Methods for NLP: Third Project Report

Vocabulary Building Chatbot

Group B

Melpomeni CHIOUTAKOU
Elisa LUBRINI
Anna MOSOLOVA
Cindy PEREIRA
Anahita POULAD

December 2020

UNIVERSITÉ DE LORRAINE

Institut des sciences du Digital Management & Cognition
COMPOSANTE DE L'UNIVERSITÉ DE LORRAINE

Contents

Abstract

The project consists in the development and evaluation of a language learning tool for adult English learners focused on vocabulary building. Such tool is presented to the user as a Telegram chatbot carrying out the main tasks of: (1) assessing the user's English vocabulary size, (2) recording their reading genre preferences, (3) providing them with suitable reading material based on such preferences and vocabulary size, and (4) provide the user with information on unknown words in order to help them enrich their English vocabulary.

For the assessment to take place, a free-to-use vocabulary size test was converted into Telegram message interactions between the chatbot (assessor) and the user (test-taker). The test percentage scores are used to assign to the user a CEFR (Common European Framework of Reference for Languages) level and exploit this data in the selection of an adequate extract from the corpus (British National Corpus) for the user to read. Another piece of information needed for the selection of the corpus is the reading preferences of the user in terms of genre, which is asked to the user after the test.

In order for the extract to be appropriate for the user's level, a lexical complexity assessment was carried out following a comparison of different tools and metrics. We considered three main methods to be compared: (1) cardinality of intersections between the corpus and CEFR word lists, (2) already existing and publicly available lexical complexity analysers, and (3) different neural network architectures trained with data retrieved from Cambridge English Assessments.

Once an excerpt of the corpus has been selected and sent to the user in form of a Telegram text, they will be able to read it and communicate to the chatbot any words they did not understand. The chatbot will provide information about the word, such as definition, part-of-speech tagging, and usage examples in the form of text (retrieved from WordNet, through the NLTK library, and spaCy), on top of a vocal message (generated from a Google Text-to-Speech audio file) with the purpose of helping the user enhance their English vocabulary.

1 Introduction

This document is a report of the group project assigned for the modules of *Methods for NLP* and *Written Corpora*. The project is an attempt to design and develop a functioning application covering at least three NLP tasks while satisfying the requirements of both the *Methods for NLP* module, by focusing on designing a solution to a problem and evaluating its implementation, and the *Written Corpora* module, by designing an optimal data structure to efficiently handle textual data.

1.1 Aims and objectives of the application

The project consists in the development and evaluation of a language learning tool for English focused on vocabulary building. Such tool is presented to the user as a Telegram chatbot carrying out the main tasks of: (1) assessing the user's English vocabulary size, (2) recording their reading genre preferences, (3) providing them with suitable reading material based on such preferences and vocabulary size, and (4) provide the user with information on unknown words in order to help them enrich their English vocabulary.

The aim of the application would be to provide adult language learners with an easily accessible and user-friendly tool to enrich their English vocabulary size by leveraging their personal interests and pre-existing knowledge to make their experience as pleasurable and smooth as possible. The application will provide the user with appropriate reading material they are more likely to enjoy and learn from. Additionally, the possibility to look up unknown words in-app is supposed to help the learning process by making it straight-forward and encouraging the user to explore new vocables without the need for additional resources.

1.2 Overview of the project progress

The first step of the project was to gather ideas for the application and then design a solution to be implemented in Python while satisfying the requirements given for the corresponding modules. In that phase, we developed a user interactions pipeline (Figure 1) and defined the features to be included in our application (Section 3.1). Once the main characteristics of the application were established, we started delineating the tasks composing the development process (Figure 5) and we assigned the task to three main branches of the project and designated group members to each one of them, in order to work on them simultaneously: two members of the group worked on the data structure, two members worked on the chatbot and one worked on the lexical complexity evaluation. Following, is the progress in each of these branches.

Data handling A script was designed to produce three CSV files storing relevant information about words, sentences and texts found in the corpora for the purposes explained in Subsection 5.2.2. Then a lexical complexity score column was added to the CSV tables and a script was written to retrieve the word information that is to be sent to the user when they want to look up an unknown term. Another CSV file was generated from a Google Forms survey enabled for the user experience evaluation (Subsection 4.2). All data was processed using pandas dataframes (more info on all the tools used in Section 3.2.7).

Chatbot The script setting up the chatbot allows it to successfully administer a vocabulary test and return a score. It assigns a CEFR[1] label to the user along to their score, suggests relevant reading material, and assist their unknown word lookup.

Evaluation: For the quantitative evaluation of the lexical complexity task, a comparison was carried out between the best algorithms listed in Subsection 4.1, obtaining extremely scarce results for each one of them, because of the small training sample (see Subsection 4.1 for more details). As for the qualitative and quantitative evaluation of the user experience, a survey was set up to be filled out by our test users in the human evaluation phase (see Subsection 4.1).

1.3 Overview of this report

This document is the last of a series of three update reports providing a current overview of our project to develop a vocabulary-building chatbot app. A fourth and final report will provide a complete overview of the project, together with a comprehensive appendix including full evaluation results and scripts produced.

This first section (Section 1) serves as an introduction to the concepts behind the application and an update on the project progress. The following section (Section 2) presents the reasons behind some choices we made while planning the application and its design. Next, the implementation of such design is discussed (Section 3) both from the developers' perspective and from the point of view of its structure and features as seen by the user. One of the processes of our development phase is discussed in detail in a separate section (Section 4) as it consists in an evaluation of different algorithms at the core of our project on top of other qualitative and quantitative evaluations of the project itself, overall. Finally, a conclusion (Section 5) summing up the report is proposed along with the required module-specific summaries.

[1] Common European Framework of Reference for Languages

2 Background Research and Design

2.1 Motivation and currently available products

By allowing users to hold conversations without the need to engage in human-to-human interactions, chatbots have become a popular tool in various domains and for various tasks. On top of some popular purposes, such as customer-service or productivity, some chatbots have been created with the objective of helping users with their language learning.

Since chatbots are supported by some popular messaging applications, it is now possible to program a bot without developing a whole GUI to present it: one can exploit the already existing messaging interface to allow interactions with the user and therefore focus on building the back-end architecture. One of the most popular chatbot-supporting messaging apps is Telegram, counting more than 400 million monthly active users[2].

