UPTEC F 23016

Examensarbete 30 hp

Juni 2023

Question-answering chatbot for Northvolt IT Support

Daniel Hjelm

Civilingenjörsprogrammet i teknisk fysik

UPPSALA UNIVERSITET

Question-answering chatbot for Northvolt IT Support

Daniel Hjelm

Abstract

Northvolt is a Swedish battery manufacturing company that specializes in the production of sustainable lithium-ion batteries for electric vehicles and energy storage systems. Established in 2016, the company has experienced significant growth in recent years. This growth has presented a major challenge for the IT Support team, as they face a substantial volume of IT-related inquiries. To address this challenge and allow the IT Support team to concentrate on more complex support tasks, a question-answering chatbot has been implemented as part of this thesis project. The chatbot has been developed using the Microsoft Bot Framework and leverages Microsoft cloud services, specifically Azure Cognitive Services, to provide intelligent and cognitive capabilities for answering employee questions directly within Microsoft Teams. The chatbot has undergone testing by a diverse group of employees from various teams within the organization and was evaluated based on three key metrics: effectiveness (including accuracy, precision, and intent recognition rate), efficiency (including response time and scalability), and satisfaction. The test results indicate that the accuracy, precision, and intent recognition rate fall below the required thresholds for production readiness. However, these metrics can be improved by expanding the knowledge base of the bot. The chatbot demonstrates impressive efficiency in terms of response time and scalability, and its user-friendly nature contributes to a positive user experience. Users express high levels of satisfaction with their interactions with the bot, and the majority would recommend it to their colleagues, recognizing it as a valuable service solution that will benefit all employees at Northvolt in the future. Moving forward, the primary focus should be on expanding the knowledge base and effectively communicating the bot's purpose and scope to enhance effectiveness and satisfaction. Additionally, integrating the bot with advanced AI features, such as OpenAI's language models available within Microsoft's ecosystem, would elevate the bot to the next level.

Teknisk-naturvetenskapliga fakulteten

Uppsala universitet, Utgivningsort Uppsala

Handledare: Vladimir Kosilko Ämnesgranskare: Ping Wu

Examinator: Tomas Nyberg

Populärvetenskaplig sammanfattning

Northvolt är ett svenskt batteriföretag som specialiserar sig på att tillverka hållbara litiumjonbatterier för elfordon och energilagringssystem. Företagets mål är att utveckla en hållbar leveranskedja för batterier och att minska tillverkningsprocessens koldioxidavtryck. 2016 grundades företaget av Peter Carlsson, tidigare chef på Tesla, och har sedan dess vuxit till att bli en av Europas ledande batteritillverkare. Northvolt har snabbt ökat i storlek och har nu över 4000 anställda från över 120 länder vilket gör att företaget har en mångfaldig personalstyrka med unika perspektiv och färdigheter. Detta gör det möjligt för Northvolt att utveckla innovativa och hållbara batterilösningar för den snabbt växande marknaden. Samtidigt står företaget inför utmaningen att hantera ökande interna och externa förfrågningar. Ökningen påverkar särsklit IT-supportteamet som har svårt att hantera den höga arbetsbelastningen som kommer med den större mängden och ökade komplexiteten av ärenden vilket minskar produktiviteten och skapar frustration bland de anställda som behöver hjälp.

I detta examensarbete presenteras en implementation av en chattbot som kan svara på IT relaterade frågor för att hjälpa IT-supportteamet med att hantera arbetsbelastningen. En chattbot är ett datorprogram som använder artificiell intelligens (AI) för att förstå användarens meddelande, tolka dess betydelse och generera relevanta svar vilket kan användas för att simulera en konversation med användaren. Förutom att hjälpa till med att hantera arbetsbelastning så erbjuder en chattbot också omedelbara och konsekventa svar till användarnas frågor, tillgänglighet dygnet runt samt möjligheten att identifiera vanliga problem och trender genom att analysera användarnas förfrågningar. Chattboten är byggd med hjälp av Microsoft Bot Framework och använder Microsofts molntjänster, som till exempel Azure Cognitive Services för intelligens och kognition, för att besvara frågor från användaren direkt i Microsoft Teams.

För att utvärdera chattbotens prestanda används ett set av mått som mäter dess effektivitet, noggrannhet och användartillfredsställelse. Boten testades av en mångfaldig grupp av anställda från olika team genom att de fick chatta med boten och fylla i formulär där de utvärderade konversationen. Genom en sammanställning och analys av både svaren på formuläret och loggar av konversationerna visar det sig att chattboten endast kunde svara korrekt på 50% av användarnas frågor med en precision på 85%. Den stora anledning till chattboten presterar dåligt på dessa mått är framför allt att den inte har blivit tränad på tillräckligt mycket högkvalitiv data vilket gör att användarna frågar frågor som är utanför botens kunskapsbas. Boten presterar dock mycket högre i mått relaterade till tillfredsställelse där användarna tycker deras konversation med boten håller högre kvalitet än deras tidigare erfarenhet med chatbottar. Chatbotens effektivitet är också imponerande där den visar på en genomsnittlig svarstid på 0.81 sekunder vilket tillsammans med att det är lätt att chatta med roboten är de främsta anledningarna till att användarna tycker om att chatta med boten. Med en större mängd träningsdata samt att beskrivning av chattbotens syfte och omfattning tydligare kommuniceras till användarna kommer botens noggrannhet och precision öka vilket kommer göra den till en servicelösning som kommer att vara till nytta för alla anställda på Northvolt i framtiden.

Sammanfattningsvis visar detta examensarbete att en chattbot kan vara ett användbart verktyg för att hjälpa IT-supportteamet hantera arbetsbelastning och ge snabba svar på IT-relaterade frågor. Trots vissa utmaningar i form av noggrannhet och precision visade chattboten positiva resultat när det gäller användartillfredsställelse och effektivitet. Med fortsatt utveckling och förbättringar kan chattboten bli en värdefull resurs för Northvolt och underlätta kommunikationen mellan anställda och IT-support i framtiden.

Acknowledgements

I would like to express my sincere gratitude to the Tools & Product team, including my supervisor Vladimir Kosilko, at Northvolt for their support and guidance throughout the duration of this master thesis project. Their expertise, dedication, and willingness to share their knowledge have been instrumental in shaping the outcome of this project. I am grateful for the fruitful discussions, insightful feedback, and constructive suggestions provided by the team members, which significantly enhanced the quality of this work. I would like to specifically thank Vladimir for his constant support, encouragement, and continuous mentorship. His expertise, patience, and valuable insights have been instrumental in shaping the direction of this project. I am thankful for his guidance, prompt feedback, and the opportunities he provided for my personal and professional growth throughout this journey.

I would also like to extend my appreciation to Ping Wu, my subject reviewer, for dedicating his time and expertise to evaluating this thesis. His critical evaluation, constructive comments, and suggestions have greatly contributed to the refinement of this work. I am grateful for his valuable input, which has helped me in developing a deeper understanding of the subject matter and improving the overall quality of this thesis.

Lastly, I am profoundly grateful to my friends and family for their unwavering support during my five years of study and the completion of this thesis. Their constant encouragement, understanding, and belief in my abilities have been invaluable. To my friends, thank you for being there for me, offering motivation, and sharing both the challenges and successes of this academic journey. And to my family, I am deeply appreciative of your constant support and the sacrifices you have made. Having such amazing friends and family by my side has been the cornerstone of my academic pursuits. Thank you for being my pillars of strength throughout this entire journey.

Acronyms

AD Active Directory.

AI Artificial intelligence.

API Application programming interface.

BERT Bidirectional encoder representations from transformers.

CEFR Common European Framework of Reference for Languages.

CI/CD Continuous integration and delivery.

CLU Conversational Language Understanding.

CNN Convolutional neural networks.

CQA Custom Question Answering.

ETL Extracted, transformed and loaded.

GPT Generative pre-trained transformer.

IAM Identity and access management.

IDE Integrated development environment.

LUIS Language Understanding Intelligent Service.

MFA Multi-factor authentication.

ML Machine learning.

NLP Natural language processing.

NLU Natural language understanding.

PVA Power Virtual Agents.

REST Representational State Transfer.

RNN Recurrent neural network.

SDK Software development kit.

SSO Single sign-on.

ULR Universal Language Representation.

UX User experience.

List of Figures

List of Tables

Contents

1 Introduction

1.1 Background

Northvolt is a Swedish battery manufacturing company specializing in producing sustainable lithium-ion batteries for electric vehicles and energy storage systems. Northvolt is committed to developing a sustainable battery supply chain and reducing the carbon footprint of the manufacturing process. The company was founded in 2016 by Peter Carlsson, a former executive at Tesla, and has since grown to become one of Europe's leading battery manufacturers. During the last years, Northvolt has seen rapid growth in size and the number of employees, resulting in a diverse workforce of over 4000 employees from over 120 countries. This diversity brings unique perspectives and skills to the company, enabling Northvolt to innovate and develop sustainable battery solutions for a rapidly evolving market. However, the rapid growth in both numbers and diversity also presents a challenge in terms of managing inquiries from a diverse range of stakeholders. IT Support is one of the teams affected the most by this, where they have to address not only a higher quantity of queries but also a rise in the complexity of the queries. As a result, IT Support struggles to manage the huge demand in workload, leading to delays, frustration, reduced productivity, etc. To enable IT Support to focus on the crucial role of ensuring that Northvolt's technological infrastructure is running smoothly and efficiently, a chatbot can help them answer frequently asked questions related to IT. A chatbot is a computer program designed to simulate conversation with human users through text-based or voice-based communication channels. It is an artificial intelligence (AI) application that uses natural language processing (NLP) and various other techniques to understand user input, interpret its meaning, and generate responses in a conversational manner. The implementation of a chatbot as an IT Support tool offers several benefits. Firstly, chatbots can provide immediate assistance to users, helping to reduce the time required to resolve issues. Secondly, chatbots can offer assistance round-the-clock, even outside regular business hours, leading to shorter resolution times and increased user satisfaction. Moreover, chatbots can be used to identify common issues and trends through the analysis of user queries, allowing the company to proactively address these issues and reduce the number of support requests. A chatbot also provides consistent responses to user inquiries, ensuring that all users receive the same level of service. It is also cost-effective since it helps to reduce the workload of IT staff and the need for additional staffing or overtime. Lastly, Northvolt's commitment to sustainability aligns with the benefits of using a chatbot. By implementing a question-answering chatbot, Northvolt can reduce its carbon footprint by reducing the need for physical IT Support representatives. A chatbot can handle large volumes of queries with minimal energy usage, making it a sustainable and environmentally-friendly solution for IT.

1.2 Purpose and goals

The purpose of this thesis project is to implement a chatbot for IT Support at Northvolt, with the aim of enhancing the efficiency and effectiveness of IT support services provided to employees while reducing the workload of the IT Support team. The project includes a set of goals to the purpose that will guide its implementation and evaluation. The primary goal is to develop an intelligent and user-friendly chatbot that leverages advanced artificial intelligence techniques, such as natural language processing, to handle frequently asked questions, allowing the IT Support team to focus on more complex issues. Utilizing a comprehensive knowledge base encompassing frequently encountered issues and their corresponding solutions, the chatbot should provide accurate and relevant responses with high accuracy and precision to stakeholders from a range of technical areas. Additionally, the bot should offer high efficiency with short response times and be seamlessly integrated into Northvolt's existing IT infrastructure and systems, including ticketing systems and monitoring tools. Lastly, the chatbot will be designed for continuous improvement and learning, leveraging user feedback and monitoring interactions and logs. This will enable continuous updates to the functionalities offered by the chatbot as well as updating its knowledge base, fine-tuning its algorithms, and refining its conversational abilities, continually evolving the bot to provide increasingly proficient and reliable IT support services.

