UCLouvain

École polytechnique de Louvain

Study and design of a chatbot as support of foreign languages learning within an e-learning platform

Author: **Carolina Eugenia Unriza Salamanca**
Supervisor: **Benoît Macq**
Readers: **Julie Chatelain, Axel Legay**
Academic year 2018–2019
Master [120] in Computer Science

Abstract

The use of artificial intelligent chatbots in industry and education has significantly increased these years. Most of them are used as customer support or as student tutoring. In both situations, the bot is trained to perform a question/answer task. In contrast, few systems are known to assist user's assessment, and even less so, language assessment.

The goal of this master thesis is to design and implement a chatbot capable of assessing the user's level into the CEFR framework while having a natural conversation with him/her. We divided the problematic of building such system into two distinct sub-problems. First, we study the different components of a chatbot and analyse various existing solutions. Second, we explore the fields of automated assessment particularly and short answers assessment. Finally, we implement a solution combining the two different parts of this work. Experimentations help us to point out critical aspects of our work that can be improved in further work.

Acknowledgements

At the end of this master thesis, I would like to heartily thank my supervisor, Professor Benoît Macq and Julie Chatelain for their supervision during the conduct of this work. Thank you for your comments, your time and the trust you gave me to conduct this work.

I also want to thank my friends Charline Outers, Romain Hubert, Robin Hormaux, Julien Gomez, Trong-Vu Tran, Corentin Surquin, Kevin Stoffler and Arnaud Gellens for their friendship, company and their good mood. You always make my day more interesting, happier and funnier. My special thanks to Vu, thank you for your time and corrections you made in this work. I also want to thank and have a though to my friend and *binome* Antoine Van Grootenbrulle, you will always be missed.

Cédric and Denis Lannoye, thank you for your support during those years. I would always be grateful to you, Antoine Zanutel, thank you for your unconditional support. Gracias a ti Lala, por tu soporte y alegria durante todos estos años. Gracias a mis abuelitos, Jorge y Eugenia, quienes siempre me dieron motivacion y soporte desde el otro lado del mundo. Para terminar, gracias a ti mami. Tu amor incondicional y confianza siempre guiaron mi camino, muchas gracias maàâma.

Contents

List of Figures

List of Tables

Introduction

Learning new languages has become, nowadays, a pivotal element in our society. As for many sectors, education is benefiting of the development of technology: there is an increasing number of applications helping us improving our skills and more particularly, languages skills. In this context, *Graal* startup is modernising our way of learning and enhance our pronunciation in a foreign language by developing a self-training app focussing on this particular aspect of the language. [28]

In order to give the best experience to the user, the app must correctly determine his/her language level. The classical approach is to fill a multiple choice, and a "fill in the blanks" quiz assessing the user grammar, vocabulary and everyday phrases knowledge. However, this approach is not very natural since there is no context/link between questions.

This thesis aims to build a chatbot capable to correctly classify the user's language level while having an almost natural conversation with him. Our goal is two-fold: build an intelligent chatbot into `unity` (the platform used by *Graal*) and correctly determine the user's level.
In this document, I will first focus on the chatbot aspect: the characteristics of such a program and the solutions available. Then, I will talk about the specificities of languages tests. Afterwards, I will discuss the solution's implementations, and finally, I will discuss limitations and improvements.

Part I

State of the art

Chapter 1

Chatbot

A chatbot or conversational agent is a computer program that attempts to simulate the conversation of a human being via text or voice interactions. [32]
The first attempt of such program goes back to the '60s with ELIZA created by Joseph Weizenbaum at MIT. [4] ELIZA create quite an enthusiasm for this domain and more chatbots were developed. However, this first generation of chatbots was made on a rule based-approach within a controlled scope. As a result, they only performed well only in constrained environments and did not retrieve any information from the user's answers. [32]

With the advancements of machine learning techniques, the performance increased well as their popularity. [2] Machine learning powered chatbots interpreting the natural language to both understand and learn better over time. [13] Their use increased, especially with the launch of chatbot platforms by big companies as Facebook, Slack and Skype [4]: by September 2016, Facebook Messenger hosted 30,000 bots and had 34,000 developers on its platform. [18] Chatbots use don't only increase for commercial reasons, researchers started to find other purposes as in medicine and education.

In this chapter, we first depict the underlying architecture of a chatbot and afterwards, a quick review of some existing solutions.

1.1 Architecture

Although there are many different chatbots nowadays, they still have the same core base. This common architecture is composed of six components (see fig.1.1). In this section we will present the *basic* and very *common* version; the organisation of the different parts may change between different systems.

1.1.1 Speech-to-text conversion

Firsts chatbots were exclusively interacting with the user by text. However, speech is the most natural and universal way of communication of humans beings. With the help of technology, more particularly, the development of speech processing, it has become quite easy to communicate directly to the device. As a result, speaking chatbots have become very common.

Speech-to-text conversion process aims to convert speech to text via Automatic Speech Recognition (ASR) and mining speech information [20]; then, the resulting text can be treated to extract the meaning of the words. [1]

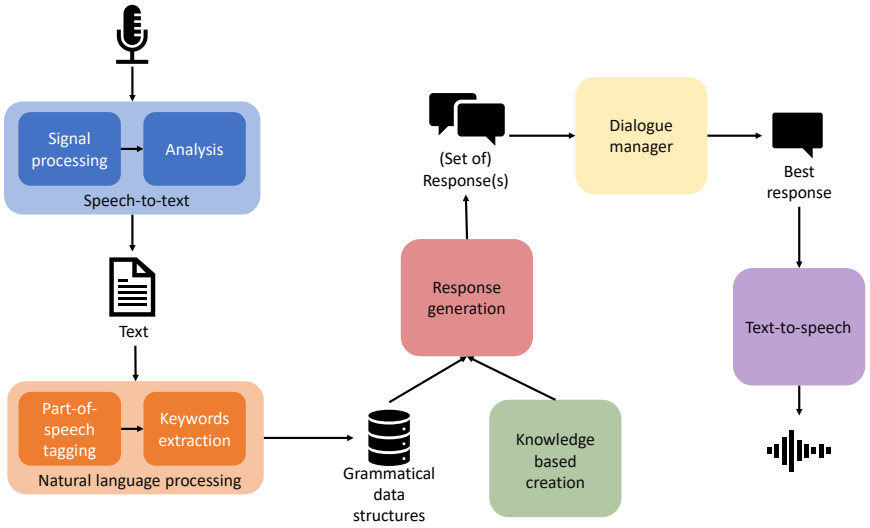

Figure 1.1: Architecture of a basic spoken chatbot

First of all, the speech is recorded into a microphone and the signal is discretised with a sampling frequency.[14] Then, the ASR system discriminates phonemes (the basic unit of speech [27]) from background noise which can be misclassified as speech and introduce error into the model [4].
After, it proceeds to feature extraction. Acoustic observations are extracted over time frames of uniform length, typically 25 milliseconds. Each of those frames is decoded: the acoustic feature vectors are mapped to the most likely corresponding words [4]. This is done with the help of the Hidden Markov Model technique which evaluates P(X|W) – given a word, the probability that we hear a sound.