Among the most popular Telegram chatbots for English language learning, we can find bots such as @andybot, which helps users learn new words by playing a guessing game, or chatbots that provide the user with English pronunciation or word definitions. However, even the most popular chatbots seem to be quite limited in their functionalities and, when more advanced functionalities are available, they require an additional app to be installed for the user to access them.

Our application aims at providing multiple functionalities within the Telegram GUI without requiring the user to download additional applications.

2.2 Design of the application

While planning our application, we decided to think about its design as an architecture of features revolving around our ideal user experience (see user interactions in Figure 1). The very first time a user interacts with the bot, they are presented with a test to detect their vocabulary size (size expressed in terms of its CEFR level[3]) and then asked to enter their genre preferences for reading material.

The bot then proceeds to propose reading material to the user, based on the previously assessed vocabulary size and genre preferences. Such reading material consists in texts (extracts from the British National Corpus) which are sent to the user in the form of one sentence per message; after each message the user has the chance to either select a word they do not know or proceed to read the next sentence. Whenever the user selects an unknown word, the chatbot replies with: word definition, POS tagging, usage examples, and an audio with the correct pronunciation. Once the last sentence of the text has been sent and the user confirms their understanding of all the words, the bot asks the user whether they are interested in reading another text.

Please refer to Section 3.1 for more information on specific features and to Section 3.2 for more information on individual development processes.

[2] urlhttps://telegram.org/blog/400-million

[3] https://www.coe.int/en/web/common-european-framework-reference-languages/level-descriptions

3 Implementation

This section provides a quick presentation of the main features of the application and illustrates the process of developing it. Unless otherwise mentioned, all code is written in Python 3. All the scripts mentioned are available on GitHub[4].

3.1 Front-end structure and features

GUI and user interactions User interactions (Figure 1) are handled by a rule-based Telegram chatbot. The typing textbox in the chat is disabled, so that the user will only be able to communicate by pressing buttons that the bot presents at each round of interaction; this prevents unexpected input and, when the user wants to ask about an unknown word, this helps discerning between different instances of the same word being used in the current sentence, since different buttons are used for different instances (see Figure 3).

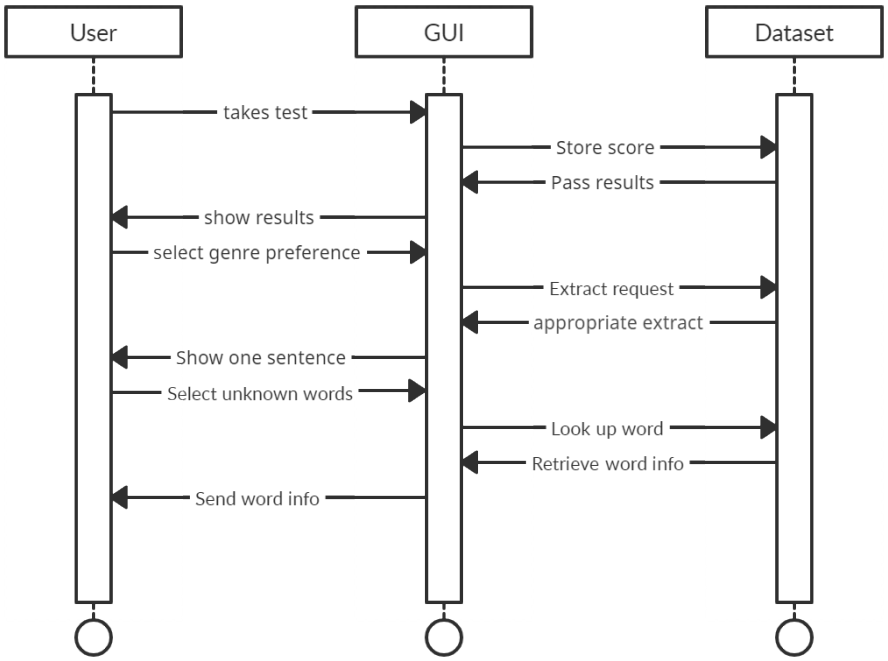

Figure 1: Chronological representation of an example of user interaction in case of a first-time user asking for information on an unknown word.

[4]https://github.com/anya-bel/vocamprove

Vocabulary test The aim of the vocabulary test is to evaluate the user's English vocabulary size for the purpose of accurately suggesting a text that will help them enrich their vocabulary. For the test, LexTale[5], a free to use web app for vocabulary assessment, was used. Its features were integrated in the chatbot conversation on Telegram. The tests consists in a series of words prompted to the user for each of which they have to express whether they think such a word exists or not (Figure 2).

Reading material The chatbot provides the user with reading material in the form of extracts from the British National Corpus. Each time the user asks the bot for a text to read, an extract from the corpus is selected based on genre preferences and lexical complexity. The extract is then divided into sentences which are sent to the user in form of Telegram messages. After each sentence the user has the choice to either ask for information on one of the words (Figure 3 and 4) or proceed to read the following sentence until the text is completed. This is supposed to help the user enrich their English vocabulary, which in turn has the potential to contribute to their overall language proficiency.

Figure 2: The vocabulary size test consists in 60 words that the user has to classify as belonging or not to the English language.

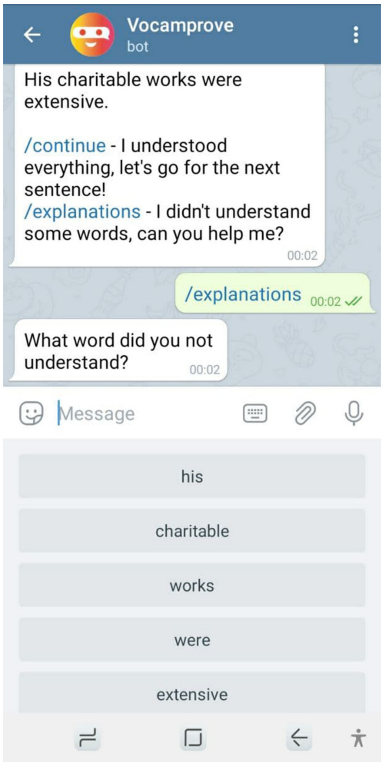

Figure 3: Words from a sentence can be selected using buttons at the bottom of the screen.

