Nico Schmid, Bsc

Developing a Chatbot for the Catrobat Wiki: Enhancing the Learning Experience for Children

Master's Thesis
to achieve the university degree of
Master of Science
Master's degree programme: Computer Science

submitted to

Graz University of Technology

Supervisor
Slany, Wolfgang, Univ.-Prof. Dipl.-Ing. Dr.techn.

Institute of Software Technology

Villach, 08 2024

Affidavit

I declare that I have authored this thesis independently, that I have not used other than the declared sources/resources, and that I have explicitly indicated all material which has been quoted either literally or by content from the sources used. The text document uploaded to TUGRAZonline is identical to the present master's thesis.

Date	Signature

Acknowledgment

First of all, I would like to express my sincere gratitude to Prof. Slany and Stefan Kutschera for their supervision of my Master's thesis. Their expertise, guidance and support have been important throughout this journey. I am also deeply thankful to Daniel Metzner for his assistance in helping me choose my thesis topic and for his mentorship in my career. His support and guidance have been crucial to my professional development. In addition I want to thank Johanna for her support during the writing process and her valuable tips on how to formulate better in a scientific context. A heartfelt thank you goes to Martin, my loyal study partner and outstanding friend. We have faced the challenges of university life together and his support and friendship have been instrumental. He has continually challenged me to think critically and grow intellectually. Above all, I am deeply appreciative of my parents, Sonja and Bertl. Their constant support and belief in me, regardless of the paths I chose, have been the foundation of my academic journey. They have been by my side every step of the way and I have learned invaluable life lessons from them that surpass any formal education. Their influence has been crucial in shaping the person I am today. I hope you know how grateful I am for you. Lastly, I want to express my deepest thanks to my life partner and best friend, Anna. Her enduring support, patience with my moods and incredible kindness have been my anchor every single day. She has been there for me on my hardest days, helping me overcome my worries and motivating me to be a better person. Her love and dedication have been a constant source of strength and inspiration and I would not have been able to finish this thesis without her. Thank you Anna for being my rock and always standing by my side.

To all of you, my heartfelt thanks. Your support and encouragement have been pivotal in my journey and I am deeply grateful to each and every one of you.

Abstract

This thesis deals with the development and the integration of a chatbot for Catrobat, that aims to improve the accessibility and the engagement of the Catrobat Wiki. Catrobat is an educational platform that intends to teach its users programming through the development of mobile apps. The thesis has several main objectives, including the development of a simple chatbot that is capable to understand and respond to user queries, the evaluation of different types of chatbots and also different NLP models for the implementation of the chatbot and finally the integration of the chatbot into the Catrobat ecosystem.

It starts with a detailed overview of the Catrobat project and the respective platforms necessary for the chatbot creation. Afterwards an overview of chatbot technologies is given, discussing their advantages as well as their disadvantages and their suitability for the educational context. Different types of chatbots, including rule-based, retrieval-based and generative models, are explored in order to determine the most effective and fitting approach for the Catrobat Wiki. The thesis is discussing the strengths of chatbots, which are driven by AI, especially those that make use of advanced NLP techniques in order to create dynamic and interactive user experiences.

In order to obtain the required data from the Catrobat Wiki for the chatbot, a custom data scraping tool was developed using Python. This scraper navigates through the structure of the Wiki, it retrieves the relevant content and organizes it into a structured data set using JSON. The challenges which occured with other data export methods, such as the limited and unfitting file formats and also the need for continuous changes and updates in the dataset, were addressed by the custom scraper to be more flexible. This enabled an efficient data collection and ensured that the chatbot always has access to up-to-date information, whenever something is added or changed in the Wiki.

The development process of the chatbot involved the creation of prototypes using two main approaches, a retrieval-based model using TF-IDF to find matching results and a generative model using the OpenAI API. The TF-IDF prototype focused on retrieving relevant texts based on query similarity, while the generative model is generating text responses that could stem from a human.

The technical architecture of the chatbot was designed to be able to be integrated seamlessly into the Catrobat Share Platform. The chatbot consists of a frontend interface, a backend server and a server handling the logic for the chatbot as well as making calls to the external OpenAI API. The user interface was designed to be simple and accessible, by sticking to a similar interface seen with Open AI's Chat

GPT or other related services, to meet the needs of the target group which mostly consists of young learners. The frontend was developed using HTML and the template engine Twig, to handle user interactions, while the backend is done with PHP and the logic for the chatbot is running on a Python server, which processes queries and generates the responses. By integrating the chatbot on the Catrobat Share Platform and therefore into the Catrobat ecosystem, users can receive support in a more intuitive way than with a traditional Wiki or FAQ.

Several limitations have been encountered, which include the scope of data scraping as well as the reliance of the chatbot on an external API. It is recommended to further work on this in the future to expand on the data covered by the scraper, explore different and more powerful models for the chatbot, conduct user testing and find ways to incorporate some form of gamification elements to further improve the user engagement.

Finally, this thesis presents an investigation of the development and implementation of a chatbot for the Catrobat Wiki. The results highlight the transformative potential of chatbot technology in the educational context and provide important insights for future innovations aimed at empowering young learners through interactive and personalized learning experiences.

Contents

List of Figures

1 Introduction

1.1 Background

In today's world, which is characterized by rapid advancements in the digital domain, technology is playing an increasingly significant role in the way we learn and how we interact and access information (Purdue-University, 2024). Educational institutions and initiatives around the world are using the power of new technologies to enable a more democratic access to learning opportunities and to promote creativity and critical thinking for learners of all ages (National-Public-School-Vidyaranyapura, 2024).

The use of digital tools and platforms has made it possible to reach new levels of collaboration and an exchange of knowledge that is able to transcend geographical boundaries. One such initiative is the Catrobat project, especially aimed at empowering children and young adults to become creators of technology on their own, through the development of mobile apps. ("Catrobat," 2024a)

The Catrobat project, which was founded about 13 years ago by the Graz University of Technology, has gathered international recognition for its innovative approach to teaching programming. By providing children as well as young adults with an intuitive visual programming language and a supportive online community, Catrobat enables young learners to design, create and share their own mobile applications in a playful manner. ("Catrobat," 2024a) One part of the Catrobat ecosystem is its Wiki, a repository of knowledge and resources designed to support and help children or learners of different ages during their app development journey. The Wiki is a source of knowledge that provides a vast amount of information, including a detailed breakdown of the available programming bricks and their respective functions. It also offers a wealth of tutorials that provide step-by-step instructions for developing mobile applications with Catrobat. If one is stuck with a problem or wants to find out what functions are available in Catrobat, there is no way around its Wiki. ("Catrobat-Wiki," 2024a)

1.2 Problem Statement

Even though the Catrobat project is very succesful and has a variety of remarkable achievements, there are still challenges concerning the accessibility and the engagement of the projects educational content, in particular the Catrobat Wiki. While the Wiki offers a large amount of information and tutorials, its current format may not fully resonate with its target audience of young learners. Traditional FAQs or Wikis, while they are informative and proven to be a good solution, often lack the interactive and engaging elements which are necessary to get the attention of users and sustain their interest in the learning process, especially for younger people. (Han & Lee, 2022)

When considering these challenges, it is necessary to explore new and innovative approaches to enhance the learning experience within the Catrobat ecosystem. By making use of new technologies such as chatbots, it may be possible to develop an interactive and personalized learning tool that not only caters to the specific needs and preferences of young learners but could also potentially replace the Wiki, which is hosted on an external service, in the future. The chatbot can be a great way to simplify the learning process and offer quick help to students stumbling upon problems during their coding journey with Pocket Code or other Apps related to Catrobat.

1.3 Objectives

The main objective of this thesis is to develop and evaluate a chatbot solution which is enhancing the Catrobat Wiki, with the overarching aim of improving accessibility and engagement for its young users. Even though a wealth of information is available on the Catrobat Wiki and users might be able to find a solution to their specific problem, navigating this repository can be quite challenging or annoying for users, especially if they are stuck on a concrete problem where they might not know how to start looking for a solution. A chatbot that is integrated into the Catrobat ecosystem may be able to simplify the access to the information in the Wiki and therefore to make the learning process more engaging and efficient. This thesis, therefore aims to achieve the following objectives:

1. **Develop a Chatbot that is able to Understand and Respond to User Queries and how to integrate it into the Catrobat ecosystem:**
   The primary objective of this thesis is to implement a chatbot that can understand natural language queries posed by users and afterwards reply with the relevant information and offer assistance within the context of the Catrobat Wiki. Achieving this, involves finding the right type of chatbot to use, gathering the necessary data from the Wiki and further instructing a generative AI chatbot to only answer Catrobat related questions. The goal is that the chatbot is able to recognize and interpret a wide range of user questions related to Catrobat like programming bricks, their functions and usage examples. The chatbot will use natural language processing techniques to deliver accurate

and contextually accurate responses. This will help users to quickly find answers to their questions without having to navigate through the complex menus or search results of the Wiki and therefore enhance their overall experience.

2. **Compare and Evaluate Different Chatbot Types and compare different NLP models:**
   To ensure the effectiveness of the chatbot, this thesis will explore and compare various types of chatbots. It will discuss their advantages and disadvantages and their suitability to be used in the context of the catrobat project. Also, for the retrieval-based chatbot, different NLP models will be evaluated and explained to find the most suitable solution for the Catrobat chatbot.

3. **How Can Using a Chatbot Affect the Accessibility and the Engagement of a Wiki for Its Target Audience?**
   Another objective of this thesis is to evaluate how chatbots can influence the accessibility and engagement of Wikis for their respective target audience. This assessment will be based on a review of existing literature, case studies and analysis of data from different educational initiatives that have integrated chatbot technology. This involves examining how chatbots improve the ability of users to efficiently find their desired information. Therefore it will be analysed how chatbots streamline navigation and reduce the time required to locate specific content within the Wiki. Furthermore, the overall user satisfaction with Wikis that incorporate chatbots will be evaluated. Lastly, it will be explored how chatbots make Wiki content more accessible to a wide variety of users, including individuals with unique or different learning needs and preferences. This also encompasses the examination of how chatbots can support personalized learning paths and provide immediate assistance.

The thesis seeks to provide an understanding of how chatbots can improve the accessibility and engagement of traditional Wikis by examining these objectives. Moreover, this thesis will highlight best practices, identify potential challenges and also suggest areas where further improvement can be achieved. Therefore it is also contributing useful insights into the usage of chatbot technology in educational applications.

1.4 Significance of the Research

This thesis aims to find useful insights into the possibilities of chatbot technology to enhance educational platforms like Catrobat and make the learning experience even more interactive than it already is, by addressing the aforementioned objectives. The development and evaluation of a chatbot for the Catrobat Wiki represents an effort to use AI-driven solutions to enhance digital literacy as well as programming skills among young learners. This thesis has several significant implications:

One of the main contributions of this thesis is to try to enhance engagement in the educational domain. Conventional educational resources are often not able to capture and maintain the attention of its users, which is especially relevant in the context of self-guided and interactive learning environments. By integrating a chatbot into the Catrobat Wiki, the thesis aims to create a more interactive and engaging learning platform. The ability of a chatbot to offer instant and personalized support has the potential to make the learning process more engaging and enjoyable. This can further encourage children to spend more time exploring and learning within the platform. (Bocian, 2024)

Furthermore, accessibility is a critical issue concerning educational technology, particularly for platforms aimed at young users who may have varying levels of digital literacy. This thesis seeks to show how a chatbot can bridge these accessibility gaps by offering intuitive and user-friendly support. The chatbot can help users navigate the Catrobat environment more efficiently and therefore also reducing the overhead which is associated with searching for information and enable learners of all backgrounds to access educational content more easily.

The ability of the chatbot to provide answers in real-time and guidance supports the concept of self-directed learning, where students themselves are responsible for their educational journey (Ali et al., 2023). This thesis also considers the importance of creating tools that empower learners to seek out information on their own, solve problems and directly apply their newly gathered knowledge. The chatbot can significantly contribute to the development of critical thinking and problem-solving skills in children by making it easier for them to find the answers they need and by encouraging them to explore the topic further. These skills are essential to become successful in the digital age.

2 The Catrobat Project and its Share Platform

2.1 Overview of the Catrobat Project

The Catrobat project is a non-profit Open Source initiative that aims to bring children as well as young adults to become creators of technology through the development of mobile apps. It has more than 1100 contributors so far, a presence in more than 180 countries and it also supports millions of users in over 60 different languages. Catrobat, founded in 2010 by the Graz University of Technology in Austria, has gained international recognition for its innovative approach to programming education. The project aims to bridge the gap between consumers and creators of technology, by providing young learners with the tools and also the knowledge they need to create and share their own mobile applications. ("Catrobat," 2024b)

The mission of the Catrobat project is to democratize the access to programming education, by making it accessible to users of all ages and especially to children worldwide. Through fostering creativity, critical thinking and problem-solving skills, Catrobat aims to provide the upcoming generation with the skills that are required in a technology-driven world. The vision of the project is to create a global community of young coders who are not just users of technology but active creators and innovators. ("Catrobat," 2024b)

Important Components of Catrobat

- **Pocket Code:** At the center of the Catrobat project is Pocket Code, a mobile app which is available for all common mobile operating systems, that allows users to create games, animations, interactive stories and various other types of apps directly on their smartphones or tablets. Pocket Code uses a visual programming language which is quite similar to the also popular „Scratch" programming language ("Scratch," 2024) which makes it accessible to beginners while it is still powerful enough for more advanced projects. It uses a drag-and-drop interface which simplifies the coding process, allowing users to focus on creativity, experimentation and the final result rather than getting overwhelmed by the complex syntax of traditional programming languages.

- **Wiki:** The Catrobat Wiki provides a vast amount of educational resources to support the learners of all different ages. These resources include tutorials, example projects and information about the specifics of the programming language. It is designed to help users get started with Pocket Code and

improve their programming skills. The resources are available in multiple languages, therefore showing Catrobat's commitment to global accessibility. ("Catrobat-Wiki," 2024b)

- **Catrobat Share:** Catrobat has a vibrant online community where users can share their projects, collaborate on new ideas and provide feedback to each other. This community aspect is a key component of the project's success, as it encourages collaborative learning and provides a platform for young coders to showcase their work to others. ("Catrobat-Share," 2024)

The global outreach efforts of Catrobat have been important in spreading its mission to diverse regions around the world. The Catrobat project collaborates with schools, educational institutions and non-profit organizations to integrate Pocket Code into their curricula. These collaborations have further enabled Catrobat to reach communities which are underserved regarding programming education and provide opportunities for children who might otherwise not have access to resources to learn programming.

2.2 The Share Platform (Catroweb)

The Share Platform of Catrobat is the dedicated place for sharing projects which have been created with Pocket Code. It further serves as an online hub for the Catrobat community, which enables users to upload their creations and explore projects from other users. Further not only whole projects can be shared but the Share Platform also has a media library included. There users can find and use previously created sounds, backgrounds and many more in their program. The Share Platform is an essential component of the Catrobat ecosystem since it supports the project's mission to give more people access to technology and also to create a global community of young coders. ("Catrobat-Share," 2024)

Relevant Technical Infrastructure

The Share Platform is built by using the programming language PHP and the Symfony framework, which provides a robust and scalable foundation for the platform and also offers a beginner-friendly entry for new students developing for the Share Platform. The use of Symfony ensures that the platform is performant and provides a great user experience. The Catrobat Share Platform is mostly developed and maintained by a team of students, studying at Graz University of Technology.

The platform uses a relational database management system (MYSQL) to store and manage the user data, project files and other necessary information. Further, the platform uses twig as a template engine, which makes it easy to create user interfaces for the platform.