The final step is a post-processing. Baseed on Bayes rule, it gives a list of the probable word sequences with their ranks. ASR chooses the most probable one as the word sequence it heard. The other top hypotheses are stored and can be used in reinforcement learning algorithms later, where they will be used to learn from and correct mistakes in the ASR phase.[4]

1.1.2 Natural language processing

The goal of this part is to extract information and meaning from the written and spoken languages to create grammatical data structures. Those grammatical data structures can be processed by the Dialogue Management unit in the next step. [4]

There are many techniques to carry out this goal as Bag of Words, Latent Semantic Analysis, Regular Expressions, Part of Speech (POS) Tagging. Since each system uses a different one, it is out of the scope of this thesis to present them all. However, they have similar sub-goals: the text given by the speech-to-text step is split into separate words for tagging with parts-of-speech labels according to their positions and neighbours in the sentence. Then, keywords are extracted from these sentences by eliminating unwanted words in chinking operations, and their rightness is evaluated. [1]

1.1.3 Response generation

This component is the cornerstone of a good chatbot. In addition to the text itself, the representation of the spoken text given by the previous step has its information about the speaker, the dialogue history, and the context. Based on this information, the goal of this component

is "straightforward": deliver a set of responses or responses generated by the Dialogue Manager. [4]

In order to make its decision, the system has to take into account:

- a knowledge database (see subsection 1.1.4)
- a dialogue history corpus (only for more complex models)
- an external data source: it provides the bot with "common sense intelligence" like "the sun is hot". This information is often gathered by allowing the bot to search through search engines.

As for the previous component, there are different ways to achieve this goal. In this subsection, we will present the most used ones briefly.

Rule base model

This response generation model contains a database of pairs `<pattern>/template`. When the bot receives an input that matches `<pattern>` it will retrieve `template` as a response. [4] This model has two meta-parameters: the design of the pairs `<pattern>/template` and the choice of a matching patterns algorithm. As for all meta parameters, the selection of those two is a major design choice.

As said at the beginning of this chapter, chatbots based on this technique are not very powerful as they only take into account the `<pattern>`.

Information retrieval

As there is more data of conversations available from social media, it has become quite obsolete to handcraft all rules since we can infer them from a large amount of data. Here, the system will build pairs of `status/response` from the gathered data. Once again, the difficulty lies into the choice of a good pattern matching algorithm.[4]

Statistical machine translation generative models

As opposed to the information retrieval model, this model builds responses on the fly. The bot is trained with machine learning techniques on real dialogues and generate responses by taking inputs as "responses". Statistical Machine Translation (SMT) models are some the most recent, and effective, models used to generate chatbot responses. [4]

Sequence to sequence model

To achieve higher accuracy, this model proposed by Cho & al.(2014), uses advances in deep learning. Their Encoder-Decoder model uses two recurrent neural networks (RNN). The first, encodes the <status>, or input sentence, and the second decodes the <status> and generates the desired <response>. The sequence to sequences is the current industry best practice for response generation and widely used.[4]

1.1.4 Knowledge base creation

As for any machine learning algorithm, the performance of a chatbot depends on the data it is using for training itself. The accuracy is not only directly linked to the quantity of data but also its quality.

Explicit confirmation	**Implicit confirmation**
User: "I want to go to New York"	**User:** "I want to go to New York"
System: "You want directions to New York?"	**System:** "What type of transportation do you want to use to get to New York?"

Figure 1.2: Explicit confirmation versus implicit confirmation [21]

In the early days of chatbots, rule base-approach was trained with human annotated corpora. As anticipated, this approach is very time consuming and needs the conversion of large corpora into corpora useful for a chatbot, i.e. divided into answer/questions pairs. Furthermore, since English is the predominant language used in NLP research, the use of already annotated corpora is likely to be restricted to the English word. [4]

With the advent of the internet, forums pages have become very popular. Straight away, researches saw a gold mine for chatbot corpora because of their question/answer structure. In the same way, E-mails have also been exploited to build corpora. Here, some preprocessing are needed since one has to extract the question/answer structure. Still, the use of forum pages is more convenient because e-mails lack clarity as to specific question-answer pairs within longer emails.[4]

1.1.5 Dialogue management

In order to appear "human", the bot must choose communication strategies and use language tricks. Only then, it deliver its message to the user.

Communication strategies

The system must elect which of the two parts shall begin the dialogue. On the one hand, the user starts the dialogue, and the system retrieves the answer. However, with this strategy, the user may provide ill-defined inputs resulting to no or bad answer from the bot. On the other hand, the bot initiates the conversation. Even if this approach considerably reduces error, it also reduces the conversation field and restricts the behaviour of the user.

Once the precursor of the dialogue selected, the dialogue management has to pick an error handling and confirmation strategy.[4] Two main approaches can be taken: explicit and implicit confirmation. In the former, the bot will merely repeat the query it thinks it heard. As one can expect, this is a time-consuming approach but ensures a good understanding of the query. Whereas, in the latter, the bot includes the query it thinks he heard as part of the next question. The two situations (fig.1.2) have been illustrated by McTear, Callejas, & Griol (2016).

Language tricks

The set of responses generated by `response generation` are most of the time generated with their corresponding confidence scores indicating how likely the response is appropriate. For responses having low probabilities, the bot can develop some language tricks that improve conversation outcomes. A set of strategies can be found in Yu & al. [40]. It includes:

- Switch topic
- Tell a joke

- Elicit more information
- End topic with an open question

Human imitation strategies

In addition to language tricks, the bot has a range of others means in order to appear more "human". The more popular are: develop a personality, make some small talk and do some "human-like" failures.

1.1.6 Text to speech

This is the final step of the chatbot journey, it converts the response generated to speech and returned to the user. As one could expect, the system will do the steps described in subsection 1.1.1. First, the text is converted into phonemes with pitch and duration, then segments of recorded speech corresponding to each phoneme are concatenated to form speech. [4]

1.2 Existing solutions

From open source solutions to commercial ones, there is a large range of build-in chatbots nowadays. It is no doubt that the choice of using a build in solution or make from scratch its own is a cornerstone at the design stage.
This section aims to introduce the reader to most known solutions. To that end, we pick two commercial systems (Watson conversation, Luis) and two open source (Wit.ai, Rasa NLU). Further information on those systems will be presented in chapter 4.

1.2.1 Watson assistant

Watson assistant [17] is the platform created by IBM and part of the IBM Bluemix cloud services. It shares Bluemix pricing schema and may access to additional features.[5]

Built on a neural network (one billion Wikipedia words), it understands intents, interprets entities and dialogues. It achieves accuracy by attempting to assess as much context as possible. The information is retrieved both at the passage of the question and from the knowledge base available. Watson teases apart the question and potential responses in the corpus, and then, examines it and the context of the statement in hundreds of ways. Watson then uses the results to gain a degree of confidence in its interpretation of the question and potential answers.[5]

Supporting over thirteen languages [16], it also has a wide range of developer tools as REST APIs, and SDKs made to facilitate integration in various systems.

1.2.2 Luis

Language Understanding Intelligent Service or LUIS is the platform developed by Microsoft. In the same way than Watson, Luis is part of Microsoft cloud, Azure, and shares the pricing schema of it.

All its applications are centred on a domain-specific topic or are content related. Active learning technology is one of LUIS's features. It is possible to use preexisting, world-class, pre-built models from Bing and Cortana.[5]

As a REST API, Luis can be integrated into any project that uses HTTP requests. Furthermore, it supports eighteen languages. [22]

1.2.3 Wit.ai

Wit.ai [39] is an open source solution, it is a natural language interface for applications capable of turning sentences into structured data. In contrast to the previous two solutions, this system doesn't work by itself, Wit needs a messaging platform.

It allows using entities, intents, contexts, actions and it incorporates natural language processing (NLP). [6] Surprisingly handling around seventy-five languages, it can be integrated through their HTPP API or by using one of their official clients. [38]

1.2.4 Rasa NLU

Rasa NLU is an open-source natural language processing tool for intent classification and entity extraction in chatbots.[31] In opposition to the other three solutions, Rasa runs locally. As a consequence, it offers the typical advantages of self-hosted open source software (adaptability, data control, etc.). [3]

Like the others, it allows intent classification and entity extraction. Rasa manages a vast range of languages. The developer has to download the right model from their GitHub page.[30]

Chapter 2

Language proficiency assessment

As stated in the introduction, the goal of this master thesis is two-fold. In the previous chapter, we study the first aspect of this work: chatbots. In this chapter, we will be interested in the other aspect: language assessment.

