Tampereen yliopisto

Perttu Autti

CHATBOT IMPLEMENTATION AND INTEGRATION

Bachelor's thesis
Faculty of Information Technology and Communication Sciences
May 2022

TIIVISTELMÄ

Perttu Autti: Chatbotien toteutus ja integrointi
Kandidaatintutkielma
Tampereen yliopisto
Tietotekniikan tutkinto-ohjelma
Toukokuu 2022

Chatbotit ovat tietokoneohjelmia, jotka kommunikoivat ihmisten kanssa käyttäen luonnollista kieltä. Niiden tärkein tehtävä on tarjota keskustelullinen pääsy johonkin palveluun tai palveluihin. Chatbot-rajapintojen tekninen toteutus ei ole yksinkertainen tehtävä kenellekään, jolle aihe ei ole entuudestaan tuttu. Tässä kanditaatintutkielmassa selvitetään, mitä yleiseen chatbotien toteutusmalliin kuuluu, ja miten chatbot voitaisiin toteuttaa käytännössä.

Tutkielma on jaettu kahteen osioon. Ensimmäisessä osiossa tutustutaan chatbotien historiaan, toteutusalustoihin ja yleisimpiin chatbotien toteutustapoihin. Osiossa perehdytään myös luonnollisen kielen prosessointiin osana käyttäjän syötteen ymmärtämiseen ja luonnollisen kielen tuottamiseen liittyviä tehtäviä.

Tutkielmassa havaitaan, että yleinen chatbotien toteutusmalli koostuu luonnollisen kielen prosessointiyksiköstä, dialoginhallintajärjestelmästä, sekä ulkoisista tietokannoista tai rajapinnoista. Luonnollisen kielen prosessointiyksikön havaitaan myös sisältävän tekstin esiprosessoinnista, luonnollisen kielen ymmärtämisestä, ja luonnollisen kielen tuottamisesta vastaavat osiot.

Havaitaan myös, että chatbot-rajapinta voidaan toteuttaa useammalle tasolle kolmikerroksista jaettujen järjestelmien arkkitehtuuria. Toteutuskohteisiin sisältyy paremmin tunnettujen toteutuskohteiden, kuten sovellusten sisäisten avustajien ja keskustelullisten IoT-rajapintojen lisäksi, myös vähemmän tunnettuja integraatiokohteita, kuten yritysprosessirajapintoja ja keskustelullisia hakumoottoreita.

Tutkielman toisessa osiossa toteutetaan oma chatbot-rajapinta avoimeen lähdekoodiin perustuvalla chatbotien kehitysalustalla, Rasalla. Toteutettava chatbot kykenee hoitamaan yksinkertaisia keskustelullisia ja hakuperusteisia tehtäviä. Rasa havaitaan helppokäyttöiseksi kehitysalustaksi, joka hoitaa luonnollisen kielen prosessointiin liittyvät tehtävät automaattisesti, jolloin kehittäjän täytyy ainoastaan määrittää opetuksessa käytettävä esimerkkidata, sekä mahdolliset toiminnot chatbotin toteuttamiseksi.

Avainsanat: chatbotit, NLP, toteutus, integrointi, Rasa

ABSTRACT

Perttu Autti: Chatbot implementation and integration
Bachelor's thesis
Tampere University
Information Technology
May 2022

Chatbots are computer programs which communicate with humans through natural language. Their main function is to provide conversational access to a service or services. The technical implementation of chatbots is not a trivial task for anyone new to the topic. In this bachelor's thesis we will find out what goes on in the chatbot implementation pipeline and how a chatbot could be developed in practice.

The thesis has been divided into two sections. In the first section we will cover the history, integration patterns, and most common ways of implementation of chatbots. In this section we also take a deeper look at natural language processing as a part of understanding user input and producing human-like responses.

It is found that a general chatbot implementation pipeline consists of a natural language processing unit, a dialogue manager and external databases or APIs. The natural language unit is also found to be further dividable into pre-processing, natural language understanding, and natural language generation sections.

We also find that chatbots can be integrated into multiple levels of the three-tiered architecture of distributed systems. This includes some better-known integration patterns like In-App assistants and conversational IoT-interfaces but also some less-known patterns such as a business process interface and a conversational query engine.