1.3 Tasks and scope

To achieve the purpose and the goals of the project, the following tasks are undertaken:

- Investigate existing chatbots, their applications and their implementation methods.
- Design and develop an AI-based chatbot that is able to provide accurate and relevant responses to questions.
- Devleop a system where feedback can be captured in order to continuously improve the bot.
- Create a pipeline for increasing the bot's knowledge base that is user-friendly and does not require in-depth knowledge of AI or machine learning.
- Implement a way to track unanswered questions to let the IT Support agents analyze them and possibly add them to the bot's knowledge base.
- Create a logging and monitoring service that can be used for analytical, diagnostic, and monitoring purposes.
- Use continuous integration and delivery (CI/CD) for the bot and its knowledge base to ease the possibilities of improving the bot.
- Integrate the bot in Microsoft Teams to enable every registered employee or consultant to chat with the bot.

The scope of the project is limited to the development and implementation of a question-answering chatbot and the specific tasks outlined above. The project aims not to replace the IT Support team but to augment their capabilities and reduce their workload by providing an efficient and sustainable service solution. In addition, the project does not include the development of other AI-powered tools or chatbots for other purposes. Finally, the project scope does not include enabling the bot to have access or allowing it to perform any changes to Northvolt's existing systems or infrastructure.

1.4 Outline of the thesis

In Chapter 2, a theoretical background to the thesis is presented, including a detailed explanation of natural language processing and chatbots. In Chapter 3, the Microsoft frameworks and services that are used in this project are presented together with Microsoft Project Turing. Chapter 4 presents the architecture of the bot and how the messages flow through the architecture. Chapter 5 presents the method used for testing the bot and the metrics used for evaluating the performance of the bot. The results of the user testing are presented and discussed in Chapter 6. Chapter 7 presents the conclusions of the project and outlines potential areas for future research and development related to the project.

2 Theory

This chapter describes the theory for key concepts relevant to the thesis. In Section 2.1, natural language processing (NLP) and understanding (NLU) as well as the transformer deep learning model are presented. In addition, an overview of chatbots is given in Section 2.2, explaining what a chatbot is, what types of chatbots there are and how chatbots can be evaluated.

2.1 Natural language processing

Natural language processing (NLP) is a branch of artificial intelligence that evolved from computational linguistics which focuses on using methods from numerous fields, including linguistics, data science and computer science, to help computers understand human language. In contrast to computational linguistics, which target is on the features of human language, NLP focuses on the use of statistical, machine learning and deep learning techniques to perform tasks and assignments, such as text summarization and topic modeling. For natural language processing to function, unstructured data must be transformed into a structured data format. This is accomplished by identifying named entities (known as named entity recognition) and word patterns using techniques like stemming, lemmatization, and tokenization [1].

The field of natural language processing can be traced back to the 1950s, when researchers, such as Alan Turing in his article "Computing Machinery and Intelligence" [2], first began exploring the idea of using computers to process and generate natural language. Since then numerous NLP algorithms have been developed and are used individually or together to support modern applications. For instance, Word2Vec [3], a neural network-based algorithm, learns distributed representations of words by analyzing large amounts of text data which can be used for anomaly detection and word similarity tasks.

2.1.1 Natural language understanding

Natural language understanding (NLU), a subfield of NLP, focuses on the development of algorithms and models that enable computers to comprehend and interpret human language at a deeper level. NLU goes beyond fundamental language processing tasks such as machine translation and word similarity to let machines understand the meaning of the text in context, infer relationships between things, and execute complicated reasoning tasks [4]. In NLU several stages are involved, such as syntactic and semantic analysis, which are essential for understanding the meaning and context of language. The syntactic analysis deals with the identification of grammatical structures in the text, such as parts of speech, phrases, and sentence structure. The semantic analysis, on the other hand, involves the identification of the meaning of words, phrases and sentences within their context [5]. Humans naturally do this in conversation, but for a machine to comprehend the intended meaning of various texts, these analyses must be combined.

2.1.2 Transformer (deep learning model)

A transformer is a deep learning model with a neural network architecture that was first introduced in 2017 by Vaswani et al. in the paper "Attention Is All You Need" [6] and has since become one of the most widely used architectures for both NLP and NLU tasks. The architecture of the transformer is specifically designed to capture long-range dependencies between tokens (an instance of a sequence of characters that are put together as a suitable semantic unit for processing in a certain document [7]) in a sequence, such as those that occur in natural language. In contrast to other sequence models, such as convolutional neural networks (CNNs) or recurrent neural networks (RNNs), the transformer architecture does not rely on sequential processing of the input tokens. Instead, it uses a combination of attention mechanisms and position-wise feedforward networks to process the input sequence.

In Figure 2.1 the architecture of the transformer is displayed. The transformer architecture consists of a stack of encoder and decoder layers. The encoder layers are responsible for processing the input sequence of text and generating a series of hidden representations that capture the meaning and syntax of the input. The decoder layers are responsible for generating the output sequence of text based on these representations.

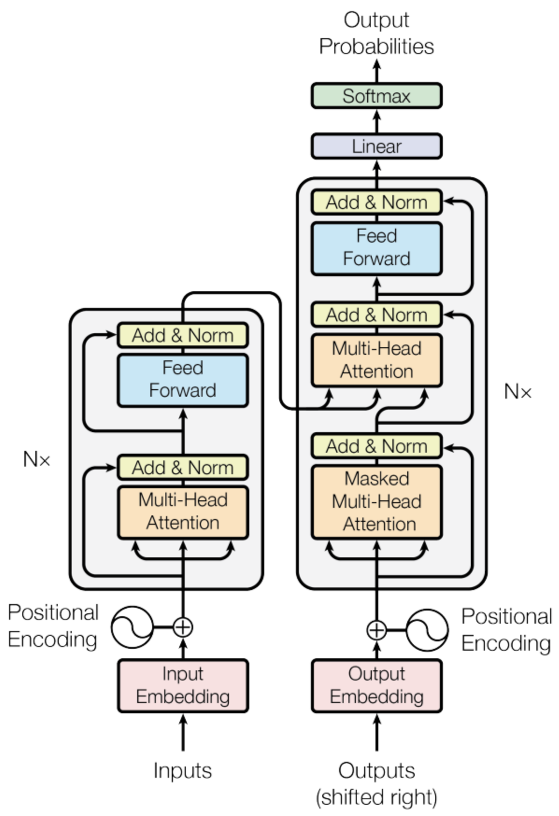


Figure 2.1: The transformer architecture using stacked fully connected, point-wise and self-attention layers for both the encoder and decoder, displayed in the left and right halves respectively [6].

The architecture of the encoder and decoder layers is based on the same structure, consisting of two sub-layers: a multi-head self-attention layer and a position-wise feedforward layer. The multi-head self-attention layer is the key component of the transformer architecture and is responsible for allowing the model to attend to different parts of the input sequence and weigh them differently based on their relevance to the current context. The self-attention mechanism works by computing a set of attention scores between each position in the input sequence and all other positions, based on the similarity of their hidden representations. These attention scores are then used to compute a weighted sum of the hidden representations, which produces a context vector that captures the most relevant information from the input sequence. The position-wise feedforward layer on the other hand applies a set of linear and non-linear transformations to each position in the sequence independently. This helps the model capture local interactions between adjacent tokens and learn more complex representations of the input sequence [6].

In addition to the two sub-layers, the transformer architecture also uses several other strategies that help improve its stability and performance during training. Firstly, residual connections are employed to solve the problem of vanishing gradients, which can occur when training deep neural networks [8]. The residual connection allows the model to learn an identity mapping, which can be added back to the output of the previous layer. This helps ensure that the gradients can flow back through the network, even when the network is very deep. Secondly, layer normalization is used to improve the stability of the network during training. It is used to ensure that the activation of each layer is normalized to have zero mean and unit variance, which can help to reduce the impact of covariate shift and improve the convergence of the network [9]. Lastly, dropout regularization is used to prevent overfitting and improve the generalization performance of the network [6]. Dropout randomly drops out a proportion of the activations in each layer during training, which can help to prevent the network from relying too heavily on any particular feature.

All in all, the transformer architecture is generally capable of capturing both short-range and long-range dependencies between tokens in the input sequence, making it well-suited for NLP and NLU tasks that call for comprehending context and meaning throughout a sequence of text. The BERT [10], GPT [11], and Transformer-XL [12] models, which have attained state-of-the-art performance on a wide range of NLP tasks, are just a few of the transformer-based models that have been developed as a result of the transformer architecture's success.

2.2 Chatbots

A chatbot, also known as a chatterbot or conversational agent, is a computer application that is designed to interpret and mimic human interaction (text or speech), enabling a human user to communicate with it as if it was a real human being. Chatbots utilize natural language to maintain a conversation with the user, understand the user's intent, and reply based on pre-defined rules or data it has been trained on. A chatbot can be as basic as a one-line program that responds to straightforward questions, or it can be as complex as a digital assistant that learns and develop over time to provide ever more individualized service as they acquire and process more data [13]. Based on the nature of the interaction between the user and the bot and the bot's scope and functionality, a chatbot can be classified into two main types: Task-oriented and Conversational.

2.2.1 Task-oriented chatbots

Task-oriented chatbots are currently the most popular chatbot and are designed as programs that handle one or a few purposes. They create purpose-focused responses in a conversational manner, using either rules, NLP or a combination of both. Think of strong, interactive FAQs when picturing interactions with task-oriented chatbots, which are extremely specialized, structured, and best suited for support and service activities. Answering common questions, like what the Wi-Fi password is, and simple tasks such as scheduling a meeting, are typical use cases for task-oriented chatbots. Although task-oriented chatbots usually are powered by advanced NLP models to enhance the conversational experience with the user, their functionality is usually limited.

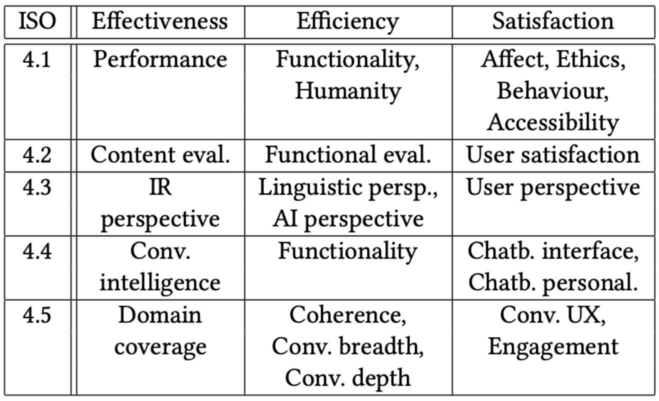
2.2.2 Conversational chatbots

Commonly referred to as digital or virtual assistants, conversational chatbots are designed to be more personalized, interactive and smart than task-oriented chatbots. Conversational bots utilize NLP and NLU to understand the context of the conversation and learn new things directly in production. For instance, a virtual assistant can offer the user personalized content, such as recommendations and mood-based answers, based on what it has learned from previous conversations. Siri, Apple's conversational bot which is integrated into the majority of Apple's flagship products, is a perfect example of a data-driven and predictive conversational chatbot [14].