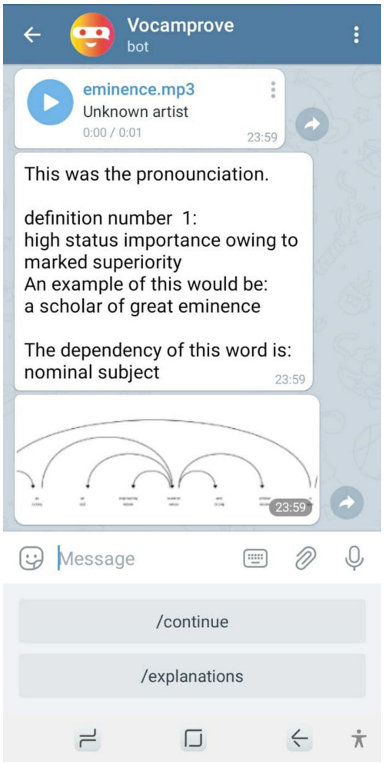

Figure 4: Pronunciation, definition, usage examples and dependency are sent to the user via text/multimedia messages.

[5]urlhttp://www.lextale.com/downloadthetest.html

Looking up unknown words When the user reads the text suggested by the bot, they may encounter some unknown words. They will be able to select those words, and the application would provide them with the definition, morphological and syntactic features in the context of the current sentence, an example of the word in use, as well as a recording of the word's pronunciation and a picture showing dependency relationships, for those interested in more technical information (Figure 4).

3.2 App development

The development of the chatbot consists in a series of processes (Figure 5), some of which are independent from one another and were carried out simultaneously by different members. For the first part of the development, we worked at the same time on: (1) the development of a dataset structure, (2) the creation of a chatbot prototype which assesses the user's vocabulary size, and (3) the evaluation of various algorithms for the lexical complexity analysis. After having completed these parts, we started collaborating in order to have our parts converge into a single final product. The final implementation of the chatbot was deployed on the Heroku platform using the code stored on the GitHub repository. Using a `Procfile` file, containing all the commands needed to start the bot, the app is now able to start and maintain the chatbot, allowing it to be continuously running on the cloud server.

3.2.1 Data structure

This project mainly handles the following datasets: a written annotated corpus, a training corpus, the algorithm evaluation results, and the user experience evaluation. After downloading the BNC Baby corpus, we created a script to read its XML content and store it, together with some metadata, in three different CSV files (Refer to Figure 6 for the structure of each file). The purpose of this data structure is storing sentences separately for individual use, while still being able to access information about the text they belong to and information about single words within the sentence (such as POS tags) by means of their `text-index` and `word-index` value respectively. Thanks to the former, We would also be able to assemble a whole text by concatenating all the individual sentences labelled with the same `text-index` tag, in case we ever needed to recreate the original corpus, up to its integrity. This means that no information is lost in the process and, therefore, there is no need to store the original corpus as well, which saves an important amount of memory. As for the metadata, in addition to the text, sentence and word indexes, a column was added to one of the CSV frameworks to store the lexical complexity (CEFR) score of each text, obtained from the text classification task (see Section 4.1).

The training corpus for the lexical complexity classifier was composed from several sources. The texts themselves were taken from the Readability Assessment dataset [7] and divided into several parts from 7 to 13 sentences each. Hence, 2480 texts were obtained from the 332 examples presented in the source dataset. After the dataset division, all the texts were processed through the TAALES framework and 462 various linguistic features were acquired. These features are stored in CSV files where each row corresponding to one file (which content is located in the `Filename` column). All these files were then processed and all the filenames were replaced with the corresponding texts so that it would be easier to treat the data during the training process. This resulted in 5 CSV files for each level of English storing all the texts and corresponding features.