In the second part we will implement our own chatbot model with an open source chatbot framework called Rasa. Our chatbot will be able to handle basic chit-chatting and retrieval-based tasks. Rasa is found to be an easy-to-use framework which handles all the natural language processing tasks automatically so that the developer only needs to specify the training data and actions to build a chatbot.

Keywords: chatbots, NLP, implementation, integration, Rasa

CONTENTS

ABBREVIATIONS

AI	Artificial Intelligence
NLP	Natural language processing
NLU	Natural language understanding
NLG	Natural language generation
AWS	Amazon Web Services
MIT	Massachusetts Institute of Technology
CNN	Convolutional neural network
RNN	Recurrent neural network
LSTM	Long short-term memory
BiLSTM	Bidirectional LSTM

1. INTRODUCTION

Chatbots have become an important part of the interaction between businesses and customers when dealing with online services. The businesses benefit from them by being able to provide personalized customer service around the clock resulting in the increase of sales and reducing the need for hired staff. Similarly, the customers benefit from chatbot interfaces by having quick access to answers regarding navigation, product information or troubleshooting questions.

Chatbots are a highly researched field of artificial intelligence (AI) with the first implementations dating all the way back to the year 1966 (Kapočiute-Dzikiene, 2020). Although a lot of new technologies have emerged to improve chatbots in that time the problem of understanding and generating natural language in a human-level of accuracy in both open and closed-domain applications hasn't yet been solved.

Due to the amount of research on chatbots with new implementations continuously trying to improve on previous ones, the number of natural language processing (NLP) techniques and application specific solutions have skyrocketed. To combat the technological complexity of these systems many chatbot frameworks and libraries such as Dialogueflow and Rasa have emerged (Perez-Soler et al., 2021). These frameworks usually offer a high level of abstraction on training a natural language understanding (NLU) model and sometimes a natural language generation (NLG) model.

It is important for developers to understand the fundamental features of chatbots to be able to make the decision on which implementation direction to take in each project. On one hand, building a good NLP model from scratch requires a deep understanding in the underlying technologies. On the other hand, frameworks and libraries often come with their own share of limitations such as cost-of-service fees, lack of language support or commitment to a single platform such as Amazon Web Services (AWS).

In this bachelor's thesis we will try to answer the questions: "what technologies are commonly used for implementing a chatbot" and "how can a simple chatbot be implemented in practice". The conducted research method is a literature review in which we will look at previous peer reviewed articles, conference proceedings, and book publications on chatbots and natural language processing.

Natural language processing was found to be a key feature in the chatbot implementation pipeline. Both the extraction of intents and entities through natural language understanding and generation of responses with natural language generation models were found to be crucial parts of the pipeline. It was also discovered that chatbots have more areas of integration, later referred to as integration patterns, than might be the common understanding among developers. It was found that chatbots can be integrated into the data, logic, and user interface levels of the three-tier architecture of distributed software. The final take from this thesis is that the quickest way to get started as a developer is to implement a chatbot with a ready-made framework like Rasa instead of implementing all the NLP processes from scratch.

In the following chapter we will look at the fundamentals of chatbots including some history, integration patterns and common features in the chatbot implementation pipeline. In Chapter 3 we will cover some natural language processing methods that are used in retrieval-based and in generative chatbots. Then in Chapter 4 we will implement our own retrieval-based chatbot which supports basic chit-chatting and data fetching actions. Finally, in Chapter 5 we will look at some chatbot performance evaluation metrics and use them to evaluate our own chatbot model.

2. CHATBOTS

Chatbots are computer programs which communicate with humans through natural language (Behera et al., 2021; Baez et al., 2021). The main function of chatbots is to offer an interface between a user and a service or multiple services. Chatbots can handle many tasks depending on where in the software architecture the interface is implemented in. On a commercial website these tasks might include fetching product information, troubleshooting a faulty product or booking a flight. Chatbots can also be deployed to assist businesses and professionals in tasks regarding data analytics and database querying.

A literature review by Sari and others (2020) found that chatbots are expected to be fast and accurate but they should also support human-like conversation by including both business-related and conversational datasets in their model. As was found in the same study, business owners believe in the benefits of integrating chatbots into their products and websites but chatbots shouldn't be left responsible of the entire business. Instead, chatbots should be working as an assistant to the seller.