Some of the most important features of the Catrobat Share Platform are:

- **Project Upload and Sharing:** Users can easily upload their Pocket Code projects to the Share Platform, where they can further be shared with the whole Catrobat community. Moreover users can also download projects created by other users. The platform supports various project types including games, animations and interactive stories and enables users to describe and categorize their projects such that they can be easier discovered.

- **Search and Discovery:** Catroweb features a search engine that allows users to find projects based on keywords, categories and other criteria. The main page features categories like „most downloaded“, „trending projects“, etc., which makes it easy for users to find successful or very popular projects. Users can get inspired by looking at projects created by other people.

- **User Profiles and Social Interaction:** Each user on the Share Platform has a profile where they can showcase their projects, achievements and contributions to the Catrobat community. The platform supports social interactions, such as commenting on projects, following other users and participating in discussions. These features foster a sense of community and encourage collaborative learning.

- **Integration of other Resources:** The Share platform is designed to be an educational resource as well as a sharing platform. It includes integration with educational content, namely the Wiki where users can find things such as tutorials that guide them through the process of creating and improving their projects. (“Catrobat-Share,” 2024)

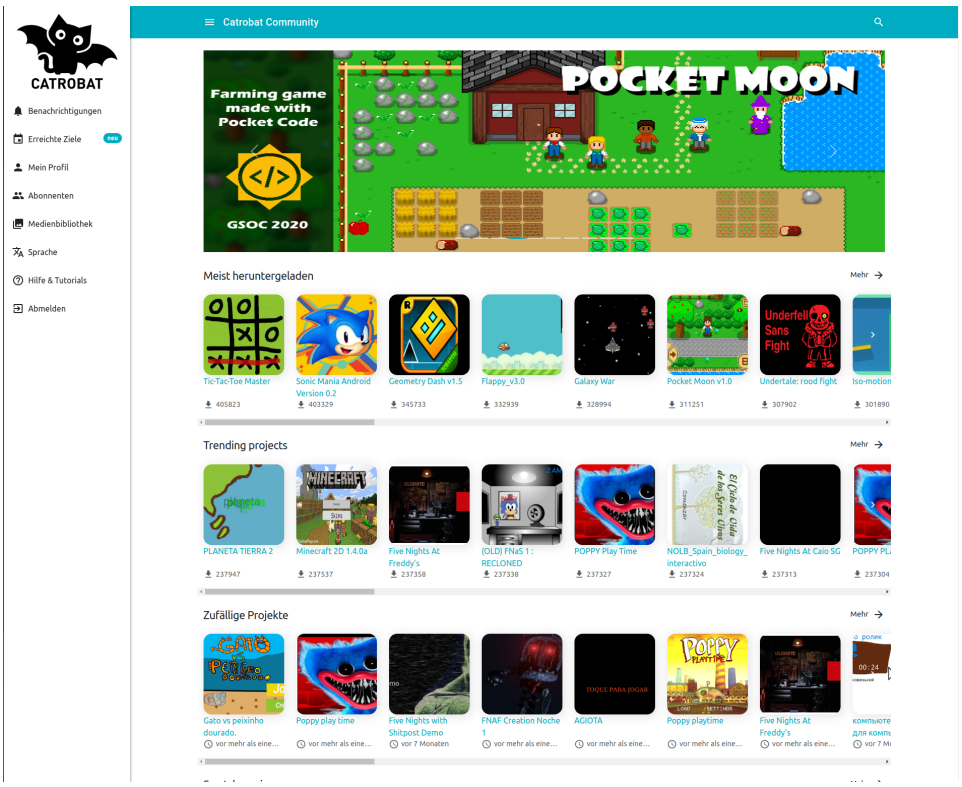

Figure 2.1: Screenshot of the Catrobat Share Platform Homepage

To integrate the Catrobat chatbot prototype into the Catrobat Share Platform was a strategic decision which aims to enhance the user experience and to optimize the information gathering process for the users. As already stated, the Catrobat Share Platform serves as a central hub where users can share projects, seek assistance and engage with the community. Meanwhile, the Catrobat Wiki, which is hosted on an external service (XWiki), serves as a repository of documentation and information about the platform. Considering this, both, the Share Platform as well as the Wiki, would be appropriate places to integrate the new chatbot. However, since the Wiki is hosted on an external service I do not have access to the codebase, implementing a custom chatbot directly in the Wiki is not possible. The decision was made to implement the chatbot on the Share Platform. By embedding the chatbot within the Share Platform rather than the Wiki, several significant benefits can be realized. Further the Wiki is also linked under „Help & Tutorials“ on the Share Platform, so the Share Platform often serves as an entry point to the Wiki.

Integrating the chatbot directly into the Catrobat Share Platform provides a unified access point for all users. Instead of having to navigate away from the platform to the external Wiki for information, users can access the knowledge base by posing queries to the chatbot without leaving the site. This enables to streamline the user journey, enhancing the convenience.

The Share Platform is a dynamic environment where users interact frequently, whether they are uploading projects, exploring the work of others, or are engaging in discussions. By placing the chatbot in this high-traffic area, users are more likely to use it, which will lead to higher levels of engagement. Additionaly, the chatbot can provide instant answers to queries, offer guidance and even suggest resources, thereby enriching the overall user experience.

The chatbot's presence on the Share Platform further allows for real-time assistance. Users currently working on projects and encountering issues can immediately ask the chatbot for help anc continue with their work. This immediate feedback loop can help to increase the user satisfaction and to foster a supportive community environment.

Given that the Share Platform is where users create and interact with projects, the chatbot can offer context-specific assistance. For instance, if a user encounters a specific problem while working on a project, the chatbot can provide targeted help based on the project's context, drawing from the extensive knowledge base in the Wiki.

3 The Catrobat Wiki

The Catrobat Wiki is a supportive online resource for the Catrobat project. The Wiki serves as the central repository of information regarding the Catrobat ecosystem. It is providing documentation, tutorials, guides and other educational materials. These resources are crucial for users participating in the Catrobat environment, especially for those working with the app Pocket code, which is the core component of the Catrobat project.

The main purpose of the Catrobat Wiki is to offer support to users and provide them with all the information they need when coding with Pocket Code. It is organised into four main categories to ensure that information is easily accessible:

1. **Educational Resources:** This section contains various information on different projects and tutorials. One project worth mentioning is the Code'n' Stitch Project, which provides embroidery tutorials that allows users to create designs for embroidery machines. You will also find the brick documentation for the Catrobat visual programming language, templates and other tutorials to make learning easier in this section.

2. **Documentation & Help:** The Documentation & Help section is the main focus for the chatbot development in this work. This section contains a wealth of information, including detailed documentation of the building bricks, the formula editor, tutorials on the functions, beginner's courses and information on game design. This is relevant for all users, whether they want to understand the basics or learn something about the more advanced functions of Pocket Code.

3. **Formula Editor:** This category provides detailed explanations of how to use the formula editor, It covers topics from basic calculations and functions to logic and also data manipulation. It serves as an important tool for users who need to understand how to use the formula editor effectively in their projects.

4. **About Catrobat:** The About Catrobat section provides general information about the Catrobat project, its mission, its goals and the motivation behind it. This helps users understand the broader context and purpose of the Catrobat initiative.

The most important section for the chatbot prototype is the 'Brick Documentation', which can be found in the Documentation & Help category. This section provides detailed explanations for each of the bricks available in Pocket Code. These bricks are the main components used to create applications with Pocket Code.

The focus on the Brick Documentation was essential for two reasons:

- **Core Components:** The bricks documented here are the primary building blocks of any app created with Pocket Code. Understanding these building blocks is crucial for users to effectively develop their projects.

- **Scope:** Given the huge number of resources published on the Wiki, it was necessary to keep the focus on a manageable scope. Scraping every resource and converting it into a format suitable for the chatbot would have been beyond the scope of this work. Prioritizing the brick documentation therefore enabled a more targeted and practical implementation.

The Brick Documentation is organized into several different categories to make it easier for users to find the information they need. These categories include:

- **Event Bricks:** Bricks that process events, e.g. when a program starts or when a certain condition is met.

- **Control Bricks:** Bricks that control the flow of the program, such as loops and conditions. An example is the if brick, which implements conditional logic (e.g., if $1 < 2$ is true, then execute a certain action).

- **Motion Bricks:** Bricks that control the movement of objects within the app.

- **Sound Bricks:** Bricks that handle sound operations, such as playing a sound or adjusting volume.

- **Looks Bricks:** Bricks that manage the appearance of objects, such as changing costumes or colors.

Each category contains specific bricks with detailed explanations of their functionality and use. This approach of structuring the information helps users to quickly find the bricks they need and understand how to use them effectively in their projects.

The Wiki will serve as the knowledge base for the new chatbot, since it contains all the relevant information for users working in the Catrobat ecosystem. By focusing on the brick documentation, the chatbot can provide users with instant and relevant information about the various programming bricks that are available in Pocket Code. This interaction ensures that users can quickly clarify their questions and continue to develop their projects without major interruptions.

The Catrobat Wiki is an important resource for users of the Catrobat project, especially those using Pocket Code. Its well-organised structure and comprehensive content ensure that users have access to all the information they need for successful

programming. By integrating a chatbot centred on brick documentation, the Catrobat Wiki will further enhance the user experience, making it easier for users to find information and develop their skills effectively.

4 Chatbots in the Educational Domain

Over the last years, the rise of digital technology has revolutionized many different sectors, including education (Purdue-University, 2024). One of those revolutionizing technologies are chatbots, which have significantly changed how content, also in the educational domain, can be delivered and accessed by its users. Since Catrobat is also an educational initiative, it is necessary to take a look at how chatbots are used in this context. (Kuhail et al., 2023)

4.1 The Evolution of Educational Chatbots

Chatbots have come a long way since their beginnings as simple rule-based systems. These so-called conversational agents have over time evolved into sophisticated AI-driven tools which are capable of understanding and responding to natural language requests posed by users. This ability is revolutionising the way students can learn and interact with educational content. (Labadze et al., 2023)

The first chatbots and therefore also educational chatbots were rule-based chatbots. These type of chatbot uses decision trees and also predefined scripts to guide the interactions between the user and the chatbot. (Caballé, 2023)

Oftentimes these chatbots make use of regular expressions to match the users input to a predefined response (codecademy, 2024). These chatbots can handle specific and predictable tasks such as answering frequently asked questions (FAQs) or providing basic tutoring in diverse subjects like mathematics and languages or just mimic a conversation. For example, already in 1966, Joseph Weizenbaum developed one of the first chatbots, called ELIZA, which simulated a Rogerian psychotherapist. It was already able to give human-like responses to questions from users. This early chatbot used a structured library where if a keyword was found using pattern matching a predefined answer related to the topic of the keyword was given as a response. (Mnasri, 2019) (Luber, 2024)

The development of AI and machine learning technologies has led to a significant leap in the development of chatbots in general and also for educational chatbots. Techniques such as natural language processing, deep learning and neural networks enabled chatbots to understand and respond to a broader range of queries with more accuracy and more contextual relevance. (Labadze et al., 2023) This provides a more engaging and personalized learning experience. One example is AutoTutor, an intelligent tutoring system, which was developed in the late 1990s which engages students in natural language conversations to help those students learn about computer literacy, physics and critical thinking. AutoTutor uses Natural

Language Processing to analyse student responses, to provide feedback and to guide the conversation towards the learning objectives. (Graesser, 2016)

Modern educational chatbots often use hybrid approaches, which are a combination of rule-based and AI-driven methods. Combining these two methods allows for the precision and reliability of rule-based systems while also incorporating the adaptability and learning capabilities of AI based technologies (Ellen & Michaelidou, 2024). For instance, a chatbot may use rule-based logic to handle common queries and to answer things like frequently asked questions and AI to be able to handle more complex, open-ended discussions. One example of a hybrid chatbot is the Jill Watson system developed by Georgia Tech, which assists students in an online course. Jill Watson uses AI to understand student questions and provide relevant information, while also relying on a knowledge base of pre-written responses for common queries. (Taneja et al., 2018)

As AI and ML technologies continue to advance and develop further, educational chatbots are becoming increasingly more sophisticated and versatile. Chatbots are now being used for a wide range of applications, such as tutoring, for assessment and even for administrative tasks (Cherniak, 2024). They are also being integrated into various educational platforms, including learning management systems, virtual classrooms and mobile apps. Looking into the future, researchers are exploring ways to make chatbots more personalized, adaptive and engaging. This includes developing chatbots that can adapt their language and content to the learning styles and preferences of individual students. Additionally, there is a growing interest to make use of chatbots to support collaborative learning, where students can interact with each other and with the chatbot to explore topics in depth. (Perihan, 2024) Educational chatbots have evolved significantly from their humble beginnings as rule-based systems. With the integration of AI and ML technologies, these conversational agents have become powerful tools for enhancing teaching as well as learning.

4.2 Applications of Chatbots in Education

Chatbots can be found in multiple, diverse applications across different educational settings, starting from primary education to higher education and further also in professional trainings.

Chatbots can provide one-on-one tutoring sessions which are tailored to the pace of individual learners and their specific learning style. They can offer explanations, answer specific questions, provide feedback and can also generate practical exercises. For example, chatbots can effectively support personalized learning by adapting their content and communication style to each student's needs. (Winkler & Söllner, 2018)

Chatbots can engage learners in real-time conversations, helping them to practice speaking, listening, reading and writing skills. They can provide instant corrections and feedback, making the learning process more interactive and effective. There is a lot of potential for chatbots in language learning, students can for example just chat with a chatbot in a foreign language and experiment with it and also get corrected if they do something wrong. There is also potential for teachers to monitor their students progress. This can be achieved for example by using a service like Jabberwacky, where a teacher can keep track of the conversations of students with the chatbot. (Fryer, 2006)

Chatbots can assist students with enrolment processes, course selection, scheduling and further accessing academic resources. By automating these routine tasks, chatbots have the ability to free up educators as well as administrative staff to focus on more important aspects of education. This not only allows educators to have more time for the more critical aspects of education but can also save educational institutions a lot of money. (Bocian, 2024) (Sanz et al., 2024)

Chatbots are also being used to support the mental health and wellbeing of students. They can provide a confidential and accessible platform for students to discuss their concerns and receive guidance. Recent studies found that chatbots can effectively support individuals with mental health issues by providing a non-judgmental and always-available outlet for their concerns. (Kelliher, 2021) (Sanz et al., 2024)

Chatbots can help with collaborative learning by moderating group discussions, managing collaborative projects and providing feedback. They can help to coordinate group activities, to track progress and to ensure that all members contribute effectively. Furthermore, chatbots can enhance collaborative learning by facilitating communication, coordination and knowledge sharing among group members. (Burkhard et al., 2022)

4.3 Benefits of Educational Chatbots

The adoption of chatbots in educational scenarios offers numerous benefits for both learners as well as for teachers or educators. In the following some important benefits are described.