Assessing the proficiency level of a non-native (L2) learners is a key point not only in the context of a learning platform but also for teachers collaboration and for the recognition of language qualifications. With the aim of having a framework of reference, the European Council created the Common European Framework of Reference for Languages (CEFR).[24] This framework decomposes learners' level in three main categories and for each, two sub-categories (see fig. 3.1). For various dimensions of proficiency (i.e. speaking, writing, etc.), it lists 'can-do' descriptors that can be used to assign a level to a learner. [34]

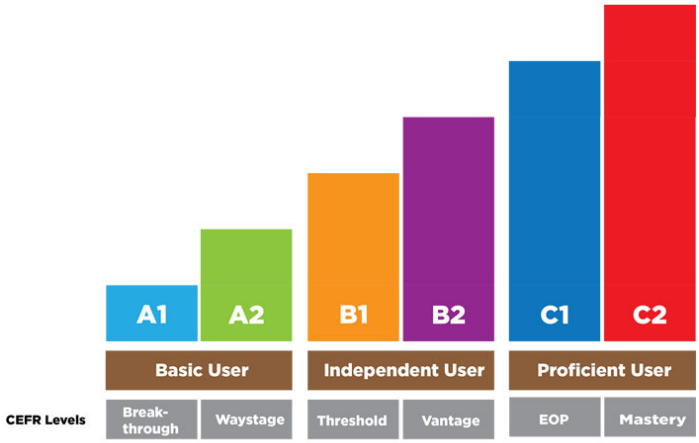

Figure 2.1: Differents levels of CEFR framework. [7]

For simplicity, in the rest of the chapter, we would focus on assessing the **writing** skills of the learner.

Traditionally, we can observe two methods for the assessment of the learner's level. In the first one, experienced and trained humans graders grade students' free text responses [36]. In the second one, the learner completes a test with multiple choice and fill-the-blanks questions. While the later can be performed by computers and thus more rapidly, the classification is not as good as in the former which has the drawback of being time-consuming.

In the context of a learning app, implementing an expert based solution could be near to

impossible. The platform would need a large panel of experts evaluating the same text to guarantee the validity of the evaluation. Furthermore, to give the best experience to the users, those experts would have to be quickly available to give feedback to the user as soon as possible. Thus, one could imagine a hybrid approach to tackle the assessment problem: implement a system that allows the user to receive feedback quickly and to give free text responses while answering the form.
This chapter is structured as follow. In section 2.1, an overview of research in automated essay assessment. In section 2.2, we will focus on short answers assessment and finally, in section 2.3 we will study language assessment.

2.1 Automated assessment

Automated assessment started in the late '60s with the work of Ellis Page (1966)[26]. This work built the bases and defined core concepts of this field. Nowadays, with the advent of technology and the popularity of distance education, automated assessment gained in popularity. [12]

One of the main difficulties faced during the implementation of automatic grading systems is to counter the subjectivity, or at least the perceived subjectivity, of the grading process. Students often perceive the variation of the final grade given by different human assessors as a source of unfairness.[37] With that in mind, first approaches defined a set of measurable features like essay length, word length, etc. and use multiple linear regressions to predict the score of the essay. This quite simple technique worked well. The predicted scores correlated about 50% with the teachers' scores, which is about as well as the teachers agreed to each other. [19]

With the progress of machine learning techniques, more sophisticated models have been developed. Currently, there is a large range of solutions using k-neighbours, neural networks, etc. developed by researchers or by commercial systems.

2.2 Short answers assessment

In an automated assessment, it is common to distinguish `automated essay grading` (AEG) and `automatic short answer grading` (ASAG). This distinction is based on the length, the type of texts and the kind of scoring method used. [34] While automated essay grading focus on the quality and proficiency of the student, automatic short answer grading aims to evaluate students correctness with respect to the initial question. The exactitude of the answer is quantified in relation to a model having a set of model answers and a set of keywords.[34]

The simplest and most popular model for language modelling is Bag-of-ngrams. The idea behind is that the presence (or absence) of a word can predict the desired output. Regardless of its simplicity, it is one of the most powerful predictors in ASAG context. Since it is a question-feature specific method, each question will have its bag of ngrams. [12]

Lexical and semantic similarity are also wildly used in ASAG systems. On the one hand, lexical similarity measures the likeness with respect to a sentence or a word of reference. On the other hand, semantic similarity retrieves from some data set (typically WordNet), information about the semantic distance between words. In other words, it evaluates if a word is a synonym or belongs to the same concept group. [23] In the same way than essay grading, ASAG models can also take into account the length of the answer and word length.

2.3 Language assessment

Language assessment can be useful either as a method of scoring them by their proficiency or for understanding the distinctive features of language at a given proficiency level. [36]

Even though research in L2 assessment has been very active in the last past years, automatic approaches for classifying language learners into standardised proficiency levels, such as CEFR, is quite a new area of interest. [35] This area suffers from a significant drawback: it is is very difficult and time consuming to gather useful data.

In the present study, we follow a hybrid approach: we focus on ASAG like questions while using in addition to that automatic essay grading techniques.

Part II

Tools & implementation

Chapter 3

Tools

3.1 Watson assistant

As stated in chapter 1, Watson assistant is one of the numerous services IBM offers. Its purpose is the design of virtual assistant - chatbot- and its deployment across multiple platforms. It is essential to take note that although Watson assistant can be deployed as a stand-alone solution, it needs to be integrated into a front-end to communicate with the user. Nonetheless, IBM has launched a significant update of this service in November/December 2018 while writing this document. We update the name of features, windows, that change name the best I could.

3.1.1 Architecture

First, the user interacts (orally or written) with Watson assistant through an integration point. This integration point can be one of the following:

- An existing social media platform as Facebook messenger, Slack
- A chatbot user interface hosted by IBM Cloud
- A custom application

The chosen integration point sends the user's message to Watson assistant which transfers the message to the dialogue skill. Next, the dialogue skill interprets the user input further, then directs the flow of the conversation and gathers any information that it needs to respond or perform a transaction on the user's behalf.[11]

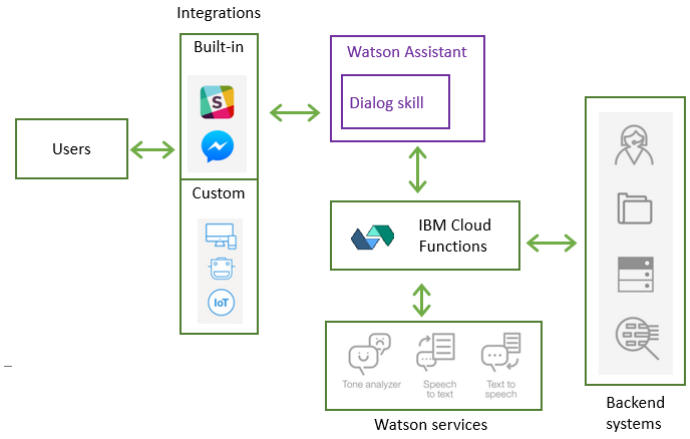

Figure 3.1: Watson assistant overal achitecture [11]

3.1.2 Conversations

Conversations are the core of this system. Through them, the user can interact and achieve a specific goal. "Skills" (formerly called "Workspaces"), are the place in IBM Watson where you can build them. They are composed of three main elements: intents, entities and dialogues.