The interaction between a user and a chatbot can be achieved through many channels. When dealing with virtual assistants like Siri on an iPhone or Amazon Alexa, the interactions happen mostly through voice commands. Correspondingly, chatbots found on more traditional websites would communicate with the user almost exclusively through text. Although the users may access chatbots through both text and voice commands, the user input is parsed into text before it is passed in to the chatbot.

In the following sections we will cover the brief history of chatbots as well as some of the chatbot integration patterns. We will also look at the most essential features in the chatbot implementation pipeline.

2.1 History

The title of the first chatbot ever created goes to ELIZA which was created in the Massachusetts Institute of Technology (MIT) in 1966 (Kapočiute-Dzikiene, 2020). It was a retrieval-based chatbot which asked questions based on keywords in user input. Other retrieval-based chatbot implementations were to follow and chatbots like PARRY and A.L.I.C.E emerged in the years 1972 and 1995 respectively. (Sutoyo et al., 2019) All of the previously mentioned chatbots were based on string matching and querying based on keywords.

The next major development step from simple retrieval-based chatbots was the introduction of recurrent neural networks (RNN) (Sutoyo et al., 2019). RNNs outperformed previous retrieval-based implementations but they still had problems handling longer input sequences. Another step up from RNNs was the introduction of long short-term memory (LSTM) models and gated recurrent unit (GRU) models but they too couldn't hold the context in longer sentences (Singh & Mahmood, 2021).

The issues with these sequential models were resolved by the invention of attention-based transformers. Transformers are trained with massive datasets, and they provide a language model that generalizes into multiple tasks well. Transformers can then be retrained to fit application specific tasks. (Singh & Mahmood, 2021) Retraining transformers was a good solution since building new transformers for every application would be computationally expensive.

Many new fields of use emerged for chatbots as they became better and more reliable at NLP tasks. These areas of integration are described in greater detail in the following section.

2.2 Pipeline

Looking at the general flow of data between a user, a chatbot, and connected services, we can define a general pipeline which most chatbots follow. The pipeline consists of a natural language processing unit, a dialogue manager, and one or more data sources. The NLP unit can be further divided into a pre-processing unit and a natural language understanding unit. (Mohamad Suhaili et al., 2021; Baez et al., 2021) If the chatbot is generative instead of retrieval-based the natural language generation module is also included in the NLP unit. The structure of the full implementation pipeline is pictured in Figure 1.

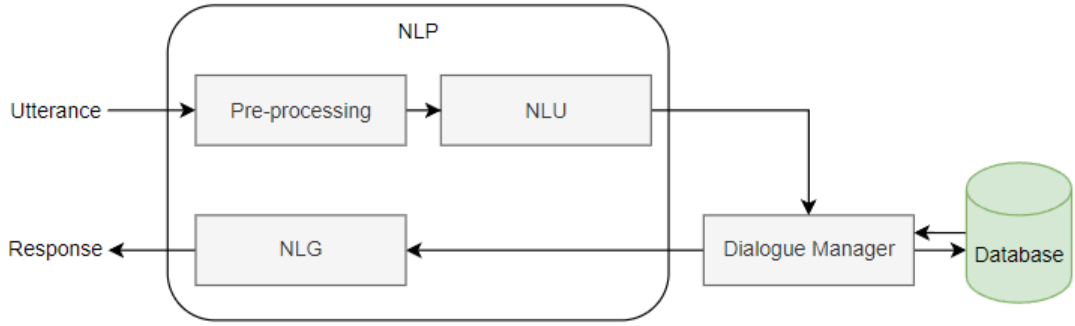

Figure 1. *A chatbot implementation pipeline (based on Mohamad Suhaili et al., 2021).*

To fully understand the division of responsibilities in the chatbot implementation pipeline we need to first define intents and slots. Intents are assigned by the developer, and they correspond to available actions like data fetching or question answering. They can take in parameters which are also referred to as slots. Slots are required to complete actions. (Perez-Soler et al., 2021; Singh et al., 2019) For example, a chatbot could have an intent for fetching weather data for a given date from an external API. This intent could have a slot for the date of which the user wants information from. Providing the chatbot with an intent and required slots through one or more utterances would result in an API call, and the chatbot would respond with the required weather information.