2.2.3 Evaluation of chatbots

Since 1966, when the first natural-language chatbot Eliza [15] was released, conversational agents have continuously improved. In recent years, there has been a significant increase in chatbot-related publications and new use cases for chatbots have emerged. However, the evaluation methods for chatbots have been falling a bit behind the rapid evolution and are today part of active research. In the paper "Evaluating Quality of Chatbots and Intelligent Conversational Agents", N. Radziwill and M. Benton review methods for evaluating chatbots from 42 papers published between 1995 and 2017 [16]. The authors conclude that evaluation methods for chatbots can be categorized into three groups; effectiveness, efficiency and satisfaction, which align with the usability concept presented in ISO 9214 [17]. In terms of chatbots, effectiveness-based methods measure the bot's ability to effectively meet the needs of users and achieve their business goals. Effectiveness can be measured by for example analyzing the accuracy and completeness of the bot's answer. Efficiency-based methods, on the other hand, refer to the evaluation of chatbots based on their ability to complete tasks efficiently and with minimal user effort. They evaluate the speed and resource utilization of the chatbot in responding to user queries and are often used in bots designed for customer service or support, where users expect quick and efficient solutions to their problems. Satisfaction-based methods refer to the evaluation of chatbots based on user satisfaction with the chatbot's performance, rather than just its ability to complete tasks effectively and efficiently. Surveying is a perfect example of a satisfaction-based method where you measure user satisfaction with a series of questions. The Godspeed Questionnaire Series [18] and Subjective Assessment of Speech System Interfaces (SASSI) [19] are examples of frameworks developed for evaluating human-bot interactions that use surveying. In addition to surveys, user satisfaction can also be collected through channels such as chat, email or social media or automatically through the bot by enabling the bot to capture feedback through for instance a rating system.

In the article ”Trends & Methods in Chatbot Evaluation” J. Casas et al. provide an overview, based on literature from 2016 to 2020 including N. Radziwill and M. Benton’s paper, of the current trends and methods used in the evaluation of chatbots [20]. The authors present a similar approach to N. Radziwill and M. Benton’s where they relate modern evaluation methods to the three keystones of the ISO 9214 concept of usability. In Figure 2.2 the relation of evaluation methods and ISO 9214 is presented, where we can see that metrics such as conversational intelligence and performance can be related to effectiveness whilst user experience and chatbot interface are more related to satisfaction.



ISO	Effectiveness	Efficiency	Satisfaction
4.1	Performance	Functionality, Humanity	Affect, Ethics, Behaviour, Accessibility
4.2	Content eval.	Functional eval.	User satisfaction
4.3	IR perspective	Linguistic persp., AI perspective	User perspective
4.4	Conv. intelligence	Functionality	Chatb. interface, Chatb. personal.
4.5	Domain coverage	Coherence, Conv. breadth, Conv. depth	Conv. UX, Engagement

Figure 2.2: Table displaying chatbot evaluation methods in relation to ISO 9214 [20].

In addition to the overview of evaluation methods, the authors conclude that there is not a single established evaluation method for chatbots but that there are trends that efficiency and satisfaction-based methods are preferred. Satisfaction-based and efficiency-based are not only being most popular single-method evaluation methods, but they are also the most widely used pairing of methods [20].

3 Microsoft Project Turing, services and frameworks

This chapter describes the Microsoft Project Turing, services and frameworks that are related to and used in this project. A brief overview of Microsoft Project Turing is presented in Section 3.1 which is the project that has developed the language models that Azure Cognitive Services (Section 3.2) and LUIS (Section 3.3) are powered by. In Section 3.4 Azure Bot Service and its components Microsoft Bot Framework (3.4.1), Bot Framework Composer (3.4.2) and Bot Framework Emulator (3.4.3), are presented, which all are used to build, test and publish the bot. In the last sections, several Azure services used by the bot, such as Azure Active Directory (Section 3.5) and Azure DevOps (Section 3.8), are presented.

3.1 Microsoft Project Turing

Microsoft Project Turing is an initiative focusing on developing state-of-the-art and large-scale models to address complex business challenges in Microsoft, including Office, Bing and Xbox [21]. The goal of the project is to extend the boundaries of areas such as computer vision, natural language understanding and transfer learning. In the past years, the team behind Project Turing has published several groundbreaking state-of-the-art NLP models. One example is the transformer-based language generation model Megatron-Turing NLG 530B. As the name suggests, the model has 530 billion parameters, which is 3 times more parameters than GPT-3 [11] and makes it one of the largest and most powerful language models ever developed. The model was trained on a massive corpus of text data, including books, web pages and other sources of text which has resulted in one of the model's key abilities which is to generate high-quality, coherent, semantically consistent long-form content, such as articles or reports. In addition, the model is capable of translating text between multiple languages, summarizing long-form text into shorter and generating natural-sounding responses to text which can be used in for example chatbots [22].

Another example of Project Turing's successful implementation of models is the set of language models called Turing Universal Language Representation (T-ULR). The family is a result of collaboration between the Microsoft Turing team and Microsoft Research and consists of several state-of-the-art models that aim to learn universal language representation. T-ULRv6 is the latest addition to the family and it demonstrates that a single multilingual model can reach state-of-the-art skills in both English and multilingual understanding challenges by taking the top spot on both the Google XTREME [23] and GLUE [24] leaderboards [25].

3.2 Azure Cognitive Services

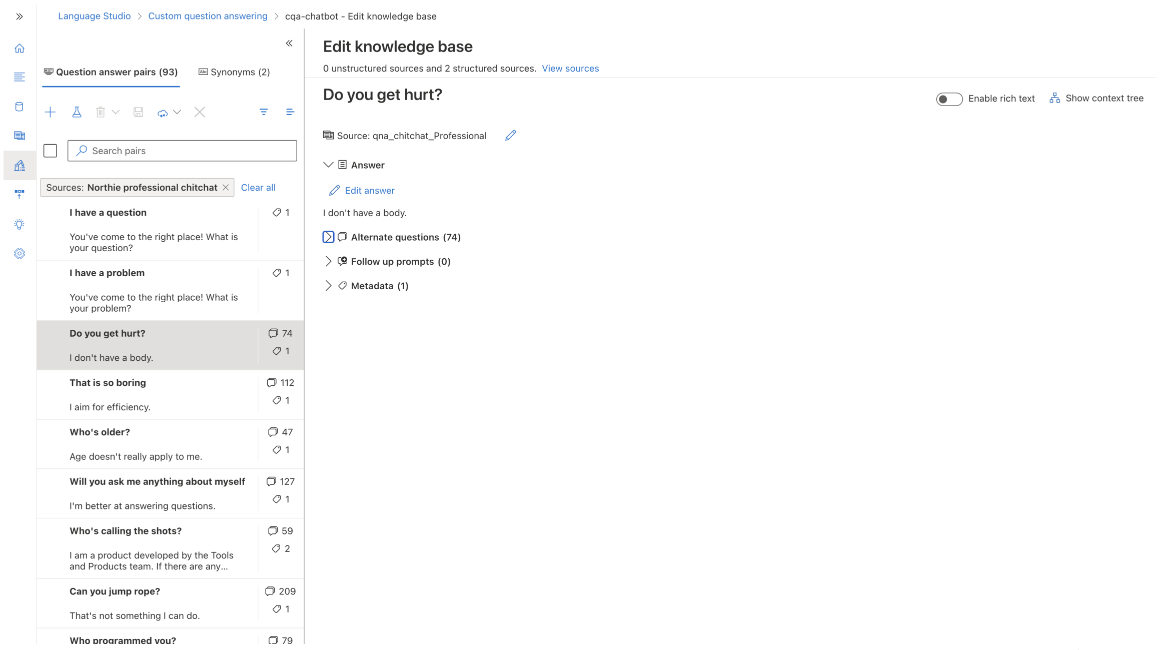
Azure Cognitive Services is a cloud-based suite of artificial intelligence and machine learning services provided by Microsoft Azure [26]. Cognitive Services, powered by models developed by Project Turing 3.1, makes state-of-art AI accessible to users with all levels of AI knowledge. Included in the suite is models with the ability to see, hear, speak, search, understand, and accelerate advanced decision-making which can be integrated into a business application with a simple API call.

3.2.1 Cognitive Service for Language

Cognitive Service for Language is Azure Cognitive Services' managed service to add natural language capabilities to applications. The service offers customizable models for tasks ranging from identifying key terms in text to sentiment analysis and language detection. The models can be built and tested directly using UI-based tools on the platform Language Studio before they are deployed to production environments [27].

3.2.1.1 Custom Question Answering

The Custom Question Answering (CQA) feature of Azure Cognitive Service for Language lets the user create a question-and-answer-styled conversational layer from data on top of a natural language model. The knowledge base of the CQA model is built from custom semi-structured content, such as documents and FAQs, plain unstructured documents or manually added questions and answers. The model then uses the knowledge base to set a confidence score between 0 and 1 for a question asked to the model and selects the answer with the highest score as a response to the question. The conversation can also be configured in a multi-turn fashioned way where the model asks for additional input in order to increase the accuracy in finding the answer the user is seeking for. Moreover, alternate questions can be added to questions to increase the confidence score for certain topics. The quality of the knowledge base can also be improved by adding synonyms or by using a feedback loop through active learning. Active learning generates alternate questions based on the users' interactions with the model which can be reviewed and added to the knowledge base [28].


Language Studio > Custom question answering > cqa-chatbot - Edit knowledge base

Question answer pairs (93) | Synonyms (2)

Search pairs

Sources: Northie professional chitchat × Clear all

I have a question	You've come to the right place! What is your question?	1
I have a problem	You've come to the right place! What is your problem?	1
Do you get hurt?	I don't have a body.	74 / 1
That is so boring	I aim for efficiency.	112 / 1
Who's older?	Age doesn't really apply to me.	47 / 1
Will you ask me anything about myself	I'm better at answering questions.	127 / 1
Who's calling the shots?	I am a product developed by the Tools and Products team. If there are any...	59 / 2
Can you jump rope?	That's not something I can do.	209 / 1
Who programmed you?		79

Edit knowledge base

0 unstructured sources and 2 structured sources. View sources

Do you get hurt?

Enable rich text | Show context tree

Source: qna_chitchat_Professional

Answer

Edit answer

I don't have a body.

Alternate questions (74)

Follow up prompts (0)

Metadata (1)

Figure 3.1: Language Studio UI for managing the questions and answers for the Custom Question Answering language model.

3.2.1.2 Language Detection

Azure Language Detection is a service part of the Azure Cognitive Services suite, which uses machine learning algorithms to automatically identify the language of a given text. Language Detection can recognize a wide range of languages and scripts, including commonly used ones such as English, Spanish, French, and Chinese, as well as less widely spoken languages such as Yoruba, Xhosa, and Amharic. The service can detect languages in various types of text data, including web pages, documents, social media posts, and more [29].

3.3 Language Understanding Intelligent Service (LUIS)

Azure Language Understanding Intelligent Service (LUIS) is a cloud-based natural language service provided by Microsoft Azure. It enables developers to build custom language understanding models for various applications, including chatbots and voice assistants. LUIS uses machine learning algorithms to understand the intent of the user's input, and then extracts relevant entities from the input, such as dates, times, and locations. It then provides a prediction of the user's intent and the corresponding entities, which can be used to trigger the appropriate action in the application. To create a LUIS language model, developers need to provide sample utterances and the corresponding intents and entities they represent. The LUIS service then uses these samples to train a language model that can accurately recognize user intents and entities in real-time [30]. LUIS is the precursor to Azure's Conversational Language Understanding (CLU) service and is therefore a stand-alone product that is not included in Azure Cognitive Services [31].

3.4 Azure Bot Service

Azure Bot Service is a cloud-based platform for building, launching and maintaining bots. With Azure Bot Service, developers can create bots in numerous programming languages, including Javascript and C#, and frameworks such as Power Virtual Agents and Bot Framework SDK. The platform offers features such as machine learning, NLP and scalable hosting environments. Azure Bot Service is also easily integrated into other services in the Azure eco-system such as Azure KeyVault to manage keys and secrets, Application Insights for logging and Azure DevOps for continuous integration and delivery. In addition, the platform offers numerous channels to connect the bot to including Slack, Facebook Messenger, and Microsoft Teams. Last but not least, Azure Bot Service offers tools for debugging, monitoring and testing the bots, helping the user to ensure the development and production versions of the bots are operating efficiently [32].

