, Vadim Savenkov, and Svitlana Vakulenko. "Open Data Chatbot." en. In: *The Semantic Web: ESWC 2019 Satellite Events*. Ed. by Pascal Hitzler et al. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2019, pp. 111–115. ISBN: 978-3-030-32327-1. DOI: 10.1007/978-3-030-32327-1_22.

[57] Sophia Keyner, Vadim Savenkov, and Svitlana Vakulenko. "Open Data Chatbot." en. In: *The Semantic Web: ESWC 2019 Satellite Events*. ISSN: 1611-3349. Springer, Cham, 2019, pp. 111–115. ISBN: 978-3-030-32327-1.

DOI: 10.1007/978-3-030-32327-1_22. URL: https://link.springer.com/chapter/10.1007/978-3-030-32327-1_22 (visited on Nov. 16, 2023).

[58] Lorenz Cuno Klopfenstein et al. "The Rise of Bots: A Survey of Conversational Interfaces, Patterns, and Paradigms." en. In: *Proceedings of the 2017 Conference on Designing Interactive Systems*. Edinburgh United Kingdom: ACM, June 2017, pp. 555–565. ISBN: 978-1-4503-4922-2. DOI: 10.1145/3064663.3064672. URL: https://dl.acm.org/doi/10.1145/3064663.3064672 (visited on Sept. 21, 2022).

[59] Ingrid Lunden. *Rasa raises $13M led by Accel for its developer-friendly open-source approach to chatbots*. en-US. Apr. 2019. URL: https://techcrunch.com/2019/04/11/rasa-raises-13m-led-by-accel-for-its-developer-friendly-open-source-approach-to-chatbots/ (visited on Oct. 11, 2023).

[60] Ben Lutkevich. *What is Natural Language Processing? An Introduction to NLP*. en. Jan. 2023. URL: https://www.techtarget.com/searchenterpriseai/definition/natural-language-processing-NLP (visited on Feb. 22, 2023).

[61] *Machine - Definition, Meaning & Synonyms*. en-US. URL: https://www.vocabulary.com/dictionary/machine (visited on Nov. 14, 2023).

[62] Mark Christopher Marino. "I, chatbot: The gender and race performativity of conversational agents - ProQuest." no. In: (Mar. 2006). URL: https://www.proquest.com/openview/3c91805eb882d2a56d58aaa6f809fa50 (visited on Oct. 11, 2023).

[63] Michael McTear, Zoraida Callejas, and David Griol. "Dialog Management." In: *The Conversational Interface: Talking to Smart Devices*. Cham: Springer International Publishing, 2016, pp. 209–233. ISBN: 978-3-319-32967-3. DOI: 10.1007/978-3-319-32967-3_10. URL: https://doi.org/10.1007/978-3-319-32967-3_10.

[64] Robert N. Moll, Michael A. Arbib, and A. J. Kfoury. *An Introduction to Formal Language Theory*. en. Google-Books-ID: tprhBwAAQBAJ. Springer Science & Business Media, Dec. 2012. ISBN: 978-1-4613-9595-9.

[65] Dimitris Mourtzis, John Angelopoulos, and Nikos Panopoulos. "The Future of the Human–Machine Interface (HMI) in Society 5.0." en. In: *Future Internet* 15.5 (May 2023). Number: 5 Publisher: Multidisciplinary Digital Publishing Institute, p. 162. ISSN: 1999-5903. DOI: 10.3390/fi15050162. URL: https://www.mdpi.com/1999-5903/15/5/162 (visited on Nov. 14, 2023).

[66] Prakash M Nadkarni, Lucila Ohno-Machado, and Wendy W Chapman. "Natural language processing: an introduction." In: *Journal of the American Medical Informatics Association* 18.5 (Sept. 2011), pp. 544–551. ISSN: 1067-5027. DOI: 10.1136/amiajnl-2011-000464. URL: https://doi.org/10.1136/amiajnl-2011-000464 (visited on Feb. 22, 2023).

[67] Naveen Naidu. *Software Engineering — Agile Software Development*. en-us. Section: Software Engineering. July 2018. URL: https://www.geeksforgeeks.org/software-engineering-agile-software-development/ (visited on Oct. 3, 2023).

[68] Sharang Narang and Aakanshka Chowdery. *Pathways Language Model (PaLM): Scaling to 540 Billion Parameters for Breakthrough Perfor-*

mance. en. Apr. 2022. URL: https://blog.research.google/2022/04/pathways-language-model-palm-scaling-to.html (visited on Oct. 16, 2023).

[69] Sebastian Neumaier, Vadim Savenkov, and Svitlana Vakulenko. "Talking Open Data." en. In: *The Semantic Web: ESWC 2017 Satellite Events*. Ed. by Eva Blomqvist et al. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2017, pp. 132–136. ISBN: 978-3-319-70407-4. DOI: 10.1007/978-3-319-70407-4_25.