The first step in the implementation pipeline is receiving textual user input which can also be referred to as an utterance. Utterances are either text or voice commands which map to intents and slots (Baez et al., 2021). For example, the utterances “What is the weather like?” and “How is the weather?” map to the same intent of fetching current weather information. Slots such as the date are included in the utterances. The utterances “What is the weather like today?” and “What is the weather like tomorrow?” both contain the information for filling the date slot.

The user utterance is passed to the natural language understanding unit through a pre-processing step where the text is processed to a more suitable form for NLU processes. The pre-processing step may include one or more ways of text processing such as tokenization and lemmatization. Some of the pre-processing steps are further explored in Chapter 3.1.

The pre-processing step is followed by the NLU unit which is responsible for extracting intents, slots, and context from the input. Natural language understanding includes many technologies like RNNs and transformers but also more general algorithms for named entity recognition and semantic analysis. The parameters extracted by the NLU model are passed on to a dialogue manager.

The dialogue manager holds the context of the user interaction and guides the conversation through pre-defined steps. A dialogue manager can be connected to one or more data sources or APIs. It triggers actions based on the current state and context of the interaction and gets processed arguments directly from the NLU unit. (Mohamad Suhaili et al. 2021) Calling actions in the dialogue manager may result to structured or unstructured data which needs to be presented to the user through a logical response in natural language. The response generation module is responsible for producing an adequate response.

There are two main categories of chatbots based on the approach to response generation: retrieval-based and generative chatbots (Mohamad Suhaili et al., 2021). Retrieval-based chatbots have pre-defined answers and only need an NLU unit for choosing the best response based on user utterances. Retrieval-based chatbots benefit from technical simplicity as well as high accuracy responses but the answers have only a little variation which makes the interactions with these chatbots less human-like.

Generative chatbots use sequence-to-sequence models such as RNNs and transformers to transform the user input into a suitable response (Mohamad Suhaili et al, 2021). While generative chatbots produce more variation in the responses and thus achieve more human-like interactions, they require large amounts of training data and can easily result in less accurate responses.

2.3 Integration patterns

A study by Baez and others (2021) describes eight integration patterns for chatbots based on the three-tier architecture of distributed systems. These integration patterns picture the different places in existing software systems where a chatbot interface can provide services to users. The integration patterns do not only cover the interactions between customers and businesses but also the use of chatbots in internal business processes.

The eight integration patterns found in the literature study are a stand-alone agent, an in-app assistant, a GUI agent, an API caller, a business process interface, an IoT interface, a query engine, and an information retriever (Baez et al., 2021). These integration patterns cover the type of intents, actions, and dialogue control in either data, logic, or

presentation level of three-tier architecture of distributed systems. The integration patterns are pictured in Figure 2 along with the implementation layer and services they are connected to.

