University of Applied Sciences and Arts Northwestern Switzerland
School of Engineering

Chatbot in English Classrooms

Encourage Negotiations of Meaning

Bachelor's Thesis of

Kelvin Louis, 8th Semester
kelvin.louis@students.fhnw.ch

Nicola Cocquio, 8th Semester
nicola.cocquio@students.fhnw.ch

University of Applied Sciences and Arts Northwestern Switzerland (FHNW)
School of Engineering
Computer Science

Supervisors

Manfred Vogel
manfred.vogel@fhnw.ch

Ivo Nussbaumer
ivo.nussbaumer@fhnw.ch

Brugg-Windisch, 19.03.2019

Abstract

Chatbots have become more prevalent in recent years, due to increasing demand as well as improvements in the fields of natural language processing (NLP) and machine learning. Within the field of education research, past studies have rightfully questioned the usefulness of chatbots as means of acquiring a foreign language. A review of the relevant literature shows that the applied chatbots were rule-based and limited to chitchatting in an open-domain.

In this thesis we propose an alternate approach to using chatbots in English as a second language (ESL) classrooms. We evaluated the current state of technology to develop a machine learning-based chatbot capable of detecting errors in the students' language. The chatbot's domain is confined to interacting with the students in a room reservation roleplay exercise. Prerecorded transcripts of ESL student interactions were used to derive wordings of intents and utterances which served to train and test the chatbot's machine learning models. Since misspellings are the most common errors in ESL students' language, a language error detection was introduced into the chatbot's architecture, providing additional feedback to the students and thereby mitigating repetitive errors.

To test the performance of our solution, usability tests and a survey were conducted. The usability tests showed that the bot understands a majority of ESL students' inquiries and is capable of responding in a comprehensible manner. The survey results revealed that the intent recognition model could have benefitted from a wider range of wordings. In a separate PhD project [1], the added value of applying our chatbot in ESL classrooms will be assessed from a language learning perspective.

Glossary

Term	Definition
AIML	Artificial Intelligence Markup Language is an XML-based standard.
API	An application programming interface offers a defined set of methods to be called by software.
CNN	Convolutional neural network – A feed-forward neural network architecture.
CRF	Conditional random field – A classification model using conditional probabilities.
DFS	Depth-first search is an algorithm used to traverse graph-like data structures.
DSL	Domain-specific language
ESL	English as a second language
GEC	Grammatical Error Correction
heteronym	Words with the same spelling, but different pronunciation and meaning
homonym	Words with the same spelling and pronunciation, but different meaning
LSTM	Long short-term memory are neural network units addressing the problem of vanishing gradients in CNN.
NLG	Natural language generation focuses on generating natural language from systems using models or other forms.
NLP	Natural language processing allows analyzing, understanding and generating written or spoken texts.
NLU	Natural language understanding is a subfield of NLP and is used for reading comprehension purposes.
OOV	Out-of-vocabulary, a non-existing word in a specific word vector
RNN	Recurrent neural network - Neural network architecture
SDK	A software development kit allows to develop programs for an existing platform.
SPA	Single-page application is a web application that dynamically rewrites its pages instead of requesting it from a server.
turn	A turn is a request/response cycle between the chatting partners.
XML	Extensible Markup Language allows to define rules encoded in documents, which are human-readable and can be processed by machines.

Author's declaration

We hereby declare that this bachelor's thesis is entirely our own work, in our own words, and that all sources used in researching it are fully acknowledged and all quotations properly identified.

Brugg-Windisch, 19.03.2019

_____	_____
Kelvin Louis	Nicola Cocquio

Table of contents

1 Introduction 1
1.1 Background 1
1.2 Problem statement 1
1.3 Objectives and scope 2
1.4 Outline 2
2 Theory 3
2.1 Chatbots 3
2.1.1 Rule-based approach 3
2.1.2 Machine learning approach 5
2.1.3 Commercial application 6
2.1.4 Application in education 6
2.1.5 Architecture 7
2.2 Natural language understanding 8
2.3 Language error detection 10
3 Analysis 11
3.1 Problem domain 11
3.2 Chatbot capabilities 11
3.3 Language errors 13
4 Evaluation 16
4.1 Chatbot services and frameworks 16
4.2 Spellcheckers 18
5 Implementation 19
5.1 Overview 19
5.2 Client 20
5.3 NLU architecture 21
5.4 Dialog manager architecture 22
5.5 Language error detection 24
5.6 Data generation 25
6 Results 27
7 Conclusions 31
8 List of Tables 32
9 List of Figures 33
10 Bibliography 34
11 Appendix 38
11.1 Room reservation exercise 38
11.2 Complete list intents 38
11.3 Complete list of entities 39
11.4 Complete list of slots 40
11.5 Complete list of actions 41

1 Introduction

1.1 Background

This bachelor's thesis is part of the PhD project of Johanna Oeschger [1]. In her project, Johanna Oeschger aims to analyze the quality of conversations between English as a second language (ESL) students and a chatbot from a language learning point of view. An especially important feature when assessing the quality of conversations in an ESL context are *negotiations of meaning.* Such negotiations occur when a speaking partner does not understand the intent or meaning of an utterance in a conversation and asks for clarification. According to Long, *negotiations of meaning* support the acquisition of a foreign language, because they are a source of feedback for students, urge them to repair unintelligible input which in turn leads to more comprehensible utterances [2].

Johanna Oeschger created two text-based conversational roleplay tasks for ESL students. In the tasks, the students are asked to enquire about available internships or event rooms at a hotel, respectively. Both tasks defined two distinct roles: the role of the client and the role of the service provider. The tasks were administered and evaluated in an initial pilot study by Johanna Oeschger. The participants – second-year students of a commercial vocational school – were randomly assigned into 7 dyads, one student carrying out the role of the client, the other student the role of the service provider. The conversations were held by typing on an instant messaging platform on the students' personal mobile devices. The test run resulted in seven recorded conversations per task. After an initial analysis of these 14 transcripts by Johanna Oeschger, where linguistic and other relevant interactional features were evaluated, it was decided to use the *room reservation* task (see appendix 11.1) and the corresponding collected data for this bachelor's thesis.

1.2 Problem statement

Over the past few years, there has been considerable progress in the field of natural language processing (NLP). This has led to technologies using NLP at their core to become more accessible and widespread – with chatbots being one of these technologies.

The use of chatbots in education is promising, because the technology offers a new and alternative opportunity for students to practice their conversational skills. Actively practicing a foreign language is crucial for becoming proficient at it. Thus, introducing a chatbot in an ESL setting could be beneficial. Previous studies have questioned the use of chatbots, stating that they are not a substantial learning instrument [3]. However, the chatbots applied in these studies were not restricted to a specific domain, nor have they made use of the latest advancements in the field of NLP.

In this thesis we consider the recent advancements of NLP and restrict the domain of the chatbot to the *room reservation* task. Based on these considerations, we aim to answer the following research questions:

- Does the chatbot understand the queries of ESL students and is it able to reply in a comprehensible manner?
- Is it possible to formulate a meaningful clarification request to trigger a rephrase from the student, in case of a misunderstanding?
- Is it possible to detect morphological, syntactical and semantical errors in the student's language?
- Do software development kits (SDKs), cloud solutions or software libraries exist to create a holistic solution in which the aforementioned language errors can be detected?

1.3 Objectives and scope

The first of two main objectives is to implement a prototype of a domain-specific chatbot which is capable of conversing with ESL students. The chatbot should be a viable chatting partner in a written roleplay exercise used in ESL classrooms. Its role is to function as a service provider of a hotel answering inquiries about event rooms. While doing so, the chatbot should encourage *negotiations of meaning* in case of misunderstandings, which in turn should lead to more comprehensible student utterances.

The second objective is to generate the necessary amount of data for Johanna Oeschger's dissertation project. As part of her project, the chatbot will be applied in four different ESL classrooms of a commercial vocational school. The students will be interacting with the chatbot simultaneously. The chat logs will be complemented with a recording of the participants' screens. Additionally, a student questionnaire will be administered before and after the chatbot interaction. So, each conversation between a student and the chatbot must be recorded, uniquely identifiable and exportable in order to match each participant's conversation, screen recording and questionnaire. Neither the screen recordings nor the questionnaires are part of the scope. Using the exported transcripts of the conversations, Johanna Oeschger will analyze the interactions between students and the chatbot to answer her research questions.

Because the exercise simulates a text-based chat conversation as opposed to a phone call, no speech-to-text and text-to-speech components are required.

To ensure that we are not restricted in logging conversations, identifying and giving feedback to a user, no integration into a messaging platform is considered. Instead, a custom client will be developed which is accessible via laptops and supports the latest browser versions.

The features regarding logging, export, identifying a user and possible load tests are not part of the research questions, thus are not considered in the chapters *Results* and *Conclusions*. However, as mentioned earlier, they influence the *Implementation*.

1.4 Outline

This thesis is structured in a way that aides a continuous flow of reading and comprehension. First, the chapter *Theory* introduces the fundamental topics of chatbots and their means of understanding text and detecting errors in language. The following chapters will keep referring to the introduced terms and definitions. Chapter *Analysis* highlights aspects that need to be considered in order to achieve the stated objectives. In *Evaluation* the necessary services, frameworks and libraries that are to be used for the *Implementation* are evaluated. The final two chapters, *Results* and *Conclusions*, measure and reflect on the results of the implemented prototype.

2 Theory

This chapter lays the theoretical foundation of our thesis by introducing the most important technological theories and definitions. Understanding the introduced technologies and terminologies is important to fully comprehend the reasoning behind the analysis, evaluation and implementation.

To create a chatbot that can be applied in English classrooms, it is important to understand what the capabilities and limitations of chatbots are and how they function. The first subchapter introduces two common approaches of chatbots and shows how they are being applied in commercial and educational areas.

The second subchapter introduces natural language processing (NLP) tasks that are relevant for text comprehension using a machine. Finally, important aspects for detecting errors in language are introduced.

2.1 Chatbots

A chatbot is a computer program designed to carry out conversations mainly with humans as their partners. The terms used to define such programs vary. They are also referred to as chatterbots, conversational agents or dialog systems. The following figure shows that chatbots are in fact understood as a subclass of dialog systems and conversational agents [4]. In this paper we will consistently refer to them as chatbots.

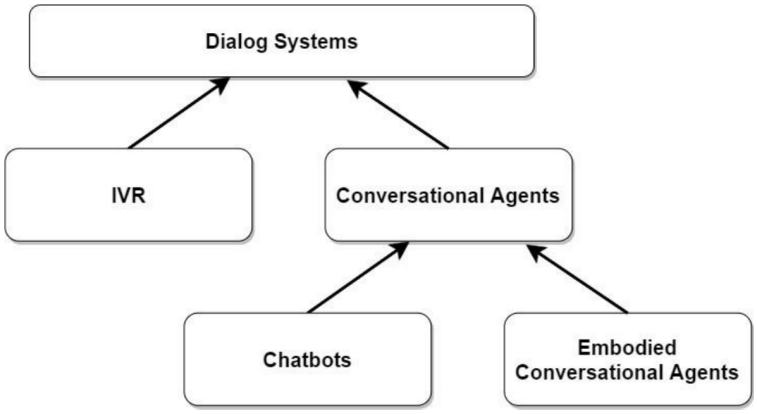

Figure 1: Classification of dialog systems [4].

2.1.1 Rule-based approach

The complexity of conversations a chatbot carries out depends on the domain and the tasks it has to fulfill. A rudimentary specimen might not focus on keeping a chatting partner interested and engaged in a conversation. Its sole purpose is to execute a specific algorithm and providing feedback after execution. Identifying which algorithm to invoke can be determined using simple pattern matching on a large set of predefined rules. In this case, if the received message does not exactly match a rule, no execution will take place, because the chatbot has no means to infer what rule should be applied.

The first chatbot that has ever been released made use of a pattern matching technique. ELIZA, released by Joseph Weisenbaum in 1966 [5], simulated a psychotherapist. Its objective is to keep its human counterpart engaged. ELIZA applies multiple strategies to accomplish this goal [6, p. 19]. For example, ELIZA continues asking questions about the user and their emotional state, in order to keep

them do the talking. Oftentimes ELIZA's question simply consists of a reflection of the user's previous message, where it swaps pronouns like "my" to "your" or "me" to "you" [7, p. 184]. An example of a conversation is given in Figure 2. By echoing keywords or parts of the user's speech and rephrasing them in the form of a question, it gives the impression that it is attentively listening. When the pattern matching algorithm is unable to apply a rule, it changes the topic or gives a neutral response.

U: Well, my boyfriend made me come here.
E: Your boyfriend made you come here.
U: I am unhappy
E: Do you think coming here will help you not to be unhappy?
U: I need some help, that much seems certain.
E: What would it mean to you if you got some help?
U: Perhaps I could learn to get along with my mother.
E: Tell me more about your family.
U: You are like my father in some ways.
E: What resemblance do you see?
U: My father is afraid of everybody
E: What else comes to mind when you think of your father?

Figure 2: Example of a conversation between a user (U) and ELIZA (E) [7, p. 184].

This sole pattern matching approach has its downsides. ELIZA is not capable of remembering parts of the dialog and has no awareness of the conversational context. Its architecture does not contain a dialog manager, a component which keeps track of the state of the conversation (see chapter 2.1.5). This limits the chatbot to basic small talk. Nevertheless, ELIZA inspired a succeeding generation of chatbots building on top of pattern matching.

AIML stands for Artificial Intelligence Markup Language. It is an open source XML-based standard that specifies how patterns and responses for chatbots can be created. This standard was introduced with the creation of Alice, a chatbot first released in 1995 by Richard Wallace [7]. Patterns defined in AIML allow keywords to be extracted and stored in variables. Thus, compared to ELIZA, Alice and other chatbots using AIML have the capability of remembering elements of a conversation. AIML patterns are processed by an interpreter and stored as a tree-shaped directed graph [7, p. 200]. Incoming messages are run through the pattern matching algorithm, which uses backtracking and depth-first search (DFS) to traverse the graph and to reply to the user [7, p. 201].

While the introduction of AIML has improved the creation process as well as the quality of chatbots, it has major disadvantages. "First of all, being a rule based system, a big set of rules need to be built and so a big fraction of time is spent analyzing the possible variation of the sentences instead of leaving it for more important tasks such as focusing on the data available" [8, p. 22]. The pattern matching algorithm will look for a rule that exactly matches word for word. This leads to redundancy, as a rule must be defined for every synonym, spelling mistake and variation of sentence structure. Figure 3 highlights this issue. For every possible synonym of "Hello", a rule has to be defined. Messages such as "Good morning" or "Hey" would otherwise not match.

```
<category>
    <pattern>HELLO</pattern>
    <template>Hi there!</template>
</category>
<category>
    <pattern>HI</pattern>
    <template><srai>HELLO</srai></template>
</category>
<category>
    <pattern>HI THERE</pattern>
    <template><srai>HELLO</srai></template>
</category>
<category>
    <pattern>HOWDY</pattern>
    <template><srai>HELLO</srai></template>
</category>
<category>
    <pattern>HOLA</pattern>
    <template><srai>HELLO</srai></template>
</category>
```

Figure 3: AIML rules to respond to "Hello" highlighting the issue of redundancy [7, p. 195].