Chatbots can provide a round-the-clock 24/7 access to educational support and therefore not being dependent on time and/or location. Learners can interact with chatbots at their convenience, whenever they need help and will receive immediate assistance and information. This makes chatbots highly accessible and convenient, because of their ability to provide support to students outside of traditional classroom hours. (Sanz et al., 2024)

Chatbots can adapt their interactions with its users based on the needs of individual learners, their preferences and the progress they make. This personalization helps to address the diverse learning styles and paces of students, enhancing their engagement and understanding. Personalized learning supported by chatbots can lead to improved learning outcomes and increased student satisfaction. (Sanz et al., 2024)

Chatbots can handle conversations with multiple learners simultaneously, which makes them highly scalable and more accessible. This scalability allows educational institutions to provide consistent support to a large number of students without the need for increases in staff, proportionate to the students. The scalability of chatbots is a key advantage, enabling them to support a large number of students while maintaining high levels of engagement and responsiveness. (Bocian, 2024)

Another of the key advantages of chatbots is that they are able to provide instant feedback to its users. Immediate feedback helps learners to identify and correct mistakes in real-time, e.g. if they are currently working on a task, reinforcing their understanding and preventing the accumulation of misconceptions. Especially concerning language learning, immediate feedback is very important and corrections as well as suggestions in real time can help to improve a students language skill. (Opeton, 2024)

Implementing chatbots can be a cost-effective solution for educational institutions. By automating routine tasks and providing scalable support, chatbots can reduce the need for additional staff and resources. The cost-effectiveness of chatbots can make educational ressources of high quality available to a wider range of learners and could further also improve the accesibility of educational ressources. (Bocian, 2024)

4.4 Challenges and Limitations

Despite the numerous benefits of implementing chatbots in the educational domain as mentioned above, educational chatbots also face several challenges and limitations. Some of the key challenges include:

Developing and maintaining educational chatbots involves various technical challenges. Ensuring accurate natural language understanding, handling diverse queries, providing contextually appropriate responses and also keeping the content safe for the target group, requires advanced and well trained AI and ML techniques. This technical complexity of developing a chatbot can be a hinderance to the widespread and adoption of chatbots in education. (Labadze et al., 2023)

Educational chatbots might handle sensitive information, which can include personal data and also academic records. To ensure the privacy of the data and security it is important to protect information from users from unauthorized access and data breaches. There is a need for robust data protection measures to build trust and confidence among users with the chatbot. (Dialzara, 2024)

While chatbots can simulate human conversation, they often lack the ability to fully understand and respond to the emotional nuances of human interactions. Furthermore, the limited emotional understanding of chatbots can be a drawback in applications such as mental health support, where empathy and emotional intelligence are very crucial. (Cluelabs, 2023)

The performance of educational chatbots and chatbots in general is very dependent on the quality and the comprehensiveness of the dataset they are trained on. Incomplete or biased data can lead to inaccurate responses and affect the overall effectiveness of the chatbot. There needs to be an emphasis on the importance of high-quality training data, to ensure the accuracy as well as the reliability of an educational chatbot. (Bawa, 2024)

Some teachers and students may be resistant to adopting chatbot technology due to concerns about either its reliability, its effectiveness and also its potentially negative impact on human interactions. Overcoming resistance to adoption is a key challenge in the implementation of educational chatbots, requiring clear communication about their benefits and limitations. (University-of-Florida, 2024)

4.5 Case Studies and Examples

To further illustrate the impact that chatbots have in the educational domain, several case studies and examples of successful implementations will be presented.

- **Duolingo:**
  Duolingo is a popular language learning platform, which also utilizes chatbots to help students with practicing their language skills. The chatbots of Duolingo like Duolingo Max engage users in simulated conversations called Roleplays, can provide instant feedback to their answer with detailed explanations of what they did right or wrong and very importantly adapt to the students proficiency level in the language they are learning. (Duolingo, 2023)

- **Stanford University's QuizBot:**
  Stanford University developed QuizBot, with the goal in mind to investigate wheter conversational AI can help students learn factual knowledge. The chatbot is designed to help students study and review course material through interactive quizzes. QuizBot generates questions, provides hints and offers explanations, creating an engaging and effective study tool. It was found that compared to students using a traditional flashcard app, students were able to recognize and recall over 20% more correct answers. Also in a further study students spent more time learning with the quizbot rather than with a flashcard app. (Ruan et al., 2019)

- **Georgia State University's Pounce:**
  Georgia State University has implemented Pounce, a chatbot which is designed to assist students with administrative tasks such as enrollment, how to get financial aid and course registration. Pounce has successfully reduced administrative workload, improved student engagement and enhanced overall satisfaction with the university services. A case study by Georgia State University (2019) reported that especially students who come from low income families or underrepresented minorities profited the most when using Pounce. (Georgia-State-University, 2022)

These case studies and examples demonstrate the potential of chatbots to enhance various aspects of education, from language learning and personalized tutoring to administrative support and student engagement. As the technology continues to evolve, we can expect to see even more innovative applications of chatbots in the educational field.

The rise of educational chatbots has in many different ways transformed how students can learn and interact with educational content. Starting from their early beginnings as simple rule-based systems, chatbots now have emerged into very sophisticated tools, often powered by artificial intelligence that can understand and respond to natural language queries. Over time they are able to increase their accuracy and contextual relevance. The integration of advanced artificial intelligence and machine learning techniques has enabled chatbots to also provide personal-

ized tutoring, engage learners in language practice in a conversational way, assist with administrative tasks, support mental health and wellbeing and finally also improve and foster collaborative learning. These diverse applications of chatbots in education come with numerous benefits, which include improved accessibility and convenience, personalized learning experiences, scalability, immediate feedback and cost-effectiveness. However, the implementation of educational chatbots also faces several challenges, such as the technical complexity of building a chatbot, privacy and security concerns, limited emotional understanding, dependence on data quality and also resistance to adoption from educators as well as educatees. As chatbots evolve further, chatbot developers are trying to address these challenges and further improve the capabilities of chatbots to better accommodate the needs of their users, whether they are learners or educators. The future of educational chatbots seems promising. With ongoing advancements in artificial intelligence and machine learning, we can expect to see even more innovative and impactful applications of these conversational agents in the educational domain. As chatbots become increasingly more personalized and adaptive, they have the potential to revolutionize the way we approach teaching and learning.

5 Different Types of Chatbots

Chatbots come in diverse forms and started out as simple rule-based systems. Each type of chatbot comes with its distinct characteristics and has different strengths and limitations. Understanding the different types is crucial for determining their suitability for specific applications, such as for this thesis, supplementing or replacing a traditional Wiki. IBM identifies five types of chatbots: menu- or button-based, rule-based, AI-powered, voice and generative AI chatbots. (Church, 2023)

For this project, the focus is on three specific types of chatbots: retrieval-based chatbots (a subset of NLP or AI -powered chatbots), generative AI chatbots and a hybrid form of these two. These types were chosen for the prototype due to their complementary strengths and potential to enhance the Catrobat Wiki's functionality and user experience. To provide context, it is also useful to understand rule-based chatbots, even though they were not used in the prototype.

5.1 Rule-Based Chatbots

Rule-based chatbots are a foundational and also the earliest form of chatbot technology that operate on predefined sets of rules and decision trees often using regular expressions to find matching predefined answers to queries. One such early example is ELIZA. ELIZA is simulating a so-called Rogerian therapist, meaning it is just conversing with the user. These kinds of chatbots follow a linear flow to process and respond to user queries. They oftentimes rely on decision trees which help to fit the chatbots responses based on the users input. (Luber, 2024) (Swain, 2024) Each possible interaction with the chatbot is mapped out in advance, which creates a logical flowchart that the chatbot follows according to a users input. Because rule-based chatbots are comparatively easy to implement, it makes them an attractive choice for organizations seeking to implement basic automated responses without delving into the complexities of advanced AI systems. (Digital, 2024)

These chatbots are particularly effective in environments where the range of user queries is narrow and predictable, or concerning only one topic, such as customer service, frequently asked questions (FAQs) or booking applications. (NeuroSoph, 2024)

5.1.1 Advantages of rule-based chatbots

One of the primary advantages of rule-based chatbots is, as already mentioned, their simplicity. They are straightforward to design and implement. The creation process involves defining a series of rules and mapping out potential interactions, which can be done without specialized knowledge of machine learning or advanced programming. Further, today many platforms like Joonbot (Franceschi, 2020) provide user-friendly interfaces and tools for developing rule-based chatbots, which further simplifies the implementation. These tools often include drag-and-drop builders and visual editors, which make them easily accessible. Rule-based chatbots also offer a high degree of control. Since the responses are predefined, there is total control over what the chatbot says and how it behaves in different situations. This ensures a high level of consistency and predictability (Kalka, 2024). The behaviour of rule-based chatbots can be precisely designed to meet specific needs, which allows organizations to script responses to cover particular scenarios and ensure that the chatbot meets their desired protocols (Singh et al., 2019). Furthermore, rule-based chatbots are cost-effective. The development of these chatbots typically comes with lower costs compared to AI-driven chatbots due to the absence of complex algorithms and the straightforward nature of the development process. The maintenance of the chatbot is also simpler in general, as updates usually just involve the modification of existing rules or the addition of new ones, which can be done without reprogramming or retraining.

5.1.2 Limitations of rule-based chatbots

Despite the advantages, rule-based chatbots also have multiple limitations. Firstly, they are restricted to the scenarios and responses that have been explicitly defined in their programming. This makes them unable to handle queries that are outside these predefined paths. Any query or scenario that is deviating from the expected or programmed interactions will result in the chatbot failing to provide a relevant response, which has the potential to reduce the satisfaction of its users. (Petruk, 2023)

While it is simple to add and modify the rules of the chatbot, if the range of tasks, queries and scenarios increases, the rule-based approach can become a problem, making it very annoying and time-consuming to manage and update a vast network of rules. With the increasing complexity, the performance can also suffer, as the chatbot needs to navigate through a large set of rules, which might slow down response times and make the system therefore harder to maintain. (BotPenguin, 2024)

Another limitation of rule-based chatbots is their lack of learning capabilities. They do not have the ability to adapt their responses in the future based on user feedback or new data fed to it, and any improvements or updates to the chatbot's knowledge base must be manually implemented by developers, which can be slow and labour-intensive as well. (Nuruzzaman & Hussain, 2018)

5.1.3 Suitability for Wikis

Despite their previously discussed limitations, rule-based chatbots can be suitable for specific use cases within educational platforms or to support or replace Wikis, such as the Catrobat Wiki. They can achieve this by handling FAQs and straightforward informational queries, quickly providing answers to common questions. As rule-based chatbots provide consistent and reliable responses, they can ensure that users receive the same information every time, which helps in maintaining accuracy and trust. (Kalka, 2024)

As an addition to Wikis, rule-based chatbots can act as a first line of interaction, which is handling basic queries and is directing users to more detailed Wiki pages when needed. This can enhance the user experience compared to just using the Wiki by providing immediate answers while also guiding users to more comprehensive resources for more complex questions. Additionally, rule-based chatbots can be a good solution for existing Wiki platforms, since it can retrieve its relevant information directly from the specific sections of the Wiki in order to respond to user queries. They enable users to interact with the chatbot to navigate the content of the Wiki more intuitively by asking for specific information and receiving direct links to the relevant sections in the Wiki. This makes the process more engaging for those who prefer conversational interfaces over traditional text-based searches. In order to update a rule-based chatbot it is necessary to add or modify its rules, which is generally straightforward and does not require advanced technical knowledge or even programming knowledge. However this has to be done manually and can therefore be more cost- and labour-intensive. While rule-based chatbots may not work for highly complex tasks or tasks that are outside of the defined ruleset, they can scale effectively for basic and repetitive queries, making them a viable option for handling a significant portion of user interactions without overwhelming the system. Rule-based chatbots can offer a practical and initially cost-effective solution for specific use cases within a Wiki. Their simplicity and ease of integration can make them suitable for handling straightforward informational queries and therefore supporting and enhancing the traditional Wiki content. However, their limitations in flexibility, scalability and learning capabilities are reasons to not use this type of chatbot for the Catrobat chatbot.

5.2 Retrieval-Based Chatbots

Retrieval-based chatbots are a type of conversational AI system that provides responses to user queries by selecting the most appropriate answers from a predefined repository. Similar to rule-based chatbots they are also mostly used in closed domain scenarios. (Neha, 2024) These chatbots make use of advanced techniques such as Term Frequency-Inverse Document Frequency (TF-IDF), cosine similarity and neural networks to match user inputs to the most relevant pre-defined answers. Their design and functionality make them especially effective in delivering precise and contextually appropriate responses in a closed domain scenarios. (Lee, 2023)

The core concept of retrieval-based chatbots is the selection of responses from a fixed knowledge repository. This repository is comprised of a usually large set of predefined answers, which are somehow indexed and organized to enable quick retrieval of a response based on user queries. When a user inputs a query, the chatbot processes the text and calculates the similarity between the query and the stored responses using different techniques. (Pandey & Sharma, 2023)

TF-IDF, or Term Frequency-Inverse Document Frequency, is a statistical measure which evaluates the importance of a word in a document relative to a corpus. The TF-IDF-Score is calculated for a word in a document out of a collection of documents. It is used to identify key terms in the user query and matching them with the most relevant responses. (MIKGroup, 2023) Cosine similarity measures the cosine of the angle between two vectors in a multi-dimensional space, which in this case is representing the query and the potential responses. High cosine similarity indicates a closer match between the query and the stored responses. (Miesle, 2023) Furthermore, advanced neural network models, particularly those designed for natural language processing, can be used to improve the accuracy of response matching by understanding the context and semantics of the query.

5.2.1 Advantages of Retrieval-Based Chatbots

One of the main advantages of retrieval-based chatbots is their ability to provide relevant and accurate information for the context they are used in. By making use of sophisticated matching algorithms, these type of chatbots can identify the most fitting responses from a vast pool of potential replies. Therefore, this retrieval of relevant responses is particularly valuable in areas where accurate information retrieval is critical, such as customer supports or knowledge bases like Wikis.

Retrieval-based chatbots are consistent in the information they provide. Since the responses are drawn from a repository of predefined answers, users receive the same information for identical or similar queries, which is therefore ensuring reliability. The fixed nature of the response repository allows to easily control the available answers, which ensures that only accurate and approved information is delivered to users. (Laba, 2024)

Furthermore, the quality control aspect is significant in environments where information accuracy is very important. The predefined repository can be curated and updated on a regular basis to always contain the most accurate and current information. This ensures that users are always provided with reliable data, in contrast to e.g. generative chatbots. This enhances the overall user experience and trust in the system.

5.2.2 Limitations of Retrieval-Based Chatbots

However, retrieval-based chatbots also have limitations. They are restricted to the responses available in their repository and cannot generate new or creative responses like a generative approach would be able to. If a user query does not closely match any of the existing responses, the chatbot may fail to provide an answer that the user finds satisfying. Henceforth, the effectiveness of a retrieval-based chatbot heavily relies on the quality and comprehensiveness of its response repository. Incomplete and outdated repositories can lead to bad user experiences and decrease the effectiveness of the chatbot. (Talkative, 2023)

5.2.3 Suitability for Wikis

Despite all of these challenges, retrieval-based chatbots can be a valuable addition or replacement for Wikis. They can, similar to rule-based systems, serve as a more interactive and user-friendly interface for accessing the information. By providing a conversational interface, these chatbots can make the process of finding information more engaging and intuitive. They can also enhance the user experience by offering quick and relevant responses to common queries, acting as a first point of contact and guiding users to more detailed articles.

Retrieval-based chatbots offer a great solution for enhancing the user experience and information accessibility in various domains and also for the Catrobat Wiki. While they have limitations, such as their reliance on the quality of the response repository and the challenges associated with scalability and maintenance, their ability to provide relevant and consistent information makes them a good choice for the Catrobat Wiki and for other educational institutions and their knowledge-bases and can improve the information delivery and user engagement.