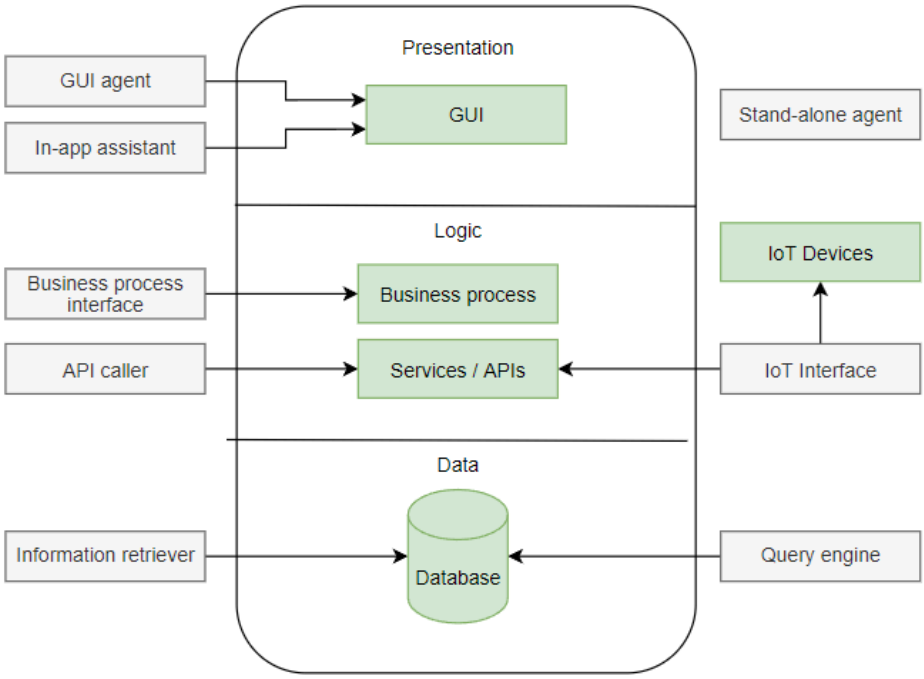

Figure 2. *Chatbot integration patterns (based on Baez et al., 2021).*

In-app assistants, IoT interfaces and information retrievers are some of the better-known integration patterns and they can often be seen on commercial platforms. In-app assistants are integrated into the graphical user interface of mobile applications and websites. They revolve around frequently asked question (FAQ), navigation, exploration, and chit-chatting tasks and often provide product information to the user based on public product data as well as previous question-answer pairs asked by other users.

IoT interfaces provide access to hardware often through voice commands (Baez et al., 2021). The intents are limited to the actions available on each connected device. IoT interfaces have gained popularity through the introduction of smart home devices. Some good examples of IoT interfaces are Google Home and Amazon Alexa.

Conversational information retrievers can be used for document summarization and for fetching answers to questions regarding documents. While information retrievers may not be as well-known as the previously mentioned integration patterns in commercial use, they have applications for example in the medical sector where there is a lot of documented medical data that needs to be summarized.

2.4 Personalization

Personalizing the interaction between a chatbot and a user is an essential part of providing a satisfying user experience. It can be achieved by tailoring the generated responses to each user based on information like current location of the user, time of day, and previous conversation history. This data is then used for constructing a context for a conversation.

Chatbots aren't limited to form the interaction context solely on information regarding system information like time and location. The user's mood can be extracted from utterances through sentiment analysis and emotion detection as part of the NLU model in the chatbot implementation pipeline (Kulkarni et al., 2021).

A study by Behera and others (2021) suggests a definition of a cognitive chatbot being the combination of an intent-based and a flow-based chatbot. Such a chatbot would be able to track the flow of the conversation while collecting the required intents and slots to perform actions. A cognitive chatbot would be able to provide a more personalized conversation than a pure intent-based chatbot.

One problem with personalizing chatbot-customer interactions is the collection and safekeeping of user data. As highlighted in a book by Singh and others (2019), there are many pitfalls in chatbot development regarding data protection. The General Data Protection Regulation (GDPR) requires that any identifying user information can't be exposed when training or using the chatbot. In the context of personalization, the use of previous conversations between the user and the chatbot may contain such personal information and thus especial care needs to be taken when designing the personalization features of a chatbot.

2.5 Evaluation metrics

It is difficult to objectively evaluate the performance of chatbots due to lack of standardized implementation patterns and evaluation methods. Some evaluation metrics that have been used in previous studies are F1-Score, BLEU, precision, accuracy, recall and human evaluation (Mohamad Suhaili et al., 2021). The overall performance of a chatbot can also be evaluated through business metrics such as ROI, customer satisfaction and retention rates after the chatbot has been tested in a real environment.

The evaluation metrics directly linked to NLU models are precision, accuracy, recall and F1-Score. Accuracy implies the overall classification success of a model by calculating how many test cases are correctly classified. While it is an evaluation metric that is easy

to understand, it runs into problems when the test cases do not cover all the actual interactions that happen in production with actual users. Accuracy is also insensitive to class imbalance since a high accuracy with a class that has many examples can overshadow a smaller class performing badly (Mohamad Suhaili et al., 2021).

The F1-Score measures the harmonic mean of precision and recall of a classifier (Caldarini et al., 2022). In the context of chatbots, F1-Score can be described as a balance between correctly retrieved responses and the chatbot's ability to find all relevant responses to a given utterance.

The bilingual evaluation understudy, also referred to as BLEU, is another automated evaluation metric. It is calculated as a number of overlapping word sequences in a computer-generated response compared to a human-made reference response. (Caldarini et al., 2022)

Lastly, human evaluation can be used to assess the overall performance of a chatbot in a production environment. Unlike the other evaluation metrics human evaluation is highly subjective. Human evaluation can be performed by asking for feedback on the conversation flow and user experience from a test group or even from real customers.