Figure 3 reveals another issue. Every synonym is referencing the original rule “Hello” by using the *srai* tag. This allows to reuse the text defined in the *template* tag of “Hello”. In the example above, this means that the chatbot replies with “Hi there!”. As the set of rules becomes larger and more complex, it gets more difficult to trace the list of applied rules on a message.

Maintaining a set of rules, which in some cases reference each other, is not feasible as the set becomes larger. To eliminate the need for rules, a different approach can be taken.

2.1.2 Machine learning approach

As shown in the previous chapter, maintaining a set of patterns to decide how to respond to a message can be troublesome. When a user asks a question or provides information in form of a sentence, it is difficult to create rules that cover all possible variations of such an utterance. Trying to encompass them leads to even larger sets. Even worse, if no pattern exists, the chatbot will not respond as expected. This is especially problematic for goal-oriented chatbots.

Compared to small talk chatbots, whose goal it is to keep a user engaged, goal-oriented chatbots serve a specific purpose. The domain of the conversation is therefore limited. A chatbot who schedules hotel room reservations, does not need to engage in chitchat. Its goal is to provide a service to the user. It needs to identify the intent of a user’s request in order to handle a related task. If it receives a message such as “I would like to reserve a deluxe room for the 28th of September.”, it should be able to classify the intent correctly. However, there are multiple ways to formulate such a request. Depending on the complexity of an intent, there a few to several thousand possible wordings [9, p. 8]. Covering them with rules is a difficult and tedious undertaking. With the help of machine learning, models can be trained to statistically infer what intent applies without the need to exactly match a set of rules.

Statistically inferring which algorithm to execute demands a more sophisticated architecture. Such architectures usually make use of natural language processing (NLP) techniques and machine learning. NLP is a subfield of computer science devoted to analyzing and processing natural language to recognize, understand and generate texts. As we will see in chapter 2.2, where natural language understanding (NLU) is introduced, understanding received texts is of utmost importance when creating chatbots. NLU is not responsible for the generation of responses. To understand a received message, the text needs to go through a pipeline of processing tasks, where it is decomposed into

parts. Once this step is completed, depending on the architecture, machine learning will be applied to classify the intent of the message and to extract important entities. The model of the classifier is trained on a set of predefined messages. Depending on the applied machine learning model, the chatbot can learn from new conversations and adjust its model.

Learning can be achieved through supervision, where the supervisor enters a message yet unknown to the chatbot and corrects its classification if necessary [10, p. 5]. Another form of learning can be attained through reinforcement learning. In this case, there is no supervisor training or correcting the model. Instead, a reward function is used to optimize the model when selecting a proper response to a message [11].

As has been outlined, there are benefits in using machine learning to understand a user's utterance. There are however also some challenges. In a journal article by Facebook AI Research, the authors point out that creating models for a specific domain requires a lot of handcrafting [12]. This problem was also highlighted in Martino Mensio's master thesis, where he states: "The bot prototype required a domain specific training corpus to correctly categorize the intents of the user and to identify the entities involved with their role. No existing publicly available collections of annotated sentences existed for the bike sharing domain, so it has been required to personally collect it. Reaching good size and quality of collected data is not easy and some circular dependencies in the workflow can occur, […]" [8, p. 99]. So, it becomes clear that enough data for training and testing machine learning models needs to be available in order to create useful intent classifiers.

Nevertheless, the most useful applications of dialog systems such as digital personal assistants or bots are currently goal-oriented and transactional [12] and they are increasingly being applied in commercial areas.

2.1.3 Commercial application

Chatbots are used in many different areas: commerce, banking, insurance and travel, to name a few. The list of companies covering existing services through chatbots is growing. There are multiple reasons for this. First, by virtue of cloud providers, it has become more accessible to create a chatbot. Instead of developing a chatbot from scratch, these providers hide the implementation details behind well documented SDKs and application interfaces (APIs). Second, there is no need for setting up a hosting infrastructure as this is taken care of by the cloud provider. Third, the chatbots can be seamlessly integrated into popular messaging platforms that their customers are already using. Finally, and most importantly, they can have a noticeable impact on saving time and costs, as pointed out in a report by Juniper Research [13].

As a consequence to the increasing use of chatbots, more people have been exposed to chatbots and their acceptance has grown. According to a study conducted by PIDAS and ZHAW [14], 70% of the questioned have interacted with chatbots or are willing to do so in the future. This is a 30% increase compared to the previous year's study. Unsurprisingly, the acceptance rate is higher for the younger demographic (ages 18 to 30). 53% of them have already used chatbots in the past [14].

2.1.4 Application in education

An effective way of learning a foreign language is through conversing with a native speaker. In foreign language classrooms, however, the teacher is often the only proficient conversation partner available to learners. Due to the student/teacher ratio, opportunities for one-to-one dialogs are rare in the language learning classroom. A chatbot could potentially be a useful alternative to provide additional speaking opportunities for students, due to its capability of holding multiple conversations

simultaneously and its availability. Thus, it is unsurprising that several attempts of applying chatbots in the educational realm have been made previously.

Already in the early 2000s, several studies were conducted to examine the effects of chatbots in language classrooms. All these chatbots followed the rule-based approach (see chapter 2.1.1). The research team either used versions of Alice [15, p. 35], [16] or created their own chatbot building on similar principles of AIML [17, p. 183], [18]. This is not surprising due to AIML being state-of-the-art and the lack of alternatives at the time. As a result, the chatbots were limited to chitchatting in an open-domain dialog on a finite set of rules. In the study of Shawar and Atwell [15, p. 38], the set was insufficient for the chatbot to generate satisfying responses related to a mentioned topic. Jia [19, p. 6] discovered similar shortcomings and ascribed them to the rule-based pattern matching approach. "Developing open-domain conversational dialogue systems is difficult, since the huge variety of user utterances makes it harder to build knowledge resources for generating appropriate system responses" [20, p. 334]. Another issue was the chatbot's inability to make use of linguistic knowledge because, unlike machine learning-based chatbots (see chapter 2.1.2), it had no awareness of the conversational history [15, p. 38].

No studies were conducted in recent years evaluating the impact of machine learning-based chatbots in the field of education. Even a recent book evaluates exclusively AIML-based systems [21]. Studies devoted to the novelty effects of chatbots in a language course [3], [22, p. 463] used Cleverbot, which does not apply a machine learning model yet [23] and is not restricted to a specific domain.

2.1.5 Architecture

A generic architecture of a modern machine learning-based chatbot consists of an NLU, a dialog manager and a response generator. The responsibility of the NLU is to map spoken or written language to intents and entities. Chapter 2.2 will elaborate further on NLU. The dialog manager tracks the context of the conversation. A response is composed by the response generator, a natural language generation (NLG) component, using the current state of the dialog manager. Figure 4 shows how these components interact with each other.

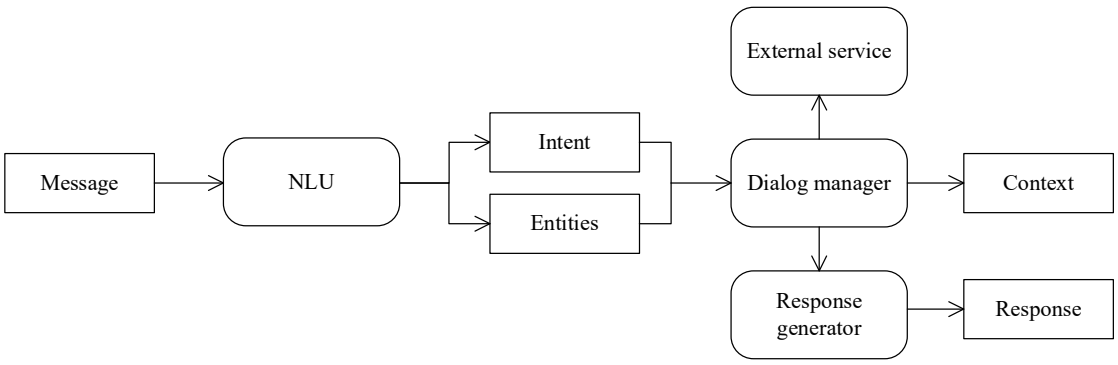

Figure 4: A generic machine learning-based chatbot architecture.

The dialog manager tracks the conversation by remembering its central aspects. Simple architectures could simply record the extracted entities. More advanced dialog managers however retain the history of uttered intents and responses as well. These aspects are stored and referred to as the context. Using the context, the dialog manager steers the conversational flow [24, p. 795] by telling the response generator how to respond. Depending on the architecture of a response generator, it may pick from a set of templates or generate the language completely by itself. The templates may contain placeholders, which can be filled with data stored in the context. The template-based approach is the most common strategy. For example, Pandorabots' Artificial Intelligence Markup Language (AIML) makes use of it [25], [26, pp. 124–131]. Dynamically generated responses on the other hand can be

achieved with neural networks [27], [28], [29, pp. 87–95]. However, this strategy is rarely used in chatbots and is more commonly applied in areas where reports, documentations or summaries are generated [30].

Without a dialog manager, multi-turn dialogs would not be possible. A multi-turn dialog is a sequence of request/response cycles, where the chatting partners keep referring to entities that were mentioned previously in the dialog. These mentioned entities can be retrieved from the context and used in responses. In this way, a chatbot may for example mention the user's name at the very end of a transaction by uttering "Thank you for your order, *Marc*", even if the user has stated his name at the very beginning of the conversation. The dialog manager also allows the chatbot to successfully react to utterances such as "I will order *the other one instead*", because it retained the alternative object mentioned in the previous turns of the conversation [24, p. 795].

The dialog manager also enables users to converse in a non-linear fashion. In linear dialogs, every conversation follows the same sequence, whereas non-linear conversations branch off at different moments, allowing users to obtain the same information in various ways. Non-linearity is an important property of chatbot conversations, because it makes them feel more natural to users.

Besides accessing the context to retrieve information, external services can be queried to answer questions or to complete transactions. These services usually provide application programming interfaces (APIs). This procedure enables chatbots to provide Q&A features, where the bot searches a knowledge base before answering [31], or to complete tasks such as booking a flight or finding a rental bike nearby [8].

2.2 Natural language understanding

Since the early days of computing, humans have relied on special input devices to interact with computers. As time goes by, so do input devices, as illustrated by the transition from punch cards to modern touchscreen devices, for instance. Natural language understanding, a subfield of natural language processing, is intended to use naturally spoken and written language as input for a computer program.

There is no definitive architecture for an NLU. Based on the intended use case, different components are combined to master a specific task. For this reason, we detail common NLP components instead of a complete architecture. A specific NLU architecture is described in chapter 5.3.

A tokenizer is applied at an early stage in the NLU pipeline. It splits sentences, words or characters into tokens. Simple rule sets split by whitespace and punctuation characters, whereas sophisticated rulesets consider floating point numbers (0.12), abbreviations (e.g.) or dates (12.11.2018) as one token [32].

Tokens can be assigned a word type (noun, verb, etc.) by using a part-of-speech (POS) tagger. More fine-grained approaches distinguish the tense of a verb, the noun number (singular or plural) and the different forms of adjectives (adverb, comparative, superlative, etc.). Hence, different POS tag conventions exist. Smaller sets such as the *Universal POS Tagset* [33] contain 12 tags, whereas the *Brown Corpus* includes 85 defined tags [34]. An example of tagged words can be seen in Figure 5, where the word type is indicated below the corresponding word.

Dependency parsing extends POS tags with syntactic labels, which describe the dependencies between the tokens. Modern greedy, transition-based dependency parsers incorporate neural network classifiers to achieve a balance of speed and accuracy [35]. Figure 5 shows the dependencies between the words as arrows with their corresponding labels.

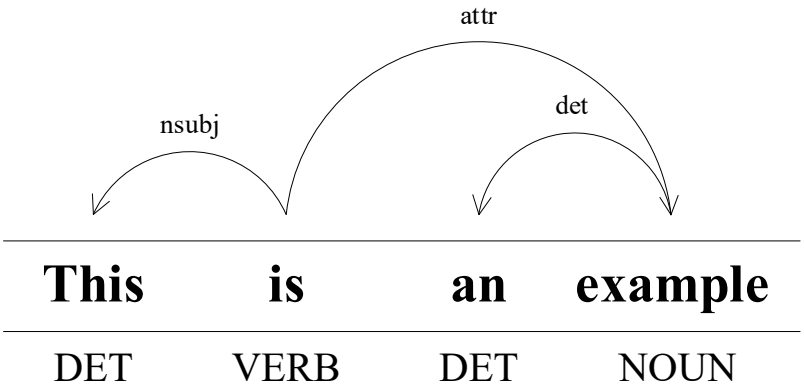

Figure 5: Example sentence parsed with spaCy, a popular NLP library.

Lemmatization is an NLP task that considers the morphology of words. Morphology studies how words are formed from its parts (e.g. word stems, prefixes, suffixes). In the process of lemmatizing, each word is reduced to the *lemma*, the dictionary or root form. To identify the root form, a language-specific vocabulary and morphological analysis is conducted. A simpler form is *stemming*, a heuristic-based approach, which tries to form the root by removing the word ending [36, p. 30]. Both of them are widely used in text mining, summarization or analysis to limit the variation of word forms and to make them more consistent [37]. Table 1 shows an example of how lemmatization works.

Original	the	boy's	cars	are	different	colors
Lemmatized	The	boy	car	be	differ	Color

Table 1: Lemmatization example sentence [36, p. 30].

The named entity recognition (NER) component identifies real world objects in tokens. Recognized objects typically include people *(PERSON),* organizations *(ORG)*, countries *(GPE)*, dates *(DATE, TIME)* or amounts of something *(MONEY, CARDINAL)*.

A: I'd like to reserve a hotel room.
B: That should be no problem. May DATE I have your full name, please?
A: My name is John Sandals PERSON .
B: Hello, Mr. Sandals PERSON . My name is Michelle PERSON . What days DATE do you need that reservation, sir?
A: I'm planning to visit New York GPE from Friday DATE , April 14 DATE until Monday, April 17 DATE .
B: Our room rates recently went up. Is that okay with you, Mr. Sandals PERSON ?
A: How much per night are we talking about?
B: Each night TIME will be $ 308 MONEY .

Figure 6: Example dialog after spaCy tagged the recognized entities [38].

The majority of NER implementations use supervised machine learning. The accuracy of these systems heavily depend on the labeled data, also known as a corpus [39, p. 7].

Word embeddings attempt to represent language in a numerical form. Simple embeddings have a finite vocabulary as a one-dimensional vector and each column represents a single word. A more

complex example is the pre-trained model GloVe [40] with 50 to 300 dimensional vectors available per embedded word. Mikolov et al. showed the interesting phenomenon of similarity (e.g. boat – ship) and relatedness (e.g. boat – water) of words in a vector space [41, p. 5].