3.4.1 Microsoft Bot Framework

Microsoft Bot Framework is a collection of tools and libraries for building and deploying chatbots. In the framework, the flexible and extendable software development kit Bot Framework SDK is included, enabling developers to efficiently build bots that understand natural language, use speech and much more [33].

3.4.2 Bot Framework Composer

Built on the Bot Framework SDK, Bot Framework Composer is an open-source integrated development environment (IDE) that enables users to create, test and deploy conversational agents. Composer offers both full-code and low-code development experiences where developers can create dialog experiences by writing code in C# and JavaScript or using the built-in visual authoring canvas [34].

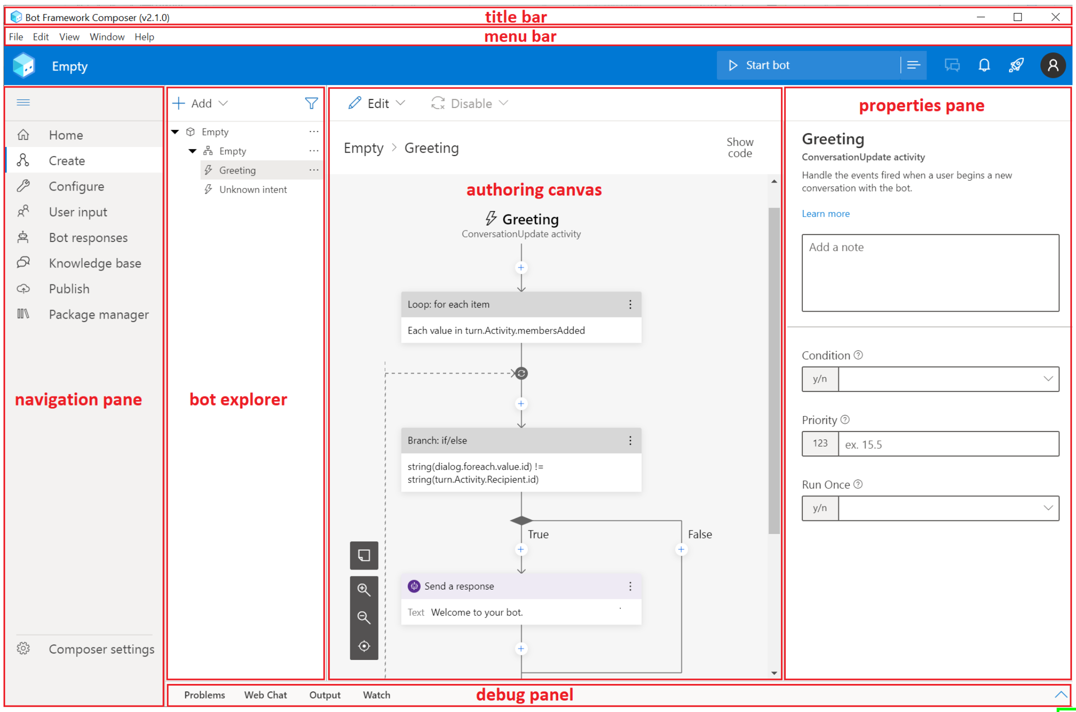
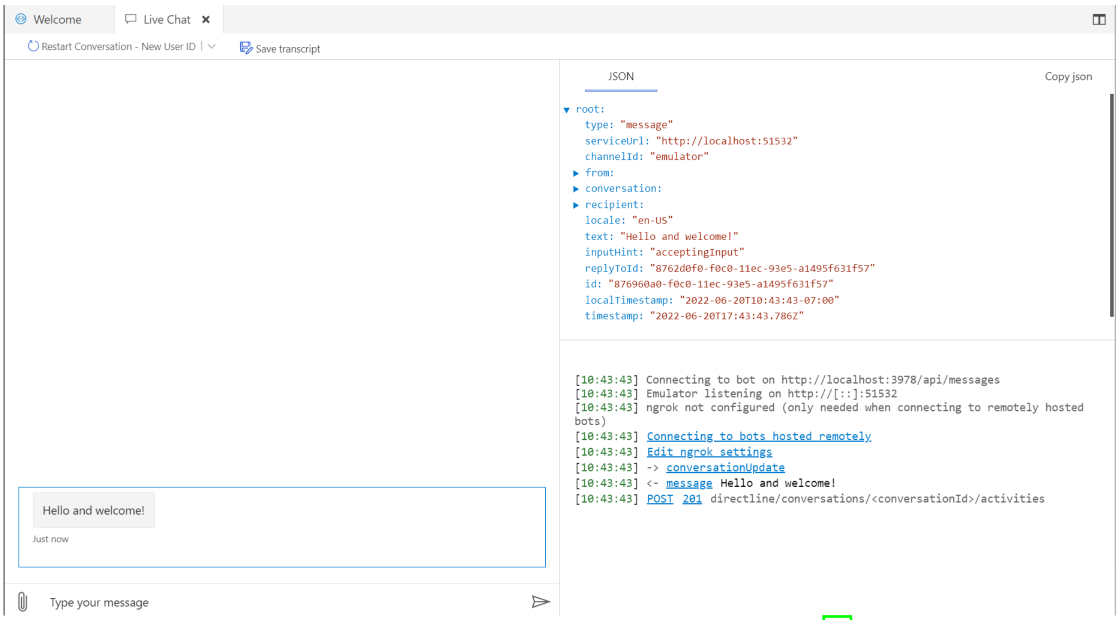


Figure 3.2: Overview of the low-code built-in visual authoring canvas in Bot Framework Composer [34].

3.4.3 Bot Framework Emulator

Before deploying to the cloud and the organization, developers usually want to run, debug and test their programs locally. This can be done for Microsoft Bot Framework conversational agents in the Bot Framework Emulator desktop application. In Emulator, messages are displayed as they would appear in any UI for web chatting and it is easy to inspect logs for all the requests and responses that are included in the bot [35].



```
▼ root:
    type: "message"
    serviceUrl: "http://localhost:51532"
    channelId: "emulator"
  ▶ from:
  ▶ conversation:
  ▶ recipient:
    locale: "en-US"
    text: "Hello and welcome!"
    inputHint: "acceptingInput"
    replyToId: "8762d0f0-f0c0-11ec-93e5-a1495f631f57"
    id: "876960a0-f0c0-11ec-93e5-a1495f631f57"
    localTimestamp: "2022-06-20T10:43:43-07:00"
    timestamp: "2022-06-20T17:43:43.786Z"
```

```
[10:43:43] Connecting to bot on http://localhost:3978/api/messages
[10:43:43] Emulator listening on http://[::]:51532
[10:43:43] ngrok not configured (only needed when connecting to remotely hosted
bots)
[10:43:43] Connecting to bots hosted remotely
[10:43:43] Edit ngrok settings
[10:43:43] -> conversationUpdate
[10:43:43] <- message Hello and welcome!
[10:43:43] POST 201 directline/conversations/<conversationId>/activities
```

Figure 3.3: Bot Framework Emulator user interface [35].

3.5 Azure Active Directory

Azure Active Directory, in short Azure AD, is a cloud-based identity and access management (IAM) service that provides a set of features for managing user identities and access to resources. It allows organizations to manage and control user access to various resources such as applications, data, and other services both within and outside their organization. In Azure AD, features such as single sign-on (SSO), multi-factor authentication (MFA), device management, application management, and security monitoring are available to ensure secure access to resources [36].

3.6 Azure Key Vault

Azure Key Vault is a cloud-based service provided by Microsoft that enables a secure solution for storing and managing cryptographic keys, secrets, and certificates used for securing data and applications. The service provides a secure and centralized location for storing sensitive information, such as passwords, connection strings, and API keys, which are often used by applications and services to authenticate with other resources. It supports multiple key types, including asymmetric and symmetric keys, and can be used to manage the lifecycle of keys, including creation, rotation, and revocation [37].

3.7 Azure Storage

Azure Storage is a cloud-based storage service provided by Microsoft that enables customers to store and manage large amounts of data in the cloud. In Azure Storage a wide range of different storage options is supported, including blob storage for unstructured data such as images, videos, and documents, file storage for sharing files and queue storage for messaging between application components. The service also provides high availability and durability of data through replication and redundancy. Data can be managed efficiently and securely in Azure Storage thanks to the collection of data management features available, such as lifecycle management, tiered storage, and encryption. Azure Storage can be accessed in numerous ways, including REST API, SDKs and integrations with other Azure services [38].

3.8 Azure DevOps

Azure DevOps is a cloud-based, unified and integrated collaboration platform provided by Microsoft that enables teams to work together more efficiently and effectively throughout the software development lifecycle. It consists of a suite of services and tools that help teams to plan, develop, test, and deploy software applications and services in a more efficient and effective way. One of many key services in Azure DevOps is Azure Pipelines which is a CI/CD system that enables teams to build, test, and deploy their applications and services to any platform or cloud. Azure DevOps integrates seamlessly with other services available in Azure, such as Azure Application Insights, providing end-to-end insights into the health and performance of applications [39].

3.9 Azure Application Insights

Azure Application Insights is a cloud-based application performance monitoring service offered as an extension of Azure Monitor [40]. It helps developers and teams to detect and diagnose issues in their applications and services, as well as monitor the overall performance and usage. With Application Insights, it is easy to track and analyze various metrics and telemetry data, such as request rates and response times, server and client exceptions, dependencies and logs, and custom events and traces. Alerts and notifications are also available which can be triggered based on specific conditions, such as thresholds or anomalies, and integrate with other Azure services such as Azure DevOps. Additionally, Application Insights provides several features and tools to help visualize and explore data, such as dashboards, queries, and custom charts. It can also be used to perform advanced analytics and machine learning scenarios, such as predictive maintenance and anomaly detection [41].

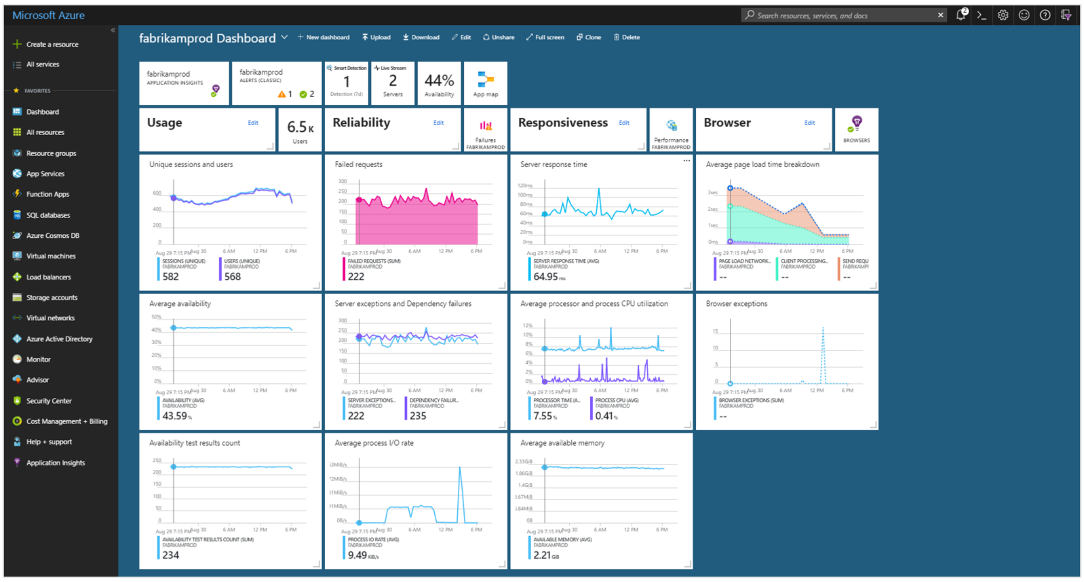


Figure 3.4: Example of a dashboard in Application Insights where for instance failed requests and server response time are displayed [41].