[70] Sebastian Neumaier, Vadim Savenkov, and Svitlana Vakulenko. "Talking Open Data Figure." en. In: *The Semantic Web: ESWC 2017 Satellite Events*. ISSN: 1611-3349. Springer, Cham, 2017, pp. 132–136. ISBN: 978-3-319-70407-4. DOI: 10.1007/978-3-319-70407-4_25. URL: https://link.springer.com/chapter/10.1007/978-3-319-70407-4_25 (visited on Nov. 16, 2023).

[71] OECD. *Frascati Manual 2002*. 2002, p. 256. DOI: https://doi.org/https://doi.org/10.1787/9789264199040-en. URL: https://www.oecd-ilibrary.org/content/publication/9789264199040-en.

[72] *OpenAI*. en-US. URL: https://openai.com/ (visited on Oct. 16, 2023).

[73] Aravindpai Pai. *What is Tokenization — Tokenization In NLP*. en. May 2020. URL: https://www.analyticsvidhya.com/blog/2020/05/what-is-tokenization-nlp/ (visited on Feb. 22, 2023).

[74] Shishir G. Patil et al. *Gorilla: Large Language Model Connected with Massive APIs*. arXiv:2305.15334 [cs]. May 2023. URL: http://arxiv.org/abs/2305.15334 (visited on Oct. 15, 2023).

[75] Priya Pedamkar. *JSON vs CSV — Know The Top 6 Important Comparison*. en-US. Nov. 2018. URL: https://www.educba.com/json-vs-csv/ (visited on Nov. 10, 2023).

[76] Simone Porreca et al. "Accessing Government Open Data Through Chatbots." en. In: *Current Trends in Web Engineering*. Ed. by Irene Garrigós and Manuel Wimmer. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2018, pp. 156–165. ISBN: 978-3-319-74433-9. DOI: 10.1007/978-3-319-74433-9_14.

[77] Amit Prakash. *GPT-4 vs GPT-3.5: A Full Breakdown of The Differences*. en. Mar. 2023. URL: https://www.thoughtspot.com/data-trends/ai/gpt-4-vs-gpt-3-5 (visited on Oct. 16, 2023).

[78] *Prissammenligning av Dagligvarer - Kassalapp®*. no. URL: https://kassal.app (visited on Oct. 16, 2023).

[79] Asif Razzaq and Arham Islam. *Top Large Language Models (LLMs) in 2023 from OpenAI, Google AI, Deepmind, Anthropic, Baidu, Huawei, Meta AI, AI21 Labs, LG AI Research and NVIDIA*. en-US. Feb. 2023. URL: https://www.marktechpost.com/2023/02/22/top-large-language-models-llms-in-2023-from-openai-google-ai-deepmind-anthropic-baidu-huawei-meta-ai-ai21-labs-lg-ai-research-and-nvidia/ (visited on Oct. 16, 2023).

[80] *React – A JavaScript library for building user interfaces*. en. URL: https://legacy.reactjs.org/ (visited on Oct. 12, 2023).

[81] *Regjering — Stortinget — Åpne data*. URL: https://data.stortinget.no/dokumentasjon-og-hjelp/regjering/ (visited on Sept. 23, 2023).