The intent classifier is an essential component of chatbot and voice assistant applications. It maps different wordings of a request to a common intent. This reduces the types of requests to a finite set of intents the application supports.

2.3 Language error detection

As Coniam concluded in his study [42], all of the evaluated chatbots are not yet suitable for a conversational ESL setting. Vocabulary or grammatical errors in the learners' language confuse the chatbots and halt the conversation. At the same time, Jia [17, p. 188] noted that the conversation partners tend to focus on the flow rather than following grammar and spelling rules. These two facts combined present a challenge to the development of a chatbot for an ESL setting: While the chatbot relies on correct language input, learners tend to disregard language rules when communicating with chatbots.

According to Lyster and Ranta [43, p. 56], feedback that is provided in the form of explicit corrections is the least effective. This is because the language learner is not actively confronted with their error and tends to simply accept the teacher's rephrasing. One goal of our chatbot is to detect and inform the students of language errors and try to encourage them to correct or rephrase the message, rather than simply marking the error or providing a corrected version. To keep the conversational flow the learner should however not be prompted for every mistake.

The types of errors that might occur in the chatbot conversations are partly depending on the ESL students' first language. Students with German as their first language, for instance, already know a rule set for articles and might make mistakes by applying the same rules to English (e.g. "My sister is doctor."), but still perform better than Russian native speakers who do not have articles in their first language [44, p. 45]. In the Cambridge Learners Corpus First Certificate in English (CLC FCE) dataset, 2488 essays written by learners from 16 different language backgrounds were corrected and annotated. In an analysis of the CLC FCE dataset by Leacock et al. [45] spelling was the most common type of error, followed by word choice, preposition and determiner errors.

Spellchecking is available in most word processors, messaging applications and web browsers. Most implementations are not context-aware and use the Levenshtein distance to measure the similarity between words. This can lead to false positives, as seen in Table 2.

Original	In the room for 200 people you *habe* round tables for 6-8 peoples with *chears*. […]
Corrected	In the room for 200 people you *have* round tables for 6-8 peoples with ***cheers***. […]

Table 2: Example of a false positive correction of a spellchecker. Chairs is the correct substitution.

In a context-aware spellchecker, *chairs* would have a higher precedence because it is more common in a sentence with *tables*.

Error detection and correction is a popular shared task, such as CoNLL [46], [47] and HOO [48], [49]. The introduction of the CoNLL shared tasks states: "This task is challenging since for many error types, current grammatical error correction systems do not achieve high performance and much research is still needed." The submitted papers provide a wide range of solutions to the problem.

3 Analysis

Previous chapters have laid the theoretical foundation and established a shared understanding of the key definitions. The goal of the present chapter is to analyze the problem domain and derive the capabilities of the prototype to be developed. These capabilities must satisfy the objectives described in the problem statement (see chapter 1.2).

First, the problem domain is analyzed by highlighting the most important aspects of it. Understanding the problem domain is necessary to derive requirements for the chatbot and the language error detection. The capabilities of these modules are analyzed separately in their respective subchapters.

The data used for the analysis stems from the *room reservation* task of the pilot run (see chapter 1.1).

3.1 Problem domain

The software system's target audience are commercial trainees who are undertaking their basic vocational training at commercial vocational schools in Switzerland. In the German-speaking part of Switzerland, these schools are known as *Kaufmännische Berufsfachschulen*. Basic vocational training programs are generally taken up by young people after graduating from secondary school. Completing secondary school in Switzerland is usually achieved after 9–12 years of formal education. Thus, a student of a vocational school is typically around 16–18 years old. Their acceptance rate for using chatbots is assumed to be higher than in other age groups (see chapter 2.1.3). The students are not English native speakers, thus are categorized as ESL students. For their basic vocational training programs, commercial vocational schools target the language proficiency levels B1 Intermediate according to the Common European Framework of Reference for Languages [50, p. 1].

In an ESL classroom, students tend to have a limited vocabulary and commit more grammatical errors than native speakers. They also tend to compensate their lack of vocabulary by using words from their native language vocabulary. Both aspects are highlighted in Figure 7.

```
S:  i have to organisate a room for our companie
S:  and i read about your rooms
S:  can i have some informations about?
H:  Sure!
H:  We hav three rooms for you Event. Room A is for 270 people, With
    ha big stage and a fix delivery. The technical (ausstatung) are
    modern with eh big with wand
```

Figure 7: Chat dialog from the room reservation exercise of two ESL students highlighting language errors.

This is an excerpt of a conversation that was held during the test run between two ESL students who were roleplaying a room reservation scenario. Besides the lack of vocabulary, determiners and preposition mistakes are the most common errors of English learners [46, p. 1].

3.2 Chatbot capabilities

In the seven logged conversations of the *room reservation* exercise (see chapter 1.1), a mean of 39 messages were sent and 12.9 turns were taken. A turn is a single request/response cycle between the chatting partners. Per turn a mean of three messages were sent. The mean duration of a conversation was 21.5 minutes. In total 273 messages were sent.

When analyzing the interactions in these conversations, we identified 29 different types of intents, with each uttered intent containing one or multiple entities. In fact, we further noted that the students used up to nine different entities within intents. As a consequence, the chatbot would not only need to

be able to respond to a specific intent but do so in varied ways, depending on the used entities. Table 3 shows a reduced list of intents that were derived from the *room reservation* exercise pilot run. A complete list can be found in the appendix (see chapter 11.1).

Intent	**Examples**
affirm	Yes | Yes, please | That is correct
ask_for_options	What rooms do you have? | Do you have meeting rooms?
ask_for_room	Tell me about room Alpha | What about the others? | What about Alpha?
ask_for_room_equipment	Is it possible to show a presentation in Gamma?
greet	Hi there! | Hey, how are you? | Hello
greet+ask_for_options	Hello. What rooms do you have?
reserve_room	I want to reserve room Beta | I want to book this room

Table 3: A reduced list of intents derived from the room reservation exercise of the pilot run.

The following table shows a reduced list of entities that are used within the intents. A full list is in the appendix (see chapter 11.3).

Entity	**Examples**
budget	1'500.- | 1200 | CHF 1500 | 1600 | $1200
date	25.04.2019 | 24th of May | 1st December 2019
name	Yves | Max Muster | Mr. Muster
nr_of_people	150
room	Alpha | Beta | Gamma

Table 4: A reduced list of entities derived from the room reservation exercise from the pilot run.

Additionally, it was determined that the chatbot must support multi-turn dialog structures. It is essential in a conversation to know what subject or object the conversational partner is currently referring to. If these properties can be inferred from context, the conversational flow becomes more natural. In Figure 8 both students refer to "Room A" without explicitly mentioning it. Using the term "other rooms" only works if both parties know which specific room they are currently talking about.

```
H: Room A is for 270 people, With ha big stage and a fix delivery.
   The technical (ausstatung) are modern with eh big with wand.
H: The price for this room is 1400.-
S: Does it have a projector to present the staff a presentation?
H: Yes!
S: ohh cool
S: are the other rooms cheaper?  bigger or smaller?
H: Room B is 900.- for 130 people inside and ouside +60 peple
H: Room C is 1100.- for 200 people
```

Figure 8: Chat dialog of two ESL students referring to an already uttered word in room reservation exercise.

Another important aspect concerns the user language. A study found that rule-based chatbots experience difficulties if words are misspelt [42, p. 105] or in an incorrect order [42, p. 109]. Rule-

based chatbots need to incorporate these mistakes into their rule sets in order to understand a user's message. It is however difficult to capture all of these errors using pattern matching. For our domain this is problematic, because ESL students do in fact tend to commit these types of errors, as seen in Figure 9. According to the same study [42, p. 109], a chatbot needs to handle these kinds of errors in order to be useful to ESL students. As noted in chapter 2.1.2, machine learning models use probabilities to classify the intent of a received text. Therefore, erroneous text input is less likely to be misunderstood, as long as there are other significant features. Furthermore, some utterances depicted in Figure 9 contain multiple clauses, which makes pattern matching even more complicated [42, p. 111]. For these reasons, solely a machine learning-based approach is considered in this thesis.

```
S1: but can we use there in room c the audio and visual functions
S2: Does it have a projector to present the staff a presentation?
S3: ehm 150 persons and ih need e room who i can show a presentation and i can make
    littel groups for speek
S4: That‘s great, we should be able to do some things in one of these room, to present
    our presentations, disscus in small teams, good atmospher, and the guests should
    feel impressed but not with extreme things, we have maximum CHF 1500 to pay
S5: We need in the room something to show a presentation. But also we need to can make
    little groups. Have one of this room a good athmosphere? We try to impress our
```

Figure 9: Examples from the room reservation exercise of users asking for room equipment.

3.3 Language errors

The issue of the language errors ESL students commit was introduced in chapter 2.3. In this chapter, the committed mistakes from the pilot run and the first usability tests of the chatbot are analyzed. When the first usability tests were carried out, the chatbot did not yet contain a language error detection module. All utterances of users who carried out the client role (see chapter 1.1) were considered. The found errors were classified by type. The error typology was taken from the *CoNLL-2014 Shared Task on Grammatical Error Correction* [47, p. 3]. Table 5 lists the resulting subset of error types. Punctuation and capitalization were disregarded for the type *Mec*, as a lack of correct punctuation and capitalization are common practice for the chatting environment. Filler words such as "ah", "ehm" and "uhm", also known as hesitation words, were ignored as well. People's names were substituted with the word "unknown".

Type	Description	Example
Mec	Spelling (ignoring punctuation and capitalization)	This knowledge [**maybe relavant** → may be relevant] to them.
Vm	Verb modal	Although the problem [**would** → may] not be serious, people [**would** → might] still be afraid.
V0	Missing verb	However, there are also a great number of people [**who** → who are] against this technology.
Nn	Noun number	A carrier may consider not having any [**child** → children] after getting married
ArtOrDet	Article or determiner	It is obvious to see that [**internet** → the internet] saves people time and also connects people globally.
Pform	Pronoun form	A couple should run a few tests to see if [**their** → they] have any genetic diseases beforehand.
Prep	Preposition	This essay will [**discuss about** → discuss] whether a carrier should tell his relatives or not.

SVA	Subject-verb agreement	The benefits of disclosing genetic risk information [**outweighs** → outweigh] the costs.
Vform	Verb form	A study in 2010 [**shown** → showed] that patients recover faster when surrounded by family members.
Vt	Verb tense	Medical technology during that time [**is** → was] not advanced enough to cure him.
WOinc	Incorrect word order	[**Someone having what kind of disease** → What kind of disease someone has] is a matter of their own privacy.
Wci	Wrong collocation/idiom	Early examination is [**healthy** → advisable] and will cast away unwanted doubts.
Wform	Word form	The sense of [**guilty** → guilt] can be more than expected.
Rloc	Redundancy	It is up to the [**patient's own choice** → patient] to disclose information.

Table 5: Subset of error types from the CoNLL-2014 Shared Task on Grammatical Error Correction [47, p. 3].

Out of 240 messages, 41.7% of them contained errors. These 100 messages had a total of 203 errors. Most mistakes (85.7%) were committed in the pilot run. This is likely to be attributed to the testing environment, where students were chatting on mobile devices with each other. Nevertheless, as depicted in Figure 10, spelling errors were the most common errors in both testing environments. They made up 45.4% of errors in the pilot run and 34.5% in the usability tests. This clear margin indicates that the error detection module of the chatbot should contain a spellchecker.

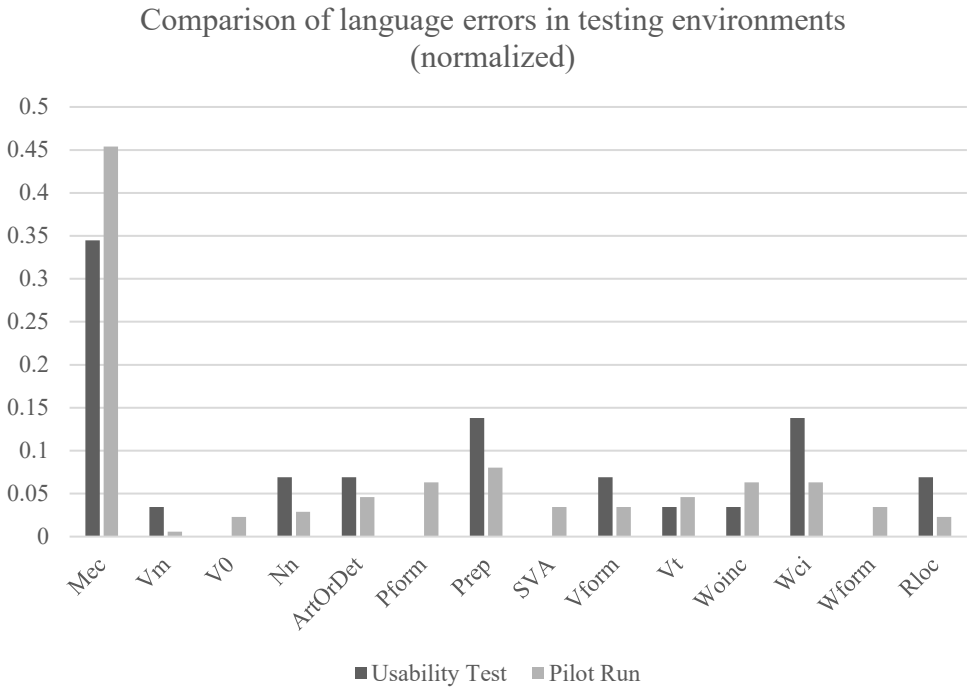

Figure 10: Comparison of language errors in testing environment (normalized).

This high percentage of spelling errors is a potential problem for the chatbot's intent classifier. Depending on the architecture, a machine learning-based classifier could still correctly classify the user's intent if the text contains some misspelled words. For example, by using the remaining

correctly spelled words, a bag-of-words model would still have a high enough frequency to ascribe them to an intent. A rule-based approach could overcome this challenge by including common errors into the set of rules. This however does not cover all possible errors and would complicate the set of rules even more.

The analysis also shows that some students repeatedly made the same errors. This happens if students are unaware of their mistakes. Introducing a language error detection module could provide helpful feedback to students, diminishing the chance of repeating the same errors. A chatbot could mimic the role of a teacher by highlighting grammatical errors to its conversation partner. In Jia and Chen, a chatbot equipped with a grammar checker was not as frequently used as the one without [17, p. 188]. This comes as no surprise as being corrected after each turn might be considered unbearable by the student. Therefore, language correction should be carefully applied to keep the student engaged and motivated for the conversation.

4 Evaluation

The previous chapter outlined what essential aspects a chatbot and a language error detection module must cover. These aspects are evaluated in this chapter. Firstly, chatbot services and frameworks are analyzed. The evaluation focuses solely on NLU components, testing the intent and entity recognition capabilities. Rule-based chatbots are not considered for the reasons stated in chapter 3.2. Furthermore, since the previous chapter revealed that misspellings are the most common language errors committed by students, publicly available spellcheckers capable of detecting these errors are evaluated.

4.1 Chatbot services and frameworks

As stated previously (see chapter 3.2), chatbots need to be able to correctly classify intents and its entities. Dialog managers must support multi-turn conversations.