Intents

An **`intent`** is a purpose or goal expressed in a customer's input. By recognising the intent expressed in a customer's input, the Watson Assistant service can choose the correct dialogue flow to responding to it.[9]

Once a goal is defined, the development team can then create the stated intent and add examples to it. Those examples are examples of possible user inputs that should be recognised as the intent hand by the assistant. Based on those examples, Watson will infer others possible user inputs corresponding to this intent.

For instance, a goal could ask for the weather, ask to switch off the lights or say hello. In fig.3.2 we can see an example of intent. Here the intent is "greetings", we can see the examples are "aloha", "Bonjour". The number of examples to introduce to Watson don't have to be very large, but they have to be as varied as possible.

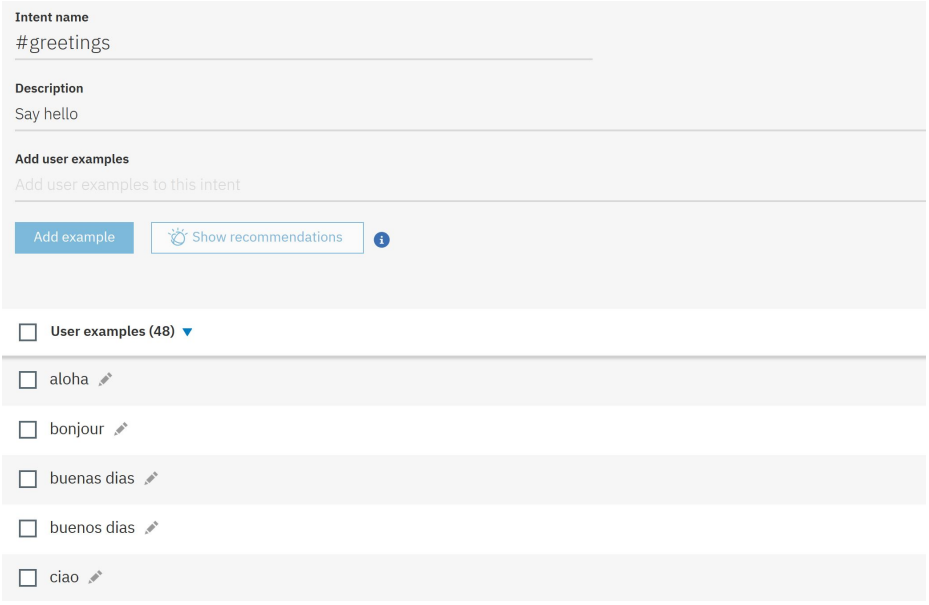

Figure 3.2: An example of intent: "greetings"

Entities

An **`entity`** represents a term or object that provides context for an intent. For example, an entity might be a city name that helps the dialogue to distinguish which store the user wants to know store hours. [8]

One can add all the examples manually, as in fig.3.3 with an example of animals recognition. However, it is possible to ask Watson for recommendations as we can see on fig.3.4.
Furthermore, Watson allows to implement custom pattern matching with regular expressions and use built-in entities such as date, currency, time.

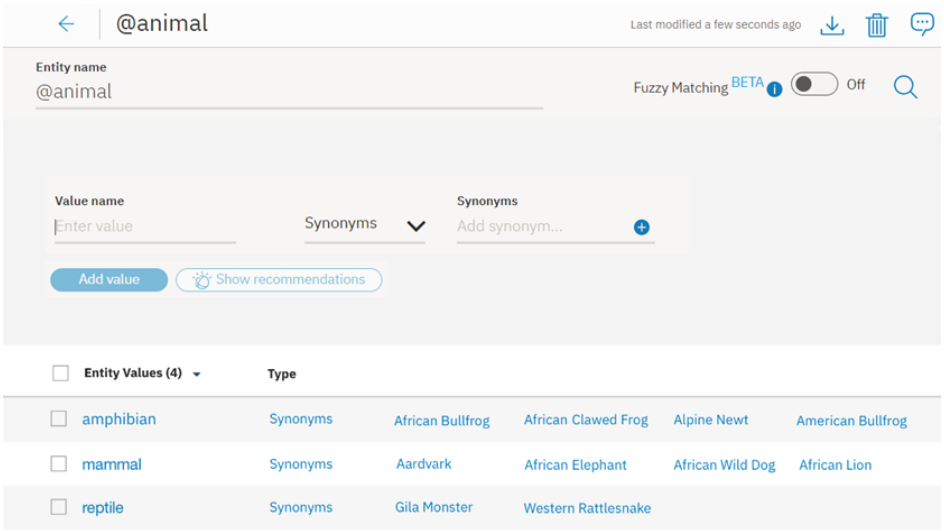

Figure 3.3: Example of entity: animal recognition [8]

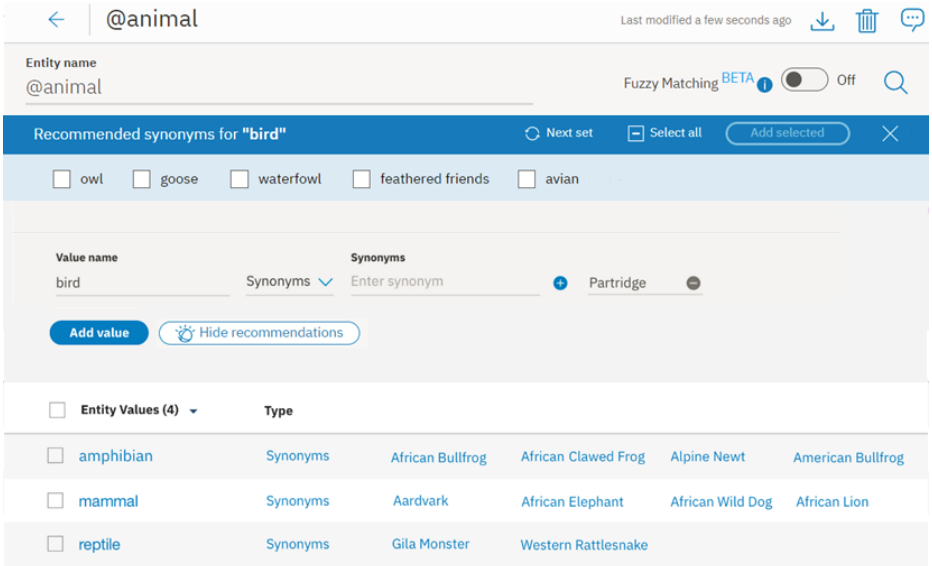

Figure 3.4: Example of entity: animal recognition recommendation [8]

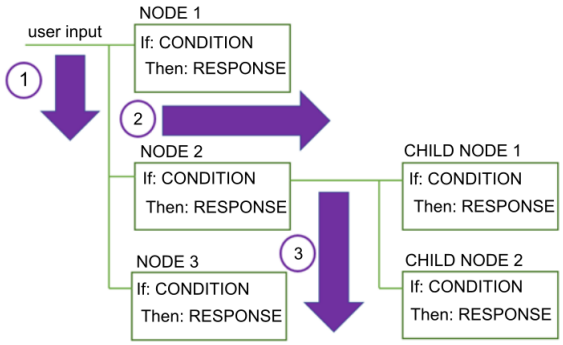

Figure 3.5: dialogue flow navigation [8]

Dialogues

The `dialogue flow` is represented graphically in the tool as a tree. One can add a branch to process each of the intents that you want the service to handle. Then, one can add branch nodes that handle the many possible permutations of a request based on other factors, such as the entities found in the user input or information that is passed to the service from an external service.[10]
The tree is composed by `nodes` and each node is composed by a condition and a response (for the simplest ones).