3. NATURAL LANGUAGE PROCESSING

As a subfield of Artificial Intelligence (AI), natural language processing is used for understanding and generating natural language. It is in the heart of the chatbot implementation pipeline. NLP can further be divided into three processes. Pre-processing converts input text into a machine understandable format, natural language understanding model takes the pre-processed text and extracts the meaning behind the input, and finally a natural language generation unit converts the retrieved data into a response in natural language. (Sari et al., 2020; Mohamad Suhaili et al. 2021)

In this section we will look at some pre-processing steps, NLU methods and NLG in more detail. Since natural language understanding is present in almost all chatbot models while natural language generation is only present in generative chatbots the emphasis of this section is in NLU.

3.1 Pre-processing

The raw user input is pre-processed before it is handed in to the NLU unit. Pre-processing is necessary because neural networks can only work with numerical vectors instead of raw text as input. Good pre-processing also increases the accuracy of the trained models by removing information that might not be needed for some retrieval tasks.

Pre-processing includes multiple techniques of which only some are applied at a time. The selection of which pre-processing method to use depends highly on the models selected for the NLU unit and the available training data for the chosen implementation.

Some of the most common pre-processing steps are tokenization, normalization, stemming and lemmatization. Tokenization refers to the process of splitting input text into smaller units such as words or characters. The normalization step includes removal of some characters such as punctuation to simplify the format of input words. Stemming and lemmatization try to convert variations of a word into a single root word. Stemming isn't bound to actual root words. As an example, the words "cat" and "cater" might be stemmed to the base unit of "cat" as they both contain the same identifying characters. Lemmatization tries to achieve the same result, but it looks up the actual root words from a dictionary. Variations like "cats" and "cat" would map to the base word "cat". (Abd Elaziz et al., 2020)

3.2 Natural language understanding

NLU is responsible for extracting the contextual content from preprocessed user input. More specifically, NLU methods are used to gather the intent and entities from a user utterance. It is a quickly evolving branch of artificial intelligence research and is often regarded as the main function of various chatbot libraries and frameworks.

The main tasks in the NLU process are named entity recognition, sentiment analysis and emotion extraction. Named entity recognition is the process of extracting predefined datatypes such as dates, numbers, and place names from input text. Sentiment analysis refers to the process of categorizing utterances to good, bad, and neutral based on the frequency of similarly categorized words in the sentence. (Sari et al., 2020) Emotion detection is similar to sentiment analysis, but it is focused on extracting the actual mood of the user (Kulkarni et al., 2021). Sentiment analysis and emotion detection can be used in chatbot personalization to build a fuller context of the ongoing interaction with a user.

Transformers such as the Bidirectional Encoder Representations from Transformers (BERT) are commonly used for NLU tasks. (Singh & Mahmood, 2021; Kulkarni et al., 2021). Although transformers require massive amounts of data for training, they can be included into smaller applications like chatbots through transfer learning.

3.3 Natural language generation

Chatbots can respond to intends either by using predefined responses or generating new responses. Natural language generation is important in the latter case. When chatbots are used in customer service they tend to use predefined responses to keep the quality of responses high and accurate. Having dynamic response generation also requires a lot of training data which is not available in the case of small businesses.

Some neural networks such as RNNs and LSTMs as well as transformers are based on sequence-to-sequence learning. Sequence-to-sequence models take a vector of words or other pieces of text and then transform them into an output sequence of varying length. Training sequence-to-sequence models requires a massive amount of training data as input-output pairs of these vectors. (Kapočiute-Dzikiene, 2020)

4. IMPLEMENTING A CHATBOT

In this section we will implement a simple retrieval-based chatbot. Our chatbot is going to handle basic chit-chatting tasks as well as fetching weather data for a given date. The weather data will not be fetched from a real API, but the implementation can be easily modified to suit that need later. All the source code will be available on GitHub (Autti, 2022). As can be detected from previous chapters, training a good retrieval-based chatbot requires a good NLP pipeline. To simplify the development process of our application we will be using a chatbot framework called Rasa.