Three offerings were considered for evaluation. Google Dialogflow [51] and Microsoft LUIS [52] are both cloud services and therefore do not require any self-hosted infrastructure or installation. They lack however the possibility to inspect or adapt the source code. Dialogflow contains an NLU and a dialog manager. LUIS is simply an NLU, thus does not contain a dialog manager. However, Microsoft offers Azure Bot Service [53] where a dialog manager can be developed using an SDK. The third offering we considered for evaluation is Rasa [54], a popular open source project, which offers an NLU pipeline and a dialog manager named Rasa Core. Rasa NLU has a configurable Tensorflow pipeline that allows developers to customize the model parameters for a given domain. The evaluation only covers the mentioned NLUs.

All dialog managers listed above support non-linear multi-turn dialogs (see chapter 2.1.5) in various ways. In Rasa Core and Dialogflow these dialog managers follow a machine learning-based approach and have to be trained. Using the SDK provided by Microsoft Azure Bot Service, multi-turn dialog support has to be manually developed.

The corpus used for evaluating the NLUs consisted of 400 generated queries containing five intents and four different entities. The procedure used to generate the data is discussed in chapter 5.6. The set of intents represent a basic and shortened conversation from the *room reservation* exercise (see chapter 1.1). Table 6 lists the evaluated intents. Each NLU is trained with 300 queries and the other 100 queries are used for the evaluation.

A recent evaluation by the TU Munich [55, p. 6] concluded that there is no absolute NLU and that the performance depends on the used corpus. Because the authors shared the automated evaluation scripts online, we have been able to evaluate the services with data from our evaluation corpus.

Intent	**Entity**	**Number of training / testing queries**
greet	Name, Company	60 / 20
provide_name	Name, Company	60 / 20
ask_for_room	Number of people	60 / 20
provide_nr_of_people	Number of people	60 / 20
Reserve_room	Room	60 / 20
Total		**300 / 100**

Table 6: Intents and entities in evaluation corpus.

The results of the evaluation are depicted in Figure 11. The used F_1 score is a standard measure in classification tasks [56]. LUIS and Rasa NLU both classified all intents and entities successfully. Dialogflow had issues with similar looking queries. For example, it classified "Good morning, my name is Mark Muller. I am in an internship at ABCD Corp" as an *Introduction* instead of a *Greeting.* Furthermore, it returned empty entities for certain queries. Unfortunately, Dialogflow does not provide confidence scores for its intents, which prevents a more detailed error analysis. Because of the test results and the limited troubleshooting capabilities, we decided against Google's Dialogflow.

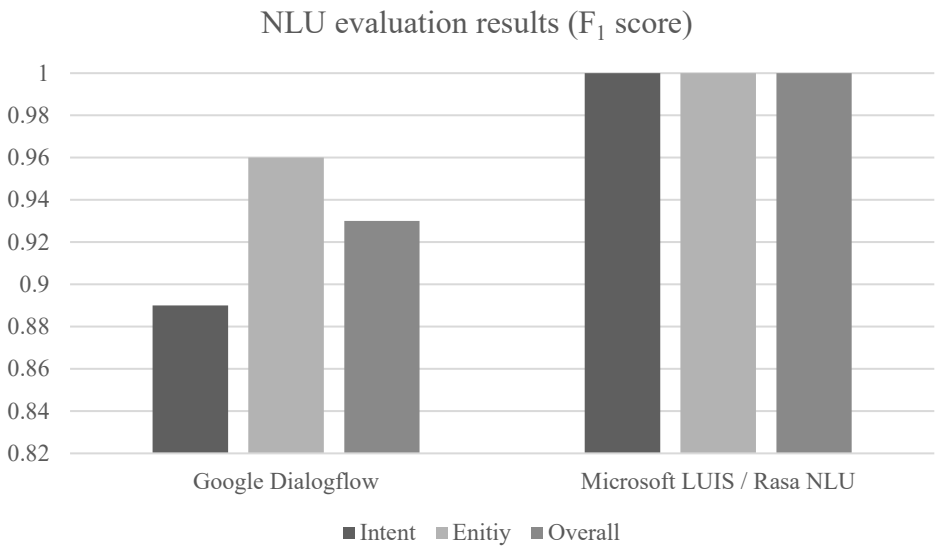

Figure 11: NLU evaluation results (F_1 score).

As LUIS and Rasa NLU performed equally well, we considered additional criteria for the evaluation, which can be seen in Table 7.

Criteria	**LUIS**	**Rasa NLU**
Source availability	Closed Source	**Open Source**
Cost	**None (within limits)**	Hosting cost
Rate limit	5 transactions per second	**No limit**
Data processing location	USA/EU	**CH**
Installation	**None**	Needs installation
Customize pipeline	Not possible	**Possible**
Multiple models	**Yes**	**Yes**
Web GUI	**Yes**	Third party project

Table 7: Additional criteria for the NLU evaluation.

An open source project has major advantages compared to closed source software. Even though time has to be invested in setting up environments, the possibility of inspecting, troubleshooting and customizing the pipelines favors an open source approach. For these reasons, we decided against Microsoft LUIS and selected Rasa's NLU and dialog manager for this project.

4.2 Spellcheckers

The error analysis in chapter 3.3 indicated that a spellchecker pointing out language mistakes to the students would be a helpful addition to the chatbot. As mentioned in chapter 2.3, automatically correcting a mistake might however not be the most effective measure for the student's learning process. Rather, the student should be encouraged to rephrase their erroneous statements, after having been made aware of them. In other words, the chatbot should detect an error but not correct it. Following this argumentation, our evaluation of spellcheckers did not consider the spellcheckers' suggestions for word corrections. Likewise, the evaluation did not account for available Grammatical Error Correction (GEC) systems [57]–[59], as these systems provide the corrected version of an erroneous sentence without knowing what types of errors have been detected.

We evaluate the following open source spellcheckers: Hunspell [60], SymSpell [61] and LanguageTool [62]. Hunspell and SymSpell are single word spellcheckers and hence have no awareness of the context in which a word is used. LanguageTool is configured to use their provided Google Books n-gram language model [63], enabling it to find contextual errors. Compared to the other spellcheckers, LanguageTool's rich set of rules allows it to find other types of errors as well. For an unbiased evaluation, only misspellings are considered.

As a basis for the evaluation, the same data and definitions from the language errors analysis are used (see chapter 3.3). The data was enhanced by labeling the word positions where spelling errors are expected to occur. Additionally, a frequency lookup table was generated containing keywords used in the context of the *room reservation* exercise. The purpose of the lookup table is to increase the accuracy by introducing context to context-unaware spellcheckers such as Hunspell. Using the table, these spellcheckers should be able to detect contextual errors such as "*witch → which*" or *"tablet → tables"* given that neither "witch" nor "tablet" are referenced in the lookup table. Even though both are correct English words, they are deemed incorrect in this context, because the intended word was inadvertently misspelled as a homonym. The code for the evaluation can be found in the GitHub repository [64].

Table 8 lists the calculated scores of the spellchecker evaluation script. When detecting errors, it is important to obtain a low number of false positives (FP). FPs indicate that the spellchecker incorrectly detects an error. Since missing errors is more preferable to the user than raising false alarms, the precision and $F_{0.5}$ scores are the most important metrics, as they favor lower FPs [56, p. 3].

Spellchecker / Score	Accuracy	Recall	Precision	F_1	$F_{0.5}$
Hunspell	**0.98**	0.83	**0.99**	0.90	**0.95**
Hunspell + Context	**0.98**	**0.88**	0.96	**0.92**	0.94
SymSpell	0.97	0.86	0.86	0.87	0.87
LanguageTool*	**0.98**	0.83	**0.99**	0.90	**0.95**

*Table 8: Scores of the spellchecker evaluation. * indicates that some misspellings were classified as other types, which lead to incorrect false negatives. They were manually corrected.*

The results show that integrating a context-aware lookup table increases the recall while decreasing precision. This leads to a lower $F_{0.5}$ score. It does not improve accuracy either. Hence, the integration of the lookup table can be omitted. SymSpell and the context-aware Hunspell detect more misspellings (higher recall) but produce more FPs (lower precision). LanguageTool and Hunspell perform the best, as their precision and $F_{0.5}$ scores indicate. Either one of them would make a good choice for detecting spelling mistakes. However, the prospect of identifying additional error types favors LanguageTool.

5 Implementation

Previous chapters laid the foundation for the implementation. The analysis highlighted important aspects of the problem domain and capabilities of the chatbot. A defined set of frameworks and tools were evaluated for the implementation. This chapter details how these components work and are put together to form a functioning chatbot.

First, a brief overview of the architecture is given. The following subchapters explain the components in more detail. The final subchapter explains how the necessary data was generated for the machine learning-based components.

The code of the implementation is located on the publicly available GitHub repository [64].

5.1 Overview

The user interacts with the chatbot using the client, a web application served by the nginx server. The chatbot consists of six services. Each service is provided via a Docker container. Figure 12 depicts the services and how they interact with each other. Boxes in bold represent Docker containers. Messages from the client are sent as events to the chatbot using Socket.IO. The nginx server redirects incoming and outgoing events via a reverse proxy. It is the only exposed service of the chatbot. The conversational logger persistently stores the conversations and provides an export for the transcripts.

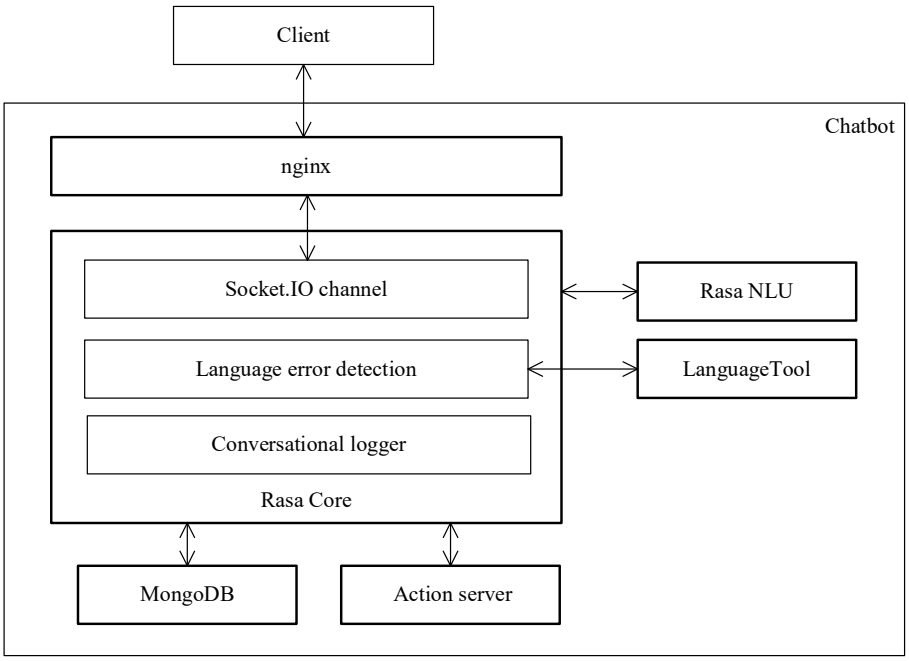

Figure 12: Overview of the chatbot architecture. The bold boxes represent Docker containers.

5.2 Client

The client is single-page application (SPA) built on top of React, a popular JavaScript library for building user interfaces. Using React, the entire user interface is composed of separate components. The communication between the components strictly follows the principle of unidirectional data flow described by the Flux architecture. This principle is realized using the library Redux. In Redux the entire application state is stored in a global store. Changes to the store are dispatched via actions. Actions can be dispatched by components or incoming events from external services. Reducers intercept the incoming actions and map their changes to the state inside the store. Changes to the store lead to partial redraws of components, if they subscribed to the store. Figure 13 depicts the described flow.

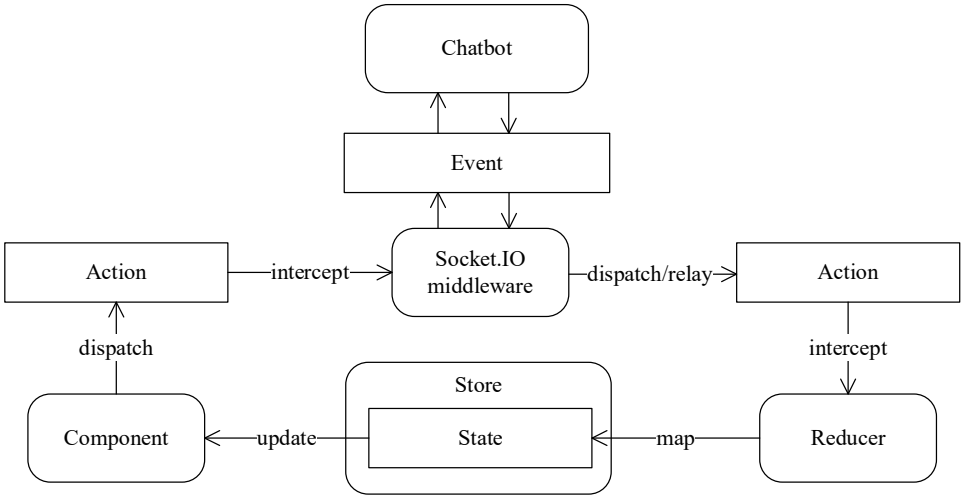

Figure 13: Unidirectional data flow within the client.

As mentioned earlier, the client communicates with the chatbot in form of events. These events are sent via Socket.IO and contain a payload in form of JSON. The table below lists all the events exchanged between the two parties.

Event	Description
connect	Creates a connection between client and chatbot.
user_uttered	A message written by the user. It contains the text, a unique id and the participant id.
bot_uttered	A message generated by the chatbot. It contains the text and a unique id.
bot_found_errors	A list of language errors the chatbot found analyzing a message of a user.
disconnect	Closes the connection between client and chatbot.

Table 9: A list of events sent between client and chatbot.

Composed messages of the user and incoming messages from the chatbot are stored inside the store. If the client receives a *bot_found_errors* event, the list of errors will be appended to the existing message in the store, which leads to a redraw of that message.

Before users start interacting with the chatbot, they need to be authenticated. Authentication is done by entering a unique *participant id*. The entered code is compared to statically deposited list in the client. No stringent security measures are necessary.

5.3 NLU architecture

Rasa NLU offers a set of 20 components. These can be combined to an NLU pipeline. Furthermore, it provides an API to build custom components. Each component declares what kind of data it requires and provides. We are using a customized version of the Rasa NLU Tensorflow pipeline. The developers of Rasa recommend this approach for our use case [65]. Figure 14 depicts the data flow of the customized pipeline.