5.3 Generative Chatbots

Currently there is an enormous hype about generative AI, including generative chatbots. Tools like ChatGPT or Google's Bard are all over the media and are known by the broad public. Generative chatbots represent a more sophisticated class of conversational agents that use advanced machine learning models, particularly neural networks, to generate responses which are based on the context of user queries rather than answers with predefined replies (Shridhar, 2017). Unlike rule-based and retrieval-based chatbots, which rely on predefined rules or existing responses, generative chatbots create new and contextually appropriate responses. They make use of deep learning and neural networks, to process and generate natural language in order to provide novel responses. (Hetler, 2024)

Generative chatbots are trained on huge amounts of text data, in order for them to learn patterns, structures and nuances of the human language. They further use this knowledge to predict and generate contextually relevant replies to the input given by the user. This dynamic approach allows them to handle a wide range of queries, making them highly versatile, interactive and not restricted to a specific setting. (Innodata, 2024)

5.3.1 Advantages of Generative Chatbots

One of the main advantages of generative chatbots is their flexibility. These chatbots are great at working with a wide range of different requests, ranging from simple questions to complex, ongoing conversations. The ability of generative chatbots to understand and generate language enables them to adapt to various contexts and still be able to provide meaningful responses and can therefore also be used in all kinds of domains. Additionally, generative chatbots are able to maintain the context throughout a conversation, allowing them to deliver more accurate and relevant responses. For instance, if a user asks a follow-up question, a generative chatbot can reference the previous question or context to provide a coherent answer. (Bansal et al., 2024)

Generative chatbots can also learn and improve over time. By analysing interactions and feedback, they can refine their language models and through that become more accurate and effective with each conversation. This continuous learning process enables them to adapt to new information and change to be more applicable to a specific users needs. With enough time, they can personalize their responses based on individual user interactions, enhancing the user experience by making conversations more relevant and engaging to each single user. (Spair, 2024)

Another advantage that should be noted for generative chatbots is their creativity. Unlike retrieval-based chatbots that pull their answers from a fixed set of responses, generative chatbots can create new and unique replies. This capability makes interactions more dynamic and engaging, as users receive answers that are not confined to a predefined knowledge base. The ability to generate creative and

varied responses helps to keep users engaged and can simulate more natural and human-like conversations, making the interaction feel more authentic. (Bani, 2024) (Spair, 2024)

5.3.2 Limitations of Generative Chatbots

Despite their numerous advantages, even generative chatbots have some significant challenges. Developing this kind of chatbot is complex and requires expertise in machine learning, natural language processing, as well as software engineering. The models must be carefully designed, trained and fine-tuned to ensure that they are able to perform effectively. In order to maintain generative chatbots, it is necessary to have regular updates and retraining, to feed it with new data and to improve its performance, which also requires significant technical expertise and resources.

Generative chatbots also need a lot of resources. The training and the deployment of these models is demanding for vast amounts of computational power. The computational costs can be high, especially for large-scale models. Additionally, in order to be effective, generative chatbots require vast datasets for their training. Gathering and keeping these datasets up-to-date can be resource-intensive as well, and the quality of the data directly impacts the performance of the chatbot.

Furthermore, there is also a risk of inaccuracies with generative chatbots. They can produce incorrect or inappropriate responses, especially if they are not adequately trained or if the training data contains biases. These errors can negatively influence the trust a user has in the system and lead to dissatisfaction. To minimize inaccuracies, generative chatbots need constant monitoring and fine-tuning. Implementing mechanisms to detect and correct wrong or insufficient responses is crucial for maintaining reliability. (Hu et al., 2020)

5.3.3 Suitability for Wikis

Generative chatbots can serve as a powerful replacement for traditional Wikis by providing dynamic, contextually relevant responses to a wide range of user queries. Their ability to generate natural language explanations and even answer follow-up questions makes them a highly effective tool. As with other types of chatbot, the conversational interface of generative chatbots can significantly enhance the user experience by making the process of finding information more intuitive and interactive. Users can ask questions in natural language and receive detailed and also personalized responses, reducing the need to click through multiple Wiki pages.

As an addition to existing Wikis, generative chatbots can offer a conversational layer that helps users navigate and understand complex information more effectively. They can act as guides, providing summaries, explanations and clarifications whenever needed. Integrating generative chatbots with the content of a Wiki ensures that users receive both immediate answers and access to more detailed information when needed.

Generative chatbots can further provide advanced search capabilities by allowing users to search for information using natural language queries, which can be more efficient and more user-friendly than traditional keyword-based search engines. They are able to maintain the context throughout a conversation, therefore generative chatbots can handle more complex queries and provide nuanced answers.

Moreover, the dynamic nature of generative chatbots promotes interactive learning. Users can engage in back-and-forth dialogues, ask follow-up questions and explore topics in depth, leading to a more engaging and educational experience. Generative chatbots can tailor their responses based on individual user interactions, providing personalized assistance that enhances user satisfaction and makes the learning process more enjoyable and effective. (Shukla, 2024)

5.3.4 Future Potential

As machine learning and NLP technologies continue to advance, the capabilities of generative chatbots will improve, making them even more effective and applicable in more domains. Beyond Wikis, generative chatbots have the potential to transform various domains, including customer support, education, healthcare and many more. Their ability to generate contextually relevant and engaging responses makes them a valuable tool across diverse applications.

In conclusion, generative chatbots offer a flexible and powerful solution for both replacing and supplementing Wikis. Their ability to generate new and contextually relevant responses enhances user engagement and satisfaction and further leads to dynamic and interactive user experiences. However, the complexity, resource requirements and potential for inaccuracy pose significant challenges that must be kept in mind. Despite these drawbacks, the continuous improvement and future potential of generative chatbots make them a great tool for enhancing information retrieval and user interaction in various contexts.

5.4 Hybrid Chatbots

Hybrid chatbots are an advanced approach to conversational agents by combining elements of rule-based, retrieval-based and generative chatbots. This combination merges the strengths of each type, while being able to mitigate their individual weaknesses. This creates in return a more versatile, accurate and efficient system. (Arz von Straussenburg & Wolters, 2023) Hybrid chatbots can follow predefined rules, retrieve relevant information from a repository and generate new responses based on user input, making them highly adaptable to various applications and user needs.

Hybrid chatbots are able to either switch between or combine different methodologies depending on the context of the interaction. For example, they might use rule-based responses for handling frequently asked questions, retrieval-based techniques for fetching some specific information from a database or some form of knowledge repository and generative models for more complex or multi-round queries. This approach guarantees that the chatbot can efficiently and more flexibly tackle a wide range of different queries.

5.4.1 Advantages of Hybrid Chatbots

One of the primary advantages of hybrid chatbots is their versatility. They can manage a large array of queries, from simple factual questions to more complex ongoing conversations. By making use of different methodologies, hybrid chatbots can adapt their responses to suit the intent of the query, which is ensuring that users receive the most appropriate and relevant information. This can enhance user satisfaction by ensuring that queries are handled according to their needs.

Hybrid chatbots can also improve the accuracy by combining the strengths of different approaches. Rule-based components ensure consistency for common queries, while retrieval-based and generative components provide flexibility and more depth. (Tiffany, 2023) This ability to switch between rule-based, retrieval-based and generative approaches allows hybrid chatbots to provide the most appropriate response in any given situation.

Moreover, these chatbots can learn from interactions over time, improving their responses and performance. The generative component in particular benefits from continuous learning. This enables hybrid chatbots to become better and better at handling diverse and evolving user queries. (Sandeep, 2024)

5.4.2 Limitations of Hybrid Chatbots

As mentioned beforehand, hybrid chatbots come with many advantages, however, they also come with some challenges of their own, particularly in terms of complexity in their design as well as their implementation. Developing these chatbots is inherently more complex than creating a chatbot that is just using one technique, as integrating rule-based, retrieval-based and generative elements requires careful planning, coordination and expertise in multiple domains. The algorithms and models used in hybrid chatbots need to be finely tuned to work together seamlessly, which can increase the initial development time and effort as well as the potential for integration challenges.

Hybrid chatbots also have significant resource requirements. They need a lot of computational resources to run effectively, if they are using advanced generative models, which can lead to higher infrastructure costs, especially for large-scale applications. Additionally, comprehensive training data is essential for the generative and retrieval-based components of hybrid chatbots. Collecting and maintaining this data can be very resource intensive, since it is also adding to the overall cost of development and maintenance. (Tiffany, 2023)

Maintaining hybrid chatbots involves ongoing management to ensure the seamless integration and performance of different components. Regular updates, fine-tuning and monitoring are necessary to maintain performance and to address any issues that may come up. Debugging can be more complex due to the interplay of different models and systems, requiring a deep understanding of each component and their interactions to identify and resolve problems.

5.4.3 Suitability for Wikis

Hybrid chatbots are highly suitable as replacements for traditional Wikis. They can handle a wide variety of queries with high accuracy and relevance, which can provide users with quick and precise answers. Their ability to generate responses based on context ensures that users receive the information most fitting to their queries. By offering a conversational interface, hybrid chatbots can transform the static nature of traditional Wikis into a dynamic, interactive experience. Users can engage in natural language conversations, making the process of finding information more intuitive and user-friendly.

As additions to existing Wikis, hybrid chatbots can significantly enhance the user interaction. They can provide quick answers to common questions, guide users to more detailed articles and assist with more complex inquiries, which enriches the overall user experience. Hybrid chatbots can supplement Wikis by offering a more dynamic approach to the traditional information retrieval process. They can interpret user intent and context, providing tailored responses that direct users to the most relevant sections of the Wiki.

Hybrid chatbots also excel at advanced search capabilities. They can understand nat-

ural language queries, making it easier for users to find the information they need without having to use specific keywords or navigate complex menus. This capability improves accessibility and user satisfaction. By maintaining context throughout a conversation, hybrid chatbots can handle queries over multiple rounds of dialogue and provide more nuanced answers, which ensures that users receive comprehensive and accurate information.

The conversational nature of hybrid chatbots promotes interactive learning. Users can engage in dialogue, ask follow-up questions and explore topics in depth, leading to a more engaging and educational experience. Hybrid chatbots can personalize their responses based on individual user interactions, which has also the potential to increase user satisfaction. This personalized approach makes the learning process more enjoyable and effective, as users receive information tailored to their specific needs and interests.

Integrating hybrid chatbots into existing Wikis means a significant effort in their design, their implementation and ongoing maintenance. Ensuring compatibility with the existing infrastructure and content is crucial for a seamless integration. The management of user data and ensuring privacy and security are critical concerns when developing hybrid chatbots. It is necessary to implement robust data protection measures in order to maintain the trust of users and to comply with different regulations.

5.4.4 Future Potential

Since hybrid chatbots also make use of generative methods the advancements in machine learning and also in NLP technologies, will improve the capabilities of hybrid chatbots as well, which will make them even more versatile and effective. Again, hybrid chatbots not only have the potential to have a positive impact on Wikis in the furture, but also on various other domains.

Hybrid chatbots can provide a versatile, accurate and efficient solution for either replacing a Wiki or enhancing it. By combining the strengths of rule-based, retrieval-based and generative approaches, they provide a comprehensive and dynamic information retrieval experience. Despite the challenges of complexity, resource requirements and maintenance, the continuous improvement and future potential of hybrid chatbots makes them a promising tool for enhancing user interaction and information accessibility in various contexts.

6 How can using a chatbot affect the accessibility and engagement of the Catrobat Wiki?

The integration of a chatbot into the Catrobat Wiki is trying to improve both its accessibility and its engagement. Building on the previously discussed findings about the use of chatbots in educational domains and by investigating their advantages and limitations, it becomes clear that chatbots have the ability to offer a wide range of benefits that have the ability to enhance the user experience. By using conversational interfaces, chatbots can provide immediate support, personalized interactions and therefore a more user-friendly experience. (Seidor, 2023) In the following, we will examine the ways in which these characteristics can have a beneficial influence on both the Catrobat Wiki and the overall Catrobat project.

6.1 Increased Accessibility

Improved accessibility is one of the main benefits of implementing a chatbot. Conventional Wikis like the Catrobat Wiki often require users to navigate through menus or search for a specific topic, which can be overwhelming, especially for audiences to which the Catrobat project is tailored. In contrast, chatbots offer a conversational interface that allows users to ask questions in natural language, which makes it easier to find the information they need. Research indicates that users often prefer interactive systems that mimic human conversation, as they feel more approachable and less intimidating than traditional search interfaces. (Labadze et al., 2023)

Furthermore, chatbots can be designed to support multiple languages, which can further broaden the reach of the Catrobat Wiki. Given that Catrobat aims to serve a global audience, including children from diverse linguistic backgrounds, a multilingual chatbot can lower their barriers to start working. It can foster inclusivity and encourage participation also from non English speakers. By offering assistance in various languages, chatbots can ensure that language is not a barrier to participate in the Catrobat ecosystem or to access its information. This makes the Wiki more inclusive and user-friendly. (Algomo, 2024)

Additionally, Chatbots are able to handle conversations with multiple users or learners simultaneously. This scalability of chatbots therefore enables Catrobat to provide consistent support to a large number of students without the need for increases in staff, proportionate to the students using catrobat. (Bocian, 2024)

6.2 Increased Engagement

Chatbots can significantly enhance the user engagement by providing a more interactive experience. Unlike classic FAQs or Wikis, which present their information in a linear way, chatbots can engage users in a dialogue, by asking clarifying questions and tailoring responses based on individual user needs. This personalized interaction can lead to a deeper understanding of the content and increased motivation to explore the Wiki further. This way of interactive learning can improve the knowledge retention and the user satisfaction, particularly for younger learners. By transforming passive reading into an active conversation, the chatbot can make learning more engaging and effective in the Catrobat ecosystem. (Perihan, 2024)

6.3 Immediate Support

Another significant advantage of chatbots is their ability to provide immediate support. Users often may encounter challenges or problems while using the programming tools of Catrobat. A chatbot can offer real-time assistance to address these issues. By providing 24/7 support, the chatbot ensures that users can find answers to their questions without having to wait for human intervention. This immediacy is crucial for maintaining the user engagement and satisfaction, as delays in the support can lead to frustration in users. For example, if a user encounters a bug or has a question about a programming concept, the chatbot can provide instant guidance or direct them to relevant resources. (Sanz et al., 2024) (Cunningham-Nelson et al., 2019)

Another key advantage of integrating a chatbot into the Catrobat ecosystem is that it is able to provide instant feedback to its users. This direct feedback can help learners to identify and correct mistakes in real-time. Immediate feedback is very important, especially with regards to language learning, and corrections as well as suggestions in real time can help improve a students language skills. (Fryer, 2006)

The integration of a chatbot into the Catrobat Wiki has the potential to significantly enhance both the accessibility and the engagement of the Catrobat Wiki and the entire Catrobat project. By providing real-time assistance and simplifying navigation, a chatbot can make the Wiki more accessible to a more diverse audience. Additionally, through interactive learning and a more personalized experience, the chatbot can boost the user engagement for participants in the Catrobat environment. By making use of the aforementioned capabilities of chatbots, the Catrobat Wiki can be replaced or supported by a more inclusive, engaging and user-friendly system.

7 Decision-Making Process for Implementing the Chatbot for the Catrobat Wiki

When starting the journey to enhance the Catrobat Wiki with chatbot technology, the several types of chatbots mentioned above were considered. Among these, rule-based chatbots were initially evaluated since they are a widespread form of chatbots, are comparatively easy to implement and would be suitable to retrieve the predefined contents of the Catrobat Wiki. The decision was made to start by implementing the Chatbot in Python, including a Python GUI to be able to test its capabilities faster and only afterwards integrate this chatbot in the share platform of Catrobat. The decision to not implement the prototype as a rule based Chatbot was made for various reasons concerning its limitations.

- Rule-based chatbots operate on a predefined set of rules, making them inflexible when it comes to handling queries outside their programmed scope. The dynamic and diverse nature of queries from users of the Catrobat Wiki required a more adaptable solution.

- As the complexity and volume of content within the Catrobat Wiki grows, maintaining and updating the rules for a rule-based chatbot would become increasingly more time intensive. This scalability issue would hinder the chatbot's ability to keep up with the expanding knowledge base.

- Rule-based chatbots are static and do not improve over time unless manually updated. This lack of learning capability would limit the chatbot's effectiveness in providing up-to-date and contextually relevant information to users.