[82] *Regular Expression Tutorial - Learn How to Use Regular Expressions.* URL: https://web.archive.org/web/20161101212501/http://www.regular-expressions.info/tutorial.html (visited on Oct. 13, 2023).
[83] György E. Révész. *Introduction to Formal Languages.* en. Google-Books-ID: CSOSsLVAYvIC. Courier Corporation, Jan. 1991. ISBN: 978-0-486-66697-6.
[84] *Rule Based Approach in NLP.* en-US. Section: Machine Learning. Apr. 2023. URL: https://www.geeksforgeeks.org/rule-based-approach-in-nlp/ (visited on Oct. 27, 2023).
[85] Kevin Scott. *Microsoft teams up with OpenAI to exclusively license GPT-3 language model.* en-US. Sept. 2020. URL: https://blogs.microsoft.com/blog/2020/09/22/microsoft-teams-up-with-openai-to-exclusively-license-gpt-3-language-model/ (visited on Oct. 16, 2023).
[86] *Scrum Guide — Scrum Guides.* en. URL: https://scrumguides.org/scrum-guide.html (visited on Oct. 3, 2023).
[87] BEN SMITH. *Beginning JSON.* en. Google-Books-ID: ZYYnCgAAQBAJ. Apress, Feb. 2015. ISBN: 978-1-4842-0202-9.
[88] Sunny Srinidhi. *Stemming of words in Natural Language Processing, what is it?* en. May 2020. URL: https://towardsdatascience.com/stemming-of-words-in-natural-language-processing-what-is-it-41a33e8996e2 (visited on Feb. 22, 2023).
[89] *Stack Overflow Developer Survey 2018.* URL: https://insights.stackoverflow.com/survey/2018/?utm_source=so-owned&utm_medium=social&utm_campaign=dev-survey-2018&utm_content=social-share (visited on Nov. 15, 2023).
[90] Lee Stanton. *How to Run Code in VS Code.* en-US. URL: https://www.alphr.com/vs-code-run-code/ (visited on Oct. 12, 2023).
[91] Allan Steckler et al. "Toward Integrating Qualitative and Quantitative Methods: An Introduction." en. In: *Health Education Quarterly* 19.1 (Apr. 1992). Publisher: SAGE Publications Inc, pp. 1–8. ISSN: 0195-8402. DOI: 10.1177/109019819201900101. URL: https://doi.org/10.1177/109019819201900101 (visited on Oct. 11, 2023).
[92] *The Software Challenges of Building Smart Chatbots.* en-US. URL: https://ieeexplore.ieee.org/document/9402515 (visited on Oct. 9, 2023).
[93] *TheCocktailDB.com.* URL: https://www.thecocktaildb.com/ (visited on Sept. 22, 2023).
[94] David R. Traum and Staffan Larsson. "The Information State Approach to Dialogue Management." In: *Current and New Directions in Discourse and Dialogue.* Ed. by Jan van Kuppevelt and Ronnie W. Smith. Dordrecht: Springer Netherlands, 2003, pp. 325–353. ISBN: 978-94-010-0019-2. DOI: 10.1007/978-94-010-0019-2_15. URL: https://doi.org/10.1007/978-94-010-0019-2_15.
[95] Cambridge University. *Stemming and lemmatization.* 2008. URL: https://nlp.stanford.edu/IR-book/html/htmledition/stemming-and-lemmatization-1.html (visited on Feb. 22, 2023).
[96] Vilmate. *How to Create a Chatbot for Your Business in 7 Steps — Vilmate.* en-US. Jan. 2021. URL: https://vilmate.com/blog/how-to-develop-a-chatbot/ (visited on Oct. 9, 2023).

[97] Prajwal Wadhwa. *50+ Vital Chatbot Statistics for 2023 Every Marketer Should Know*. en-GB. Aug. 2023. URL: https://outgrow.co/blog/vital-chatbot-statistics/ (visited on Oct. 9, 2023).

[98] Nic Weber and Brandon Locke. *Ethics of Open Data*. May 2022. DOI: 10.48550/arXiv.2205.10402.

[99] . *What Do Your Customers Actually Think About Chatbots? [Research Study]*. en. Dec. 2022. URL: https://www.userlike.com/en/blog/consumer-chatbot-perceptions (visited on Oct. 9, 2023).

[100] *What is a Large Language Model? — A Comprehensive LLMs Guide*. en-us. URL: https://www.elastic.co/what-is/large-language-models (visited on Oct. 3, 2023).

[101] *What Is an IDE?* en. Apr. 2022. URL: https://www.codecademy.com/article/what-is-an-ide (visited on Oct. 8, 2023).

[102] *What is BERT (Language Model) and How Does It Work?* en. URL: https://www.techtarget.com/searchenterpriseai/definition/BERT-language-model (visited on Oct. 16, 2023).

[103] *What Is GitHub? A Beginner's Introduction to GitHub*. en-US. Oct. 2023. URL: https://kinsta.com/knowledgebase/what-is-github/ (visited on Nov. 15, 2023).

[104] *What is Java? — IBM*. en-us. URL: https://www.ibm.com/topics/java (visited on Oct. 8, 2023).

[105] Carrie Williams. "Research Methods." en. In: *Journal of Business & Economics Research (JBER)* 5.3 (Mar. 2007). Number: 3. ISSN: 2157-8893. DOI: 10.19030/jber.v5i3.2532. URL: https://clutejournals.com (visited on Oct. 11, 2023).

[106] *Xatkit*. en-US. URL: https://www.eu-startups.com/directory/xatkit/ (visited on Sept. 29, 2023).

[107] xatkit. *How to build a custom Named-Entity Recognition system for your chatbot engine - Xatkit*. en-US. Section: Features. May 2022. URL: https://xatkit.com/custom-named-entity-recognition-for-chatbot/ (visited on Nov. 17, 2023).

[108] xatkit. *How to build your own chatbot intent classifier - Xatkit*. en-US. Section: Releases. Mar. 2022. URL: https://xatkit.com/how-to-build-chatbot-intent-classifier/ (visited on Oct. 16, 2023).

[109] xatkit. *The full tech stack to run a chatbot - behind the scenes*. en-US. Section: eCommerce. July 2022. URL: https://xatkit.com/tech-stack-developing-chatbot/ (visited on Oct. 16, 2023).

[110] Antonello Zanini. *A Complete Guide to Lombok*. en. June 2021. URL: https://auth0.com/blog/a-complete-guide-to-lombok/ (visited on Oct. 16, 2023).

[111] Andrew Zola. *What is a strongly typed programming language?* en. URL: https://www.techtarget.com/whatis/definition/strongly-typed (visited on Sept. 24, 2023).