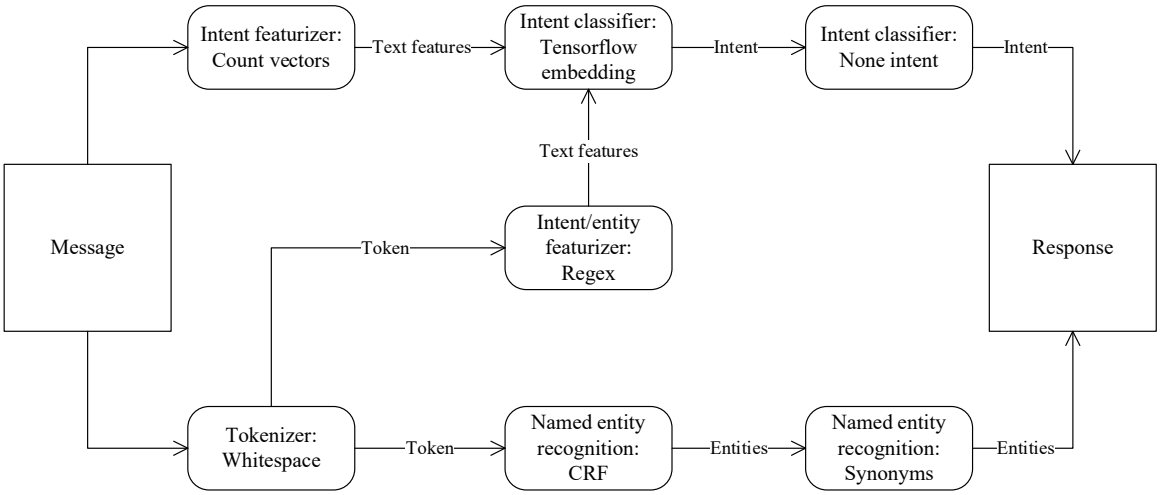

Figure 14: Visualized data flow between NLU components in the pipeline.

The intent classification is implemented with Tensorflow embeddings, which are trained on a one-hot encoded vector (text features) consisting of the domain vocabulary and the results of supplied regular expression patterns. To generate this domain vocabulary, *CountVectorizer* of sklearn is used. This is a one-dimensional vector representing every word seen in the training data.

These text features are the input layer for an adapted StarSpace [66] neural embedding model with two additional hidden layers (256 and 128 neurons) and an embedding layer with 20 neurons. The output are one of 29 intents. If the input consists of mostly out-of-vocabulary (OOV) tokens, the classifier falls below a configured threshold. This results in an undefined intent which our custom *None intent* classifier tries to label based on the entities found. This improves the detection of user messages containing only names or booking dates.

For *Named entity recognition*, the pipeline performs a tokenization of the message with a simple whitespace ruleset. A probabilistic model called conditional random field (CRF) [67] is trained with various subsequences of our labeled data. It calculates the probability of the entity *name* given the sequence “I am”, which, in this example, is very high. A list of all featured entities is provided in the appendix (see chapter 11.3). *Synonyms* is a standard component to condense multiple found entity values into one. We do not use this capability.

The response that the NLU returns is a structured representation of the user message. An example of it is depicted in Figure 15. It consists of the top-ranking intent, the detected entities, the full intent ranking of all intents with a confidence above zero and the original supplied text.

```
{
    "intent": {
        "name": "greet+provide_name",
        "confidence": 0.9047530889511108
    },
    "entities": [
        {
            "start": 12,
            "end": 17,
            "value": "David",
            "entity": "name",
            "confidence": 0.9912837177405575,
            "extractor": "ner_crf"
        }
    ],
    "intent_ranking": [
        {
            "name": "greet+provide_name",
            "confidence": 0.9047530889511108
        },
        {
            "name": "provide_name",
            "confidence": 0.2864721417427063
        },...
    ],
    "text": "Hello, I am David"
}
```


Figure 15: An abridged response object of Rasa NLU.

5.4 Dialog manager architecture

The dialog manager of the chatbot uses Rasa Core. The interacting components of the dialog manager are depicted in Figure 16.

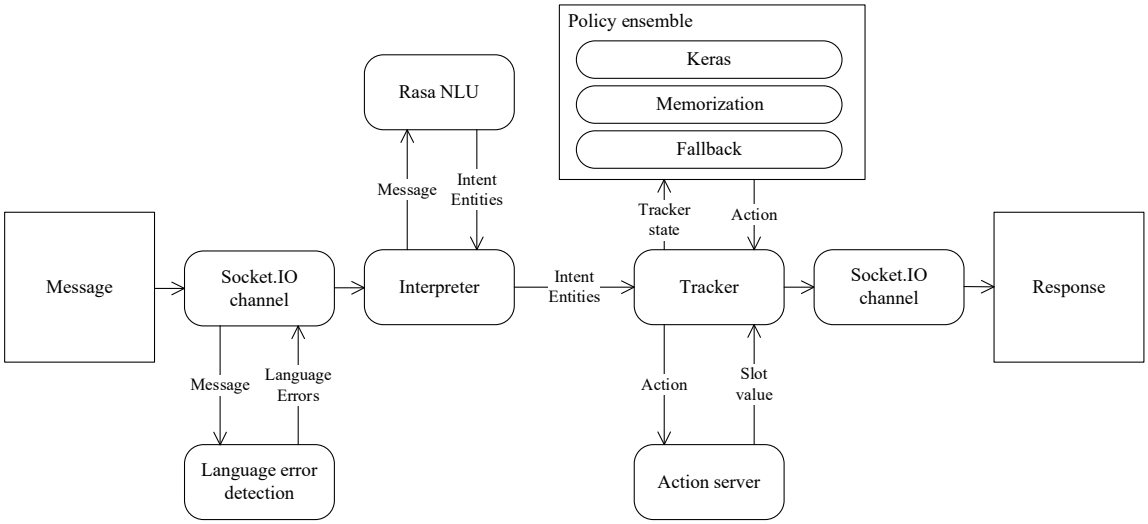

Figure 16: Visualized data flow between Rasa Core components.

The *interpreter* processes the incoming *Socket.IO channel* messages and connects the dialog manager to the NLU.

The *tracker* component, as the name indicates, keeps track of the state of every user. The state includes user messages, executed actions of the bot, and slot values.

Actions are tasks that a bot runs in response to a user message. There are three types of actions. Listening for a user message or running a fallback option, if the bot does not have high enough confidence, are referred to as *default* actions. *Utter* actions allow the bot to send messages to the user.

Custom actions can be used to query an API or change the state of the tracker. These *custom* actions run on a separate *Action server* component. We use these custom actions to fill slots in the tracker component. A complete list of actions is provided in the appendix (see chapter 11.5).

Slots are the long-term memory of the chatbot. Slots can be filled and are exposed as simple key-value pairs. They are important for multi-turn conversations and play a key role in the prediction of the next action, as will be demonstrated later. Each mentioned entity is automatically stored in a slot. However, for the entities *budget* and *nr_of_people* categories representing ranges are stored in one-hot encoded form, instead of their raw integer value (see chapter 11.3). This optimization leads to better predictions. Additionally, we use two custom slots *current_room* and *topic* to retain what the user is currently talking about. The slots are filled by custom actions which are invoked on certain intents. Setting these slots allows the bot to respond to messages such as "what about the others?", even when no explicit mention of the topic and rooms in question are present.

In order to select the appropriate action to a user message the chatbot needs a brain. This is encapsulated in the policy ensemble, a stack of policies. A policy decides what the next action should be, based on the input and its implementation. Our stack of policies consists of the *Keras policy*, *Memorization policy* and *Fallback policy*.

Figure 17 is a simplified representation of what the *Keras policy* model takes as input and produces as output. Max history *h* is a configured numerical value, which states how much of the conversational history should be considered for the prediction. P_i is a row vector and is created after receiving a user message. P_0 represents the most recent message. Its features are made up of the predicted intent and entities of the NLU, the previous action it executed and the current state of the slots. The features are all one-hot encoded, resulting in 141 features. A complete list of intents, entities, actions and slots are provided in the appendix (see chapter 11). The *Keras policy* uses the P_0 vector, which represents the most recent state of the conversation, and takes the previous *h-1* vectors it constructed to create the input matrix *I* for the model.

$$\mathrm{h} = 8 \tag{1}$$

$$\mathrm{P}_{Entities} \in \mathbf{R}^{1\times 9} \qquad P_{Intents} \in \mathbf{R}^{1\times 29}$$

$$P_{PrevActions} \in \mathbf{R}^{1\times 74} \qquad P_{Slots} \in \mathbf{R}^{1\times 29} \tag{2}$$

$$P_i = \left[P_{Entities}, P_{Intents}, P_{PrevActions}, P_{Slots}\right] \tag{3}$$

$$I \in \mathbf{R}^{h\times \mathrm{len}(P_i)} \tag{4}$$

$$I = \begin{bmatrix} P_0 \\ P_1 \\ \vdots \\ P_{h-1} \end{bmatrix} \tag{5}$$

$$O \in \mathbf{R}^{1\times \mathrm{len}(\mathrm{Actions})} \tag{6}$$

Figure 17: Simplified representation of the input and output the model of the Keras Policy.

The policy uses a long short-term memory (LSTM) based recurrent neural network (RNN) implemented in Keras. This type of neural network allows the eight pervious messages (matrix *I*) to influence the current prediction [68, p. 164]. Comparatively, the convolutional neural network (CNN) in the NLU component predicts the intent solely on the most current message.

The prediction is an output vector *O*, which contains the confidence scores for 74 possible actions. The action with the highest score is considered as the next action.

Next in the stack is the *Memorization Policy.* During training it generates a hashed lookup table of the same input matrices as the *Keras Policy.* If it encounters the same hash, the prediction of the *Keras Policy* will be overwritten with a confidence score of 100%.

If the policies above have confidence scores lower than a configured threshold of 40%, the *Fallback Policy* will utter a fallback message prompting to rephrase the previous statement. Otherwise, the NLG simply selects a matching utterance out of a list of templates. The occurring placeholders are replaced with their slot values and the message is sent to the user.

5.5 Language error detection

The evaluation of spellcheckers (see chapter 4.2) showed that the spellchecking capabilities of LanguageTool were on par with Hunspell. Since LanguageTool can detect other error types as well, we decided to use it for our prototype.

Per default, LanguageTool parses a text on a set of 2,216 rules [69]. Additionally, a compatible language model can be provided, which is based on the Google's Books Ngram database [70]. The language model considers n-grams up to *n=3*. We integrated LanguageTool as a standalone HTTP server running in a Docker container.

Figure 18 shows a simplified version of the language error detection process. An incoming user message is sent to the Rasa NLU service to extract entities. The message is subsequently passed to the LanguageTool service. Its response provides a list of detected errors. Among other things, an error contains data about the type, position in the text, suggestions and a user-friendly message to correct the mistake. This list is passed to a wrapper class, which extracts the relevant information and preprocesses it for the client. As stated in chapter 3.3, LanguageTool and other spellcheckers mark hesitation words (e.g. "uhm" and "ehm") and names of individuals as false positives. Based on this consideration, we created methods in the wrapper class to ignore these errors. Hesitation words are filtered using a regular expression and a lookup table. Errors related to names of individuals are ignored by considering the list of entities from the NLU. The list of errors is returned to the client, which highlights the affected parts of the text accordingly. If the error-to-word ratio equals or exceeds 25%, the chatbot will ask the user to rephrase their message.

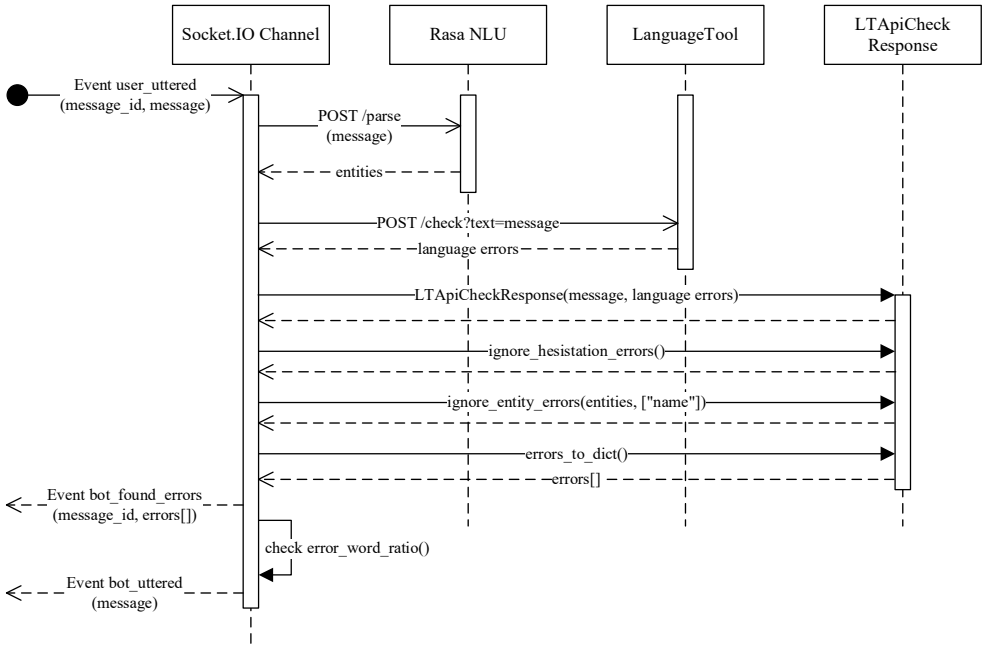

Figure 18: A simplified depiction of the language error detection process.

5.6 Data generation

As mentioned in the previous chapters, the NLU and dialog manager use machine learning-based models for their classification tasks. A considerable amount of data is required to train such models.

The NLU requires a data set of intents including entities. Depending on the variability of how an intent can be worded, a few to a several thousand are required [9, p. 8]. Unfortunately, the amount of data generated from the pilot runs does not suffice to train such models. For all 29 derived intents of the analysis phase (see chapter 3.2), a multitude of possible wordings were generated using a domain-specific language (DSL) [71]. The figure below shows an excerpt of a DSL file that generates wordings for the intent *ask_for_room_price*. The generator replaces the placeholders and creates a randomized set of permutations. In the case below, the placeholder *~[can_we]* is replaced by one of the words assigned to that placeholder.

```
%[ask_for_room_price]
    ~[whats] the price of ~[specific_room]?
    ~[is_there] ~[a_room] ~[for_singular] @[budget]?
    ~[can_we] ~[rent] ~[a_room] ~[for_singular] @[budget]?
    ~[is_there] ~[a_cheap] ~[room]?

~[can_we]
    Can we
    Could we
    Is it possible to
    Do you have
```

DSL file generates wordings

- What's the price of room [alpha](room)?
- Is there an even room for [1500 CHF](budget)?
- Is there a room for [1200.-](budget)?
- **Can we** book a room for [1500](budget)?
- **Is it possible to** reserve a room for [1500](budget)?
- Is there an inexpensive room?
- Do you have a cheap event room?

…

Figure 19: An excerpt of the DSL file generating wordings for the intent ask_for_room_price.

The NLU pipeline described in chapter 5.3 treats every word equally. It does not differentiate between stop words or keywords of a specific intent. If an intent is not trained on stop words that are used in different intents, incorrect predictions could occur due to a bias in the trained model. For example, the user message “Are there TVs in room Alpha?” might not match with the expected intent *ask_for_room_equipment,* even though the keyword *TVs* has been used, because the tokens “Are”, “there” and “in” were not part of its training set. Stop words cannot be ignored entirely however, as some intents rely on these words.