Given these limitations, it was clear that a more sophisticated approach was necessary. The Catrobat Wiki required chatbots capable of understanding a wide range of natural language queries, providing accurate and also contextually appropriate responses and adapting to the evolving content within the Wiki.

After rejecting the rule-based approach, two main types of chatbots were selected for implementation: a retrieval-based chatbot using Natural language processing and TF-IDF (Term Frequency-Inverse Document Frequency) and a generative chatbot integrated with the ChatGPT API.

7.1 Retrieval-Based Chatbot Using TF-IDF

Recap of the advantages of a retrieval-based chatbot:

- Retrieval-based chatbots perform great at finding the most relevant responses from a predefined repository of information. Since we already have the Catrobat Wiki as our predefined repository of information, this is an important advantage. By leveraging sophisticated matching algorithms, the chatbot can provide highly accurate responses based on the similarity between user queries and the content on the Wiki.

- These chatbots ensure consistent responses by retrieving predefined answers and maintaining a high level of reliability and predictability in their interactions, which is also relevant for the new Catrobat chatbot since we want to be reliable in our answers.

- Furthermore, retrieval-based chatbots are efficient in providing quick responses to well-defined queries, enhancing the overall user experience.

7.1.1 Why Choose TF-IDF for Query Matching?

When considering how to implement the matching of user queries to the content of the Catrobat Wiki, several techniques were evaluated, including Latent Semantic Analysis (LSA), Word2Vec and BERT. Each of these methods has its own strengths and weaknesses:

- **Latent Semantic Analysis (LSA):**
  LSA extends beyond the traditional term frequency-inverse document frequency (TF-IDF) by further capturing the underlying structure of the data. It iis transforming TF-IDF vectors to a lower-dimension represantation. While LSA can reveal deeper semantic meanings, it is more computationally intensive and complex to implement. (Pocs, 2020)

- **Word2Vec:**
  Word2Vec is a neural network-based technique which also processes text by transforming words to vectors. Word2Vec is particularly effective for understanding context and can enhance retrieval accuracy. However, it requires a large corpus for training and furthermore demands significant computational resources. (Nicholson, 2024)

- **BERT (Bidirectional Encoder Representations from Transformers):**
  BERT is a significant advancement in natural language processing technology. BERT was developed by Google in 2018 and is able to consider the full context of a user query. BERT is able to capture nuances of the human language and can therefore provide highly relevant responses. However BERT is also very resource intensive and complex and therefore not ideal for a prototype. (Luna, 2023)

The Case for TF-IDF

Given the advantages and disadvantages of these techniques, it was decided to use TF-IDF for the initial prototype of the Catrobat chatbot. The decision to use TF-IDF was based on several key factors:

- **Simplicity and Effectiveness:** TF-IDF is a well-established method for text retrieval that balances simplicity with effectiveness. It calculates the importance of words in a document relative to a collection of documents, making it suitable for identifying relevant information in the Catrobat Wiki. (Karabiber, 2024b)

- **Computational Efficiency:** Compared to the more complex algorithms, TF-IDF is computationally efficient and can be implemented with relatively low resource requirements. This practicality makes it an ideal choice for the initial phase of the project.

- **Ease of Implementation:** TF-IDF is straightforward to implement using existing libraries and tools in Python, which enables rapid development and deployment. This allows for quicker iteration and testing of the chatbot's capabilities. (Karabiber, 2024b)

In summary, the decision to use TF-IDF for the initial prototype of the Catrobat chatbot is grounded in its simplicity, efficiency and ease of implementation, making it a fitting choice for the development of a prototyoe.

Why Not Machine Learning-Based Intent Classification?

While considering various approaches on implementing a chatbot, the idea of using machine learning-based intent classification was evaluated. This method typically involves training a model, such as a decision tree, to classify user queries into predefined intents (helpshift, 2024). However, several factors led to the decision against using this approach for the initial phase:

- **Manual Upkeep of Labels:** Machine learning-based intent classification requires a significant amount of manual labelling of training data. Each potential query needs to be categorized into an appropriate intent, which is a time-consuming and ongoing process(Braun et al., 2017). The need for constant updating and maintenance of these labels to always be able to handle new queries and evolving user needs can be quite labour-intensive.

- **Complexity and Maintenance:** Managing an intent-based system involves more than just the initial setup. It is necessary to have continuous training and retraining of the model, to maintain accuracy as the variety of user queries grows. This can be labour-intensive and requires dedicated resources to ensure the system remains effective.

- **Moderation Effort:** To maintain accurate responses, human moderation would be required to review and correct the responses of the chatbot, especially in the case of ambiguous or misclassified intents. This adds another layer of complexity and ongoing workload.

- **Flexibility and Scalability:** A retrieval-based approach using TF-IDF allows for greater flexibility and scalability. As the underlying repository of information grows, the system can seamlessly integrate new content without the need for extensive retraining or labelling of data. This makes it easier to expand the knowledge base and improve the chatbots capabilities over time.

In summary, while machine learning-based intent classification can offer high accuracy and a good understanding of user queries, the corresponding manual upkeep, complexity and moderation efforts make it less suitable for the prototype of the Catrobat chatbot. A retrieval-based approach using TF-IDF is a good balance between simplicity, effectiveness and ease of implementation, making it the preferred choice for deploying a reliable and efficient chatbot.

How does TF-IDF work in combination with cosine similarity for a retrieval based chatbot?

7.1.2 TF-IDF and Cosine Similarity for Retrieval-Based Chatbots

Retrieval-based chatbots in general rely on sophisticated techniques to match user queries with the most relevant responses from a predefined repository containing the information. Two possible components to achieve this functionality are Term Frequency-Inverse Document Frequency (TF-IDF) and cosine similarity. (Lee, 2023)

7.1.3 Term Frequency-Inverse Document Frequency (TF-IDF)

TF-IDF is an oftenly used statistical method for quantifying the importance of words within a document or a collection of documents (Karabiber, 2024b). TF-IDF tells us how significant a term is for a text in a collection of documents (corpus).

The TF-IDF value for a word is calculated as follows:
TF-IDF(t, d) = TF(t, d) × IDF(t)
Where:

- TF(t, d) is the so called term frequency, which represents the number of times the word t appears in the document d. It is calculated by dividing the number of words in document d by the total number of words in the document d.

- IDF(t) is the inverse document frequency, which measures the importance of the word t across the entire corpus of documents. It is calculated as follows:
  IDF(t) = log(N / df(t))
  Where:

- N is the total number of documents in the corpus.
- df(t) is the number of documents containing the word t.

The TF-IDF value is higher for words that appear frequently in a specific document but are less common across the entire corpus. These words are considered more informative and relevant for that document. (GeeksforGeeks, 2023)

7.1.4 Cosine Similarity

Cosine similarity is a commonly used measure of the similarity between two non-zero vectors, such as the TF-IDF vectors of a user query and a document. It measures the cosine angle between two vectors. If the directions of two compared vectors are identical, this would yield a cosine similarity of 1, if they are pointing in opposite directions -1 and 0 if they form a 90 degree angle. The closer the value is to 1 the higher is the similarity of the vectors. A 1 would indicate a perfect match between the two vectors and therefore the texts would be the same. (Lee, 2023) The cosine similarity between the two vectors, A and B, is calculated as shown below:

$$\text{Cosine Similarity}(A, B) = \frac{A \cdot B}{\|A\| \times \|B\|} \tag{7.1}$$

Where:

- $A \cdot B$ is the dot product of the two vectors.
  Which is calculated as:

$$\mathbf{A} \cdot \mathbf{B} = a_1 b_1 + a_2 b_2 + \cdots + a_n b_n = \sum_{i=1}^{n} a_i b_i \tag{7.2}$$

- $\|A\| and \|B\|$ are the magnitudes of the vectors.
  Where the magnitude is computed as:

$$\|\mathbf{A}\| = \sqrt{a_1^2 + a_2^2 + \cdots + a_n^2} = \sqrt{\sum_{i=1}^{n} a_i^2} \tag{7.3}$$

(Karabiber, 2024a)

7.2 Combining TF-IDF and Cosine Similarity for Retrieval-Based Chatbots

In the context of a retrieval-based chatbot, the TF-IDF and cosine similarity techniques work together to identify the most relevant responses from the chatbot's knowledge base (Lee, 2023). Below this process is shown on the example of the new Catrobat chatbot:

1. **Preprocessing and TF-IDF Vectorization:** The chatbot's knowledge base, which consists of the data retrieved from the Wiki by a scraper and therefore contains the predefined answers, is preprocessed by tokenizing, converting the text to lowercase, removing stop words and filtering out non-alphabetic tokens. The preprocessed text is then transformed into a so called TF-IDF matrix representation using the TfidfVectorizer from the scikit-learn library. The TF-IDF matrix is a vector where each word of the text represents a number (The calculated TF-IDF value for the word).

2. **User Query Processing:** When a user query is received, it undergoes the same preprocessing steps as the knowledge base. The preprocessed query is then also transformed into a TF-IDF vector using the same TfidfVectorizer.

3. **Cosine Similarity Calculation:** The cosine similarity between the user query's TF-IDF vector and each document's TF-IDF vector in the knowledge base is calculated. This step identifies the most relevant documents or responses based on the similarity scores.

4. **Response Selection:** The chatbot selects the document or response with the highest cosine similarity score as the most relevant answer to the user's query. If the highest similarity score is below a predefined threshold, the chatbot indicates that no suitable response was found.

By combining TF-IDF and cosine similarity, retrieval-based chatbots can effectively match user queries with the most relevant information from their knowledge base, providing accurate and contextually appropriate responses. This approach has several advantages, including:

- **Relevance:** The TF-IDF and cosine similarity techniques ensure that the most relevant responses are retrieved, based on the similarity between the user's query and the chatbot's knowledge base.

- **Efficiency:** The computational efficiency of these methods allows the chatbot to provide quick responses, enhancing the overall user experience.

- **Scalability:** As the knowledge base grows, the TF-IDF and cosine similarity calculations can scale to handle larger volumes of information without compromising performance.

(Pradeep, 2023)

The integration of TF-IDF and cosine similarity is a powerful technique for retrieval-based chatbots, enabling them to efficiently match user queries with the most relevant information from their knowledge base. This approach, when combined with other advanced natural language processing techniques, can significantly enhance the effectiveness and user experience of educational chatbots. However, it is important to note that retrieval-based chatbots have limitations, such as their inability to handle complex, open-ended queries or generate novel responses. To address these limitations, many chatbot systems employ a hybrid approach, combining retrieval-based and generative models to provide a more comprehensive and adaptive solution (Arz von Straussenburg & Wolters, 2023).

7.2.1 Generative Chatbot Using OpenAI API

Recap of the advantages of generative chatbots:

- Generative chatbots can handle a wide range of queries and generate novel responses. This allows them to manage complex and varied user interactions.
- These chatbots can learn from interactions and improve over time, becoming more effective in providing contextually relevant answers.
- Unlike retrieval-based models, generative chatbots are not limited to predefined responses, making interactions more engaging and dynamic.

Why ChatGPT API?

- ChatGPT, developed by OpenAI, is one of the most advanced language models available. Its ability to understand and generate human-like texts makes it ideal for providing contextually relevant and high quality responses to user queries.
- The ChatGPT API is easily integrable with existing systems, especially in our case with Python. By providing instructions specific to Catrobat-related questions, the generative chatbot can focus its responses on relevant topics.
- The API is backed by ongoing research and improvements by OpenAI, ensuring that the chatbot remains at the cutting edge of natural language processing capabilities.

7.2.2 Implementation Using Python

Python was chosen for this project due to its versatility, popularity and extensive support within the programming community. It is known for its readability and simplicity. Therefore Python is accessible to developers of all skill levels. Its widespread use ensures a robust support network and an extensive amount of libraries which are suitable for a variety of tasks, including natural language processing and machine learning.

One of the significant advantages of Python is its vast ecosystem. Libraries such as NLTK, SpaCy and scikit-learn simplify the text processing tasks, while TensorFlow and PyTorch are great for developing machine learning models. These tools collectively can support the efficient development and deployment of both retrieval-based and generative chatbots, making Python an ideal choice for this project.

Moreover, Python's ability to setup a simple HTTP-Server to communicate with other technologies and platforms further underscores its suitability. This ensures that the chatbot can interact effectively with the Catrobat Share Platform and other components of the Catrobat ecosystem, and integrates smoothly within the existing systems without significant reengineering.

In terms of performance, Python is capable of handling complex computations efficiently, ensuring that chatbot solutions are both responsive and scalable. This performance is crucial for providing a real-time, interactive experience for users, maintaining a balance between speed and accuracy in delivering responses.

7.3 Decision Process and Final Implementation

The decision to implement both, retrieval-based and generative chatbots in combination, was driven by the specific needs and objectives of the Catrobat Wiki. The primary goal was to enhance the user interaction by delivering accurate as well as contextually appropriate responses to concrete questions by the user. Furthermore it is beneficial to incorporate the unique replies of generative models to make the chatbot more engaging and to handle more complex conversations. This hybrid approach ensures that the chatbot can handle a broad spectrum of inquiries, from simple information retrieval to more complex questions.

To meet these needs of Catrobat Wiki users, a combined approach involving both retrieval-based and generative chatbots was implemented.

The TF-IDF (Term Frequency-Inverse Document Frequency) based retrieval chatbot was developed to handle queries with a concrete question by matching user inputs with the most relevant content from the Wiki. This approach ensured that users could quickly access accurate and specific information, effectively addressing straightforward queries.

To manage more complex and varied queries, a generative chatbot powered by the ChatGPT API was integrated. This chatbot was instructed to focus its responses on Catrobat-related topics to maintain relevance and accuracy. This capability allows the chatbot to generate context-sensitive and very detailed responses, which helps with conversations that require more understanding and flexibility.

To take advantage of the strengths of both approaches, a hybrid chatbot was developed. This hybrid model returns the generative response from the ChatGPT API and appends the four most relevant results from the TF-IDF retrieval-based bot. This combination of the two approaches ensures that users receive comprehensive answers, that provide both depth and breadth of information, and thereby enhance the overall user experience.

The chatbot was designed to fit in with the design of the Catrobat Share Platform interface. A user-friendly frontend, implemented in HTML and Twig, allows users to interact with the chatbot in a simple way. The backend of the Share Platform communicates over HTTP with a server developed in Python, which handles the logic and processing, ensuring efficient communication between the frontend and the chatbot. This architecture ensures a smooth and responsive user experience, and enables easy access to information.

Benefits

The development of both, retrieval-based and generative chatbots, combined into a hybrid model, significantly enhances the user experience on the Catrobat Wiki. By addressing a wide range of user queries, from simple information retrieval to complex, context-sensitive interactions, the chatbot ensures users can access the information they need quickly and efficiently.

Moreover, the generative part of the chatbot and therefore the chatbot in general benefits from continuous improvement, by learning from interactions and adapting to the needs of users over time. This capability to learn improves the effectiveness of the chatbot, by providing increasingly relevant and accurate responses.

The decision to implement a hybrid chatbot solution for the Catrobat Wiki was driven by the need for a flexible, accurate and also scalable system. By taking advantage of the strengths of both TF-IDF for retrieval-based responses and the ChatGPT API for generative responses and using Python for development, the hybrid solution successfully addresses the limitations of traditional rule-based chatbots. It ensures that the Catrobat Wiki can cater to a diverse range of user needs, offering efficiency in retrieving known information and flexibility in handling more complex queries. The hybrid chatbot model, by combining generative responses with the most relevant retrieval-based results, provides a robust and user-friendly solution that can significantly enhance the experience users have with the Catrobat Wiki.