The **conditions** specify the information that must be present in the user input for this node in the dialogue to be triggered. The information is typically a specific intent. It might also be an entity type, an entity value, or a context variable value. The **response** is the sentence that the service uses as an answer to the user. As users often have more complex requests, a single node may not be able to handle complex answers. For this reason, it is possible to add child nodes that ask the user to provide any additional information that the service needs.

Navigation through the tree is done from top to bottom as we can see on fig.3.5. If the service finds a condition that is met, it then moves along the triggered node to check the user input against any child node conditions. As it checks the child nodes, it moves again from the first child node to the last.

3.2 NLTK

Natural Language Toolkit or `NLTK` is a python library for natural language processing (NLP) [29]. It was developed by Steven Bird and Edward Loper in the Department of Computer and Information Science at the University of Pennsylvania. NLTK provides a huge amount of corpora and lexical resources as well as a wild range of NLP functions.

Before opting for this solution, we check for compatibility between `Unity` and `Python`. Luckily, we found various third-party libraries allowing to execute `Python` code in `Unity`.
For this master thesis, we only use a small part of this library. More particularly, we use some of the corpora given by NLTK and the tokenise class used for tokenising the user input. The specificities will be explained in more details in the next chapter.

Chapter 4

Architecture and implementation

The goal of this master thesis is to propose to the user an entraining way to assess its language level. Because this final application depends on different parts and technologies that interact with each other, the first section presents the architecture and the different parts. Following this, sections 4.2 to 4.4 describe more in more details these different parts, their implementation choices and the algorithms used.

4.1 Architecture

Fig.4.1 shows an overview of the program and the different parts involved. The right part is dedicated to the processing of the user's input while the middle and left parts focus on the interaction with the user.
As the reader can see, the analysis of the input is divided into two components: the top one (IBM Watson), handles the conversation, the bottom one (Python script) that evaluates the user's level. The interface with the user is managed by `Unity` engine with a chat-like interface.

In the next sections, we detail the different parts as they came in the development of this system. First the choice of the service handling the conversation, then the selection of the interface part and finally, the method used for assessing the user's level.

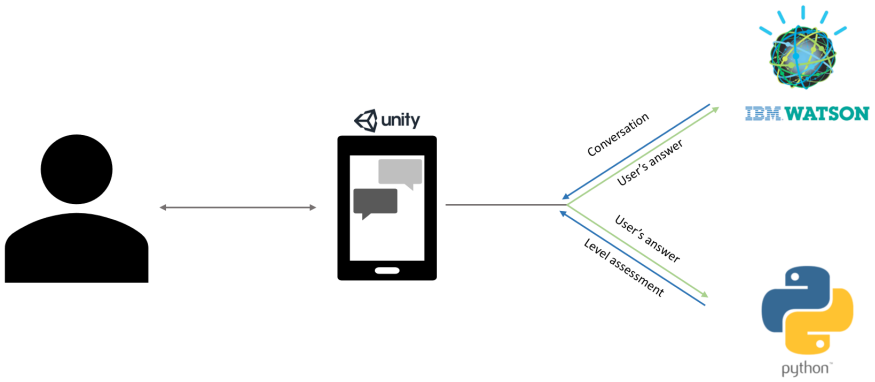

Figure 4.1: Architecture of the program

4.2 Conversation service

Seeing that the conversation flow was a crucial element of the system, we begin the implementation by exploring all the chatbots available and evaluating the cost of building one from scratch.

With the aim of fairly compare bots, we selected the following criteria:

- **Integration** because the chatbot is meant to be integrated into various platforms (mobile and web), this criterion was quite critical for the re-usability of the solution,
- **Licence** the system is designed to be a part of the *Graal* app. Hence it is important that the licence of the service is compatible with the purposes of *Graal*,
- **Languages** as *Graal* is meant to grow and support more languages, using a service with different languages already supported was a plus,
- **Data policy** with the new regulation in the European Union (GDPR) and the importance of controlling users' data, the service has to be as transparent as possible about how they handle the collected data,
- **Community and documentation** having support and documentation to lay on is a big plus for any developer.
- **Pricing** considering that this work is an exploratory one, choosing a service that didn't offer a free trial or a good sample of its capabilities seem senseless. Moreover, as it may be integrated into *Graal* project, open source and low pricing solutions were preferable.

Selected solutions

Among all the existing solutions, we choose to investigate further five of them. Table 4.1 summarise all the different services we consider. Those five were selected based on their capacities and popularity in the related literature. In the beginning, we also consider `DialogFlow` of Google. Its data policy was so confused that we rapidly remove it from the list.

The first service discarded was Luis. Even though it can be integrated into one of the current target platform (Android), its pricing and the reviews across forums and journals put *Luis* out of the game. *Almond* was also quickly eliminated because it only supports English.
At first glance, we were opting for an open source solution, that is `rasa NLU` or `wit.ai`. However, both systems require training on its data, and unfortunately, we didn't have that kind of data, nor we could find it. For this reason, and because it has an integration directly in `Unity`, we choose to develop the chatbot with `IBM Watson Conversation`.

First steps with IBM Watson Conversation

Before anything, we had to create an account into IBM Cloud and create an instance of Watson Conversation. Luckily, IBM provides students accounts, and we could use more functionalities than in the free plan. In order to get familiar with Watson we create a toy example of a conversation. During those experiments, we discover that some functionalities are only available in English.

Watson final dialogue

In order to define the questions, we refer to the guideline of the Council of Europe [25] and the paper of Tack & al. (2017). For each level, we choose two themes and ask several questions about it (see tab 4.2). We try to ask as many questions as possible in a theme intending to having more data for the classification. Moreover, we took advantage of the possibility to jump from a node to another in Watson conversation to, at any time, jump to another node or stop the conversation if the user doesn't understand any more. The final conversation is illustrated on fig.4.2.

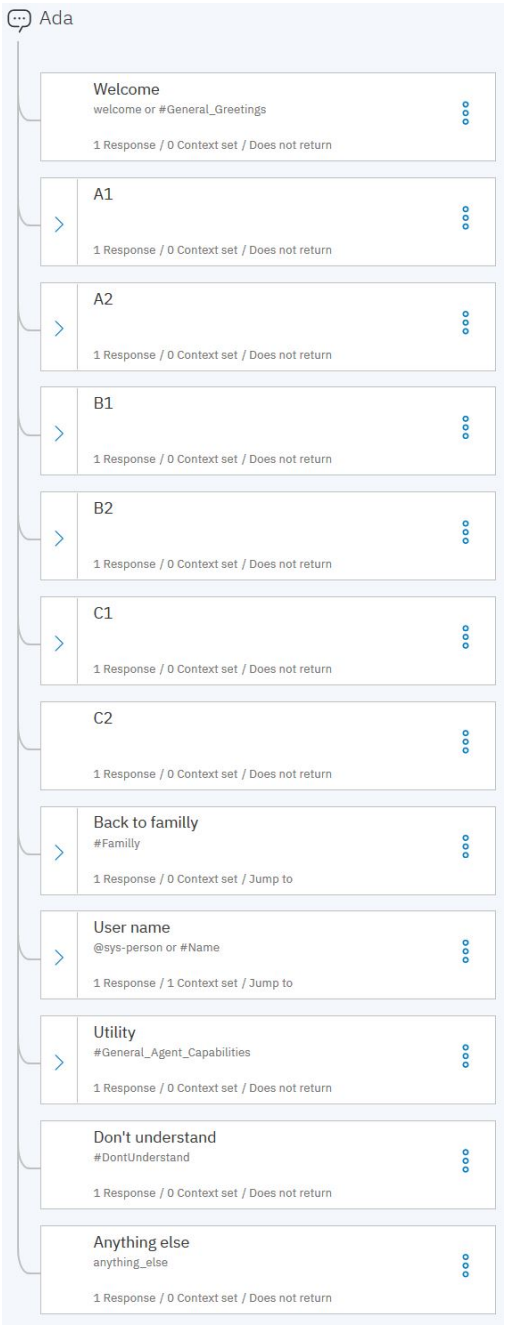

Figure 4.2: Final conversation

4.3 Unity

`Unity` is a cross-platform game engine developed by Unity Technologies. The engine can be used to create both three-dimensional and two-dimensional games as well as simulations for its many platforms. This component is responsive of the direct communication with the user.