The dialog manager is trained on so-called *stories*. A *story*, as seen in Figure 20, is composed of a sequence of intents including entities and actions that should be executed in response. They either represent an entire conversation or parts of a conversation that are tightly coupled. Enough stories must be provided for the chatbot to learn how to respond under certain circumstances.

```
* greet
    - utter_greet
    - utter_ask_for_name
* provide_name{"name": "Yin"}
    - slot{"name": "Yin"}
    - utter_ask_for_service_with_name
* ask_for_room_size{"nr_of_people": "150"}
    - slot{"nr_of_people": "150"}
    - action_set_topic
    - slot{"topic": "size"}
    - utter_available_rooms_150_people
* ask_for_room_price
    - action_set_topic
    - slot{"topic": "price"}
    - utter_ask_for_room
```

Figure 20: An example of a story showing intents (*) with mentioned entities ({key: value}) and actions (-) the bot takes. A complete list of intents, entities, actions and slots are provided in the appendix (see chapter 11).

6 Results

This chapter presents the results of the implementation, by answering the research questions stated in the *Problem statement*.

Does the chatbot understand the queries of ESL students and is it able to reply in a comprehensible manner?

Overall, 70% of the user messages (n=146) were classified correctly in the usability tests. Test 2 and 3 were conducted on the final models of the NLU and dialogue manager.

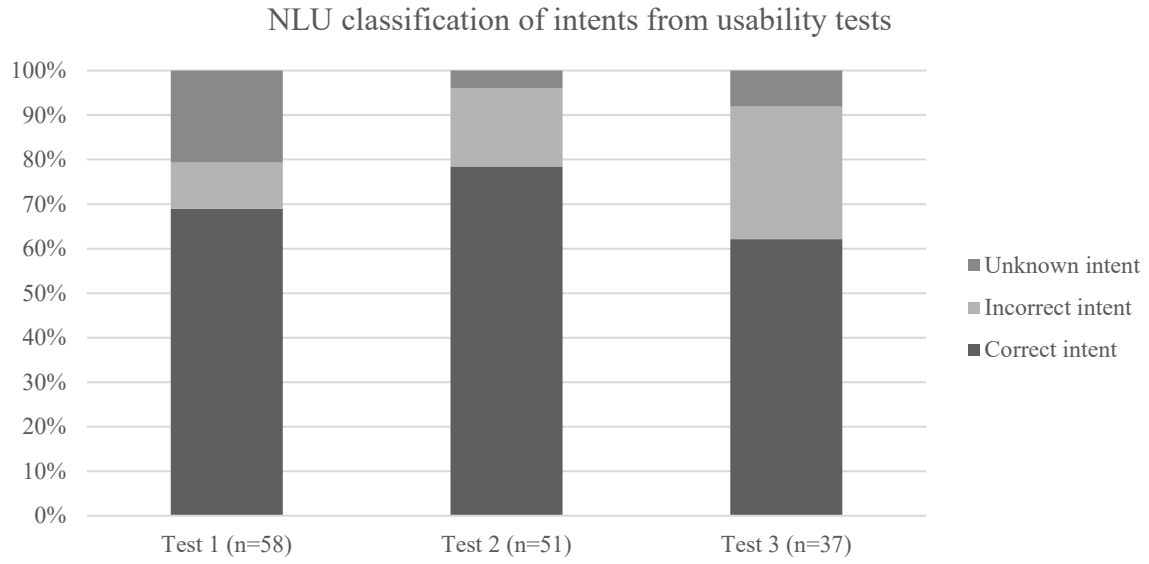

Figure 21: Proportions of classified intents per usability test.

Messages are labeled as *unknown intent* in Figure 21 when the user's intent does not exist in our model. This occurs in instances where no training data is available or where three or more intents were mixed together in one message. After the first usability test, the model was augmented by introducing compound intents (*greet+...*) and *ask_for_room_catering*, to mitigate the classification of *unknown intents*.

The chatbot's utterances (n=208) are correct in 75% of the cases, as shown in Figure 22. Incorrect utterances, meaning wrong or non-coherent answers account for 19%. Fallback messages triggered by confidence thresholds of NLU, dialog manger or language error detection account for 5.6%.

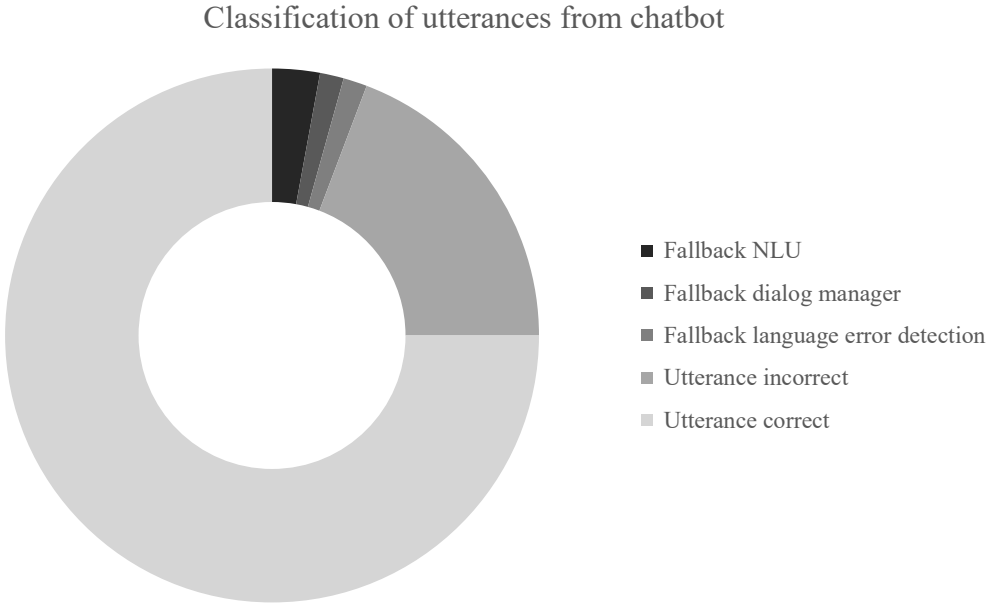

Figure 22: Chatbot utterances labeled by correctness and origin of fallback messages.

As an additional measure to separately test the performance of the NLU, a survey was conducted to collect wordings of intents for a test set. The survey was sent primarily to college students and commercial trainees completing their basic vocational training at commercial vocational schools. Students were asked to formulate intents in English. The questions and writing prompts were given in German, to mitigate the risk of influencing answers. A total of 4479 wordings were raised from the survey. The gathered data was not grammatically or otherwise corrected. Each wording was assigned to one intent. The featured entities in the wordings were not labelled. For the intents *deny*, *farewell*, *disagree, affirm* and *ask_for_room_catering* no wordings were explicitly collected. The scores of the test are depicted in Table 10. The *micro* averages the total of TPs, FNs and FPs. Both *macro* and *weighted* calculate the scores per intent and find their means. However, the *weighted* considers the number of wordings per intent as the weight. An excerpt of the confusion matrix from the test is portrayed in Figure 23.

	Recall	**Precision**	**F_1**
Avg. Micro	0.68	0.68	0.68
Avg. Macro	0.53	0.53	0.51
Avg. Weighted	0.68	0.73	0.69

Table 10: Scores from the intent classification on survey data.

Confusing unrelated intents during a conversation are the most irritating issues. An uneven distribution of stop words (see chapter 5.6) and missing keywords in the training set are contributing factors. This is especially true for the intent *ask_for_room_atmosphere*. The participants of the survey used keywords such as "luxurious", "atmospheric", "swanky" and "pretentious", to ask about a room's atmosphere. Similarly, for the intent *ask_for_room_highlight*, the expression

"salient/particular feature" was applied unexpectedly often. None of these words are part of the training set. Compared to the usability tests, the range of vocabulary was wider. This indicates that some participants might have used dictionaries (prohibited in the *room reservation* exercise) for completing the survey or have a higher English proficiency (see chapter 3.1).

As mentioned before, unknown words cause issues for the NLU. This is also the case for similarly worded intents where the unknown word is the differentiating keyword. For example, the intent *ask_for_room* is often predicted where a more specific intent would be expected, as illustrated in the third column in Figure 23. Introducing these unknown words into the training set would lead to better predictions.

When intents sharing the same underlying topic are confused by the NLU, the dialog manager can decrease the impact of the confusion. For example, if the NLU "wrongly" predicts *provide_nr_of_people* or *provide_room* instead of *ask_for_room_size* respectively *ask_for_room,* the chatbot will still be able to provide reasonable and related responses to the user. In the event of mistaking "I like room Alpha" (*provide_preference)* for *reserve_room*, the chatbot will eagerly ask if it should reserve the room. Therefore, misunderstandings of this type are not as consequential as others.

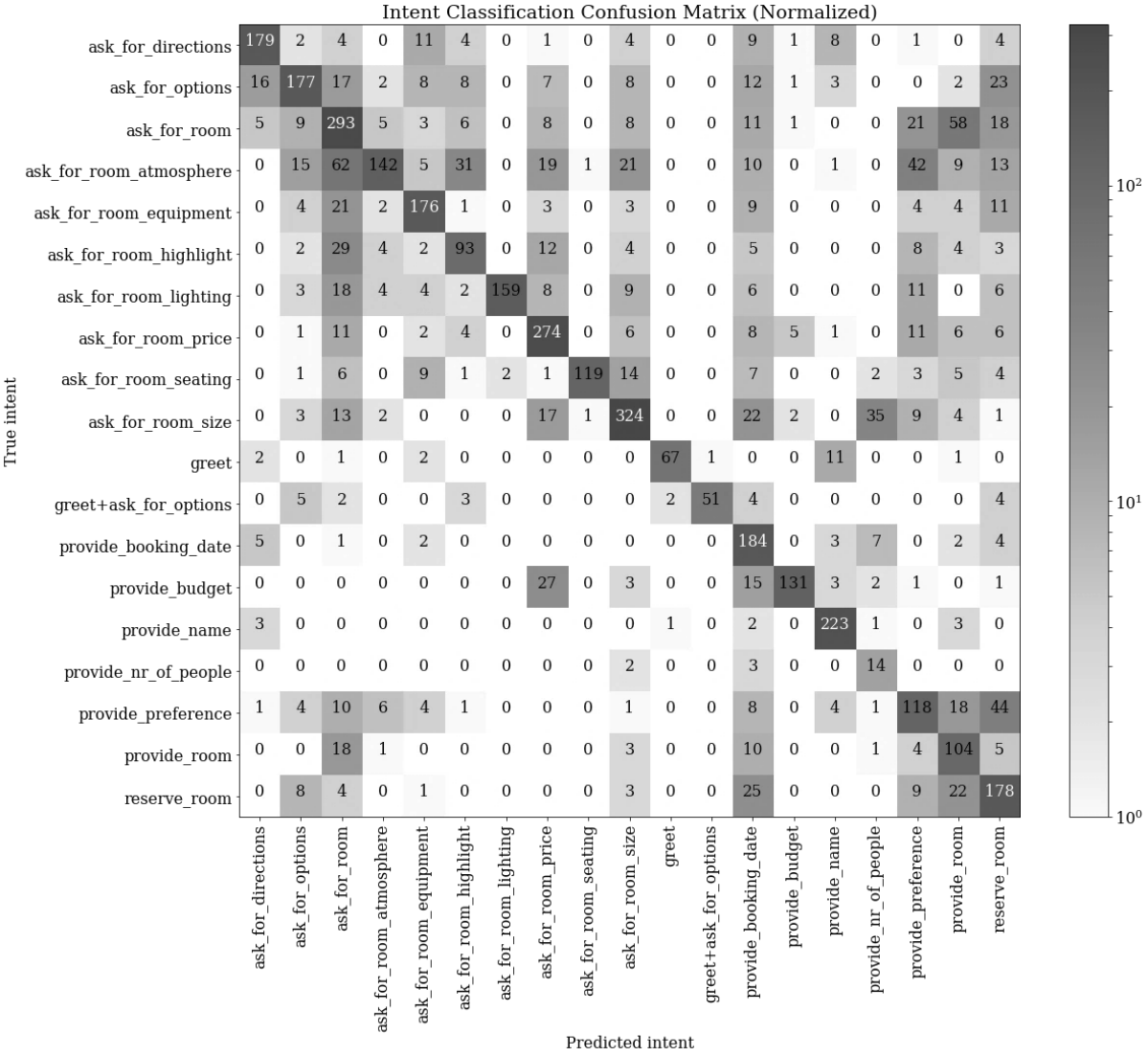

Figure 23: Excerpt of confusion matrix of intent classification on survey data.

Is it possible to formulate a meaningful clarification request to trigger a rephrase from the student, in case of a misunderstanding?

Rephrases are triggered when one of the confidence scores produced by the language error detection, NLU, or dialog manager falls below their threshold (see Figure 24). In this case, a message is picked from a list of fallback messages.

The dialog manager is trained to ask for clarification, in case it does not know which room the user is referring to. In this case it will ask the user which room was meant by firing the action *utter_ask_for_room* (see chapter 11.5).

A truly meaningful explanation of why a rephrase is necessary is not provided to the user. This would demand a more sophisticated NLG (see chapter 5.4), which the implementation does not include. However, the language error detection offers a potential source for meaningful explanations to the user. Erroneous parts of the user's message are highlighted, and explanatory hints are displayed. We chose this option over generating responses to not impede the flow of the conversation.

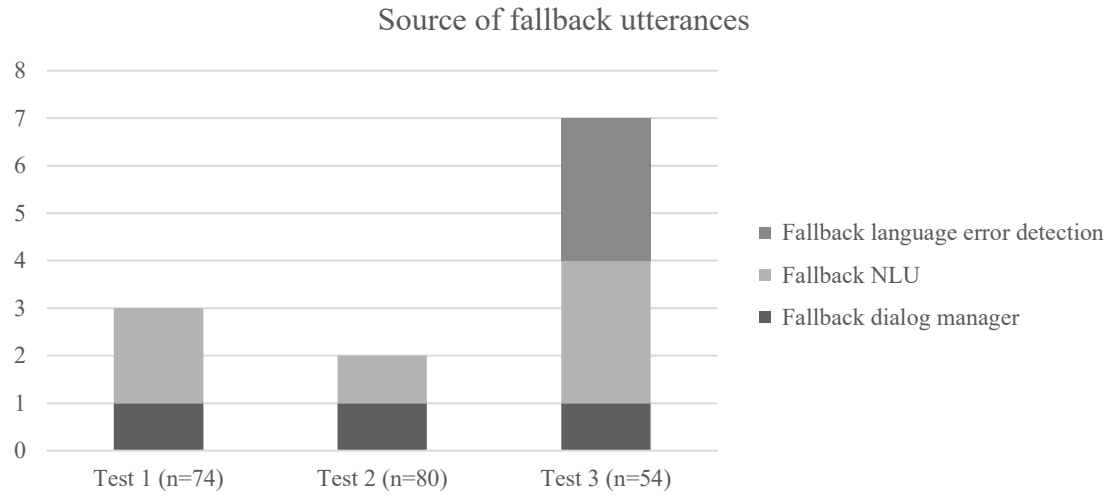

Figure 24: Source of fallback utterances by the chatbot.