3.10 Azure Logic Apps

Azure Logic Apps is a cloud-based service provided by Microsoft that allows developers to create and run workflow-style applications. In Logic Apps, developers can build apps that automate business processes and integrate disparate systems and services across different platforms, both within and outside of Azure. A visual designer is used to create and configure the workflows using a drag-and-drop interface, without the need for extensive coding. A wide range of pre-built connectors is available, enabling easy integration with various services, such as Azure Storage and Office 365. Each app can also be defined to run manually, on a schedule or triggered by pre-built or custom-built triggers. Overall, Logic Apps is a scalable, reliable, flexible and secure tool that can help organizations streamline workflows which improves productivity and efficiency [42].

3.11 Azure App Service

Azure App Service is a comprehensive and fully managed platform offered by Microsoft Azure that facilitates the development, deployment, and scalability of web applications and APIs. It provides support for a wide range of programming languages and frameworks, alleviating the burden of infrastructure management. With Azure App Service, developers can effortlessly host web applications, create robust APIs, establish mobile backends, automate workflows, and execute serverless functions. The platform seamlessly integrates with other Azure services, offering automatic scaling, continuous deployment, and robust monitoring capabilities. Azure App Service streamlines the deployment and management processes, empowering developers to concentrate on application development and delivery, without being overloaded by infrastructure concerns [43].

4 Implementation

This chapter describes the implementation of the bot. A detailed explanation of the architecture of the bot is presented in Section 4.1 followed by a deep dive into the message flow in Section 4.2.

4.1 System architecture

The architecture of the chatbot is illustrated in Figure 4.1. It consists of the building blocks:

- Bot logic and UX
- Security and governance
- Storage, logging and monitoring
- Bot cognition and intelligence
- Quality assurance and enhancements
- Data and data ETL

which are explained in more detail in the following subsections, respectively.

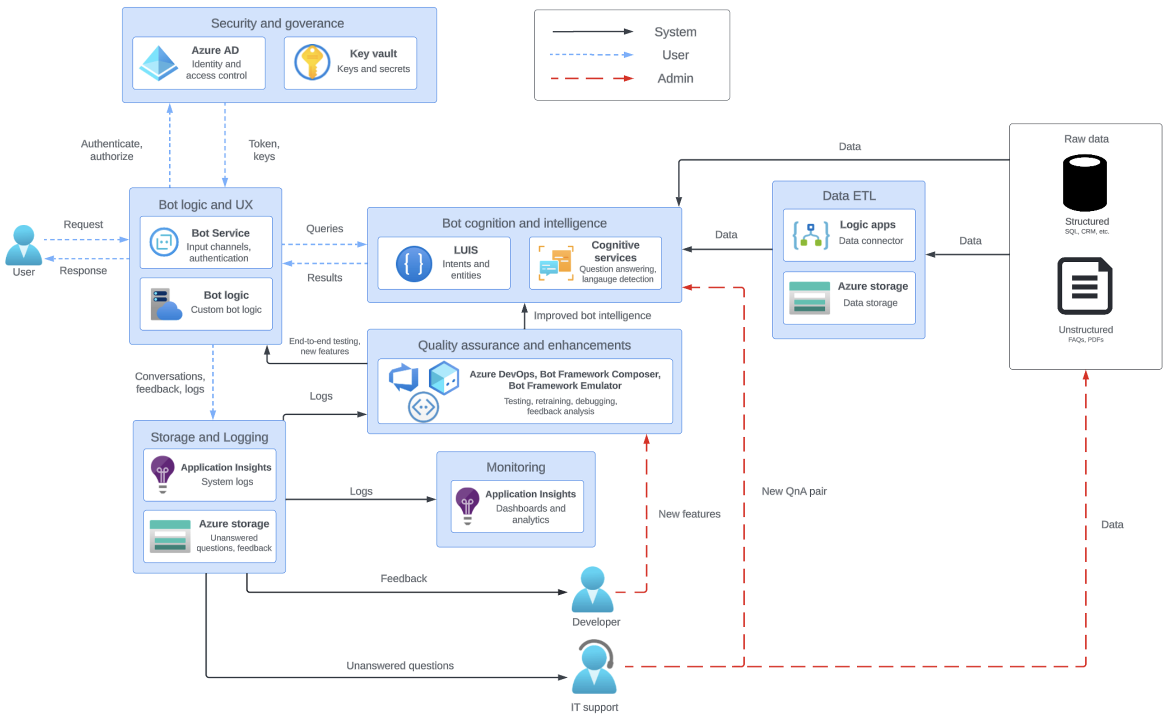


Figure 4.1: The architecture of the chatbot. The color and shape of the arrows represent if the data flow was initiated by the system, user or an admin of the system.

4.1.1 Bot logic and UX

The bot logic and user experience (UX) part of the architecture is responsible for facilitating the communication between the bot and its user. Azure Bot Service connects the bot logic hosted in Azure App Service to the communication app Microsoft Teams where the user can chat with the bot.

4.1.2 Security and governance

Azure Active Directory and Azure Key Vault handle the security and governance of the bot. An Azure AD application is registered for the bot which handles the bot identity and registers the bot as a resource in Azure. This enables the bot to access secured resources inside Azure, such as Microsoft Graph, which is used to for instance fetch the user information needed in the conversation. Azure Key Vault stores secrets and keys used in the bot such as keys for the bot intelligence and storage resources.

4.1.3 Storage, logging and monitoring

Application Insights and Azure Storage handle the logging and storage of the logs. In Application Insights, system logs such as requests and their status, response time, etc are logged for analytical, diagnostic, and monitoring purposes. In Azure Storage, written feedback to the bot and unanswered questions are stored. These are stored so the developer can get feedback for new features of the bot and IT Support can track unanswered questions which can be added to the knowledge base of the bot or to other company data sources.

4.1.4 Bot cognition and intelligence

The bot's intelligence and cognition are handled by the LUIS, CQA and language detection models. LUIS allows the bot to identify the intents of the conversation with the user which is used for example when the user wants to provide feedback or stop the conversation. The Custom Question Answering model, on the other hand, is responsible for answering questions and handling chitchat conversations with the user. The CQA knowledge base is built using a professional chitchat dataset [44] and question-and-answer pairs provided by the IT Support team and their internal documentation. The language detection model plays a crucial role in handling non-English user input. This is vital because the LUIS and CQA models are specifically trained to process English text.

4.1.5 Quality assurance and enhancements

To assure the quality and continuous enhancement of the bot, Azure DevOps, Bot Framework Composer and Bot Framework Emulator are used. Azure DevOps enables a pipeline for continuous integration and delivery of the bot while Bot Framework Composer is used for building the bot and Bot Framework Emulator for local testing.

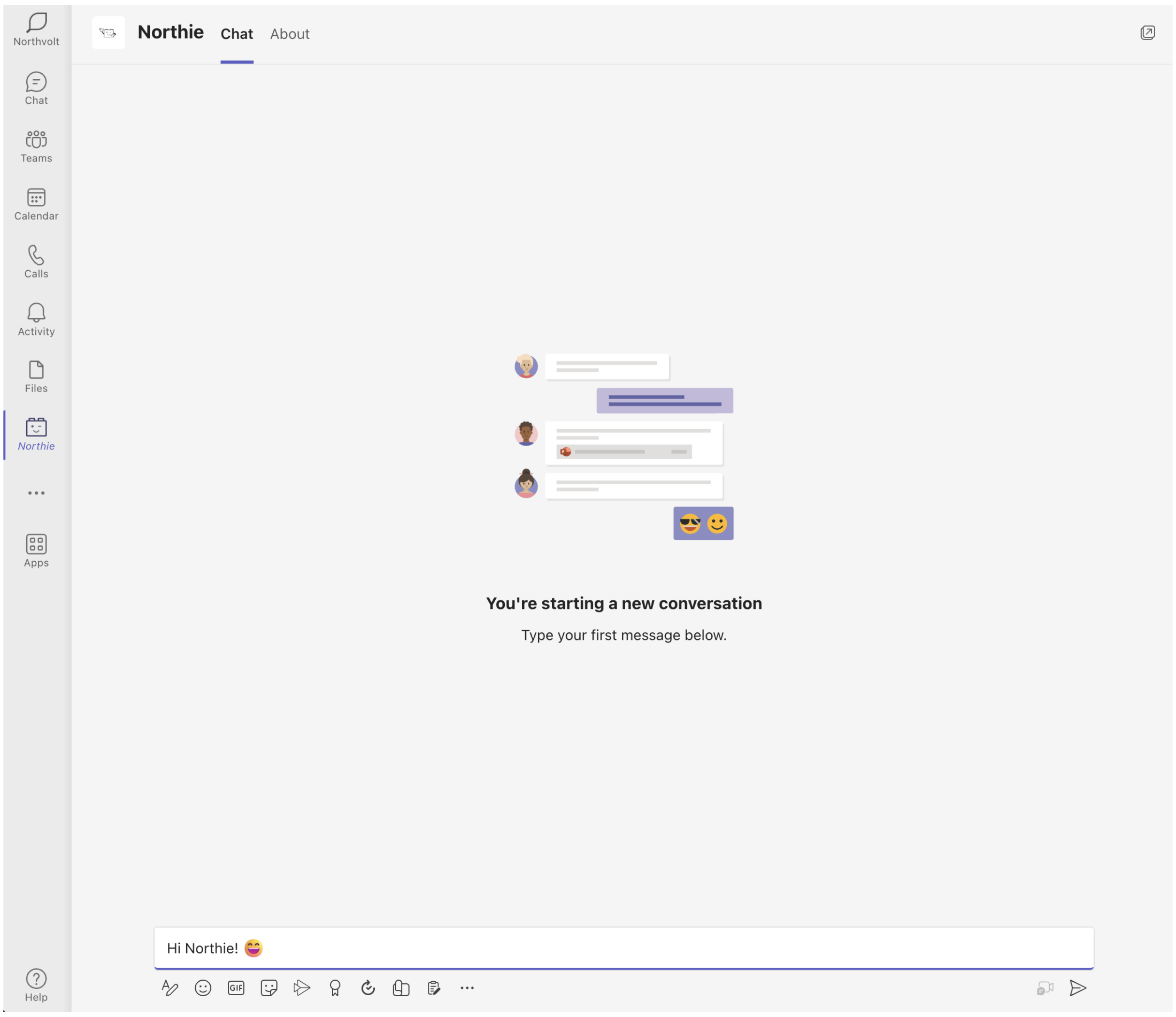
4.1.6 Data and data ETL

The data needed for the knowledge base of the Custom Question Answering model is ingested into Language Studio in several ways. Firstly, it can be directly ingested by an IT Support agent or developer who has access to Language Studio. In addition, structured and unstructured data can be extracted, transformed and loaded (ETL) through Azure Logic Apps and Azure Storage and then added to the model in Language Studio. In some cases, based on for example the security level or the structure of the data, the data can be directly extracted and loaded into Language Studio.

4.2 Message flow

The message flow starts with users authenticating themselves by logging into Microsoft Teams. Once the user is logged in and authenticated, the user can find the bot as a registered Microsoft Teams application as displayed in Figure 4.2. In the application, the user can send a message directly to the bot and based on the content of the message, it is routed to different services or workflows inside the bot. The message is first routed to the Custom Question Answering resource where the model tries to find an answer to the user's message. If the language model is confident enough in its response, the bot constructs an answer and sends it back to the user. If the CQA model is uncertain of the message, the message can be caught by the LUIS language model. The LUIS model tries to understand the intent of the user's message to determine if the user wants to, for example, provide feedback, raise a support ticket to IT Support or stop the conversation. If the model is confident in the intent of the message, the bot routes the message to the matching workflow where the conversation continues.