8 Data Scraping for Content Acquisition

The acquisition of data from the Catrobat Wiki is a crucial step in developing an effective retrieval-based chatbot. This chapter discusses the challenges that were encountered during the data export, the development of a custom data scraping tool to address these challenges and the benefits of choosing this approach. The repository on github for the scraping tool can be found here: https://github.com/bonbuo/Catrobat-Wiki-Scraper

8.1 Challenges with the Data Export

Extracting the needed data from the Catrobat Wiki posed several significant challenges, primarily due to the limitations of the available data export methods. The Wiki, hosted on Xwiki, only supports exports in formats such as ODT, PDF, RTF and XAR. ("Xwiki," 2024) While these formats are useful for human readability, they are not ideal for automated data extraction and analysis. The key challenges encountered include:

- The available export formats are not ideal for easy data parsing and manipulation. For instance, ODT, PDF and RTF are document formats that prioritize formatting over structured data representation, making it difficult to extract specific pieces of information programmatically.

- Some of these formats do not inherently support the hierarchical and relational structure of the Wiki content, which is essential for building a chatbot that understands the context and relationships between different pieces of information.

- The Catrobat Wiki is regularly updated with new information, whether something is added or existing content is edited. A one-time static export would quickly become outdated. Therefore it is necessary to have a dynamic and repeatable data extraction process to keep the chatbot's knowledge base current.

- Different subcategories within the Wiki have a unique HTML structure, which requires tailored approaches to accurately extract the relevant data. This variability of the subcategories further adds complexity to the data scraping process and asks for a more flexible and adaptable solution.

8.2 Development of the Data Scraper

To overcome the above mentioned challenges, a custom data scraping tool was developed using Python. The scraper was designed to navigate the structure of the Catrobat Wiki, retrieve the relevant content and organize it into a structured dataset suitable for the chatbot to retrieve the data. There are a few important aspects for the scraper's development:

- The scraper makes use of Python libraries such as BeautifulSoup and requests. BeautifulSoup is used to parse HTML content, which allows for the extraction of specific elements based on tags and attributes. The requests library enabled the HTTP requests, necessary to fetch the Wiki pages.

- The scraper was programmed to systematically navigate through the Wiki, identifying and accessing pages of interest. This automated navigation reduced the need for manual intervention, which simplifies the data collection process.

- Given the diverse HTML structures across different Wiki subcategories, the scraper has to consider numerous edge cases to handle variations in the content layout. Custom parsing rules were developed for each subcategory to ensure accurate data extraction.

- As this thesis is about developing a prototype, the initial phase of the scraper focused on extracting data from the Brick Documentation section of the Wiki. This section contains detailed information about the various programming bricks, which are necessary for answering user queries related to specific functionalities of Pocket Code.

- While the initial scraper focuses on the Brick Documentation, the design is scalable to accommodate for future expansions. This may include adapting the scraper to extract data from other Wiki categories, which enables the chatbot to answer a broader range of questions beyond specific bricks and cover the entire content of the Catrobat Wiki.

8.3 Benefits of the Custom Scraping Solution

The development of a custom data scraping solution offered several significant advantages over the available data export methods. Firstly, by programmatically navigating the Catrobat Wiki, the scraper provides better flexibility and control over the data collection process. This ensured that the exact information needed for the chatbot could be extracted and structured appropriately.

The scraper was designed to adapt to changes in the Wiki's structure or content. This adaptability is important for maintaining the reliability and scalability of the data collection process over time, ensuring that the chatbot's knowledge base remains up-to-date.

The custom scraper enables the acquisition of a comprehensive dataset, encompassing detailed information on programming bricks and their functions. This dataset is essential for the chatbot to understand and respond accurately to user queries.

By automating the data extraction process, the scraper significantly reduced the time and effort a person would have to invest to keep the data up-to-date. This allows for more frequent updates and continuous improvement of the chatbot's capabilities.

The scraper is able to handle diverse HTML structures and edge cases which ensures that all relevant data can be extracted and structured appropriately. Accurate and structured data is important for the chatbot's performance, as it can directly impact the accuracy and relevance of the responses provided to users.

8.4 Implementation Details

To provide a better understanding of the scraper's implementation, the following steps show the process in more detail:

1. **Identifying Target Pages:** The first step was to identify the specific pages within the Catrobat Wiki which contained relevant information. This included pages related to programming bricks, tutorials and user guides.

2. **Retrieving HTML Content:** Using the requests library, the scraper fetched the HTML content of each target page. This step involved sending HTTP GET requests to the Wiki's URLs and retrieving the HTML responses.

3. **Parsing HTML Structure:** BeautifulSoup was used to parse the HTML content. The parser identified specific HTML elements, such as divs, spans and paragraphs that contained the desired information. Custom parsing rules were applied to handle variations in the HTML structure across different pages.

4. **Storing the Data:** The extracted information was organized into a structured format, in form of a JSON file.

5. **Data Cleaning:** The stored data in the JSON file underwent a cleaning and validation process to ensure its accuracy and consistency. This step also involved the removal of duplicates, correcting formatting issues and verifying the extracted information against the original Wiki content.

8.5 Future Enhancements

While the current scraper focuses on the Brick Documentation, future enhancements will involve extending the scraping capabilities to other sections of the Wiki. This includes other sections like the Formula Editor, the About Catrobat section and also the tutorial section. Since the tutorial section includes also a lot of videos, it would be necessary to adapt the chatbot in order to handle them correctly.

9 Implementation and Integration

As previously discussed, the chatbot's core logic is implemented in Python. A GitHub repository was created to host the code, accessible via: https://github.com/bonbuo/Catrobat-Chatbot. This repository includes not only the final version of the chatbot prototype but also an initial prototype used for experimentation. The initial prototype features a standalone graphical user interface (GUI) developed entirely in Python and does not communicate with the Share Platform.

9.1 How the chatbot works

At first, a new page was added to the Catrobat Share Platform using HTML and Twig templates, which serves as the frontend interface for the chatbot.

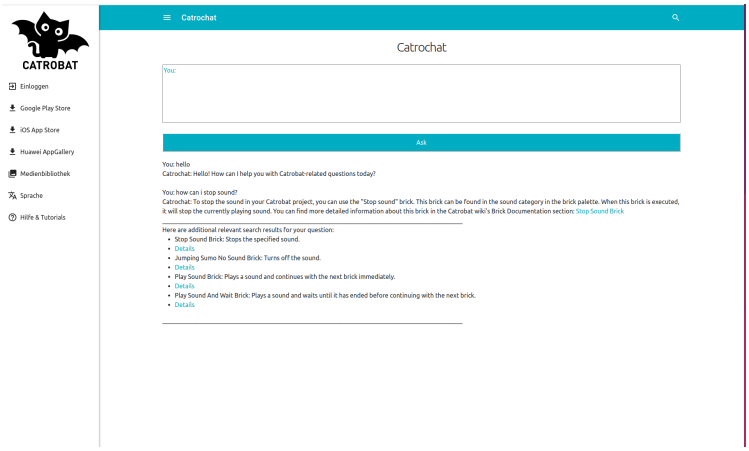

Figure 9.1: Screenshot of the GUI of the Chatbot integrated into the Catrobat Share Platform

The GUI depicted above includes a text window where users can input their questions and a display area where previous questions and answers appear. It also features a button that, when clicked, submits the query of the user. This text window and button are part of a form that, upon submission, sends the user query to the backend of the Catrobat Share Platform via a POST request. The backend then processes this request and forwards it to a Python server, which manages the chatbot logic. In order for the chatbot to function correctly, the Python server must be operational. The chatbot consists of three main components: the server, the logic and the data.

9.1.1 Server

The server is responsible for managing POST requests containing user queries from the Catrobat Share Platform. It processes these queries and returns appropriate responses. There are three servers in total, although two are primarily for demonstration purposes. The primary server, which is found in server_chatbot_hybrid.py, integrates both retrieval-based and generative AI models to provide comprehensive responses. We will take a look at this sever.

The server is set up using Python's http.server module, which enables the interaction with the Catrobat Share Platform, where the chatbot's frontend is located. The server handles both GET and POST requests, although the GET requests are mainly for testing purposes.

The retrieval-based logic of the chatbot is managed by the CatrochatRetrievalBased-Hybrid class from the chatbot_retrieval_based_hybrid module, which is imported on the server. Additionally, the server uses the JSON library to read data from a JSON file containing information scraped from the Catrobat Wiki.

To integrate with the OpenAI API, the server also includes the openai library. This integration requires an API key, which is stored as an environment variable on the server for security reasons. In future versions, it might be beneficial to allow users to provide their own API keys. This would ensure that they bear the cost of their queries, rather than the Catrobat project, helping to manage query volumes and associated expenses more effectively.

The server also includes predefined messages that serve as instructions to the OpenAI API. These messages set the context for the chatbot, specifying that it should only respond to Catrobat-related questions and provide relevant links from the Catrobat Wiki when possible. This ensures the chatbot remains focused and accurate in its responses.

How does this Server work?

Starting the Server (run_server Method)

- The server is configured to listen on all available IP addresses (0.0.0.0) on port 8000, which will be replaced in a live setting with the actual IP address and port.
- An HTTPServer instance is created with the server address and the RequestHandler class.
- The run_server function is called if the script is run directly, starting the server.
- The server is started, is running until the server is shut down and waits for incoming requests.

Handling GET Requests (do_GET Method)

When a GET request is made to the answer_user path, the server responds with a simple greeting. All other paths result in a 404 error. This is only used for testing purposes.

Handling POST Requests (do_POST Method)

- **Reading and Parsing Request Data:**
  - The server reads the length of the incoming request data.
  - It reads the actual data from the request and decodes it from bytes to a string.
  - The decoded data is parsed into a dictionary using parse_qs to extract the user_query.
- **Processing the User Query:**
  - If a user_query is present, the server prints the received query to the console.
  - It then sends a 200 OK response back to the Share platform.
  - Since openAI keeps track of previous instructions and queries with the previously mentioned messages list, the new query is appended to this list and flagged as a question posed by the user.
  - Then a new request to the OpenAI API is created and the model to use (gpt-3.5-turbo) as well as the list of messages is passed as a parameter.

  - The response of the API may contain multiple different answers, therefore randomly one of the answers is chosen.
  - The response is also saved to the message list.
  - The four most relevant search results related to the user_query are retrieved from the extracted data from the Wiki. Afterwards this response is appended to the response of the OpenAI API.
  - This concatenated response is then formatted and the server sends it back to the Share Platform for display in the chatbot interface.
- **Error Handling:**
  - If the user_query is missing, the server responds with a 400 Bad Request error.
  - If any other exception occurs, the server responds with a 500 Internal Server Error, including the error message.

The above mentioned two other servers for demonstration purposes work essentially in the same way, however one of them is only calling the retrieval-based part and the other is only making the call to the OpenAI API. Therefore, the not needed modules in each of these are not included and they only consist of the basic functionality to make their respective calls either to the retrieval-based module or to the OpenAI API work. The codes for these other two servers are in the files server_chatbot_retrieval_based_only.py for the retrieval-based version and in server_chatbot_openai_only.py for the version using the openAI API.

9.1.2 Logic

As the chatbot is a combination of a retrieval-based approach and a generative one by using the OpenAI API, the logic for the generative part is outsourced to the API. However, the logic for the retrieval-based part of the chatbot had to be implemented. The logic for this can be found in the class CatrochatRetrievalBasedForHybrid which is using TF-IDF and cosine similarity to provide relevant responses to user queries.

When an instance of the class is initialized, the first step is to process the data scraped from the Wiki. The texts are extracted from the JSON file and then undergo preprocessing and tokenization. This involves several steps: converting the text to lowercase, removing stop words and filtering out non-alphabetic tokens. These preprocessing tasks are accomplished using the spaCy library, which is well-suited for natural language processing tasks.

Once the text has been preprocessed, it is converted into a TF-IDF matrix using the TfidfVectorizer from the sklearn library. TF-IDF stands for Term Frequency-Inverse Document Frequency and is a numerical representation that shows the importance of a word in a document relative to a collection of documents (the corpus).

By converting the texts from the scraped Wiki data into a TF-IDF matrix, we create a data structure that enables the efficient comparison between the user queries and the existing texts in the Wiki. This transformation allows the system to quantify the relevance of the texts in the knowledge base compared to the query of the user, which enables the accurate retrieval of information during the interactions with the chatbot.

When the server calls the answer_user function of the CatrochatRetrievalBasedForHybrid class with a user query, the program attempts to recognize the intent of the user. This function is relatively straightforward, it identifies the users intent based on predefined keywords such as greetings, requests for assistance or farewells. If one of these intents is recognized, the function returns nothing and the server will rely solely on the response from the OpenAI API. This is because the generative chatbot is more proficient at handling such conversational phrases.

If no specific intent is identified, the program proceeds to search for appropriate responses from the knowledge base. It does this by taking advantage of the previously preprocessed data from the JSON file.

Next, the user query undergoes the same preprocessing and conversion into a TF-IDF matrix. The two matrices, one for the knowledge base and one for the user query, are then compared by computing their cosine similarities. The similarity scores between the entries in the knowledge base and the user query are calculated and stored.

The top four scores, representing the most relevant responses, are obtained. The corresponding original texts from the knowledge base are retrieved using their indices. However, if all the similarity scores fall below a certain threshold, indicating that there is not a fitting response in the knowledge base, the function again returns nothing and the server will use the OpenAI API's answer.

If the scores are high enough, indicating a good match, the four best-fitting replies are formatted and returned for usage by the server. This method ensures that the server can provide accurate as well as contextually relevant responses when possible while still using the generative capabilities of the OpenAI API for more general conversational needs. This hybrid approach combines the strengths of both retrieval-based and generative models, enhancing the overall effectiveness and user experience of the chatbot.

In the following the functionality of the CatrochatRetrievalBasedForHybrid class is explained for each function.

1. **Initialization (__init__ method):**
    - **Loading the NLP Model:** The class initializes an instance of the spaCy model (en_core_web_md) for the natural language processing tasks.

- **Loading the Data:** It loads the structured data from the JSON file (scraper-data/wiki_data.json) containing the information scraped from the Catrobat Wiki.

- **Extracting the Text:** The extract_text method extracts the text data from the JSON, it iterates through all hierarchical levels of the JSON to get to the innermost entries tagged with 'text' and constructs URLs based on the hierarchical keys.

- **Preprocessing the Texts:** It preprocesses the extracted text data by tokenizing, converting to lowercase, removing stop words and filtering out non-alphabetic tokens using functionalities of the spaCy library.

- **TF-IDF Vectorization:** The TfidfVectorizer from sklearn is used to convert preprocessed texts into TF-IDF (Term Frequency-Inverse Document Frequency) matrix representation, which quantifies the importance of words in the texts relative to the entire corpus.

2. **Text Preprocessing (preprocess_text method):**
   - **Tokenization and Normalization:** Each text undergoes tokenization and normalization (lowercasing, stop word removal and filtering non-alphabetic tokens) to prepare it for further analysis.
3. **Calculating Similarity (calculate_similarity method):**
   - **TF-IDF Transformation:** Transforms a user query into a TF-IDF vector using the TfidfVectorizer.

   - **Cosine Similarity:** Computes cosine similarity scores between the TF-IDF matrix of the query and the TF-IDF matrix of the knowledge bases texts, to identify the most relevant texts based on the resulting similarity scores.
4. **Finding Most Similar Entry (find_most_similar_entry method):**
   - **Identifying the Best Match:** The document with the highest cosine similarity score to the user query is identified.

   - **Threshold Handling:** If the highest similarity score is below a specified threshold (0.15), it indicates no relevant match found and a corresponding message is returned.
5. **Finding Multiple Similar Entries (find_multiple_similar_entries method):**
   - **Top Four Matches:** Retrieves the top four matches based on the cosine similarity scores.