As the app developed by *Graal's* team is meant, for now, for mobile devices, we faced two choices: build the solution for Android or create one in Unity. Even though the creation of an Android app handling Watson was more straightforward, it seems more interesting to build a solution that Graal's team may reuse easily. For this reason, we choose Unity for building the interface with the user.

The first task was to build a scene in a new Unity project capable of taking care of the conversation flow. That is the user's input and the answer from IBM Watson. The result is shown on fig.4.3. As it shows, we chose a chat-like interface to keep the illusion to be interacting with someone rather than a machine.

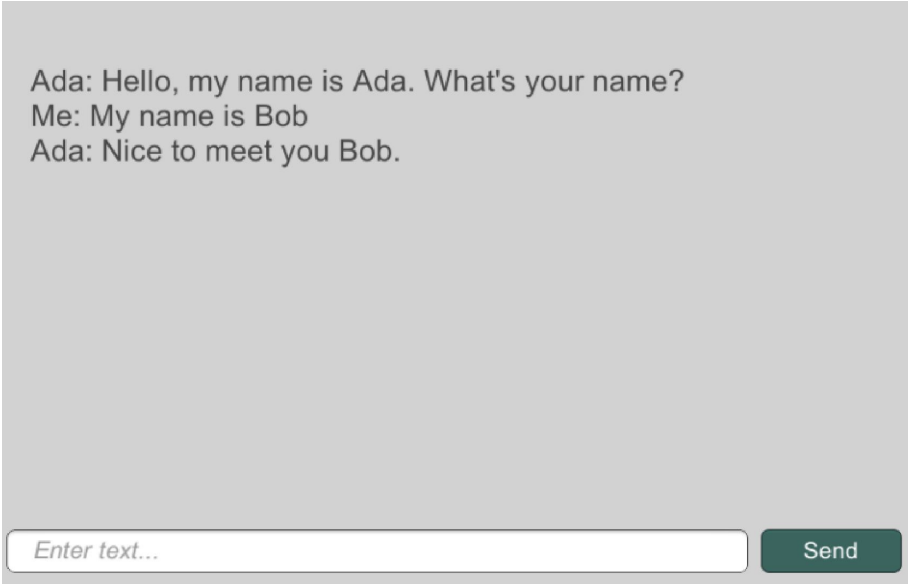

Figure 4.3: Interface with the user on Unity

Once this interface was ready, the next step was to connect the scene to the first conversation build in IBM Watson conversation service. As said previously in section 4.2, Watson offers a humongous number of API including Unity. However, because Watson contains more than a dozen services, the cornerstone was to find the correct script corresponding to Watson assistant service. In addition to that, the documentation of Watson for Unity is quite sparse, so it was also quite difficult to know where exactly in the script modify the credential values. At first, we misunderstood the difference between *speech-to-text/text-to-speech* services and *assistant* service. As a consequence, we pick the script corresponding to *speech-to-text* service **and** *text-to-speech*. Since the credentials for the different services are all different (even if they are created in the same account), we were unsuccessful at the beginning. We finally manage to understand my error and connect Unity to my conversation.

4.4 Level assessment

Once Watson and Unity start to communicate, the next step was building level assessment system. In this section, we first present the first sketched model, then the corpus used. Afterwards, we describe the adapted model based on the corpus used and the performances of the model. After that, we detail the questions selected for the conversation of Watson. Finally, we present the integration of the model into Unity.

First model

As stated in chapter 2, automatic assessment of short answers for classification into CEFR levels is a quite new field of interest. We based our model on architectures found in the literature; in particular, on the paper of Tack & al(2017) [34]. The first model is divided into two phases: preprocessing and classification.

Preprocessing

As preprocessing step, we perform tokenisation (chopping it up into pieces, called tokens, the sentence) and part-of-speech tagging on the texts of the data set. This process will be executed with the help of the `NLTK` library.

Classification

Once the data is preprocessed, we can now train a model using machine learning techniques. To facilitate the classification, CEFR levels will be transformed into numbers from one to six, representing A1, A2, B1 etc.
The classifier will be trained by word/tag couples labelled with their corresponding CEFR level. As we go through the literature, we notice that linear regression did not perform well by contrast to support vectors machine (SVM), decision trees, naive Bayes and k-nearest-neighbours (KNN). Resulting from this observation, we decided to train these four methods and perform a soft majority vote with the result of those models.

Soft majority vote consists on computing the average of the results and then decide to which category it belongs. As opposed to that, a hard majority vote will count the number of results belonging to a specific category and pick the one having the most votes. Soft voting can improve on hard voting because it takes into account more information; it uses each classifier's uncertainty in the final decision.

In addition to that, using Naive Bayes technique, we will compute the probability of a non-tokenized user's input to belong to a specific category (a CEFR level) given the words present in the input. Then, we will compare the results obtained by the two techniques and compute a soft majority vote again.

Corpus

Now that the model is sketched, we were able to determine precisely the type of data that we needed. As the goal is to classify small texts, we were looking for an important number of small texts of different CEFR levels and labelled with their corresponding level. The text could be either, textbook texts or students texts labelled by an expert. Unfortunately, we couldn't find such corpora. We looked in platforms such as `google datasets`[1] and `kaggle`[2] known for distributing open data sets with no success. A solution to this problem was to create our labelled data set by gathering texts of different CEFR levels across learning websites. Nevertheless, doing so would bring inconsistency to the data since all the websites may classify a word, a sentence construction in different levels.

Hence, we modified our search and looked for datasets classifying any corpora or words into CEFR level. As a consequence, we found a dataset classifying a pair word/tag in a CEFR level.

This dataset was produced by Guzey & al. (2014)[15] in a study aiming to provide a methodology for making educational standard aligned language level predictions for all English words.[33] Because words can have different meanings hence PoS tags and thus levels, determining the level of a word is not straightforward.[33] In this study, they select thirty English as a second language teachers and select 7000 words chosen randomly from British National Corpus. Three teachers classified each word/tag pair and then a soft majority vote was applied to defined the final CEFR level.

[1] `https://toolbox.google.com/datasetsearch`
[2] `https://www.kaggle.com/datasets`

The published dataset is composed of eight files: raw data and processed data. For this study, we kept the dataset containing the word/tag pair, the average of all three levels given to the word/tag pair, the CEFR level corresponding to the rounded average of the three teachers. To compute the average, they transformed the categorical levels (A1, B1, etc.) into numerical ones, from one to six. As for the tags used by this study, they are the same used by the `NLTK` library which facilitated the construction of the model.

Adapted model

Because we couldn't find reliable texts data sets, the model was modified. On the contrary of the classification part, we did not modify the preprocessing part.
The classification of the non-tokenised user's input using naive Bayes was now beyond the bounds of possibility since there were no training texts available. As for using machine learning techniques, training those classifiers just on a list of pairs word/tag was senseless. For this reason, the model was transformed into looking up in the data set for each pair word/tag, retrieve its corresponding level and then compute a soft majority vote on the answers.

Model performance

Because we couldn't gather texts of different levels labelled with their corresponding CEFR level, testing the system was complicated. However, since for testing we need less data than for training, we gather data from learning website[3]. For each level, we concatenate the available texts and split the text such as there is one sentence per line. As a result, we end up with six files, one for each level. We then try to predict the level of a sentence and compute the measure the performance of the model.