4.1 Rasa

Rasa is an open source chatbot framework for creating chatbots with the Python programming language. It provides the NLU model based on developer defined intents and entities and allows the developer to write responses or trigger custom actions when those intents are activated. All the intents, entities, and stories can be configured by through YAML files, but Rasa also supports defining custom actions through Python classes. We will be using Rasa version 3 in this project which is the most recent version of Rasa at the time of writing. (Rasa, 2021)

4.2 Implementation

Rasa requires a developer to provide examples of user utterances corresponding to each intent. It then uses these examples as training data for the NLU model. Our chatbot is going to have three intents. Two of the intents serve basic chit-chatting actions by greeting the user when the user says hi, and by saying goodbye when the user utters goodbye. The third intent is going to trigger an action for fetching weather data.

Now that the intents have been specified, we need to provide training examples for building the NLU model. Rasa automatically provided the project with examples for greet and goodbye intents so the only examples that need to be defined are for the ask weather intent. The action to fetch weather data requires a time slot to extract information for a correct date. The time entity is included in multiple examples. A special feature extractor called Duckling is ran on a separate docker container. Duckling is pretrained to extract time entities from text so there is no need to cover all possible examples of stating the time. The word “weather” is also often mentioned in the examples. However, to have the

model cover more variations of potential user input some examples were provided without the word “weather” too.

Lastly, we need to define the action for fetching weather data. The action is implemented as a Python class which can handle custom application logic. The action reads the time slot and then sets slots for a random weather status such as “+24 °C, clear” or “+10 °C, overcast”. It also sets a slot for formatted date which allows the chatbot to rephrase the input date in a more readable format. Any code can be run within custom actions and a real weather API could be called from the action instead of returning a random weather status.

Rasa allows the user to configure stories which mimic conversations between a user and the agent. The stories describe all the steps in a conversation between a user and the chatbot. The stories are then used for training the NLU model to respond correctly when it faces a similar order of user input. Each step in a story can be an intent which was triggered, an action that was called, or a slot that was requested or filled. Our model is going to have only one story which covers all the steps from responding to user’s chit-chat and fetching the weather data on request.

All that is left to do is to train the model after which the chatbot can be accessed through the Rasa shell command line interface. A transcript of a short conversation with the chatbot is shown in Figure 3.

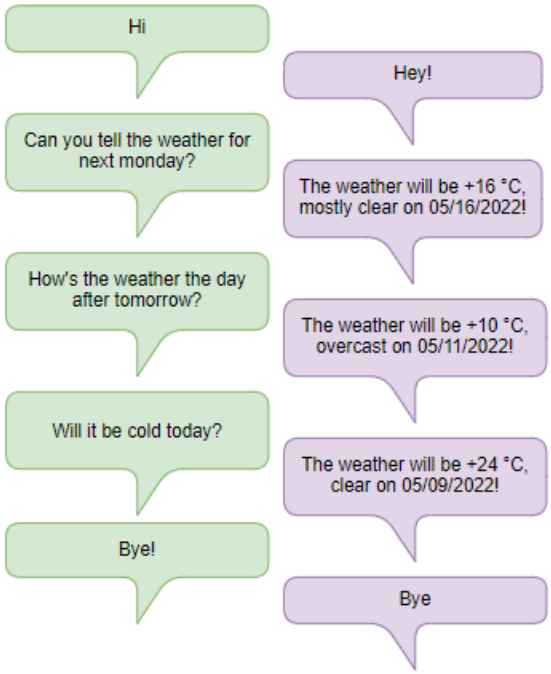

Figure 3. *A transcript of chatting with our chatbot. The user’s utterances are marked green and the chatbots responses are marked purple.*

As can be seen from the Figure 3, our model is able to detect the fetch weather intent even when presented with sentences the chatbot hasn’t seen before. The action for fetching the weather data returns a random weather status along with the date specified by the user utterance.

5. EVALUATION

As was found in the Chapter 2.5, evaluation of chatbots is not easy. Fortunately, Rasa allows the user to write test stories in a similar way to the stories used in training. The developer can then run the tests and Rasa will calculate the F1-score, precision, and accuracy for the model. To evaluate our model a test story was created. The resulting scores were 0,875 for F1-Score, precision, and accuracy. While evaluating rasa models through test stories can provide numerical evaluation scores, they might not cover all the test cases of potential user input. Since our chatbot model is very simple not many tests could be written.