If neither of the language models finds a suitable response or intent to the user's message, the message is classified as "Unknown Intent". In this case, the bot routes the message to the language detection model to check if the user's message is in English. If the detected language is not English, the bot responds that it is only proficient in English and asks the user to try again. If the detected language is indeed English, the question is logged as an unanswered question and the bot responds with a message explaining that it doesn't understand. If the bot doesn't understand the user's messages three times in a row, it asks if the user wants to raise a support ticket to IT Support instead. If the user responds negatively, the bot will reply with a message acknowledging that the user's question has been logged and expressing its hope to provide an answer in the future.



You're starting a new conversation

Type your first message below.

Hi Northie! 😄

Figure 4.2: Interface in Teams for the bot's registered application.

5 User testing

In this chapter, the user testing of the bot is presented. The purpose of user testing is to see how the bot performs and how people interact with the bot, and then to discover how people perceive the bot. The aim is also to collect feedback and reveal possible bugs and faults in the bot for further development. In Section 5.1 the method for the testing is described and in Section 5.2 follows a description of the metrics used to evaluate the bot.

5.1 Method

The testing period of the bot was open for five work days and simple instructions on how to start testing and what to test were given to the users. The users were asked to test the bot's ability to answer questions related to IT and ordinary questions such as "How are you?". In addition, they were asked to create a support ticket to IT Support by chatting with the bot and also give feedback directly to the bot. The instructions were posted to a test group consisting of employees from different teams with professions ranging from IT Support agents to senior software developers and project managers. The intentional diversity of professions was aimed at capturing how various professional backgrounds interact with and perceive the bot. This is crucial because the bot must deliver satisfactory performance for all individuals across the company.

5.2 Evaluation metrics

Based on the theory for evaluation methods presented in Section 2.2.3, the performance of the chatbot is evaluated using a collection of effectiveness, efficiency and satisfaction-based metrics. The metrics are measured either automatically by the system, for example, response time, or found through a survey conducted on the test group. Evaluation of metrics such as accuracy and the intent recognition rate is done manually from the logs of the users' interactions with the bot.

5.2.1 Effectiveness

The following metrics are used to evaluate the effectiveness of the bot:

- **Accuracy**: Accuracy measures the percentage of messages answered correctly by the chatbot. It is calculated by dividing the number of correct answers by the total number of messages sent to the bot.
- **Precision**: Precision measures the proportion of relevant answers provided by the bot out of the total number of answers provided. This metric is useful to ensure that the bot is not providing irrelevant answers to questions.
- **Intent recognition rate**: Measures the percentage of queries where the intent of the user's message is correctly identified. A high intent recognition rate signifies a strong ability to understand the user's query and deliver an appropriate response effectively.

5.2.2 Efficiency

The following metrics are used to evaluate the efficiency of the bot:

- **Response Time**: The time it takes for the chatbot to respond to the user's message. Shorter response times are generally considered more efficient and indicate that the chatbot can provide quick answers to the user's questions.
- **Scalability**: This metric measures how well the chatbot can handle an increasing number of users and queries. A scalable chatbot can handle a high volume of traffic without experiencing performance issues such as an increased response time.

5.2.3 Satisfaction

A survey is used to measure the user's overall satisfaction with the chatbot. The survey consists of several items that ask users to rate their satisfaction with various aspects of the chatbot, such as its responsiveness, accuracy, and helpfulness. The survey consists of a section of statements where the user is instructed to rate their agreement on a five-point Likert scale from "Strongly disagree" to "Strongly agree". The statements are:

- I enjoyed the overall experience with the bot. (Enjoyment)
- It was easy to chat and use the bot. (Ease)
- The bot was able to give sufficient answers to my questions. (Accuracy)
- The bot's answers are related/relevant to my question. (Relevancy)
- The time it takes for the bot to answer is acceptable. (Responsiveness)
- The chatbot explained its scope and purpose well. (Purpose)
- In the future, I will use the chatbot instead of contacting IT Support. (Retention)

where the options are transferred into integer values, 1-5, where 1 represents "Strongly disagree" and 5 "Strongly agree". Moreover, the users are asked to provide their gender and also their age in the gender gaps 18-24, 25-34, 35-44, 45-54, 55-64 and 65 or older. They are also asked to provide their role and what team they belong to as well as their English proficiency. The options for English proficiency are the following:

- Native
- Fluent (C1-C2)
- Advanced (B2-C1)
- Intermediate (B1-B2)
- Beginner (A1-A2)

where the scale is inspired by the Common European Framework of Reference for Languages (CEFR) [45]. The options are transferred into integer values, 1-5, where 1 represents "Beginner" and 5 is "Native". In addition, they were also asked how frequently they have IT-related questions at work, if there were anything in particular they liked about the interaction with the bot and if the chatbot could improve upon something. Lastly, the survey asked the user how likely is it that they will recommend using the chatbot to their colleagues on a scale of 0-10, with 0 being "not at all likely" and 10 being "extremely likely". This question is used to calculate the so-called Net Promoter Score (NPS). NPS is calculated by partitioning the scores into groups where users providing ratings of 10 or 9 are "promoters", 7 or 8 are "passives" and 6 or lower are "detractors". The NPS is then calculated according to

$$\text{NPS} = (\text{Percentage of promoters - Percentage of detractors}) \cdot 100. \tag{5.1}$$

6 Results and discussion

During the testing period, a total of 20 employees across six teams interacted with the bot and completed the survey. The diverse testing group consisted of employees with professions ranging from content producer to senior software engineer, with ages ranging from 18 to 54. 11 of the participants identified as male and 9 as female, resulting in an almost perfect split between the genders of the participants. The majority evaluated themselves to be fluent English speakers with an average English proficiency score of 3.75. Out of the participants, 60 % reported having one or multiple IT-related questions on a daily basis, while 20% indicated having them weekly, and the remaining 20% reported having such questions on a monthly or rare basis. A total of 202 messages were sent to the bot, resulting in an average of 10.1 messages per user, with the maximum number of messages sent by a user being 20 and the minimum being 5. These messages exclude the messages in the workflows where for example the user can create an issue ticket or provide feedback, as these workflows follow a predefined structure where the user responds to questions asked by the bot. However, the results of these workflows are tracked to for instance analyze the response time across all bot messages and will be discussed in the following subsection together with a detailed presentation of the results of the bot's effectiveness, efficiency and satisfaction.

6.1 Effectiveness

A detailed summary of the messages directed to the question-answering model is displayed in Table 6.1. Out of the 187 total messages directed to the model, the model successfully found an answer in its knowledge base and sent back a response to the user for 112 of them. By manually analyzing these messages, 86 were evaluated to be answered correctly and 17 wrongly. The nine remaining were hard to evaluate since the messages were very unclear in their intent. From the analysis of the incorrectly answered messages, it was found that six of them were from users attempting to reach out to a real person, be it a colleague or a specific member of the IT Support team, through the bot. Moreover, two of the messages were valid, but their answers were not present in the knowledge base. These messages included non-English terms, triggering the intervention of the language detection model, which categorized them as a language other than English. The remaining nine inaccurately answered messages were the result of the model responding even when it lacked confidence in its response. If the required confidence score for the bot to answer was set higher, the bot wouldn't answer these questions and instead ask the user to reformulate their question to be able to find a more confident answer.

Description	Number of messages
Answered	**112**
Correctly answered	86
Wrongly answered	17
Responded despite low confidence	9
Contact a real person	6
Non-English terms	2
Not answered	**75**
Valid IT questions not in KB	57
Chitchat questions	9
Issue ticket creation	7
In KB but not confident enough	2
Total	**187**

Table 6.1: Summary of message analysis for the CQA model.

The remaining 75 messages directed to the question-answering model did not yield a confident answer from the language model's knowledge base. Among these, 57 messages were evaluated as valid IT-related questions that the bot should answer, but they were not part of the knowledge base during the testing period. Two messages contained questions that had valid answers in the knowledge base, but they were formulated in an unconventional manner with poor grammar, leading to the model lacking confidence in its response. Additionally, nine messages were chitchat questions that fell outside the scope of the bot's conversational abilities, such as queries like "Would you like to share a piece of cheese with me?" or "What's the most memorable vacation you've ever had?". In the remaining seven messages, the user attempts to create an issue ticket for IT Support, but they were formulated in a way that the LUIS model couldn't confidently interpret and guide the user through the ticket creation workflow.

The 15 messages that were not directed to the question-answering model, were messages where the LUIS model stepped in and started a workflow for creating an issue ticket to IT Support or stopping the conversation. There were 9 messages for stopping the conversation and 6 for creating an issue ticket, and in all of them, the LUIS model found the correct intent of the message and redirected the user to the correct workflow.

From the evaluation of the 202 messages, the accuracy is calculated by dividing the number of correct answers by the total number of questions asked:

$$\text{Accuracy} = \frac{\text{Number of correct answers for QnA and LUIS model}}{\text{Total number of messages}} = \frac{101}{202} = 50\% \tag{6.1}$$

An accuracy score of 50% is not impressive for a question-answering bot since high accuracy is essential in ensuring effective communication with users. However, in this calculation, for instance, complex chitchat questions and messages where the intent could not be understood from the manual evaluation are included, which lowers the accuracy. To improve accuracy, both the quality and quantity of the data ingested into the question-answering language model need to be increased since the data is currently not diverse enough, resulting in the model having issues understanding user queries and providing accurate responses. This can be done by allowing several persons, preferably from different parts of the company, to populate the knowledge base. This would require both training in Azure Cognitive Services and in information security since we do not want to for example break the language model whilst it is in production or spread personal and confidential information.

For the precision metric, the bot performs much better. Only 17 out of the 112 answered messages were answered wrongly, resulting in a precision of 85%. An important note here is that if the user read the scope of the bot in more detail, there would be no messages about getting in contact with a real person. Also, if the confidence score threshold was set higher, the bot would not answer when it was unsure and hence the precision would increase. Although the precision of the model is relatively high, it can still be improved to increase the effectiveness of the bot. Similarly to accuracy, the precision will increase with the amount and quality of data ingested into the question-answering language model. This can for example be done by adding more alternate questions to the already present questions and adding synonyms.

Moving on to the intent recognition rate, the LUIS model found the correct intent for 15 messages. In total there were 22 messages where the intent of the message should have been captured by the LUIS model, resulting in an intent recognition rate of 68.2%. Analyzing the seven messages where the intent wasn't found, it is noticeable that these were both not written in well-formulated English or in a similar way to the utterances the LUIS model was trained on. If the model were trained on a wider variety of utterances, the intent of these messages would have been detected which would result in a significant increase in the intent recognition rate. Another option would be to migrate to the next generation of LUIS, Conversational Language Understanding (CLU) which is included in Azure Cognitive Services [31]. CLU offers several advantages over LUIS including improved accuracy for less amount of data, the ability to predict intent across 96 languages for a model trained in one language and easy integration with other Azure Cognitive Services projects such as Custom Question Answering [46].

6.2 Efficiency

The response time for the bot was evaluated using the logs in Application Insights for all messages sent during the testing period. The average response time for the bot was 0.81 seconds, with a minimum response time of 0.17 seconds and a maximum response time of 3.401 seconds. The distribution of response times was approximately normal, with the majority of responses falling within the range of 0.25 to 0.75 seconds. However, there were some outliers with response times above 1.5 seconds, which may indicate areas for optimization in the bot's architecture. All of the messages which had response times over 2 seconds were messages where when the user created an issue ticket to IT Support where dynamic information is fetched through an API call to Jira REST API. Response times between 1.5 and 2 seconds were messages which were classified as Unknown Intent resulting in the bot sending an API call to the language detection model and triggering nested conditional statements which increases the time it takes for the bot to answer.