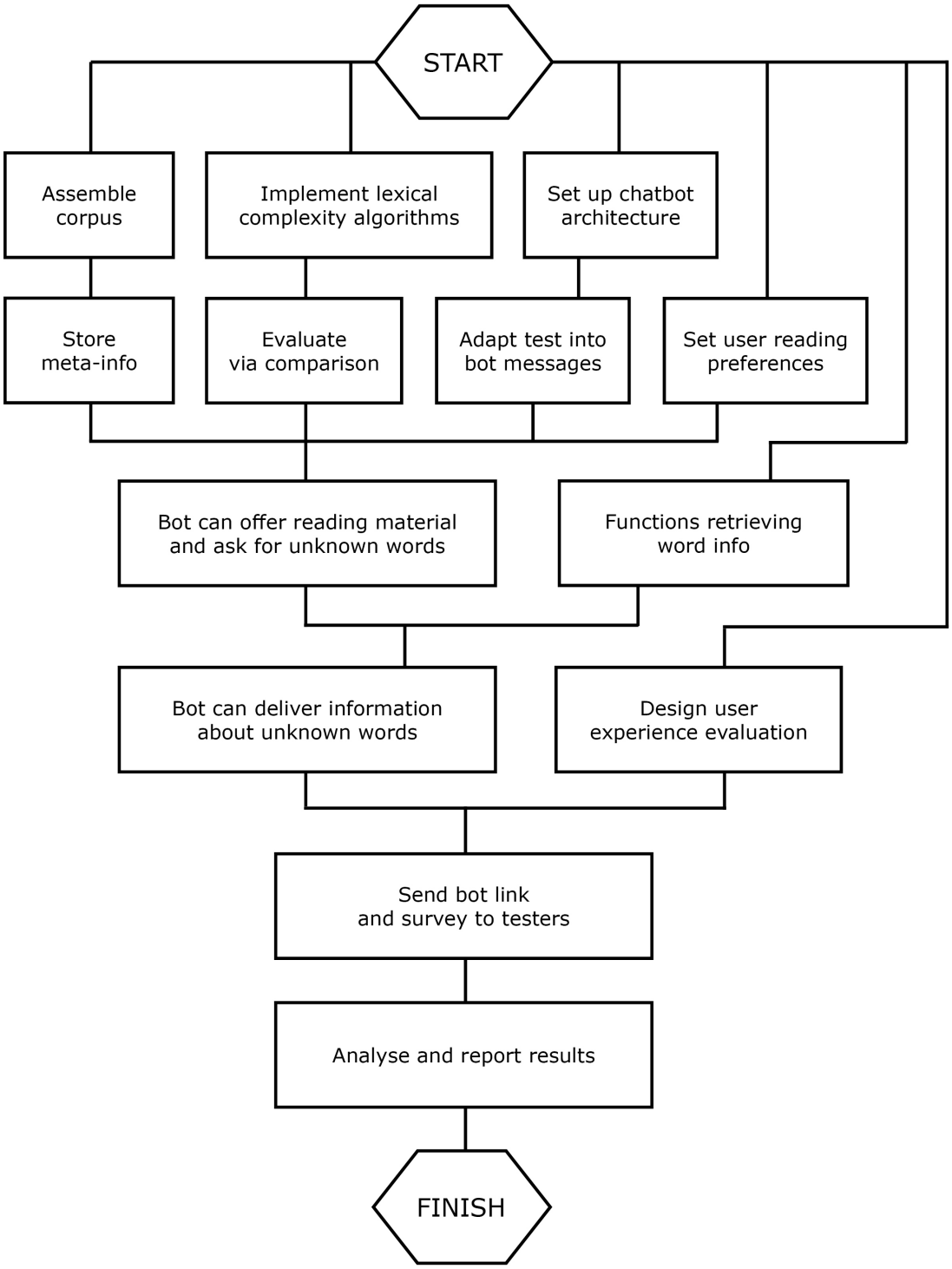

Figure 5: Development pipeline.

words.csv

word-index	word	sentence-index	text-index	POS
1	Hi	1	1	INTERJ
2	Anna	1	1	NOUN
3	I	2	1	PRON
4	told	2	1	VERB
5	you	2	1	PRON
...	...	...	...	...

sentences.csv

sentence-index	sentence	text-index
1	Hi Anna! I told you...	1
2	Today a new buildi...	1
3	Maybe you didn't k...	1
...	...	...

texts.csv

text-index	genre	complexity
1	conversation	B1
2	news	B2
3	fiction	A2
...	...	...

Figure 6: Data structure

Finally, an additional CSV table was generated from our user experience survey to store qualitative and quantitative feedback. For the user experience evaluation algoritm to treat the data, the pandas module was used to filtering only the numerical values from the CSV file.

Written corpus The corpus for this project, from which extracts are going to be selected, is a section of the BNC (British National Corpus)[6], more specifically its Baby Edition[7]. Using the BNC will alleviate the text analysis tasks in this project, given its already existing morphological annotations and its variety in genres, composing the texts of the corpus. The user will be proposed to choose from four different genres of texts: fiction, academic, conversations and news, these genres being well represented in the British National Corpus. The lexical complexity assessment of these texts is discussed in the Lexical complexity assessment subsection. Syntactic annotation will be generated with the spaCy library[8], since the corpus is not syntactically annotated[9].

Training corpus The training data comes from the Cambridge English Assessments[10]and is composed of 332 CEFR-labelled texts. This corpus is being used to test various evaluation frameworks (see Section 4.1). Unfortunately - due to the limited resources allocated for this project - for the time being, the only CEFR-labelled texts that are available for training purposes might not entirely match in terms of genre the corpus that the feedforward neural network is supposed to classify. Additionally, following the algorithm evaluation (see Section 4.1), the training corpus appears to be too small to serve its purpose. It appears that the small size of the CEFR-labelled

[6]The British National Corpus, available at http://www.natcorp.ox.ac.uk/, is a 100 million word collection of samples of written and spoken language from a wide range of sources, designed to represent a wide cross-section of British English, both spoken and written, from the late twentieth century.

[7]BNC Baby, available at https://ota.bodleian.ox.ac.uk/repository/xmlui/handle/20.500.12024/2553, is a sub-corpus of the BNC that consists of four sets of samples, each containing one million words tagged as they are in BNC itself. The words in each sample set correspond to a specific genre label. One sample set contains spoken conversation and the other three sample sets contain written text: academic writing, fiction and newspapers respectively.

[8]https://spacy.io

[9]Publicly available syntactic annotations for the BNC were not found. The only attempt to create a syntactic annotation that could be accessed was described by Lehmann et al. in 2012[3]. The authors stated that their dependency bank was tested by a small group of early adopters and claimed that a general release for a wider audience would happen in mid 2016. However, the final version of the bank appears to have not been published yet.

[10]http://www.cl.cam.ac.uk/~mx223/cedata.html

corpus of texts was the main factor contributing to the failure in producing an effective algorithm and to the diminished value of our overall application.

3.2.2 User's vocabulary size and genre preferences

For the evaluation of the user's vocabulary size, LexTale[11], a tool developed by the Radboud University, was used, being it publicly available and free to use for any purposes. The test evaluates the user's vocabulary by requiring them to confirm or deny the existence of 50 English words according to their current knowledge. After having answered all the questions, the user receives a score from 0 to 100 that is converted to the CEFR system using the LexTale vs. CERF correlation table presented by Lemhöfer and Broersma [4].

3.2.3 Chatbot

The development of the chatbot was supported by the use of the *Telegram API*[12] and the *Telegram Bot* library[13]. The bot proposes to the user different features and tasks, such as a vocabulary test, genre preference selection, reading material retrieval and word lookup. The chatbot is able to work simultaneously with several users, by storing their information separately, and since it is deployed on the Heroku[14] cloud platform, the conversation is available at any time.

3.2.4 Vocabulary test and genre preferences

The chatbot part has evolved during the application development process, so two main versions of the user-bot interactions will be described in this section. The general pipeline of the chatbot works as follows: initially, 60 questions about the existing and not existing words are sent to a user consequently. These questions are stored in a JSON file `Questions.json` as well as the correct answers. Once the questionnaire is done, the result is computed and sent to a user. Meanwhile, this score is converted in CEFR using a conversion table (Figure 7) similar to the one found in the LexTale paper[5]. This table was edited in order to split the C1 and C2 level into two different groups and assign A1 and A2 to a score below 50%, which is the score which giving random answers tends to. The CEFR level is stored in the `context` element in the variable `user_data` under the keyword `level`. After receiving the score, a user is suggested to choose one of the four genres in order to read the text of the favorite category. Once the needed genre is chosen, the function `search_text` is called with the genre as an argument. The *labeled_texts.csv* file is then opened, where all the texts are sorted by genre, and the indexes of texts matching with the genre and the level are taken from it in a python list. The `random.choice` function is applied on this list in order to have a random corresponding text. If no text matches the level and the genre, the user is asked to choose another genre. The final text index is then stored in the `user_data` variable under the `text` keyword as well as the number of the current sentence, which is used every time the algorithm goes to the next sentence.