Is it possible to detect morphological, syntactical and semantical errors in the student's language?

As pointed out in the analysis (chapter 3.3), ESL students commit predominantly morphological errors. For this reason, the focus here lies on spelling errors. LanguageTool ($F_{0.5}$=0.95) proves to be an adequate tool for our needs, as demonstrated in the spellchecker evaluation (see chapter 4.2).

The detection of syntactical and semantical errors is still a difficult task for computers, as we pointed out in our research chapter 2.3. LanguageTool contains grammar rules such as "Sentence is a fragment", which can detect syntactical errors [69]. The introduction of the n-gram language model, allows us to find semantical errors such as homonyms and heteronyms. However, no measures are provided to quantitatively prove its effectiveness regarding syntactical and semantical errors. No added value would have resulted from it because these types of errors were comparatively insignificant (see chapter 3.3).

Do software development kits (SDKs), cloud solutions or software libraries exist to create a holistic solution in which the aforementioned language errors can be detected?

As our implementation demonstrates, it is possible to create a holistic solution where language errors can be detected. However, we have not found a sufficiently performing solution to identify syntactical or semantical errors in English texts.

7 Conclusions

In this final chapter, we conclude this thesis with a summary of the results, lessons learned and suggestions for improvement for future releases.

As demonstrated in the results, it is possible to build an adequately performing domain-specific machine learning-based chatbot for ESL classrooms. In our tests with the target audience, our prototype understood 70% of the exchanges and was able to generate matching responses. In case of a misunderstanding, due to language errors or an unclear intent, negotiations of meaning are promoted, by requesting the user to rephrase. Shortcomings in the users' language are highlighted, allowing them to learn from their mistakes and improve their English. The added value of introducing such a chatbot into ESL classrooms will be evaluated in Johanna Oeschger's thesis [1].

More meaningful responses in case of misunderstandings could be provided to the user by integrating the language error detection as an integral step in the NLU pipeline. By feeding the errors to the dialog manager, a model could be trained to respond based on the number and type of errors in the current context.

Testing the NLU with the data set collected from the survey revealed some deficiencies in its model. We are certain that the performance of the NLU could be improved by integrating parts of that data into the model. Keeping some of the language errors that exist in the data could increase the confidence of the intent classification. This would not impair the *negotiations of meaning* if the language error detection influences the dialog manager as described in the previous paragraph.

Unfortunately, no preexisting domain-specific corpus or models existed for our endeavor. The time it took to generate the necessary data for our machine learning models was greatly underestimated. We believe that conducting the survey after the initial analysis would have been beneficial in many ways. Firstly, instead of spending much time to invent possible wordings for intents, they could have been deduced from the survey responses. Secondly, the training and test sets would have been more diverse, possibly leading to more reliable test results during implementation. Lastly, the NLU might have performed better much earlier thanks to higher diversity. It is important to note however, that solely relying on collected data for the models has its downsides too. Instead of generating data, time would have been reallocated to cleansing it.

8 List of Tables

Table 1: Lemmatization example sentence 9
Table 2: Example of a false positive correction of a spellchecker. 10
Table 3: A reduced list of intents derived from the room reservation exercise of the pilot run. 12
Table 4: A reduced list of entities derived from the room reservation exercise from the pilot run. 12
Table 5: Subset of error types from the CoNLL-2014 Shared Task 14
Table 6: Intents and entities in evaluation corpus 16
Table 7: Additional criteria for the NLU evaluation. 17
Table 8: Scores of the spellchecker evaluation. 18
Table 9: A list of events sent between client and chatbot. 20
Table 10: Scores from the intent classification on survey data. 28

9 List of Figures

Figure 1: Classification of dialog systems 3
Figure 2: Example of a conversation between a user (U) and ELIZA (E) 4
Figure 3: AIML rules to respond to "Hello" highlighting the issue of redundancy 5
Figure 4: A generic machine learning-based chatbot architecture. 7
Figure 5: Example sentence parsed with spaCy, a popular NLP library 9
Figure 6: Example dialog after spaCy tagged the recognized entities 9
Figure 7: Chat dialog from the room reservation exercise 11
Figure 8: Chat dialog of two ESL students 12
Figure 9: Examples from the room reservation exercise 13
Figure 10: Comparison of language errors in testing environment (normalized). 14
Figure 11: NLU evaluation results (F_1 score). 17
Figure 12: Overview of the chatbot architecture 19
Figure 13: Unidirectional data flow within the client. 20
Figure 14: Visualized data flow between NLU components in the pipeline. 21
Figure 15: An abridged response object of Rasa NLU. 22
Figure 16: Visualized data flow between Rasa Core components. 22
Figure 17: Simplified representation of the input and output the model of the Keras Policy 23
Figure 18: A simplified depiction of the language error detection process. 25
Figure 19: An excerpt of the DSL file generating wordings for the intent ask_for_room_price. 25
Figure 20: An example of a story 26
Figure 21: Proportions of classified intents per usability test. 27
Figure 22: Chatbot utterances labeled by correctness and origin of fallback messages. 28
Figure 23: Excerpt of confusion matrix of intent classification on survey data. 29
Figure 24: Source of fallback utterances by the chatbot. 30

10 Bibliography

[1] J. Oeschger, "Johanna Oeschger | Institut für Bildungswissenschaften." [Online]. Available: https://www.bildungswissenschaften.unibas.ch/de/doktorat/laufende-dissertationen/oeschger/. [Accessed: 19-Sep-2018].

[2] M. H. Long, "The role of the linguistic environment in second language acquisition," in *Handbook of Second Language Acquisition*, vol. 2, 1996, pp. 413–468.

[3] L. K. Fryer, K. Nakao, and A. Thompson, "Chatbot learning partners: Connecting learning experiences, interest and competence," *Computers in Human Behavior*, vol. 93, pp. 279–289, Apr. 2019.

[4] N. M. Radziwill and M. C. Benton, "Evaluating Quality of Chatbots and Intelligent Conversational Agents," *arXiv:1704.04579 [cs]*, Apr. 2017.

[5] 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.

[6] M. L. Mauldin, "CHATTERBOTS, TINYMUDS, and the Turing Test: Entering the Loebner Prize Competition," in *AAAI*, 1994.

[7] R. S. Wallace, "The Anatomy of A.L.I.C.E.," in *Parsing the Turing Test: Philosophical and Methodological Issues in the Quest for the Thinking Computer*, R. Epstein, G. Roberts, and G. Beber, Eds. Dordrecht: Springer Netherlands, 2009, pp. 181–210.

[8] M. Mensio, "Deep Semantic Learning for Conversational Agents," Master of Science in Computer Engineering, Politecnico di Torino, 2018.

[9] A. Coucke *et al.*, "Snips Voice Platform: an embedded Spoken Language Understanding system for private-by-design voice interfaces," *arXiv:1805.10190 [cs]*, May 2018.

[10] T. Bocklisch, J. Faulkner, N. Pawlowski, and A. Nichol, "Rasa: Open Source Language Understanding and Dialogue Management," *arXiv:1712.05181 [cs]*, Dec. 2017.

[11] I. V. Serban *et al.*, "A Deep Reinforcement Learning Chatbot," *arXiv:1709.02349 [cs, stat]*, Sep. 2017.

[12] A. Bordes, Y.-L. Boureau, and J. Weston, "Learning End-to-End Goal-Oriented Dialog," *arXiv:1605.07683 [cs]*, May 2016.

[13] S. Dhanda, "Chatbots: Banking, eCommerce, Retail & Healthcare 2018-2023," Juniper Research, Mar. 2018.

[14] PIDAS AG, "Chatbot-Studie - Die digitalen Helfer im Praxistest," 10-Feb-2018. [Online]. Available: https://page.pidas.com/chatbot-studie. [Accessed: 03-Oct-2018].

[15] B. A. Shawar and E. Atwell, "Chatbots: Are they Really Useful?," p. 21, 2007.

[16] B. Heller, M. Proctor, D. Mah, L. Jewell, and B. Cheung, "Freudbot: An Investigation of Chatbot Technology in Distance Education," presented at the EdMedia + Innovate Learning, 2005, pp. 3913–3918.

[17] J. Jia and W. Chen, "Motivate the Learners to Practice English through Playing with Chatbot CSIEC," in *Technologies for E-Learning and Digital Entertainment*, 2008, pp. 180–191.

[18] L. Fryer, "Bots as Language Learning Tools," *Language, Learning and Technology*, vol. 10, pp. 8–14, Jan. 2006.

[19] J. Jia, "The Study of the Application of a Web-Based Chatbot System on the Teaching of Foreign Languages," presented at the Society for Information Technology & Teacher Education International Conference, 2004, pp. 1201–1207.

[20] H. Sugiyama, T. Meguro, R. Higashinaka, and Y. Minami, "Open-domain Utterance Generation for Conversational Dialogue Systems using Web-scale Dependency Structures," in *Proceedings of the SIGDIAL 2013 Conference*, Metz, France, 2013, pp. 334–338.

[21] S. Roos, *Chatbots in education : A passing trend or a valuable pedagogical tool?* 2018.

[22] L. K. Fryer, M. Ainley, A. Thompson, A. Gibson, and Z. Sherlock, "Stimulating and sustaining interest in a language course: An experimental comparison of Chatbot and Human task partners," *Computers in Human Behavior*, vol. 75, pp. 461–468, Oct. 2017.

[23] Existor, "Cleverbot Data for Machine Learning," 15-Jan-2016. .

[24] S. Luperfoy, D. Loehr, D. Duff, K. Miller, F. Reeder, and L. Harper, "An Architecture for Dialogue Management, Context Tracking, and Pragmatic Adaptation in Spoken Dialogue Systems," May 2002.

[25] "Pandorabots: Home." [Online]. Available: https://home.pandorabots.com/home.html. [Accessed: 31-Oct-2018].

[26] Y. F. Wang and S. Petrina, "Using learning analytics to understand the design of an intelligent language tutor–Chatbot lucy," *Computer Science Journals | IJACSA*, vol. 4, no. 11, pp. 124–131, 2013.

[27] H. Zhou and M. Huang, "Context-aware natural language generation for spoken dialogue systems," in *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, 2016, pp. 2032–2041.

[28] T.-H. Wen, M. Gasic, N. Mrksic, P.-H. Su, D. Vandyke, and S. Young, "Semantically Conditioned LSTM-based Natural Language Generation for Spoken Dialogue Systems," *arXiv:1508.01745 [cs]*, Aug. 2015.

[29] M.-C. Jenkins, "Designing Service-Oriented Chatbot Systems Using a Construction Grammar-Driven Natural Language Generation System," doctoral, University of East Anglia, 2011.

[30] E. Reiter and R. Dale, *Building natural language generation systems*. Casmbridge, U.K. ; New York: Cambridge University Press, 2000.

[31] V. Q. Liao *et al.*, "All Work and no Play? Conversations with a Question-and-Answer Chatbot in the Wild," in *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems (CHI'18). ACM, New York, NY, USA*, 2018, vol. 13.

[32] Y. He and M. Kayaalp, "A Comparison of 13 Tokenizers on MEDLINE," *Bethesda, MD: The Lister Hill National Center for Biomedical Communications*, vol. 48, 2006.

[33] S. Petrov, D. Das, and R. McDonald, "A universal part-of-speech tagset," *arXiv preprint arXiv:1104.2086*, 2011.

[34] W. N. Francis and H. Kucera, "Brown Corpus Manual," *Brown Corpus Manual*, Jul-1979. [Online]. Available: http://clu.uni.no/icame/manuals/BROWN/INDEX.HTM. [Accessed: 03-Nov-2018].

[35] D. Chen and C. Manning, "A Fast and Accurate Dependency Parser using Neural Networks," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Doha, Qatar, 2014, pp. 740–750.

[36] H. Schütze, C. D. Manning, and P. Raghavan, *Introduction to information retrieval*, vol. 39. Cambridge University Press, 2008.

[37] A. Nenkova and K. McKeown, "A Survey of Text Summarization Techniques," in *Mining Text Data*, C. C. Aggarwal and C. Zhai, Eds. Boston, MA: Springer US, 2012, pp. 43–76.

[38] "displaCy Named Entity Visualizer · Demos · Explosion AI," *Explosion AI*. [Online]. Available: https://explosion.ai/demos/displacy-ent. [Accessed: 10-Nov-2018].

[39] D. Nadeau and S. Sekine, “A survey of named entity recognition and classification,” *Lingvisticæ Investigationes*, vol. 30, no. 1, pp. 3–26, Jan. 2007.

[40] J. Pennington, R. Socher, and C. Manning, “Glove: Global vectors for word representation,” in *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, 2014, pp. 1532–1543.

[41] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean, “Distributed Representations of Words and Phrases and Their Compositionality,” in *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2*, USA, 2013, pp. 3111–3119.

[42] D. Coniam, “Evaluating the language resources of chatbots for their potential in English as a second language,” *ReCALL*, vol. 20, no. 01, Jan. 2008.

[43] R. Lyster and L. Ranta, “CORRECTIVE FEEDBACK AND LEARNER UPTAKE: Negotiation of Form in Communicative Classrooms,” *Studies in Second Language Acquisition*, vol. 19, no. 1, pp. 37–66, Mar. 1997.

[44] M. Swan and B. Smith, *Learner English: a teacher’s guide to interference and other problems*, Second rev. ed. [repr.]. Cambridge: Cambridge University Press, 2002.

[45] C. Leacock, M. Chodorow, M. Gamon, and J. Tetreault, “Automated Grammatical Error Detection for Language Learners, Second Edition,” *Synthesis Lectures on Human Language Technologies*, vol. 7, no. 1, pp. 1–170, Feb. 2014.

[46] H. T. Ng, S. M. Wu, Y. Wu, C. Hadiwinoto, and J. Tetreault, “The CoNLL-2013 Shared Task on Grammatical Error Correction,” in *Proceedings of the Seventeenth Conference on Computational Natural Language Learning: Shared Task*, Sofia, Bulgaria, 2013, pp. 1–12.

[47] H. T. Ng, S. M. Wu, T. Briscoe, C. Hadiwinoto, R. H. Susanto, and C. Bryant, “The CoNLL-2014 Shared Task on Grammatical Error Correction,” in *Proceedings of the Eighteenth Conference on Computational Natural Language Learning: Shared Task*, Baltimore, Maryland, 2014, pp. 1–14.

[48] R. Dale and A. Kilgarriff, “Helping our own: The HOO 2011 pilot shared task,” in *Proceedings of the 13th European Workshop on Natural Language Generation*, 2011, pp. 242–249.

[49] R. Dale, I. Anisimoff, and G. Narroway, “HOO 2012: A report on the preposition and determiner error correction shared task,” in *Proceedings of the Seventh Workshop on Building Educational Applications Using NLP*, 2012, pp. 54–62.