The response time of a chatbot can be influenced by many factors, including the number of users interacting with the bot at the same time, the complexity of the queries being processed, and the capacity of the chatbot's hosting environment. During periods of high load, where a larger number of users are interacting with the bot simultaneously, it is common to experience longer response times due to increased processing demands. When comparing the response time during a time period with high load, in this case, seven concurrent users, the average response time is only two hundredths of a second longer than the average response time when only one user was interacting with the bot during the same time period the next day. While it is unusual for a chatbot to maintain the same response time during high load periods and it might not be able to maintain this behavior when the number of users is significantly higher than during the testing period, it is a testament to the design and optimization of the bot's architecture and hosting environment.

Overall, the response time during the testing period indicates that the bot performs within acceptable limits for most user requests, with a fast average response time and a relatively small range between the minimum and maximum values during both high and low loads, which is an important factor in ensuring a positive user experience. However, further optimization is necessary to improve the performance of edge cases with high response times. To solve the problem with response times above 2 seconds, the handling of requests sent to Jira REST API needs to be optimized. Currently, the bot sends the request to the external API and waits for the response, which can result in the response time being affected by for instance network latency and rate limit. To solve this, the requests to the external API can be sent at the beginning of the workflow and while the bot waits for the response, it can let the user proceed through the workflow and perform actions that are not dependent on the responses of the requests. Then, when the responses are received, they can be stored in the memory and used whenever the user has finished the other actions, resulting in a shorter response time. This can unfortunately not be done for the Unknown Intent messages that had a response time between 1.5 and 2.0 seconds, since all actions after the result of the evaluation of the language detection model rely on whether the message was recognized as English or not. In this case, the optimization should be focused on the actions followed by the evaluation of language detection, where the number of conditional statements and actions can be minimized.

6.3 Satisfaction

In Table 6.2 the aggregated results of statements where the user rate their agreement are displayed. From the table, it can be concluded that on average the users enjoyed the overall experience with the bot. A score of 3.95 is also higher than what the user rated their previous experience with chatbots, which was on average 3.56. The overall experience with the bot was also scored higher when the bot asked the user directly in the chat to give a rating between 1-5 where the average was 4.4. The high overall enjoyment of the bot is probably dominated by the ease and responsiveness of the which both scored over 4 on average with a low standard deviation. This claim is supported by the written feedback part of the survey, where several users pointed out that having a question-answering bot with such responsiveness available in Microsoft Teams where most of their communication is hosted, is a great solution for the future.

Question	Average	Std
Enjoyment	3.95	0.60
Ease	4.20	0.62
Accuracy	3.20	0.83
Relevancy	3.70	0.73
Responsiveness	4.60	0.50
Purpose	3.95	0.60
Retention	3.65	0.93

Table 6.2: Result of the survey displaying the average and standard deviation for the agreement statements.

The knowledge base is a dominant topic in the section where the users are asked to suggest what the bot can improve on. The majority of the users think that the bot has too many gaps in its knowledge base as of right now which is supported by the accuracy score presented in Section 6.1 and in the average user rating of accuracy presented in Table 6.2. However, several users point out that they are impressed with how the model finds relevant answers even when bad grammar and sentence syntax are used, which is also what the precision score in Section 6.1 and the relevancy score in Table 6.2 display. Users also mention that collecting valuable information for the knowledge base of the model should not only depend on one junior employee from one team but rather be a collective responsibility across several teams. The importance of involving multiple teams in the process of building a great knowledge base is clearly visible in the questions users ask the bot where the users focus on topics related to the systems they are working with. As an example, a user working with digital productivity asks questions related to their systems while a user from the IT Support focuses on FAQs that they receive daily such as "How can I get an iPhone case?" or "What should I do with my broken phone"?.

Overall the users think that the bot explained its scope and purpose well. However, some users had a neutral agreement on how the scope and purpose of the bot are communicated. Analyzing the logs of these users' interactions with the bot, it is noticeable that they have sent messages to which the bot couldn't find a confident answer. As an example, a majority of these users sent messages where they try to come in contact with a real person which is out of the scope of the bot. The main reason why we don't want to let the bot hand off a conversation to an IT Support agent is that the IT Support team needs to track the support issues in Jira that require more knowledge or complex actions than what the bot has. If the conversation is allowed to continue in Microsoft Teams, it may not be properly documented and tracked, which can lead to issues such as loss of information and difficulty tracking the status of the support request. Moreover, the survey displays a clear correlation between the score of purpose and accuracy, where users that score a low purpose tend to score low on accuracy as well. The precision is also depending on the success of communicating the purpose and scope of the bot. If the scope and the purpose are clear for the user, the user will hopefully interact with the bot within its scope, minimizing the number of unnecessary messages and increasing both accuracy and precision. A clear scope and purpose are also important for the overall experience of the bot, since communicating with a bot that does not understand what you are saying is both frustrating and a waste of time.

The retention score displayed in Table 6.2 is surprisingly low since the overall enjoyment is significantly higher. However, the retention score includes the scores given by IT Support agents who probably will not use the bot but only maintain its knowledge base. If the scores from the users in the IT Support team are removed, the average retention score increases to 3.93 which is in agreement with the overall enjoyment of the bot. Somewhat connected to the retention is the Net Promoter Score where the user state how likely they are to recommend the chatbot to one of their colleagues. 50 % of the users scored 9 or 10 for the likelihood of them recommending the bot to fellow colleagues, while 40 % scored 7 or 8, and the remaining 10% scored a 6. This results in an NPS score of

$$\text{(Percentage of promoters - Percentage of detractors)} \cdot 100 = \left(\frac{10}{20} - \frac{2}{20}\right) * 100 = 40 \tag{6.2}$$

which can be considered good since the bot has significantly more promoters than detractors. A good sign is also that there were no scores below 6 which states that the detractors are closer to being passives and possibly even promoters rather than being "true" detractors who are completely against the implementation of the bot.

Although the effectiveness of the chatbot was low, several users provided feedback about how good the support the bot gave them was and how easy it was to get support by simply chatting with the bot in Microsoft Teams. One of the reasons for this is the workflow where the bot helps the user create an issue ticket to IT Support, which is triggered when the bot does not understand several messages in a row or the user asks the bot to create the ticket. In total 5 issue tickets were created by users during the test period and three of these users mentioned in the feedback that it was something that they particularly liked about the bot.

Altogether the survey did not display any significant correlation between gender, age, role, team and the scores of the statements where the user rated their agreement except for the retention where users from IT Support tend to score lower than others. This indicates that the bot is providing consistent and reliable responses that are not biased toward any particular group. This is important since if the performance ratings vary significantly between different groups, it could mean that the bot is not effectively addressing the needs and preferences of certain users, which could result in dissatisfaction and reduced usage. Having consistent ratings across different groups also suggests that the bot is delivering responses that are clear, concise, and relevant to a wide range of users, which is essential for its success in providing useful and effective assistance.

7 Conclusion and further work

7.1 Conclusion

In this thesis, a question-answering chatbot for IT Support at Northvolt has been implemented. The chatbot is built in Bot Framework Composer and its intelligence and cognition come from language models which are deployed in Azure Cognitive Services and LUIS. The bot is deployed through an Azure DevOps pipeline and Azure Bot Service to a Microsoft Teams application where employees can chat with it and get answers to their IT-related questions. Important performance metrics are monitored through Azure Application Insights and stored in Azure Storage. The testing of the chatbot was conducted through a 5-day-long testing period where a diverse group of employees interacted with the bot. From the results of the testing, efficiency, effectiveness and satisfaction metrics were collected, analyzed and evaluated. Through the analysis and evaluation of the conversation logs and the user feedback survey, it can be concluded that the effectiveness of the bot in terms of accuracy, precision and intent recognition rate was relatively low. Although the bot's effectiveness was lacking, the overall satisfaction of the bot was high which mainly comes from the ease of chatting with the bot and its short response time. With a greater set of high-quality data ingested into the bot's cognition and intelligence services and a clearer description of the bot's purpose and scope, the bot's accuracy and precision will increase, resulting in an increase in retention and overall enjoyment.

7.2 Further work

As for further work, the main focus should be on increasing the knowledge base of the bot. It can be done in several ways, with the easiest being adding more question-and-answer pairs and connecting the base to additional data sources in Language Studio. Two data sources that are essential for Northvolt are Jira and Confluence where teams create and track issues and store technical documentation. However, the problem with these sources in this context is that they are very complex in their nature and have no real data standardization that ranges across the organization. For instance, Jira provides a structured format for capturing information but the content of each support ticket can vary widely based on the preferences of the individual user who created it. Some users may include detailed descriptions and comments, while others may provide only minimal information which makes it challenging to locate the relevant questions and answers within the ticket. Confluence, on the other hand, provides a structured format for creating pages but the content of each page can contain a wide variety of information, including text, images, videos, and other multimedia content. Some users may organize their Confluence pages in a logical, hierarchical manner, while others may use a more free-form approach which can make it challenging to locate the relevant data to give to the bot. An attempt to remedy this would be to use NLP techniques and possibly other AI tools to automatically extract questions and answers from Jira issue tickets and Confluence pages. For example, NLP models can be trained to identify and extract specific types of information, such as questions and answers, from unstructured text data.

Another option to increase the knowledge base of the bot would be to extend the bot with Azure OpenAI Service. Azure OpenAI Service is a partnership between Microsoft Azure and OpenAI, a leading research organization in artificial intelligence which for instance developed the GPT-3 language model, providing several services and tools that leverage OpenAI's state-of-the-art language models [47]. An example of how this can be done is by creating a solution using a combination of Azure OpenAI Service and Azure Cognitive Search [48] as presented by P. Castro in his blog post "Revolutionize your Enterprise Data with ChatGPT: Next-gen Apps w/ Azure OpenAI and Cognitive Search" [49]. Castro's solution utilizes Azure Cognitive Search's ability to index, understand and retrieve the correct data and then one of the OpenAI large language models constructs an answer based on that data. There are however some potential risks to consider with this approach, including data privacy and security and a risk that the answers provided may be inaccurate or unreliable. Overall, there is not a single correct way of increasing the knowledge base. However, the most important part is that information fetched to the knowledge base originates from multiple data sources that have been created and maintained by a range of employees and teams. This will help ensure that the bot has a diverse range of perspectives and expertise which will improve the accuracy and completeness of the answers provided by the bot. Working together on the knowledge base can also facilitate knowledge sharing and collaboration among employees and teams which can help to break down silos and ensure that knowledge is shared across the company.

As of right now, the bot is designed and trained to handle specific queries related to IT, such as technical support or software issues. However, as Northvolt becomes more interconnected and reliant on technology, employees from various departments may have questions that fall outside of the scope of IT. By rebranding the bot to handle questions from the entire organization, it can become a valuable resource for all employees, regardless of their role or team. This not only increases the efficiency of the organization by providing a centralized knowledge base but also promotes a culture of collaboration and knowledge sharing. Additionally, by expanding the bot's capabilities, it can help reduce the workload for other employees and teams as well.

To effectively communicate the purpose and scope of the bot in the future, the Teams application that the bot communicates through should be improved. It should have a concise and clear description of the bot's purpose and scope. Currently, it only consists of a description of the bot's capabilities but not its limitations. The description should also be written in easy and understandable language, making it accessible for every employee no matter the technical expertise or previous experience with chatbots. Moreover, providing examples of the types of questions the bot can answer and the types of information it can provide will be a helpful way to demonstrate its capabilities. This can for example be done through workshops, written documentation or short videos that would grasp the interest of a user and help them gain quick learnings.