After retrieving the text index, its content is opened by means of the *sentences.csv* file where all the sentences from all the texts are stored, so every sentence could be extracted by its in-

[11] http://www.lextale.com/index.html
[12] https://core.telegram.org/api
[13] https://python-telegram-bot.readthedocs.io/en/stable/
[14] https://heroku.com/

CEFR Level	LexTALE Score	
	LexTale Paper	Vocamprove
A1	<59	0-50
A2		
B1		51-59
B2	60-80	60-80
C1	80-100	81-90
C2		91-100

Figure 7: LexTale score to CEFR conversion table

dex and the index of a current text. This sentence is also stored so that it could be used in the `split_words` function to display a button for each word of the sentence after removing punctuation. This list, appearing as a keyboard on a screen, allows the user to choose the words they didn't understand and the bot to give them the pronunciation, the definition, the part of speech tag and the dependency.

The implementations described above concern all the versions of the code, and we will describe below what crucial changes were made for the present moment.

The first attempt to implement the chatbot was made using the `Quiz` class, a new instance of which was created every time a user started the tests by interacting with the chatbot. A poll function[15] was used to pass the questions and receive the user's answers. Each time a question received an answer, the next question was provided through a new poll. The next word to be shown was fetched from the instance and passed to the poll. The answers were passed to the `Quiz` instance to update the test result. Once all the questions were asked, the `calculate_result` method was called from the `Quiz` instance and the final result was communicated to the user. This implementation of the chatbot forced a restriction regarding simultaneous multiple users. If the bot was communicating with more than one user at once, the functionality and the test result was not reliable. After some research we decided to handle this issue by modifying the code and using the `context` element in the bot object, instead of having the `Quiz` class. The bot is able to perfectly interact with several users at the same time now. Moreover, each user is no longer required to take a test at the beginning of each interaction with the bot and the user score is stored for further text suggestion.

[15]https://python-telegram-bot.readthedocs.io/en/stable/telegram.poll.html

3.2.5 Lexical complexity

The lexical complexity evaluation could be considered the core of the project. It consists in assigning a CEFR label to each of the corpus extracts so that a suitable one can be proposed to the user depending on their vocabulary size. In order to optimise the results of this process, an evaluation of different algorithms was carried out. Refer to Section 4.1 for more information about the implementation and results of the evaluation.

3.2.6 Unknown words

Once the test is performed and the application received user's genre preferences it outputs a sentence from a text of a given genre and level. A user may choose to continue reading, if everything in the sentence is clear, or ask for explanations. If the explanation is required, the application prompts to the user a list of buttons where all the words from the sentence are shown, so that the user can choose the one(s) they are unfamiliar with and receive more information. After being asked to look up a word, the chatbot sends to the user three messages: (1) a vocal message with the pronunciation, (2) a text message with definitions, usage examples, part-of-speech tag and syntactic tag, and (3) a picture of the dependency tree of the current sentence.

The recording of the word is created using the gtts library, which generates it using Google Translate's text-to-speech API. The definitions of the word and the usage examples are provided by the wordnet corpus from the nltk library. When a unknown word is chosen, all the information available for such word is retrieved from the WordNet corpus; however, only the relevant that have the same part of speech as the required word are sent to the user. Usage examples are sorted in the same way.

Part-of-speech tags are taken from the British National Corpus where all the texts were annotated automatically using CLAWS[16] and then corrected manually. Specifically, in the Baby Edition of the BNC that was used in the current project, the simplified POS tags were provided for each word, containing only 10 tags.

Syntactic tags, according to dependency grammar, are automatically generated using the SpaCy library, which uses a standard Universal Dependency format to mark each words in the sentence with their relationships to one another. The picture that is sent to the user, presenting the dependency tree of the sentence, is also generated with SpaCy. We used the displacy module[17] for generating a picture and then saving it as an SVG file that is then converted into PNG format (this conversion was required by the fact that only some formats are compatible with Telegram). The dependency tree picture provides a deeper insight into the roles played by each word in a given sentence.

3.2.7 Tools

The main Python libraries used in this projects are the following:

- spaCy[18]: we will use them for the syntactic annotation of the corpus extracts;

[16] http://ucrel.lancs.ac.uk/claws/
[17] https://spacy.io/usage/visualizers
[18] https://spacy.io/

- telegram-bot-api[19]: this package will be used for creating a simple rule-based chatbot;
- nltk[20]: we retrieved a tool for interaction with the WordNet corpus from this library
- gtts[21]: we will use this package for generating a file with the pronunciation of an unknown word;
- pandas[22]: this library will be used for working with the data stored in the csv files;
- bs4[23] (BeautifulSoup): we will use this library for dealing with British National Corpus files that are stored in xml format.
- sklearn[24]: this library will be used for training classical machine learning algorithms in order to classify texts by their lexical complexity.
- Heroku[25]: a cloud platform to allow the app to run continuously, allowing users to interact with it from the Telegram app without running the source code themselves.

[19]https://github.com/python-telegram-bot/python-telegram-bot
[20]https://www.nltk.org/
[21]https://gtts.readthedocs.io/en/latest/
[22]https://pandas.pydata.org/
[23]https://www.crummy.com/software/BeautifulSoup/bs4/doc/
[24]https://scikit-learn.org/stable/index.html
[25]https://heroku.com/

4 Evaluation

This Section illustrates the evaluation process by describing the qualitative and quantitative assessment of both: (1) the algorithmic structure of the application, and (2) the final product from the point of view of the user.

4.1 Algorithm evaluation

In this part of the evaluation, we assessed whether our code resulted in the desired output. We performed a qualitative analysis on the main features of our app and a quantitative analysis of the algorithm behind the lexical complexity task.

4.1.1 Qualitative analysis

Following is a qualitative analysis of each script.

GUI and user interactions The user interface meets all the expected requirements. Interactions between the bot and the user flow without interruptions. The user can easily interact with the chatbot through buttons, and instructions are correctly received and answered by the bot.

Vocabulary test The test meets all the requirements. It proceeds smoothly and gives an accurate score at the end. The goal to accurately asses the user's vocabulary size is achieved. The results of each user are stored in a *user_data* element.

Dataset A dataset containing the corpus extracts to be shown to the user was created with a script that is easy to reuse using a larger corpus as data source. It successfully generates and stores metadata in three the three expected CSV files. Although the script is fully functioning, the dataset seems to be too small for its purpose, given the limited amount of corpus extracts available for some of the CEFR levels.