As we can see on table 4.3, our model does not meet the requirements found during the theoretical analysis. Indeed, we can observe that this model always predicts *A1* level. This may occur since the data that is the data set of Guzey & al. (2014)[15] does not take into account any context. Additionally, the texts used for this performance assessment are texts build for **reading** skills assessment. This can introduce bias into our measures since the themes and vocabulary are slightly different from the ones used for **witting** skills assessment.

Furthermore, we also try to take all the sentences of a level as a "big" sentence and try to classify it. Unfortunately, doing so didn't change the results, and all the "big" sentences were classified as being *A1*.

Integration in Unity

Unity originally comes with support for scripting in C# and `Javascript` by default. However, as said in the previous chapter, it exists third-party libraries allowing running Python code in Unity.

We try first to use `IronPython`[4]. In the first time, we run small examples as simple calculus or display something. Once the small examples worked, we were able to launch the classification scripts. In spite of that those working examples, running all the `Python` file was arduous. Despite our best efforts, we were not able to run them. This is due to the fact that we use some `Python` libraries and `IronPython` does not support the import of libraries. We looked at the others plugins for running `Python` in `Unity` but they all seem to have the same issue.

[3] `https://www.examenglish.com/`
[4] `http://ironpython.net`

Furthermore, while going through numerous forums, we discover that even if it was possible to run `Python` in `Unity` this solution may not work once we export our `Unity` project in `Android` or `IOs`. Indeed, those plugins use the `Python` version used in the computer, and since phones don't usually have `Python`, it may don't work.

Name	Integration	Licence	Languages	Data policy	Comunity	Doc and Pricing
Microsoft Language Understanding intelligent service(Luis)	C# SDK Python SDK NodeJS SDK Android SDK	-	Spanish, English, Portuguese, Italian, German, Japoneese, Korean, French, Dutch, Chinese	Client data is used in for the services they are using. When leaving the service, data is deleted	Active community, documentation exists	Pricing of Azure, depends on the number of transactions per second
IBM Watson assistant	Swift SDK Ruby SDK OpenWhisk SDK Node SDK Java SDK Python SDK .Net SDK Unity SDK Salesforce SDK	-	English, Arabic, Chineese, Czech, Dutch, French, German, Italian, Japanese, Korean, Portuguese, Spanish	Client maintain ownership of its data, insights, training, and IP.	Active community, webinar, a lot of documentation, a slack	Free version contains good sample of their capabilities, princing of IBM Cloud
Almond	Principally for IoT but has integration with Android	OpenSource under GNU v2	English	Self hosted, client maintain ownership	Few documentation	Free
wit.ai	Node.js Python Ruby HTTP API	OpenSource, no licence	Albanian, Arabic, Azerbaijani, Bengali, Bosnian, Bulgarian, Burmese, Catalan, Chinese, Croatian, Czech, Danish, Dutch, English, Estonian, Finnish, French, Georgian, German, Greek, Hebrew, Hindi, Hungarian, Icelandic,Indonesian, Italian, Japanese, Korean, Latin, Lithuanian, Macedonian, Malay, Norwegian, Persian, Polish, Portuguese, Romanian, Russian, Serbian, Slovak, Slovenian, Spanish, Swahili, Swedish, Tagalog, Tamil, Thai, Turkish, Ukrainian and Vietnamese.	Client maintain ownership	Good documentation, community on Github	Free
rasa NLU	HTTP API Python	OpenSource Apache license	English, German, Spanish, Portuguese, Italian, Dutch, French	Self hosted, client maintain ownership	Good documentation, community on Github	Free

Table 4.1: Comparative table of chatbot services

Level	Topics
A1	family hobies
A2	holiday memories lifetime goals
B1	book reading spending 1 million euros
B2	enjoy work or earn money study abroad
C1	improve the environment social networks
C2	leading a healthy life living in the public eye

Table 4.2: Questions themes per CEFR level

		Actual class					
		A1	A2	B1	B2	C1	C2
Predicted class	A1	77	106	83	97	61	16
	A2	0	0	0	0	0	0
	B1	0	0	0	0	0	0
	B2	0	0	0	0	0	0
	C1	0	0	0	0	0	0
	C2	0	0	0	0	0	0

Table 4.3: Performance results

Chapter 5

Discussion

In the previous chapters, we have seen the different components of our system and the theory related to it. We also depict the way that all those components interact with each other to build the final application. Given the variety of the components and the fact that they are not, *a priori*, linked, we first focused on each part and then, we assembled each element with the aim of implementing the final system. Resulting from this, it remains improvements that can be applied to the different aspects.

Because of the results of the model performance, the level assessment part is the one that better suits improvements. Its limitations are discernible: it always classifies a sentence as being *A1* level. To tackle this problem, more adequate training data and the use of the first model instead of the adapted one could improve the performances of the model.
Furthermore, one could investigate the use of finite state machines in order to take into account the structure of the sentences. With this model, we could compare the structures of labelled sentences real students (this would be the training data of our model), to the user's input and so, highlight the frequent structure mistakes done in each level by the students.

Secondly, the chatbot conversation part could benefit and help the classification, if we could enable direct communication between the classification scripts and Watson. At the moment they are two different running entities, each one performing its role. By allowing direct communication, the assessment of the level would be refined as the conversation could ask more questions of a level classification script is uncertain about.

Thirdly, a better way to run the `Python` scripts would be to run a remote server instead of trying to run them on directly on `Unity`. The sentence to be analysed would be sent to this server, examined, and the feedback would be send back to `Unity` or even directly to Watson. This way, we would counter the limitations of `Unity` scripting.

Finally, the current interface of `Unity` could be modified. Currently, it looks like a chat on a website. To be more user-friendly, it could be altered and look-a-like text messages app or social networks messaging apps.

Chapter 6

Conclusion

The purpose of this master thesis is two-fold: build an intelligent chatbot into `Unity` and create a model capable of correctly classify the user's level in the CEFR levels reference. In order to propose such kind of application, several challenges have to be tackled. As there are two very distinguishable parts in this work, we divided our research into different parts.

In the first time, we study the literature to define the core concepts of a chatbot. Afterwards, we review popular existing solutions. We summarise two commercial systems and two open source.

In a second time, we focus on language proficiency assessment. We define the different levels of proficiency described by the European Council and concentrate our work on the assessment of writing skills. Then, we investigate three fields of assessments. First, automated assessment, characterise the field of automatic grading systems. The second field, short answers assessment, focus on the implementation of automated assessment in the context of small answers, that is only a few lines of input. The third field, language assessment, we briefly present this new field and present our hybrid approach.

To be able to build the required system, in the third time, we present all the tools used. First an introduction to IBM Watson conversation service, its components and structure. Then, we introduce `NLTK` library used in the classification.

The final purpose being to create an application, we establish the architecture of the whole application. After that, we depict each component of the final system. First, we explain our choice of using a built-in solution and present all the considered solutions. As a result, we choose IBM Watson since it fit better to our criteria. Then, we expose how we integrate Watson into Unity.
Finally, we described our model of level assessment. To begin, we illustrate the first model we established. Because in any assessment task requires to use data about the real world, we analyse the available sources of data. As the field of automated language assessment is quite new, we didn't find the data we were looking for. However, we find another kind of data that we could use. For this reason, the first model had to be changed and adapted to this data.

To conclude, we can say that the final application proposed at the end of this master thesis is a sketch of a future platform. On the one hand, the first goal, namely, the creation of a chatbot agent into `Unity` has successfully been achieved. On the other hand, the assessment part of this master thesis stayed at a theoretical level. However, we can now define better the structure of the data needed for this kind of application and conceive ways to collect it.