Perhaps a better way to evaluate our model is through human-evaluation. Based on the transcript shown in Figure 3, our chatbot can correctly identify intents for both chit-chatting and for fetching the weather. Also, the time slot seems to be extracted correctly from the user input.

Based on the numerical evaluation scores provided by Rasa, and on human-evaluation, it can be concluded that our chatbot is able to do all the tasks we defined in a sufficient manner. However, our evaluation methods don't cover many test areas outside the domain of our simple tasks, so a re-evaluation needs to be made after any future changes to our chatbot model.

6. CONCLUSION

In this bachelor's thesis we found natural language processing to be a key part of the chatbot implementation pipeline along with a dialogue manager. It was also found that the user utterances go through pre-processing, natural language understanding models and potentially natural language generation models within the NLP part of the pipeline.

In addition to clarifying the development process and overall pipeline of chatbot implementation, we discovered that chatbots have multiple integration patterns based on the three-tier architecture of distributed systems. While some of the better-known chatbot types such as an IoT-interface and an In-App assistant were covered, there were also some less-known integration patterns such as a business process interface and a query engine.

In the topic of natural language processing, we discovered some pre-processing methods like tokenization and lemmatization to have been used in chatbot development. Similarly, the key functions of natural language processing were found to be named entity recognition, sentiment analysis and emotion detection. Natural language generation was found to be handled mostly with sequence-to-sequence solutions such as transformers and LSTMs.

Lastly, we build our own chatbot with an open-source framework called Rasa. From the project we learned that developing a chatbot with a framework is simple compared to implementing a chatbot from scratch with some very complex NLP methods.

REFERENCES

Abd Elaziz, M., Al-qaness, M. A. A., Ewees, A. A., & Dahou, A. (2020). Recent Advances in NLP: The Case of Arabic Language. 1st ed. 2020. Springer International Publishing.

Autti, P. (2022). Source Code, viewed 9 May 2022, <https://github.com/Pertsaa/rasa-weather-api-example>.

Baez, M., Daniel, F., Casati, F., & Benatallah, B. (2021). Chatbot Integration in Few Patterns. IEEE Internet Computing, 25 (3), 52–59.

Behera, R. K., Bala, P. K., & Ray, A. (2021). Cognitive Chatbot for Personalised Contextual Customer Service: Behind the Scene and beyond the Hype. Information Systems Frontiers, 1 (1).

Caldarini, G., Jaf, S., McGarry, K. (2022). A Literature Survey of Recent Advances in Chatbots. Information (Basel), 13 (1), 41.

Kapočiute-Dzikiene, J. (2020). A domain-specific generative chatbot trained from little data. Applied Sciences, 10 (7), 2221.

Kulkarni, A., Shivananda, A., & Kulkarni, A. (2021). Natural Language Processing Projects: Build Next-Generation NLP Applications Using AI Techniques. Berkeley, CA: Apress L. P.

Mohamad Suhaili, S., Salim, N., & Jambli, M. N. (2021). Service chatbots: A systematic review. Expert Systems with Applications, 184, 115461.

Perez-Soler, S., Juarez-Puerta, S., Guerra, E., & de Lara, J. (2021). Choosing a Chatbot Development Tool. IEEE Software, 38 (4), 94–103.

Rasa (2021). Introduction to Rasa Open Source 2021, Rasa Technologies, viewed 24 April 2022, <https://rasa.com/docs/rasa/>.

Sari, A. C., Virnilia, N., Susanto, J. T., Phiedono, K. A., & Hartono, T. K. (2020). Chatbot developments in the business world. Advances in Science, Technology and Engineering Systems, 5 (6), 627–635.

Singh, A., Ramasubramanian, K., & Shivam, S. (2019). Building an Enterprise Chatbot: Work with Protected Enterprise Data Using Open Source Frameworks. Berkeley, CA: Apress L. P.

Singh, S., & Mahmood, A. (2021). The NLP Cookbook: Modern Recipes for Transformer Based Deep Learning Architectures. IEEE Access, 9, 68675–68702.

Sutoyo, R., Chowanda, A., Kurniati, A., & Wongso, R. (2019). Designing an Emotionally Realistic Chatbot Framework to Enhance Its Believability with AIML and Information States. Procedia Computer Science, 157, 621–628.