Word information The functions retrieving information about each word are fully operating and return the required information asked by the user. However, because of the strictness of the algorithm in showing only information relevant to the POS in question, many words, used in certain context, are treated as having no definition, leaving further information that the user could find beneficial to be withheld from them. Additionally, the image generated to illustrate word dependency is sometimes too wide, when the sentence is particularly long, resulting in the quality of the image dropping to the point of being unreadable.

Lexical complexity The algorithms implemented for the lexical complexity analysis seem to be poorly functioning. Corpus extracts seem not to match the vocabulary size of the user and a Visual representations of the categorisation algorithms clearly show their poor performance in creating clusters based on CEFR vocabulary level (Figure 8).

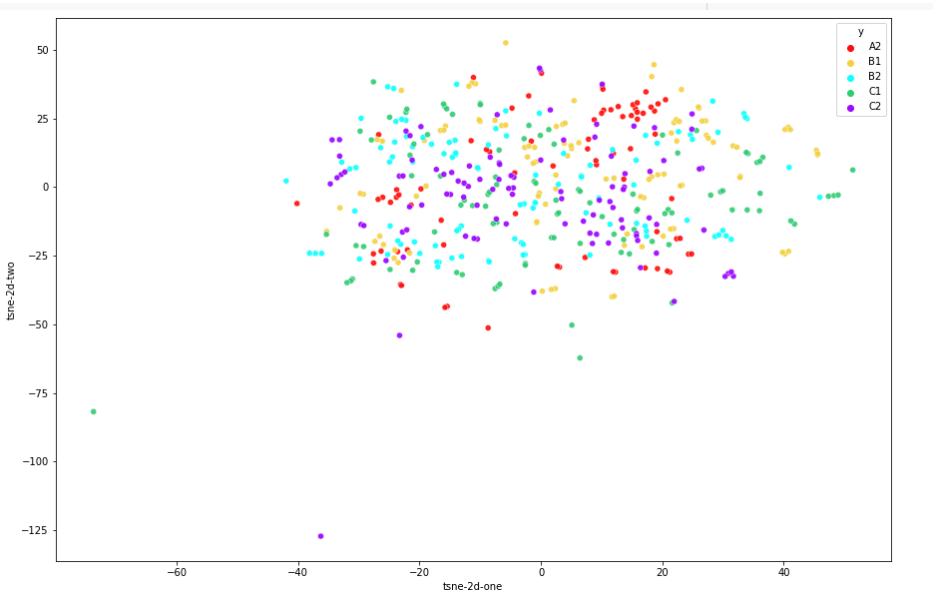

Figure 8: The most functioning algorithm still seems to perform poorly.

4.1.2 Quantitative analysis

At the time of writing, the evaluation of different algorithms for lexical complexity assessment has been completed. The lexical complexity assessment was carried out following a comparison of different tools and metrics. We considered three main methods to be compared: (1) cardinality of intersections between the corpus and CEFR word lists, (2) already existing and publicly available lexical complexity analysers, and (3) different neural network architectures trained with data retrieved from Cambridge English Assessments.

The LexTale vocabulary test administered to the user produces values that could be converted into the CEFR system by means of a table provided in the paper by the creators of the test itself. In order for the user's detected vocabulary size and the corpus complexity labels to be compatible, we decided to assess the complexity of our corpus extracts according to this system. Because of compatibility issues between the test results and the lexical complexity assessment, two of the four algorithms had to be excluded as they were not applicable to our assessment.

Cardinalities One of the lexical complexity analysers consists in the simple intersection algorithm where the level of a text is determined by means of the intersection of all unigrams and bigrams (excluding stopwords[26] like articles, auxiliary verbs etc.) of a given text with wordlists of all the levels. The results are presented in Table 1. As it can be observed, the results are not very promising, and the reason might be connected with the considerable amount of words that are not recognised as stopwords, but are nonetheless widely represented in documents from all the levels of language proficiency (e.g. "take", "know" etc.).

[26] A list of stopwords was retrieved from the sklearn library

	precision	recall	f1-score	support
A2	0.183	0.944	0.306	18
B1	0.333	0.053	0.091	19
B2	0.0	0.0	0.0	17
C1	0.0	0.0	0.0	20
C2	0.0	0.0	0.0	22
accuracy			0.188	96
macro avg	0.103	0.199	0.079	96
weighted avg	0.1	0.188	0.075	96

Table 1: Classification report for lexical intersection

Existing frameworks The next attempt to evaluate our texts consisted in using two publicly available Lexical complexity analysers: (1) the TAALES[27] (Tool for the Automatic Analysis of Lexical Sophistication), which uses indices related to word frequency, word range, n-gram frequency, n-gram range, n-gram strength of association, contextual distinctiveness, word recognition norms, semantic network, and word neighbors and (2) the Lexical Complexity Analyser by Xiaofei Lu[6], which is an implementation of the system described by Lu in *The Modern Language Journal (2012)*, using 25 different measures of lexical density, variation and sophistication. After the evaluation of our texts with these frameworks we understood that almost all measures that they propose are meaningless for our purposes as they do not give any explicit indications of the text level (almost all the measures are connected with different kinds of frequencies, word counts and n-gram counts). However, these features were not discarded, but used for the machine learning classification described below.

The last attempt in implementing lexical complexity evaluation algorithm was training several neural networks in order to classify our texts. For this purpose, we found a readability dataset presented by [7] in 2019 that contains 332 texts of 5 different levels (the details about the dataset size are presented in the table 2. As it can be seen from the table, the dataset is composed of only few examples, so each text was divided into several sections containing from 7 to 13 sentences, for each section to be considered as a single example.

Level	Number of texts	Number of texts after the division
A1	332	1240
A2	64	76
B1	60	113
B2	71	331
C1	68	372
C2	69	348

Table 2: Readability dataset by [7]

Neural networks The first neural network implemented was a feed forward neural network with text vectorisation using BERT sentence representations, but the insufficient number of ex-

[27] https://link.springer.com/article/10.3758/s13428-017-0924-4

amples used for training caused the results to be close to 0.0 for all classes.