Bibliography

[1] Sameera A. Abdul-Kader and Dr. John Woods. Survey on chatbot design techniques in speech conversation systems. *International Journal of Advanced Computer Science and Applications*, 6(7), 2015.

[2] Prashanti Angara, Miguel Jiménez, Kirti Agarwal, Harshit Jain, Roshni Jain, Ulrike Stege, Sudhakar Ganti, Hausi A. Müller, and Joanna W. Ng. Foodie fooderson a conversational agent for the smart kitchen. In *CASCON*, 2017.

[3] Daniel Braun, Adrian Hernandez-Mendez, Florian Matthes, and Manfred Langen. Evaluating natural language understanding services for conversational question answering systems. In *Proceedings of the 18th Annual SIGdial Meeting on Discourse and Dialogue*, pages 174–185, 2017.

[4] Jack Cahn. Chatbot: Architecture, design, and development. *University of Pennsylvania School of Engineering and Applied Science Department of Computer and Information Science*, 2017.

[5] Massimo Canonico and Luigi De Russis. A comparison and critique of natural language understanding tools. *CLOUD COMPUTING 2018*, page 120, 2018.

[6] Olga Davydova. 25 chatbot platforms: A comparative table. `https://chatbotsjournal.com/25-chatbot-platforms-a-comparative-table-aeefc932eaff`. Accessed: 08-11-2018.

[7] ML Desnos. Présentation du cecrl (cefrl). `https://www.whatnot.fr/presentation-de-cecrl/`, 2014. Accessed: 01-12-2018.

[8] IBM Cloud Docs. Defining entites. `https://console.bluemix.net/docs/services/assistant/entities.html?locale=en#entity-described`, 2018. Accessed: 19-12-2018.

[9] IBM Cloud Docs. Defining intents. `https://console.bluemix.net/docs/services/assistant/intents.html?locale=en#intent-described`, 2018. Accessed: 15-12-2018.

[10] IBM Cloud Docs. Dialog overview. `https://console.bluemix.net/docs/services/assistant/dialog-overview.html?locale=en#dialog-overview`, 2018. Accessed: 19-12-2018.

[11] IBM Cloud Docs. Watson assistant. `https://console.bluemix.net/docs/services/assistant/index.html?locale=en#a-propos-de`, 2018. Accessed: 15-12-2018.

[12] Lucas Galhardi, Cinthyan Renata Barbosa, Rodrigo Clemente Thom de Souza, and Jacques Duílio Brancher. Portuguese automatic short answer grading. In *Brazilian Symposium on Computers in Education (Simpósio Brasileiro de Informática na Educação-SBIE)*, volume 29, page 1373, 2018.

[13] Palash Goyal, Sumit Pandey, and Karan Jain. Deep learning for natural language processing.

[14] Rainer E Gruhn, Wolfgang Minker, and Satoshi Nakamura. *Statistical pronunciation modeling for non-native speech processing*. Springer Science & Business Media, 2011.

[15] Onur Guzey, Gihad Sohsah, and Muhammed Unal. Classification of word levels with usage frequency, expert opinions and machine learning, October 2014.

[16] IBM. Documentation ibm cloudwatson assistant. `https://console.bluemix.net/docs/services/conversation/getting-started.html#gettingstarted`. Accessed: 08-11-2018.

[17] IBM. Watson assistant. `https://www.ibm.com/cloud/watson-assistant?lnk=ushpv18t3&lnk2=trial_mkt_WatAssist&psrc=none&pexp=def`. Accessed: 08-11-2018.

[18] Sara Perez Josh Constine. Facebook messenger now allows payments in its 30,000 chat bots. `https://techcrunch.com/2016/09/12/messenger-bot-payments/?guccounter=1`. Accessed: 17-10-18.

[19] Leah S Larkey. Automatic essay grading using text categorization techniques. In *Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 90–95. ACM, 1998.

[20] Chin-Hui Lee. From knowledge-ignorant to knowledge-rich modeling: A new speech research paradigm for next generation automatic speech recognition. In *Proc. ICSLP*, volume 4, 2004.

[21] Michael McTear, Zoraida Callejas, and David Griol. The dawn of the conversational interface. In *The Conversational Interface*, pages 11–24. Springer, 2016.

[22] Microsoft. Qu'est-ce que le service language understanding (luis). `https://docs.microsoft.com/fr-fr/azure/cognitive-services/luis/what-is-luis#implementing-luis`. Accessed: 08-11-2018.

[23] Michael Mohler and Rada Mihalcea. Text-to-text semantic similarity for automatic short answer grading. In *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics*, pages 567–575. Association for Computational Linguistics, 2009.

[24] Council of Europe. Common european framework of reference for languages: Learning, teaching, assessment (cefr). `https://www.coe.int/en/web/common-european-framework-reference-languages`. Accessed: 01-12-2018.

[25] Council of Europe. Council for Cultural Co-operation. Education Committee. Modern Languages Division. *Common European Framework of Reference for Languages: learning, teaching, assessment*. Cambridge University Press, 2001.

[26] Ellis B Page. The imminence of... grading essays by computer. *The Phi Delta Kappan*, 47(5):238–243, 1966.

[27] Cédrick Fairon Pierre Dupont. [lingi2263] computational linguistics. Class material, 2018.

[28] PiLab. Graal spin-off project. `https://pilab.be/about-project/?projectID=5`. Accessed: 17-09-2018.

[29] NLTK Project. Nltk 3.4 documentation. `https://www.nltk.org/`, 2018. Accessed: 19-12-2018.

[30] Rasa. Pre-trained word vectors. `https://github.com/facebookresearch/fastText/blob/master/pretrained-vectors.md`. Accessed: 08-11-2018.

[31] Rasa. Rasa nlu: Language understanding for chatbots and ai assistants. https://www.rasa.com/docs/nlu. Accessed: 08-11-2018.

[32] Margaret Rouse. Definition chatbot. https://searchcrm.techtarget.com/definition/chatbot. Accessed: 19-09-2018.

[33] Gihad N Sohsah, Muhammed Esad Ünal, and Onur Güzey. Classification of word levels with usage frequency, expert opinions and machine learning. *British Journal of Educational Technology*, 46(5):1097–1101, 2015.

[34] Anaïs Tack, Thomas François, Sophie Roekhaut, and Cédrick Fairon. Human and automated cefr-based grading of short answers. In *Proceedings of the 12th Workshop on Innovative Use of NLP for Building Educational Applications*, pages 169–179, 2017.

[35] Sowmya Vajjala and Kaidi Loo. Role of morpho-syntactic features in estonian proficiency classification. In *Proceedings of the Eighth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 63–72, 2013.

[36] Sowmya Vajjala and Kaidi Loo. Automatic cefr level prediction for estonian learner text. In *Proceedings of the third workshop on NLP for computer-assisted language learning*, pages 113–127, 2014.

[37] Salvatore Valenti, Francesca Neri, and Alessandro Cucchiarelli. An overview of current research on automated essay grading. *Journal of Information Technology Education: Research*, 2:319–330, 2003.

[38] Wit.ai. Integrate into my app. https://wit.ai/docs/recipes#integrate-into-my-app. Accessed: 08-11-2018.

[39] Wit.ai. Natural language for developers. https://wit.ai. Accessed: 08-11-2018.

[40] Zhou Yu, Ziyu Xu, Alan W Black, and Alexander Rudnicky. Strategy and policy learning for non-task-oriented conversational systems. In *Proceedings of the 17th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 404–412, 2016.

UNIVERSITÉ CATHOLIQUE DE LOUVAIN
École polytechnique de Louvain
Rue Archimède, 1 bte L6.11.01, 1348 Louvain-la-Neuve, Belgique | **www.uclouvain.be/epl**