Another suggestion for further work is to migrate the bot to Microsoft's Power Virtual Agents (PVA) [50]. Power Virtual Agents is a cloud-based service that allows the developer to create and deploy intelligent chatbots using a graphical interface. With Power Virtual Agents, no coding skills or expertise in artificial intelligence is needed to create chatbots, allowing collaborative development of the bot without weeks of training. In addition, language models from Azure OpenAI are supported natively in PVA, enabling the developer to simply describe what they would like the bot to do and Microsoft Copilot [51] will help to create entire workflows. Moreover, conversations with a bot built in PVA can be boosted by allowing OpenAI's GPT language model to create real-time responses from a website. As of right now, these features in PVA are in public preview, allowing developers to explore and test upcoming features, but they will most probably be released in the near future. Altogether, Power Virtual Agents is Microsoft's flagship service for bot building and with its advanced AI features powered by OpenAI, the service strengthens its position to take the top spot of platforms to build conversational agents in the future.

References

[1] K. R. Chowdhary, "Natural language processing," in *Fundamentals of Artificial Intelligence*. New Delhi: Springer India, 2020, pp. 603–649, ISBN: 978-81-322-3972-7. DOI: 10.1007/978-81-322-3972-7_19. [Online]. Available: https://doi.org/10.1007/978-81-322-3972-7_19.

[2] A. M. Turing, "Computing machinery and intelligence," *Mind*, vol. 59, no. 236, pp. 433–460, 1950, ISSN: 00264423, 14602113. [Online]. Available: http://www.jstor.org/stable/2251299.

[3] T. Mikolov, K. Chen, G. Corrado, and J. Dean, *Efficient estimation of word representations in vector space*, 2013. arXiv: 1301.3781 [cs.CL].

[4] S. Jung, "Semantic vector learning for natural language understanding," *Computer Speech & Language*, vol. 56, pp. 130–145, 2019, ISSN: 0885-2308. DOI: https://doi.org/10.1016/j.csl.2018.12.008. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0885230817303595.

[5] A. Chopra, A. Prashar, and C. Sain, "Natural language processing," *International Journal of Technology Enhancements and Emerging Engineering Research*, vol. 1, pp. 131–134, 2013.

[6] A. Vaswani, N. Shazeer, N. Parmar, *et al.*, *Attention is all you need*, 2017. DOI: 10.48550/ARXIV.1706.03762. [Online]. Available: https://arxiv.org/abs/1706.03762.

[7] C. D. Manning, P. Raghavan, and H. Schütze, *Introduction to Information Retrieval*. Cambridge, UK: Cambridge University Press, 2008, ISBN: 978-0-521-86571-5. [Online]. Available: http://nlp.stanford.edu/IR-book/information-retrieval-book.html.

[8] S. Hochreiter, "The vanishing gradient problem during learning recurrent neural nets and problem solutions.," *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, vol. 6, no. 2, pp. 107–116, 1998. [Online]. Available: http://dblp.uni-trier.de/db/journals/ijufks/ijufks6.html#Hochreiter98.

[9] J. L. Ba, J. R. Kiros, and G. E. Hinton, *Layer normalization*, 2016. DOI: 10.48550/ARXIV.1607.06450. [Online]. Available: https://arxiv.org/abs/1607.06450.

[10] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, *Bert: Pre-training of deep bidirectional transformers for language understanding*, 2018. DOI: 10.48550/ARXIV.1810.04805. [Online]. Available: https://arxiv.org/abs/1810.04805.

[11] T. B. Brown, B. Mann, N. Ryder, *et al.*, *Language models are few-shot learners*, 2020. DOI: 10.48550/ARXIV.2005.14165. [Online]. Available: https://arxiv.org/abs/2005.14165.

[12] Z. Dai, Z. Yang, Y. Yang, J. Carbonell, Q. V. Le, and R. Salakhutdinov, *Transformer-xl: Attentive language models beyond a fixed-length context*, 2019. DOI: 10.48550/ARXIV.1901.02860. [Online]. Available: https://arxiv.org/abs/1901.02860.

[13] E. Adamopoulou and L. Moussiades, "An overview of chatbot technology," in *Artificial Intelligence Applications and Innovations*, I. Maglogiannis, L. Iliadis, and E. Pimenidis, Eds., Cham: Springer International Publishing, 2020, pp. 373–383, ISBN: 978-3-030-49186-4.

[14] Apple, *Siri*. [Online]. Available: https://www.apple.com/siri/.

[15] J. Weizenbaum, "Eliza a computer program for the study of natural language communication between man and machine," *Commun. ACM*, vol. 9, no. 1, pp. 36–45, Jan. 1966, ISSN: 0001-0782. DOI: 10.1145/365153.365168. [Online]. Available: http://doi.acm.org/10.1145/365153.365168.

[16] N. M. Radziwill and M. C. Benton, *Evaluating quality of chatbots and intelligent conversational agents*, 2017. arXiv: 1704.04579 [cs.CY].

[17] A. Abran, A. Khelifi, W. Suryn, and A. Seffah, "Usability meanings and interpretations in iso standards," *Software Quality Journal*, vol. 11, pp. 325–338, Nov. 2003. DOI: 10.1023/A:1025869312943.

[18] C. Bartneck, D. Kulic, E. A. Croft, and S. Zoghbi, "Measurement instruments for the anthropomorphism, animacy, likeability, perceived intelligence, and perceived safety of robots.," *Int. J. Soc. Robotics*, vol. 1, no. 1, pp. 71–81, 2009. [Online]. Available: http://dblp.uni-trier.de/db/journals/ijsr/ijsr1.html#BartneckKCZ09.

[19] K. Hone, U. Ph, R. Graham, and A. Link, "Towards a tool for the subjective assessment of speech system interfaces (sassi)," *Natural Language Engineering*, vol. 6, Jul. 2000. DOI: 10.1017/S1351324900002497.

[20] J. Casas, M.-O. Tricot, O. Abou Khaled, E. Mugellini, and P. Cudre-Mauroux, "Trends & methods in chatbot evaluation," Oct. 2020, pp. 280–286. DOI: 10.1145/3395035.3425319.

[21] Microsoft, *Project turing.* [Online]. Available: https://turing.microsoft.com/.

[22] S. Smith, M. Patwary, B. Norick, *et al.*, *Using deepspeed and megatron to train megatron-turing nlg 530b, a large-scale generative language model*, 2022. DOI: 10.48550/ARXIV.2201.11990. [Online]. Available: https://arxiv.org/abs/2201.11990.

[23] J. Hu, S. Ruder, A. Siddhant, G. Neubig, O. Firat, and M. Johnson, *Xtreme: A massively multilingual multi-task benchmark for evaluating cross-lingual generalization*, 2020. DOI: 10.48550/ARXIV.2003.11080. [Online]. Available: https://arxiv.org/abs/2003.11080.

[24] A. Wang, A. Singh, J. Michael, F. Hill, O. Levy, and S. R. Bowman, "GLUE: A multi-task benchmark and analysis platform for natural language understanding," *CoRR*, vol. abs/1804.07461, 2018. arXiv: 1804.07461. [Online]. Available: http://arxiv.org/abs/1804.07461.

[25] Microsoft, *Microsoft turing universal language representation model, t-ulrv6, tops both xtreme and glue leaderboards with a single model.* [Online]. Available: https://blogs.bing.com/search-quality-insights/october-2022/Microsoft-Turing-Universal-Language-Representation-model,-T-ULRv6,-tops-both-XTREME-and-GLUE-leaderb.

[26] Microsoft, *Azure cognitive services.* [Online]. Available: https://azure.microsoft.com/en-us/products/cognitive-services/#overview.

[27] Microsoft, *Language studio.* [Online]. Available: https://learn.microsoft.com/en-us/azure/cognitive-services/language-service/language-studio.

[28] Microsoft, *Question answering.* [Online]. Available: https://azure.microsoft.com/en-us/products/cognitive-services/question-answering/.

[29] Microsoft, *Azure language detection.* [Online]. Available: https://learn.microsoft.com/en-us/azure/cognitive-services/language-service/language-detection/overview.

[30] Microsoft, *Luis.* [Online]. Available: https://www.luis.ai/.

[31] Microsoft, *Conversational language understanding.* [Online]. Available: https://learn.microsoft.com/en-us/azure/cognitive-services/language-service/conversational-language-understanding/overview.

[32] Microsoft, *Azure bot service.* [Online]. Available: https://azure.microsoft.com/en-us/products/bot-services.

[33] Microsoft, *Microsoft bot framework.* [Online]. Available: https://learn.microsoft.com/en-us/azure/bot-service/bot-service-overview?view=azure-bot-service-4.0.

[34] Microsoft, *Microsoft bot framework composer.* [Online]. Available: https://learn.microsoft.com/en-us/composer/introduction?tabs=v2x.

[35] Microsoft, *Microsoft bot framework emulator.* [Online]. Available: https://learn.microsoft.com/en-us/azure/bot-service/bot-service-debug-emulator?view=azure-bot-service-4.0&tabs=csharp.

[36] Microsoft, *Azure active directory.* [Online]. Available: https://azure.microsoft.com/en-us/products/active-directory.

[37] Microsoft, *Azure key vault.* [Online]. Available: https://azure.microsoft.com/en-us/products/key-vault/.

[38] Microsoft, *Azure key vault.* [Online]. Available: https://learn.microsoft.com/en-us/azure/storage/common/storage-introduction.

[39] Microsoft, *Azure devops.* [Online]. Available: https://azure.microsoft.com/en-gb/products/devops/.

[40] Microsoft, *Azure monitor.* [Online]. Available: https://azure.microsoft.com/en-gb/products/monitor/.

[41] Microsoft, *Azure application insights.* [Online]. Available: https://learn.microsoft.com/en-us/azure/azure-monitor/app/app-insights-overview?tabs=net.

[42] Microsoft, *Azure logic apps.* [Online]. Available: https://azure.microsoft.com/en-us/products/logic-apps/.

[43] Microsoft, *Azure app service*. [Online]. Available: https://azure.microsoft.com/en-us/products/app-service.

[44] Microsoft, *Chit chat in custom question answering*. [Online]. Available: https://learn.microsoft.com/en-us/azure/cognitive-services/language-service/question-answering/how-to/chit-chat.

[45] COE, *Cefr*. [Online]. Available: https://www.coe.int/en/web/common-european-framework-reference-languages/level-descriptions.

[46] Microsoft, *Comparison between luis and clu*. [Online]. Available: https://learn.microsoft.com/en-us/azure/cognitive-services/language-service/conversational-language-understanding/how-to/migrate-from-luis?tabs=luis-portal#comparison-between-luis-and-clu.

[47] Microsoft, *Azure openai*. [Online]. Available: https://azure.microsoft.com/en-us/products/cognitive-services/openai-service.

[48] Microsoft, *Azure cognitive search*. [Online]. Available: https://azure.microsoft.com/en-us/products/search/.

[49] P. Castro, *Revolutionize your enterprise data with chatgpt: Next-gen apps w/ azure openai and cognitive search*. [Online]. Available: https://techcommunity.microsoft.com/t5/ai-applied-ai-blog/revolutionize-your-enterprise-data-with-chatgpt-next-gen-apps-w/ba-p/3762087.

[50] Microsoft, *Power virtual agents*. [Online]. Available: https://powervirtualagents.microsoft.com/sv-se/.

[51] Microsoft, *Microsoft copilot*. [Online]. Available: https://blogs.microsoft.com/blog/2023/03/16/introducing-microsoft-365-copilot-your-copilot-for-work/.