Then we tried to use classical machine learning algorithms such as Support Vector Machine, Logistic regression, Random Forest, Decision Tree and AdaBoost classifiers. A Multilayer perceptron with 100 hidden units and 1 hidden layer was also implemented. The ratio used to divide the dataset into train and test parts was 4:1. Tf–idf and bag-of-words were used for text vectorisation for these algorithms. The algorithms were complemented by some additional features, such as: (1) all features from TAALES framework described earlier, and (2) dependency annotation using treelets method[2] and the SpaCy syntactic dependency parser.

The complete results of these experiments will be presented in the Appendix of the final report, while the results from the best algorithm are presented in Table 3.

	precision	recall	f1-score
A2	1.0 (SVC on Dep)	0.944 (MLP on BOW)	0.919 (MLP on BOW/TFIDF)
B1	0.786 (LR on BOW/BOW+Dep)	0.789 (MLP on BOW)	0.75 (MLP on BOW)
B2	0.524 (MLP on BOW+Dep)	1.0 (MLP on Features)	0.579 (MLP on BOW+Dep)
C1	0.611 (MLP on TFIDF+Dep)	0.9 (MLP on TFIDF+Features)	0.579 (MLP on TFIDF+Dep)
C2	0.571 (SVC on BOW/BOW+Dep)	0.62 (MLP on BOW+Features)	0.49 (LR on TFIDF+Dep)
accuracy			0.625 (MLP on BOW)
macro avg	0.62 (MLP on BOW)	0.638 (MLP on BOW)	0.625 (MLP on BOW)
weighted avg	0.614 (MLP on BOW)	0.625 (MLP on BOW)	0.615 (MLP on BOW)

Table 3: Best results obtained during the training

As it can be seen, the best results are obtained with only textual data vectorised with bag-of-words method, but we assumed that usage of only the most important features from each of the type of our data could improve the quality of our model. These features and their importance are presented in the Tables 4, 5, 6, 7. The results of the MLP Classifier (we chose it since it gave the best result in the experiments previously described) trained only using the most important features of each type are presented in Tables 8, 9, 10. It can be noticed that the results are comparable to the ones achieved on all the features, but don't improve them, so our assumption was erroneous, and usage of MLP Classifier with bag-of-words method of word vectorisation is the best solution in our case.

Feature	Score
Mean age of acquisition score on all words	0.013
Mean age of acquisition score on concrete words	0.013
LDA Age of Exposure (.40 cosine threshold)	0.011
Academic Word List All	0.01
Lexical Decision Time (z-score)	0.009
Lexical Decision Time (z-score) on content words	0.009
LDA Age of Exposure (inverse slope)	0.009
Lexical Decision Time on content words	0.009
Free Association Stimuli Elicited on content words	0.009
LDA Age of Exposure (inverse average)	0.009

Table 4: top 10 most important features for Random Forest Classifier

Word	Score
of	0.007
the	0.007
is	0.006
as	0.006
you	0.005
my	0.005
because	0.005
to	0.005
in	0.004
can	0.004

Table 5: top 10 most important words for Random Forest Classifier with stopwords

Word	Score
pair	0.007
truly	0.007
kendra	0.006
assume	0.006
nutrition	0.005
bend	0.005
unfortunate	0.005
intellectuals	0.004
carted	0.004
wondered	0.004

Table 6: top 10 most important words for Random Forest Classifier without stopwords

Treelet	Score
NOUN - prep - ADP - pobj - NOUN	0.022
ADP - pobj - NOUN - prep - ADP	0.02
VERB - prep - ADP - pobj - NOUN	0.016
VERB - dobj - NOUN - prep - ADP	0.012
NOUN - prep - ADP - pobj - PROPN	0.011
AUX - prep - ADP - pobj - NOUN	0.011
VERB - conj - VERB - prep - ADP	0.011
AUX - attr - NOUN - prep - ADP	0.01
VERB - xcomp - VERB - dobj - NOUN	0.01
VERB - xcomp - VERB - prep - ADP	0.009

Table 7: top 10 most important treelets for Random Forest Classifier

	precision	recall	f1-score	support
A2	0.765	0.722	0.743	18
B1	0.526	0.526	0.526	19
B2	0.35	0.412	0.378	17
C1	0.32	0.4	0.356	20
C2	0.467	0.318	0.378	22
accuracy			0.469	96
macro avg	0.486	0.476	0.476	96
weighted avg	0.483	0.469	0.471	96

Table 8: Classification report for MLP classifier on the 100 most important words

	precision	recall	f1-score	support
A2	0.714	0.556	0.625	18
B1	0.36	0.474	0.409	19
B2	0.412	0.412	0.412	17
C1	0.333	0.3	0.316	20
C2	0.364	0.364	0.364	22
accuracy			0.417	96
macro avg	0.437	0.421	0.425	96
weighted avg	0.431	0.417	0.42	96

Table 9: Classification report for MLP classifier on the 100 most important treelets

	precision	recall	f1-score	support
A2	0.375	0.5	0.429	18
B1	0.0	0.0	0.0	19
B2	0.0	0.0	0.0	17
C1	0.0	0.0	0.0	20
C2	0.235	0.727	0.356	22
accuracy			0.26	96
macro avg	0.122	0.245	0.157	96
weighted avg	0.124	0.26	0.162	96

Table 10: Classification report for MLP classifier on the 100 most important features

	precision	recall	f1-score	support
A2	0.0	0.0	0.0	18
B1	0.222	0.947	0.36	19
B2	0.25	0.118	0.16	17
C1	0.429	0.15	0.222	20
C2	0.0	0.0	0.0	22
accuracy			0.24	96
macro avg	0.18	0.243	0.148	96
weighted avg	0.178	0.24	0.146	96

Table 11: Classification report for MLP classifier on the 100 most important words, dependencies, features

Unfortunately, the results seem to be far from promising, and are likely to be linked to the limited size of the training dataset. Because of the small sample used, some common word that are frequently used across different proficiency levels are poorly represented in the corpus. This means that the tf-idf does not detect them as tokens eligible to be assigned a low weight, making them to contribute excessively to lowering the overall lexical complexity level of complex extracts.

4.2 User experience evaluation

For the evaluation of the user experience, a survey[28] was filled in by 16 testers. The survey consisted in two parts: one dedicated to the quantitative evaluation and the other to a qualitative assessment of the system overall.

4.2.1 Quantitative analysis

The quantitative section of the evaluation survey was be based on the model proposed by Escudeiro[1] for the evaluation of educational applications on mobile platforms. The quality of the app was calculated according to the Quantitative Evaluation Framework, specifically designed to assess educational software quality for digital learning, which evaluates a system in a three-dimensional space, where each dimension represents performance from one particular aspect, based on the distance from an ideal system represented by the coordinates (1,1,1).