   - **Sorting:** Sorts the matches based on similarity scores in descending order and formats them for presentation.

6. **Intent Recognition (recognize_intent method):**
    - **Intent Classification:** Identifies the user intent based on predefined keywords such as greetings, requests for assistance or farewells.
    - **Response:** Returns the appropriate response based on the recognized intent.
7. **Answering User Queries (answer_user method):**
    - **Intent Response:** Checks if the user query matches any predefined intents (e.g. greetings, help requests) and provides no response if matched.
    - **Retrieval-Based Response:** If no intent is recognized, retrieves the most relevant document entry or entries using find_most_similar_entry and find_multiple_similar_entries.
    - **Response Formatting:** Constructs and formats the response with the retrieved information, including links to relevant Wiki entries and additional search results.

In summary, the CatrochatSimpleForHybrid class includes a retrieval-based chatbot tailored for the Catrobat Wiki, which uses NLP techniques and similarity measures (TF-IDF, cosine similarity) to deliver accurate and contextually relevant responses to user queries about Catrobat-related topics. The modular design of the chatbot enables efficient data retrieval, processing and response generation, with the goal in mind to enhance user interaction and satisfaction with the chatbot. For the server which is only acting as a retrieval-based chatbot there exists another class called CatrochatSimple. This works in a similar way with some minor differences. Here, if an intent like a greeting was found, it returns an appropriate predefined greeting. Also, the format is different if matching entries were found in the knowledge base. It will use the most matching result as the answer and only append the other less relevant texts as additional search results for the question.

9.1.3 Data

In the repository, there is a directory named scraper-data that contains the JSON files obtained by the scraper. These files serve as the knowledge base for our retrieval chatbot.

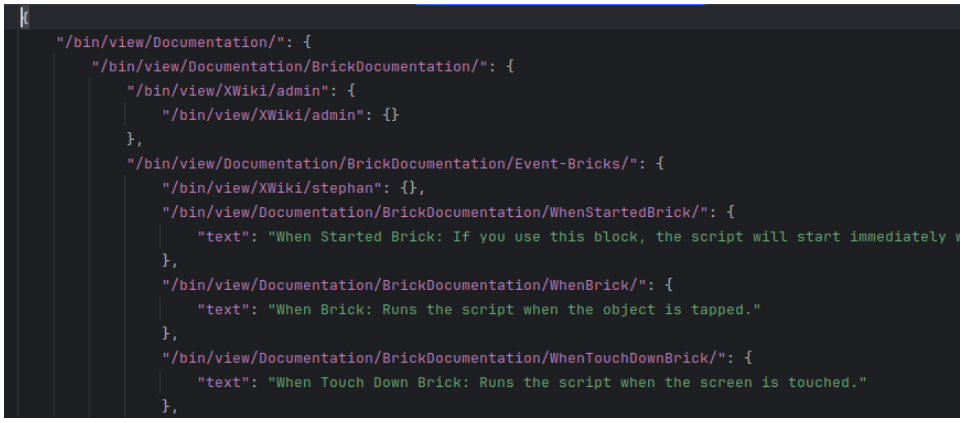

Figure 9.2: Screenshot of the JSON File containing the data scraped from the wiki.

The JSON file with the data scraped from the Wiki is structured as follows:

- **Root Element:** /bin/view/Documentation/ is the root in the Wiki where all the documentation for Catrobat can be found.
- **Next Level:** /bin/view/Documentation/BrickDocumentation/ which contains documentation about different bricks.
- **Further Levels:** /bin/view/Documentation/BrickDocumentation/..../ which contains documentation about different bricks.

The text field is the text found on the Wiki page describing a specific brick (e.g., WhenStartedBrick).
As an ecample the JSON for the "When Started Brick" is shown below:

```
{
    "/bin/view/Documentation/": {
        "/bin/view/Documentation/BrickDocumentation/": {
            "/bin/view/Documentation/BrickDocumentation/Event-Bricks/": {
                "/bin/view/Documentation/BrickDocumentation/WhenStartedBrick/": {
                    "text": "When Started Brick: If you use this block, ...)."
                }
            }
        }
    }
}
```

For the prototype, only the BrickDocumentation was initially considered. The scraper starts from /bin/view/Documentation and traverses to all the documentation sections, like Formula Editor, constructing the JSON in a similar hierarchical way.

By storing the data this way, we can always construct a link if we find the correct brick related to a user query, providing the user with a link for further information on the Wiki. Additionally, we have information about the categories this brick belongs to, such as EventBricks.

The logic part of the chatbot can then read the data from this JSON and continue to process and work with it. When using the retrieval-based chatbot, the text is the most relevant part of this JSON. The user's query and this text are compared using TF-IDF and cosine similarity to find the best match.

This way of structuring the data ensures that our chatbot can efficiently navigate and use the scraped Wiki data to provide accurate and helpful responses to user queries.

9.2 Changes on the Share Platform

Finally, we need to examine how exactly the Chatbot was integrated into the Share Platform.

- **Modifications to the frontend:** The GUI was designed to be as simple as possible and was inspired by well-known chatbots such as ChatGPT or Perplexity AI. In the assets/styles/custom/catrochat.scss file, specific styling rules were defined to enhance the visual presentation of the chatbot interface. These styles were adapted to align with the overall design of the Catrobat Share platform, ensuring a consistent and intuitive user experience. The webpack.config.js file played a crucial role in configuring the build process for the frontend JavaScript files. Here, the entry point "catrochat" was added and linked to the catrochat.js file. This setup ensured that all JavaScript components and functionalities related to the chatbot were properly bundled and optimized during the building process. By integrating "catrochat" as an entry, the build system effectively manages dependencies and enables the deployment of the frontend components within the Catrobat share platform environment.

  The catrochat.html.twig template served as the frontend interface for the Catrobat chatbot. This template extended the base template base.html.twig to inherit common styles and scripts used across the Catrobat share platform. It included essential links generated by Encore for the "catrochat" entry, ensuring that all necessary JavaScript and CSS files were properly included.

- **Modifications to the backend:** In the IndexController.php file, fundamental changes were implemented to handle the backend logic for the Catrobat chatbot. A new route "/catrochat/" was defined within the Symfony framework. It is configured to respond to both GET and POST requests. This route is associated with the catrochatView function, which serves as the entry point for processing the user queries and interacting with the Python-based chatbot servers. Inside catrochatView, a GuzzleHttp client was instantiated. This client was configured with the base URI pointing to the Python server's address (typically "172.17.0.1:8000" in a local setup), enabling the seamless communication between the Symfony application and the backend Python servers. When handling POST requests, the function retrieves user queries submitted from the frontend form and then packaged these queries into POST requests directed to "/answer_user" on the Python server. Error handling mechanisms were integrated to manage exceptions and unexpected responses from the Python server. This ensured that errors, such as connection failures or invalid responses, were appropriately handle. This helps to maintain the stability and reliability of the chatbot functionality within the Catrobat share platform.

The pull request with these changes can be found here:
https://github.com/Catrobat/Catroweb/compare/develop...bonbuo:Catroweb:catrochat?expand=1

10 The initial Prototype

As previously stated, the logic for the Chatbot is programmed in Python, in order to be able to quickly test and experiment with different NLP techniques etc. it was decided to first do a standalone chatbot which is completely coded in python, including its graphical user interface.

Similar to the actual prototype, this prototype consists of Data, Logic and a Graphical User Interface. For this first version of the prototype which was written solely in Python and with no connection to the share platform, the directory called scraper-data, which contains the JSON files that were generated by the scraper, is also needed. Therefore, the content of this directory serves as the knowledge base also for this retrieval chatbot. The scraper-data is the exact same for the initial prototype as well as for the final version explained above.

The GUI for this first prototype was done using the Tkinter library for Python. This GUI allows users to interact with the chatbot, which can respond using a simple TF-IDF and cosine similarity method. Also, in this case the GUI is based on popular conversational agents like Chat GPT.

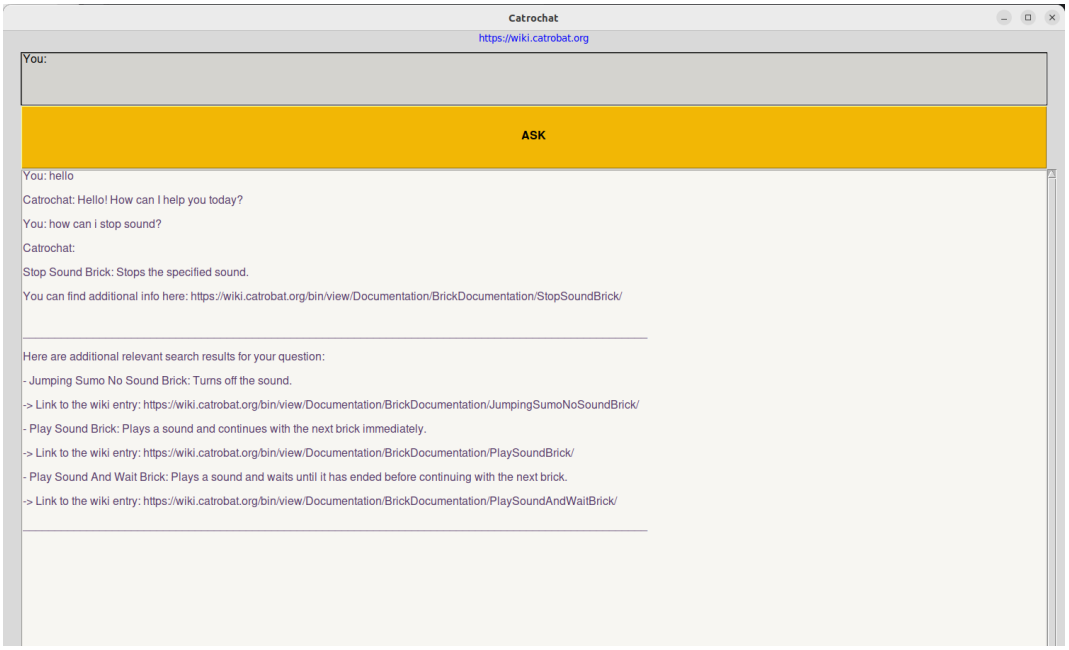

Figure 10.1: Screenshot of the GUI of the initial prototype solely in Python

GUI Elements

1. **Window:** The main application window, sized at 1500x900 pixels, serves as the container for all other GUI elements.

2. **UserQuery:** A text box where users can type their questions for the chatbot. It is initialized with a welcome message and a prompt for the user's input.

3. **SendButton:** A button that users can click to submit their queries to the chatbot.

4. **ChatWindow:** A text box that displays the entire conversation between the user and the chatbot. User queries are prefixed with "You:" and chatbot responses are prefixed with "Catrochat:".

5. **Wiki Link:** A clickable link to the Catrobat Wiki, placed at the top of the window for easy access to the Wiki itself.

6. **Scrollbar:** A scrollbar attached to the ChatWindow to allow users to scroll through the conversation.

This GUI effectively creates a user-friendly interface for interacting with the Catrochat chatbot. By separating the chatbot logic from the GUI and providing easy-to-use functions for query submission it ensures a smooth and intuitive user experience.

11 Future Work and Next Steps

The prototype that was developed for this thesis is working and fully functional, and can already be a great enhancement to the Wiki. However, as discussed in the chapters above, it can be improved to be even more engaging and capable of covering the whole range of information available on the Catrobat Wiki in the future.

First of all, the Scraper should be extended to acquire data from the entire Wiki, rather than just the Brick Documentation section. This would enable the retrieval-based part of the hybrid chatbot to retrieve more information beyond just the bricks that can be used in Pocket Code. Additionally, since the Wiki contains more than just textual information, both the scraper and the chatbot itself could be adapted to include videos available on the Wiki in its replies.

The retrieval-based version of the chatbot is currently using TF-IDF and cosine similarity to match the best responses to the user queries. However, there may be more advanced methodologies to further improve the accuracy of the chatbot. For example, BERT could be used, since it represents the state-of-the-art way to handle natural language processing. Furthermore, it would make sense to incorporate some form of gamification into the chatbot, in order to improve the engagement of the tool even more.

Furthermore, the chatbot should be tested with actual users to gather feedback in order to be able to improve the chatbot and to evaluate how it resonates with its target audience.

Finally, it should be investigated if there is a way for users to cover the expenses of the OpenAI API, rather than Catrobat providing an API key and potentially having to bear a lot of costs.

By putting in this future work, the chatbot can become an even more valuable addition to the Catrobat ecosystem.

12 Conclusion

This thesis was concerned with the development and the integration of a chatbot for Catrobat, aiming to enhance the user experience within the Catrobat Share Platform. The development began with a standalone Python-based prototype that was used for initial experimentation with natural language processing techniques and was able to already provide valuable insights. The prototype made use of a simple retrieval-based approach, which uses TF-IDF and cosine similarity to respond to user queries by retrieving data from the Catrobat Wiki. This early version of the chatbot served as a testing ground for understanding the user needs and refining the main functionalities.

The transformation from this initial prototype to the final implementation showed an increase in both scope as well as complexity. The final solution integrates the chatbot into the Catrobat Share Platform. It combines retrieval-based methods with generative ones powered by OpenAI's API. This hybrid approach improves the ability of the chatbot to provide more accurate and contextually relevant responses, by taking advantage of the strengths of both of these techniques.

Chatbots can significantly improve the user experience by providing immediate and interactive assistance and therefore enabling an intuitive access to information. They can offer different types of interactions depending on their design and application. Retrieval-based chatbots, like the initial prototype and also the one used in the final prototype, use predefined responses. They retrieve these responses from a structured knowledge base. These chatbots are effective for providing specific and factual information and are ideal for scenarios where responses can be directly mapped to data that is already existent.

Generative chatbots, in comparison to retrieval-based chatbots, make use of advanced language models to generate their responses dynamically based on the input they receive from the user. This makes it possible to have more flexible and conversational interactions, which makes them suitable for more complex queries that require a good understanding of the context. By integrating these generative capabilities, the final version of the Catrobat chatbot provides a more engaging user experience, is able to handle a wider range of queries and offers more detailed and human-like responses.

The decision to implement a hybrid approach was made because of the need to balance the precision of retrieval-based methods with the adaptability of generative responses. This combination ensures that users receive both accurate factual information and are also able to have meaningful conversational interactions. This

enables the chatbot to address a wider spectrum of different user needs.

The integration of the chatbot into the Catrobat Share Platform involved adjustments to both the frontend and backend of the Share Platform. The frontend was designed to fit in with the existing aesthetics of the platform, ensuring a consistent user experience. Backend modifications within the Symfony framework enabled the communication between the Share Platform and the Python server and therefore also incorporates error-handling mechanisms to maintain reliability.

Implementing a chatbot significantly improves the accessibility as well as the engagement of the Catrobat Wiki. Traditional Wikis, like the Catrobat Wiki itself, often require users to navigate through extensive menus or search for specific topics, which can be overwhelming, especially considering the younger audience of Catrobat. In contrast to Wikis, chatbots offer a conversational interface, that allows users to ask questions in natural language, which makes it easier to find the information they need.

Chatbots enhance the user engagement by providing an interactive and conversational assistance, which can lead to a deeper understanding of the content and might increase the motivation to explore topics even further. Additionally, chatbots provide an immediate 24/7 support, that helps users to resolve their issues in real-time and is able to maintain their engagement by preventing frustrations which come from delays.

This thesis shows an approach to develop a chatbot that enhances the user experience through a mix of retrieval-based and generative techniques. By integrating both of these methods, the thesis and the resulting chatbot not only advanced the capabilities of the Catrobat platform, but also demonstrated key principles of chatbot development. Furthermore, the integration of the chatbot significantly improves the accessibility and engagement of the Catrobat Wiki, making it more inclusive and user-friendly. The thesis also shows the future potential of the chatbot by making suggestions on how to improve it over time.