[50] U. Renold and J.-P. Lüthi, “Leistungszielkatalog Fremdsprachen: 2. Landessprache und / oder Englisch (FS B/E Profil).” Staatssekretariat für Bildung, Forschung und Innovation, 21-Nov-2014.

[51] Google, “Dialogflow.” [Online]. Available: https://dialogflow.com/. [Accessed: 09-Jan-2019].

[52] Microsoft, “LUIS (Language Understanding) – Cognitive Services – Microsoft Azure.” [Online]. Available: https://www.luis.ai/home. [Accessed: 09-Jan-2019].

[53] Microsoft, “Azure Bot Service - chatbot | Microsoft Azure.” [Online]. Available: https://azure.microsoft.com/en-us/services/bot-service/. [Accessed: 13-Feb-2019].

[54] Rasa Technologies GmbH, “Rasa: Open source conversational AI.” [Online]. Available: https://rasa.com/. [Accessed: 09-Jan-2019].

[55] D. Braun, A. Hernandez-Mendez, F. Matthes, and M. Langen, “Evaluating Natural Language Understanding Services for Conversational Question Answering Systems,” in *Proceedings of*

the 18th Annual SIGdial Meeting on Discourse and Dialogue, Saarbrücken, Germany, 2017, pp. 174–185.

[56] Y. Sasaki, "The truth of the F-measure," 2007.

[57] S. Chollampatt and H. T. Ng, "Neural Quality Estimation of Grammatical Error Correction," in *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, Brussels, Belgium, 2018, pp. 2528–2539.

[58] M. Junczys-Dowmunt, R. Grundkiewicz, S. Guha, and K. Heafield, "Approaching Neural Grammatical Error Correction as a Low-Resource Machine Translation Task," in *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, New Orleans, Louisiana, 2018, pp. 595–606.

[59] S. Chollampatt and H. T. Ng, "A Multilayer Convolutional Encoder-Decoder Neural Network for Grammatical Error Correction," in *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[60] László Németh, "Hunspell," 26-Jan-2019. [Online]. Available: https://github.com/hunspell/hunspell. [Accessed: 26-Jan-2019].

[61] W. Garbe, "SymSpell," 25-Jan-2019. [Online]. Available: https://github.com/wolfgarbe/SymSpell. [Accessed: 26-Jan-2019].

[62] LanguageTooler GmbH, "LanguageTool," 26-Jan-2019. [Online]. Available: https://github.com/languagetool-org/languagetool. [Accessed: 26-Jan-2019].

[63] D. Naber, "Finding errors using n-gram data - LanguageTool Wiki," 30-Dec-2017. [Online]. Available: http://wiki.languagetool.org/finding-errors-using-n-gram-data. [Accessed: 13-Feb-2019].

[64] K. Louis and N. Cocquio, "GitHub Repository of the Chatbot for English Classrooms," 19-Nov-2018. [Online]. Available: https://github.com/kelvinlouis/ip6-chatbot. [Accessed: 13-Feb-2019].

[65] Rasa Technologies GmbH, "Choosing a Rasa NLU Pipeline," *Choosing a Rasa NLU Pipeline*. [Online]. Available: https://rasa.com/docs/nlu/0.13.8/choosing_pipeline/. [Accessed: 07-Feb-2019].

[66] L. Wu, A. Fisch, S. Chopra, K. Adams, A. Bordes, and J. Weston, "StarSpace: Embed All The Things!," *arXiv:1709.03856 [cs]*, Sep. 2017.

[67] C. Sutton and A. McCallum, "An Introduction to Conditional Random Fields," *arXiv:1011.4088 [stat]*, Nov. 2010.

[68] Y. Bengio, P. Simard, and P. Frasconi, "Learning long-term dependencies with gradient descent is difficult," *IEEE Transactions on Neural Networks*, vol. 5, no. 2, pp. 157–166, Mar. 1994.

[69] LanguageTooler GmbH, "Browse LanguageTool Rules: 2,216 matches for English," 30-Jan-2019. [Online]. Available: https://community.languagetool.org/rule/list?offset=0&max=10&lang=en&filter=&categoryFilter=&_action_list=Filter. [Accessed: 14-Feb-2019].

[70] Google, "Google Ngram Viewer Dataset," 2013. [Online]. Available: http://storage.googleapis.com/books/ngrams/books/datasetsv2.html. [Accessed: 13-Feb-2019].

[71] R. Pimentel, "Chatito," 14-Feb-2019. [Online]. Available: https://github.com/rodrigopivi/Chatito. [Accessed: 14-Feb-2019].

11 Appendix

11.1 Room reservation exercise

The original *room reservation* exercise students are given prior to chatting with the bot. It is written in German.

> Sie sind KV-Lernende/r in einem Betrieb. Ihre Kolleginnen und Kollegen aus der Marketing-Abteilung organisieren nächsten Monat einen Networking-Event in Toronto (Kanada). Sie haben nun den Auftrag, im «One King» Hotel in Toronto einen passenden Raum für den Event zu finden.
>
> Der Raum soll:
>
> - Platz haben für mind. 150 Personen
> - eine Präsentation (Video, Audio) ermöglichen
> - Gespräche in kleineren Gruppen ermöglichen
> - eine stimmungsvolle Atmosphäre haben
> - den Gästen Eindruck machen, aber nicht zu protzig/übertrieben wirken
> - max. 1'500 Franken kosten
>
> Das Hotel «One King» hat drei Eventräume: A, B und C. Kontaktieren Sie das Hotel über ihren Online-Chat und stellen Sie Fragen zu den drei Räumen. Entscheiden Sie, welcher Raum am besten passt.
>
> Notieren Sie mindestens fünf Argumente (in Stichworten), um das Marketing-Team zu überzeugen, dass Sie den besten Raum gefunden haben.

11.2 Complete list intents

ID	Intent	Examples
1	affirm	Yes | Yes, please | That is correct
2	ask_for_directions	What is the fastest way to your hotel?
3	ask_for_options	What rooms do you have? | Do you have meeting rooms?
4	ask_for_room	Tell me about room Alpha | What about the others?
5	ask_for_room_atmosphere	Are all the rooms nice? | How is the atmosphere in room Alpha?
6	ask_for_room_equipment	Is it possible to show a presentation in Alpha?
7	ask_for_room_highlight	Are there any special things in the room?
8	ask_for_room_lighting	How is the lighting in the room?
9	ask_for_room_price	How much does room Beta cost? | Are the other rooms expensive?
10	ask_for_room_seating	Are we able to form groups for a workshop?
11	ask_for_room_size	Are the other ones bigger? | Does room Alpha have space for 150?

12	ask_for_room_catering	Do you provide lunch? | Are there refreshments in Room Gamma?
13	deny	No | Nope | No thank you | Maybe not
14	disagree	Not great | That doesn't sound good | I don't like it
15	farewell	Take care! | Bye | Have a nice day
16	greet	Hi there! | Hey, how are you? | Hello
17	greet+ask_for_options	Hello. What rooms do you have?
18	greet+ask_for_room_equipment	Hello there, do you have rooms that have audio systems?
19	greet+ask_for_room_price	Hello there how is it going? How much are the rooms?
20	greet+ask_for_room_size	Hello, how are you? Do you provide rooms for up to 200?
21	greet+provide_name	Hi, my name is Fred Pierson | Hi, I am Lynn can you help me?
22	provide_booking_date	Please book the room on the 25th of October | 24.12.19
23	provide_budget	I have a budget of CHF 1'500 | I have 1500.- available | 1'500.-
24	provide_name	My name is Ellis | Landon Donovan | Michael
25	provide_nr_of_people	We are expecting 200 people | 150 people | 150
26	provide_preference	I really like room Alpha | Room Beta sounds fantastic
27	provide_room	Room Alpha | I am talking about room Beta | Alpha!
28	reserve_room	I want to reserve room Alpha | I want to book this room
29	thanks	Thank you | Thanks! | Thanks a lot

11.3 Complete list of entities

ID	Entity	Examples
1	budget	1'500.- | 1200 | CHF 1500 | 1600 | $1200
2	company	ABC Inc. | Lindt AG
3	current_room	Alpha | Beta | Gamma
4	date	25.04.2019 | 24th of May | 1st December 2019
5	name	Yves | Max Muster | Mr. Muster
6	nr_of_people	150
7	room	Alpha | Beta | Gamma
8	start_location	Train station | Airport
9	time	24:00 | 3pm

11.4 Complete list of slots

ID	Slot	Examples	One-hot encoding
1	budget	<empty>	0 0 0 0
		x < = 900	1 0 0 0
		900 < x <= 1100	0 1 0 0
		1100 < x <= 1400	0 0 1 0
		1400 < x	0 0 0 1
2	company	<empty>	0
		<not empty>	1
3	current_room	<empty>	0 0 0
		Alpha	1 0 0
		Beta	0 1 0
		Gamma	0 0 1
4	date	<empty>	0
		<not empty>	1
5	name	<empty>	0
		<not empty>	1
6	nr_of_people	<empty>	0 0 0 0
		x < = 190	1 0 0 0
		190 < x <= 200	0 1 0 0
		200 < x <= 270	0 0 1 0
		270 < x	0 0 0 1
7	room	<empty>	0 0 0 0 0 0
		Alpha, first	1 0 0 0 0 0
		Beta, second	0 1 0 0 0 0
		Gamma, third, last	0 0 1 0 0 0
		other, others, another	0 0 0 1 0 0
		this, that, it, there	0 0 0 0 1 0
		all, every, each	0 0 0 0 0 1
8	start_location	<empty>	0
		<not empty>	1
9	time	<empty>	0
		<not empty>	1
10	topic	<empty>	0 0 0 0 0 0 0
		atmosphere	1 0 0 0 0 0 0
		lighting	0 1 0 0 0 0 0
		equipment	0 0 1 0 0 0 0

highlight	0 0 0 1 0 0 0
price	0 0 0 0 1 0 0
seating	0 0 0 0 0 1 0
size	0 0 0 0 0 0 1

11.5 Complete list of actions

ID	Action	Description / Example
1	action_correct_room	Fills the slot current_room given the entity room
2	action_default_fallback	Triggered if below threshold (provided by Rasa)
3	action_get_room_price	Not actively used
4	action_get_room_size	Not actively used
5	action_listen	Waits for user input (provided by Rasa)
6	action_restart	Restarts the dialog (provided by Rasa)
7	action_set_topic	Fills the topic slot if certain intents are triggered
8	utter_appreciation	You are welcome.
9	utter_ask_book_room	Sure. Would you like to book it?
10	utter_ask_for_additional_service	Could I help you with anything else?
11	utter_ask_for_alternative	Would you like to see another option?
12	utter_ask_for_booking_confirmation	Thank you. So, I will book {current_room} …
13	utter_ask_for_booking_date	What would be the date for the booking?
14	utter_ask_for_budget	Could you please tell me what the budget …?
15	utter_ask_for_confirmation	Could you please confirm if this is correct?
16	utter_ask_for_name	Please could you give me your name?
17	utter_ask_for_nr_of_people	How many guests are you expecting?
18	utter_ask_for_room	Please could you give me the name of the room?
19	utter_ask_for_service	How can I help you?
20	utter_ask_for_service_with_name	How can I help you, {name}?
21	utter_ask_to_narrow_options	Are you looking for a specific room?
22	utter_ask_what_info	What information can I help you with?
23	utter_available_rooms_150_people	We have three event rooms on offer for …
24	utter_available_rooms_200_people	We have two event rooms on offer …
25	utter_available_rooms_270_people	There is one room available for {nr_of_people}…
26	utter_budget_limitation_1100	We can offer you two rooms for this price.
27	utter_budget_limitation_1400	All our rooms are below {budget}.
28	utter_budget_limitation_900	There is one room on offer that is below {budget}.
29	utter_catering_options	I am afraid we do not offer a catering option.
30	utter_confirm	Yes, sure.

31	utter_confirm_booking	Room {current_room} has just been reserved …
32	utter_confirm_preference_positive	Great choice. I think this room will …
33	utter_default	Excuse me, could you rephrase this please?
34	utter_directions_with_start_location	You can easily reach us by subway from the …
35	utter_directions_without_start_location	I recommend that you travel by subway.
36	utter_dissatisfaction	I am sorry to hear that.
37	utter_enough_budget	All of our rooms are below your price limit.
38	utter_generic_atmosphere_options	Each room has its own unique ambience.
39	utter_generic_equipment_options	All of the three rooms are equipped for present …
40	utter_generic_highlight_options	One room is a spacious auditorium with all …
41	utter_generic_lighting_options	The lighting in all three rooms may be adjusted …
42	utter_generic_pricing_options	We offer competitive pricing for all our rooms…
43	utter_generic_seating_options	We have rooms for various occasions.
44	utter_goodbye	Thank you very much. Goodbye, {name}.
45	utter_greet	Welcome to One King Hotel. I am happy to …
46	utter_greet_with_name	Welcome at One King Hotel, {name}.
47	utter_options	We have multiple offerings.
48	utter_room_alpha_atmosphere	Room Alpha is an auditorium with a stage and …
49	utter_room_alpha_equipment	Room Alpha is equipped with a big screen …
50	utter_room_alpha_equipment_stage	It has a magnificent stage including …
51	utter_room_alpha_highlight	Room Alpha offers plenty of space for …
52	utter_room_alpha_lighting	Room Alpha has no natural light as …
53	utter_room_alpha_people_limit	Room Alpha provides space for up to 270 people.
54	utter_room_alpha_price	The price for room Alpha is CHF 1400.
55	utter_room_alpha_seating	Alpha has fixed seating and no tables. It is an …
56	utter_room_beta_atmosphere	The outdoor section in room Beta offers a great …
57	utter_room_beta_equipment	Room Beta has a portable screen, a projector …
58	utter_room_beta_equipment_stage	It does not have a stage. There is however …
59	utter_room_beta_highlight	Room Beta is located on the 15th floor. …
60	utter_room_beta_lighting	Room Beta has a lot of natural light from …
61	utter_room_beta_people_limit	Room Beta offers 130 seats indoors and 60 seats…
62	utter_room_beta_price	Beta is CHF 900 including the terrace.
63	utter_room_beta_seating	The indoor seating area in room Beta has tables …
64	utter_room_gamma_atmosphere	Gamma is on the first floor. Its original decora…
65	utter_room_gamma_equipment	Gamma has two large screens and loudspeakers
66	utter_room_gamma_equipment_stage	It does not have a stage. It does however offer …
67	utter_room_gamma_highlight	Room Gamma is on the first floor. It is a …
68	utter_room_gamma_lighting	Room Gamma is well-lit with lighting …

69	utter_room_gamma_people_limit	Room Gamma offers space for up to 200 people…
70	utter_room_gamma_price	The price is CHF 1100 CHF for room Gamma.
71	utter_room_gamma_seating	There are moveable tables for 6-8 guests …
72	utter_satisfied	I am pleased to hear that you like it.
73	utter_thanks	Thank you very much for choosing One King …
74	utter_thanks_with_name	Thank you, {name}!