Bibliography

Algomo. (2024). *Why you absolutely need a multilingual chatbot*. https://www.algomo.com/blog/why-you-need-a-bot (cit. on p. 31).

Ali, F., Choy, D., Divaharan, S., Tay, H. Y., & Chen, W. (2023). Supporting self-directed learning and self-assessment using teachergaia, a generative ai chatbot application: Learning approaches and prompt engineering. *Learning: Research and Practice*, *9*(2), 135–147. https://doi.org/10.1080/23735082.2023.2258886 (cit. on p. 4).

Arz von Straussenburg, A., & Wolters, A. (2023). Towards hybrid architectures: Integrating large language models in informative chatbots. https://www.researchgate.net/publication/374420909_Towards_Hybrid_Architectures_Integrating_Large_Language_Models_in_Informative_Chatbots (cit. on pp. 28, 39).

Bani. (2024). *Differences between conversational ai and generative ai*. https://hellotars.com/blog/differences-between-conversational-ai-and-generative-ai (cit. on p. 26).

Bansal, G., Chamola, V., Hussain, A., Guizani, M., & Niyato, D. (2024). Transforming conversations with ai—a comprehensive study of chatgpt. *Cognitive Computation*, *16*, 1–24. https://doi.org/10.1007/s12559-023-10236-2 (cit. on p. 25).

Bawa, Y. (2024). *The importance of high quality training data in ai*. https://mindkosh.com/blog/the-importance-of-high-quality-training-data-in-ai/ (cit. on p. 17).

Bocian, Z. (2024). *Discover how chatbots in education transform learning experiences*. https://www.chatbot.com/blog/chatbot-for-education/ (cit. on pp. 4, 15, 16, 31).

BotPenguin. (2024). *Rule-based chatbot*. https://botpenguin.com/glossary/rule-based-chatbot (cit. on p. 21).

Braun, D., Hernandez Mendez, A., Matthes, F., & Langen, M. (2017). Evaluating natural language understanding services for conversational question answering systems. https://doi.org/10.18653/v1/W17-5522 (cit. on p. 35).

Burkhard, M., Seufert, S., Cetto, M., & Handschuh, S. Educational chatbots for collaborative learning: Results of a design experiment in a middle school. In: In *International conference on cognition and exploratory learning in digital age (celda)*. 2022. https://eric.ed.gov/?id=ED626892 (cit. on p. 15).

Caballé, M. (2023). *Rule-based chatbots vs. ai chatbots: Key differences*. https://www.hubtype.com/blog/rule-based-chatbots-vs-ai-chatbots (cit. on p. 13).

Catrobat. (2024a). https://catrobat.org/ (cit. on p. 1).

Catrobat. (2024b). https://catrobat.org/about/ (cit. on p. 5).

Catrobat-share. (2024). https://share.catrob.at/app/ (cit. on pp. 6, 7).

Catrobat-wiki. (2024a). https://wiki.catrobat.org/ (cit. on p. 1).

Catrobat-wiki. (2024b). https://wiki.catrobat.org/bin/view/AboutCatrobat/ (cit. on p. 6).

Cherniak, K. (2024). *Chatbots for education: How to overcome limitations and bridge the gap between technology and teaching*. https://masterofcode.com/blog/chatbots-for-education (cit. on p. 14).

Church, B. (2023). *5 types of chatbot and how to choose the right one for your business*. https://www.ibm.com/blog/chatbot-types/ (cit. on p. 20).

Cluelabs. (2023). *Chatbots as personalized learning aids: Opportunities and challenges*. https://cluelabs.com/blog/chatbots-as-personalized-learning-aids-opportunities-and-challenges/ (cit. on p. 17).

codecademy. (2024). *Rule-based chatbots*. https://www.codecademy.com/learn/rule-based-chatbots/modules/rule-based-chatbots/cheatsheet (cit. on p. 13).

Cunningham-Nelson, S., Boles, W., Trouton, L., & Margerison, E. (2019). A review of chatbots in education: Practical steps forward. In *30th annual conference for the australasian association for engineering education (aaee 2019): Educators becoming agents of change: Innovate, integrate, motivate* (pp. 299–306). Engineers Australia. https://eprints.qut.edu.au/134323/ (cit. on p. 32).

Dialzara. (2024). *Ai chatbot privacy: Data security best practices*. https://dialzara.com/blog/ai-chatbot-privacy-data-security-best-practices/ (cit. on p. 17).

Digital, B. (2024). *Rule-based vs. ai chatbots: Key differences*. https://borndigital.ai/rule-based-vs-ai-chatbots-key-differences/ (cit. on p. 20).

Duolingo. (2023). *Introducing duolingo max, a learning experience powered by gpt-4*. https://blog.duolingo.com/duolingo-max/ (cit. on p. 18).

Ellen, S., & Michaelidou, T. (2024). *A quick guide to chatbots for learning in the age of ai*. https://www.gpstrategies.com/blog/a-quick-guide-to-chatbots-for-learning-in-the-age-of-ai/ (cit. on p. 14).

Franceschi, C. (2020). *How to make a chatbot in less than 30 minutes*. https://joonbot.com/blog/guides/how-to-make-a-chatbot/ (cit. on p. 21).

Fryer, L. (2006). Bots as language learning tools. *Language, Learning and Technology*, *10*. https://www.researchgate.net/publication/233816040_Bots_as_Language_Learning_Tools (cit. on pp. 15, 32).

GeeksforGeeks. (2023). *Understanding tf-idf (term frequency-inverse document frequency)*. https://www.geeksforgeeks.org/understanding-tf-idf-term-frequency-inverse-document-frequency/ (cit. on p. 37).

Georgia-State-University. (2022). *Classroom chatbot improves student performance, study says classroom chatbot improves student performance, study says*. https://news.gsu.edu/2022/03/21/classroom-chatbot-improves-student-performance-study-says/ (cit. on p. 18).

Graesser, A. (2016). Conversations with autotutor help students learn. *International Journal of Artificial Intelligence in Education*, *26*. https://doi.org/10.1007/s40593-015-0086-4 (cit. on p. 14).

Han, S., & Lee, M. K. (2022). Faq chatbot and inclusive learning in massive open online courses. *Computers and Education*, *179*, 104395. https://doi.org/https://doi.org/10.1016/j.compedu.2021.104395 (cit. on p. 2).

helpshift. (2024). *Intent classification*. https://www.helpshift.com/glossary/intent-classification/ (cit. on p. 35).

Hetler, A. (2024). *Conversational ai vs. generative ai: What's the difference?* https://www.techtarget.com/whatis/feature/Conversational-AI-vs-generative-AI-Whats-the-difference (cit. on p. 25).

Hu, Y., Jacob, J., Parker, G., Hawkes, D., Hurst, J., & Stoyanov, D. (2020). The challenges of deploying artificial intelligence models in a rapidly evolving pandemic. *Nature Machine Intelligence*, *2*, 1–3. https://doi.org/10.1038/s42256-020-0185-2 (cit. on p. 26).

Innodata. (2024). *How do you source training data for generative ai?* https://innodata.com/how-do-you-source-training-data-for-generative-ai/ (cit. on p. 25).

Kalka, P. (2024). *Fundamentals of rule-based chatbots*. https://www.chatomiser.com/chatbot/rule-based (cit. on pp. 21, 22).

Karabiber, F. (2024a). *Cosine similarity*. https://www.learndatasci.com/glossary/cosine-similarity/ (cit. on p. 37).

Karabiber, F. (2024b). *Tf-idf — term frequency-inverse document frequency*. https://www.learndatasci.com/glossary/tf-idf-term-frequency-inverse-document-frequency/ (cit. on pp. 35, 36).

Kelliher, R. (2021). *Can — and should — chatbots help students navigate mental health crises?* https://www.diverseeducation.com/students/article/15281699/can-and-should-chatbots-help-students-navigate-mental-health-crises (cit. on p. 15).

Kuhail, M. A., Alturki, N., Alramlawi, S., & Alhejori, K. (2023). Interacting with educational chatbots: A systematic review. *Education and Information Technologies*, *28*. https://doi.org/10.1007/s10639-022-11177-3 (cit. on p. 13).

Laba, Y. (2024). *Retrieval-based chatbots for information extractions: Next-generation instrument for businesses*. https://intelliarts.com/blog/retrieval-based-chatbot-for-enterprises/ (cit. on p. 23).

Labadze, L., Grigolia, M., & Machaidze, L. (2023). Role of ai chatbots in education: Systematic literature review. *International Journal of Educational Technology in Higher Education*, *20*. https://doi.org/10.1186/s41239-023-00426-1 (cit. on pp. 13, 17, 31).

Lee, E. (2023). Implementing a smart campus chatbot using tf-idf. https://digitalcommons.bard.edu/cgi/viewcontent.cgi?article=1042&context=senproj_f2023 (cit. on pp. 23, 36–38).

Luber, S. (2024). *Was ist eliza?* https://www.bigdata-insider.de/was-ist-eliza-a-0615a5b88923975bed5af5c317fd1232/ (cit. on pp. 13, 20).

Luna, J. C. (2023). *What is bert? an intro to bert models*. https://www.datacamp.com/blog/what-is-bert-an-intro-to-bert-models (cit. on p. 34).

Miesle, P. (2023). *What is cosine similarity: A comprehensive guide*. https://www.datastax.com/guides/what-is-cosine-similarity (cit. on p. 23).

MIKGroup. (2023). *Tf-idf*. https://www.mikgroup.ch/wissen/tf-idf/ (cit. on p. 23).

Mnasri, M. (2019). Recent advances in conversational NLP : Towards the standardization of chatbot building. *CoRR*, *abs/1903.09025*. http://arxiv.org/abs/1903.09025 (cit. on p. 13).

National-Public-School-Vidyaranyapura. (2024). *The role of technology in modern education*. https://npsvrp.com/the-role-of-technology-in-modern-education/ (cit. on p. 1).

Neha. (2024). *Retrieval vs. generative chatbots: Best choice for your business in 2024*. https://yourgpt.ai/blog/growth/retrieval-vs-generative-chatbots-best-choice-for-your-business-in-2024 (cit. on p. 23).

NeuroSoph. (2024). *Conversational ai vs. traditional rule-based chatbots*. https://neurosoph.com/conversational-ai-traditional-rule-based-chatbots/ (cit. on p. 20).

Nicholson, C. (2024). *A beginner's guide to word2vec and neural word embeddings*. https://wiki.pathmind.com/word2vec (cit. on p. 34).

Nuruzzaman, M., & Hussain, O. (2018). A survey on chatbot implementation in customer service industry through deep neural networks, 54–61. https://doi.org/10.1109/ICEBE.2018.00019 (cit. on p. 21).

Opeton. (2024). *Maximizing language learning with a language learning chatbot*. https://www.opeton.co/blog/maximizing-language-learning-with-language-learning-chatbot (cit. on p. 16).

Pandey, S., & Sharma, S. (2023). A comparative study of retrieval-based and generative-based chatbots using deep learning and machine learning. *Healthcare Analytics*, *3*. https://doi.org/10.1016/j.health.2023.100198 (cit. on p. 23).

Perihan. (2024). *Chatbots for educational institutions- benefits, applications*. https://livechatai.com/blog/using-machine-learning-chatbot-in-educational-institutions (cit. on pp. 14, 32).

Petruk, M. (2023). *Rule-based vs ai chatbot: What is the difference?* https://wesoftyou.com/ai/rule-based-vs-ai-chatbot-what-is-the-difference/ (cit. on p. 21).

Pocs, M. (2020). *Lovecraft with natural language processing — part 4: Latent semantic analysis*. https://towardsdatascience.com/lovecraft-with-natural-language-processing-part-4-latent-semantic-analysis-70aa2fa2161b (cit. on p. 34).

Pradeep. (2023). *Understanding tf-idf in nlp: A comprehensive guide*. https://medium.com/@er.iit.pradeep09/understanding-tf-idf-in-nlp-a-comprehensive-guide-26707dbocec5 (cit. on p. 38).

Purdue-University. (2024). *How has technology changed education?* https://education.purdue.edu/2024/01/how-has-technology-changed-education/ (cit. on pp. 1, 13).

Ruan, S., Jiang, L., Xu, J., Tham, B. J.-K., Qiu, Z., Zhu, Y., Murnane, E. L., Brunskill, E., & Landay, J. A. (2019). Quizbot: A dialogue-based adaptive learning system for factual knowledge. *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, 357:1–357:13. https://doi.org/10.1145/3290605.3300587 (cit. on p. 18).

Sandeep. (2024). *Hybrid chatbots: Redefining the customer experience with ai and human touch*. https://fastbots.ai/blog/hybrid-chatbots-redefining-the-customer-experience-with-ai-and-human-touch (cit. on p. 28).

Sanz, E., Mariscal, G., Redondo Duarte, S., Jiménez-García, E., & REQUEJO, S. (2024). Ai-driven student assistance: Chatbots redefining university support. https://doi.org/10.21125/inted.2024.0221 (cit. on pp. 15, 16, 32).

Scratch. (2024). https://scratch.mit.edu/about (cit. on p. 5).

Seidor. (2023). *How chatbots and conversational interfaces are changing the way we use technology*. https://www.seidor.com/en-de/blog/how-chatbots-and-conversational-interfaces-are-changing-way-we-use-technology (cit. on p. 31).

Shridhar, K. (2017). *Generative model chatbots*. https://medium.com/botsupply/generative-model-chatbots-e422ab08461e (cit. on p. 25).

Shukla, D. (2024). *Transforming remote learning for edtech with generative ai*. https://www.gupshup.io/resources/blog/remote-learning-for-edtech-with-generative-ai (cit. on p. 27).

Singh, J., Joesph, M., & Abdul Jabbar, K. (2019). Rule-based chabot for student enquiries. *Journal of Physics: Conference Series*, *1228*, 012060. https://doi.org/10.1088/1742-6596/1228/1/012060 (cit. on p. 21).

Spair, R. (2024). *Revolutionizing conversations: The rise of generative ai chatbots*. https://medium.com/@rickspair/revolutionizing-conversations-the-rise-of-generative-ai-chatbots-d29d5cb115d2 (cit. on pp. 25, 26).

Swain, C. (2024). *Creating an effective chat and voice-enabled decision tree for improved customer service*. https://yellow.ai/blog/chatbot-decision-tree/ (cit. on p. 20).

Talkative. (2023). *The limitations of chatbots (and how to overcome them)*. https://gettalkative.com/info/limitations-of-chatbot (cit. on p. 24).

Taneja, K., Maiti, P., Kakar, S., Guruprasad, P., Rao, S., & Goel, A. K. (2018). Jill watson: A virtual teaching assistant for online education. https://dilab.gatech.edu/test/wp-content/uploads/2022/06/GoelPolepeddi-DedeRichardsSaxberg-JillWatson-2018.pdf (cit. on p. 14).

Tiffany. (2023). *Hybrid chatbot: Everything you need to know (2023 complete guide)*. https://www.chatinsight.ai/chatbots/hybrid-chatbots/#part5 (cit. on pp. 28, 29).

University-of-Florida. (2024). *Chatbots and artificial intelligence in education*. https://teach.ufl.edu/resource-library/chatbots-and-artificial-intelligence-in-education/ (cit. on p. 17).

Winkler, R., & Söllner, M. (2018). Unleashing the potential of chatbots in education: A state-of-the-art analysis. *Academy of Management Proceedings*, *2018*, 15903. https://doi.org/10.5465/AMBPP.2018.15903abstract (cit. on p. 15).

Xwiki. (2024). https://www.xwiki.org/xwiki/bin/view/Documentation/UserGuide/Features/Exports (cit. on p. 42).