Each dimension, namely functionality, efficiency and adaptability[29], is comprehensive of various factors contributing to it, as shown in Figure 12.

Following are the formulas we used to calculate the performance score associated with each dimension(Dim_j) and that of each factor, where p is the weight associated to the score by which it is multiplied, q_m is the average of the scores that question m has obtained in the survey, and n is the number of factors.

$$Dim_i = \sum_n (p_n \times factor_n), \sum_n (p_n) = 1, p_n \in [0, 1]$$

$$factor_n = \frac{\sum_m (p_m \times q_m)}{\sum_m p_m}$$

The scores obtained by each dimension can be visualised as coordinates in a space of i-dimensions, in this case a three-dimensional space. Once we obtained the coordinates, we calculated the euclidean distance D from the ideal set of coordinates (1,1,1).

$$D = \sqrt{\sum \left(1 - \frac{Dim_j}{100}\right)^2}$$

Finally, we calculated the quality of the whole system as a percentage.

[28]The survey was delivered through Google Forms and the results were exported as a CSV file and are now fully available on the GitHub project repository.

[29]For more information on what each of these dimensions represent, please refer to the original paper [1].

Dimensions	Factors
Functionality	• Ease of use • Content quality
Efficiency	• Versatility • Pedagogical aspect • Didactic structure • Stimulates initiative and self-learning • Cognitive effort
Adaptability	• Audiovisual quality • Technical elements • Navigation and interaction • Originality and use of advanced technology

Table 12: Structure of the factors composing each dimension.

$$q = \left(1 - \frac{D}{\sqrt{n}}\right)$$

The results of the survey, completed by a total of 30 users, show a 90.47% overall system quality, with *adaptability* being the best performing dimension.

4.2.2 Qualitative analysis

The final part of the survey prompted the testers to write their general opinion about the chatbot's current performance and their desired future improvements (refer to the Appendix, for the full survey and to the GitHub repository for the full unprocessed results). The main advantages of our application according to users' opinions are the following:

- Taking into account user's genre preferences;
- Possibility to choose unknown words for explanation;
- Word pronunciation being sent as a voice message;
- Easy-to-follow user interface;
- Chatbot high answering speed;
- Originality of the idea;

- Suggested texts are particularly gripping;
- Easy access through the Telegram application, which does not require any additional installations;
- Everything is automated;
- Possibility to evaluate English level;
- Game-like vocabulary size test;
- Concept of learning through messages.

Users then identified areas needing improvement proposed the following features to be added or improved:

- Front-end
  - Adding explanations about the */explanations* and */continue* buttons or making their names more descriptive;
  - Adding an explicit conversion of the numerical test results into CEFR standard;
  - Considering changing *conversations* corpus excerpts as they are not always appropriate for learning;
  - Adding */quit* button to the reading text interface;
  - Removing stopwords from the buttons listing words to be defined;
  - Ensuring that all the indexes in the questions are correct;
  - Considering adding a special section for the references in the academic texts so that only the interested users could read it;
  - Upgrading the feedback form by means of adding some questions considering personal feelings about the interaction with the application and adding some information about the team who created this project;
  - Adding a feedback on the second stage of the game as it is not clear whether there is an ending of the sentences;
  - Adding the possibility to change a text inside genre if the suggested one is not suitable;
  - Simplifying the language used by the chatbot itself as the explanations of the process might be poorly understood by beginners in English;
  - Adding an explicit explanation that all the suggested texts are based on the English level of a test taker;
  - Adding a main menu for changing the configurations (current text, current genre etc.);
  - Offering a chance to retake the test a second time;
- Back-end
  - Expanding the number of the texts and the genres suggested;

- Expanding definition corpus;
- Adding the disambiguation of the definitions so that it would be easy to understand which one is suitable for a particular text;
- Adding an audio version of the whole sentence or of the text;
- Adding a third stage where the progress is checked and the memorisation success is evaluated;
- Adding sign language to the word definition phase.

Most of the feedback provides valuable insights about the future development of the chatbot. In our conclusion we will explore our usage of this feedback in order to improve the bot.

5 Conclusion

The quantitative and part of the qualitative assessments have been carried out. The quantitative evaluation of various algorithms identified the best algorithm to carry out the main NLP task of our project: MLP Classificator with Bag-of-Words method resulted being the best solution for our Lexical Complexity assessment. The quantitative evaluation of the user experience allowed us to check whether the performance of the system was overall satisfying; it provided promising results, showing that our application has potential for further development and success. The qualitative analyses illustrated the ways our application could be improved.

Although no problem arose in following the methodology and all the processes composing our projects were being carried out smoothly while respecting the deadlines, the quantitative results of the algorithm evaluation, were particularly scarce. A reason for the scarce performance of the algorithms was briefly explained in Section 4.1. However, as we stated since the beginning of this project, our aim has always been to create a product that focuses on a positive user experience, and the relative evaluation seems to have provided outstanding results with a considerably high quality score of the overall system.

To sum up, learning English through the Telegram chatbot seems to the testers original, attractive and engaging, therefore, our application may be regarded as a convenient tool for learning English online that provides most of the necessary features needed for reading a text in a foreign language with the purpose of improving the level of proficiency. In addition, users appreciated not only the chatbot itself but also the fact that it was deployed on Telegram, so they do not need to download a new application, to install another tool on their smartphone or to interrupt abruptly their daily communication-related activities for the sake of enhancing their English abilities.

5.1 Plans for improvement

According to the qualitative feedback, we found areas of improvement on which we already started acting. We already made the algorithm providing definitions less strict in order to minimise the "definition not found" instances and We are planning to expand the vocabulary in order to provide more definitions, Next, we would like to allow the user to be more in control of their learning activities: we would like them to be able to switch text, genre, level, etc. freely. In order to check their progress we might also consider allowing the bot to be compatible with apps making use of innovative learning methods, such as spaced-repetition (e.g. AnkiDroid or Cram).

5.2 Module-specific summaries

The purpose of this subsection is to describe some relevant aspects of the project from the point of view of *Methods for NLP* and *Written Corpora*, the two modules for which this project has been developed.

5.2.1 Methods for NLP

The requirements expressed for this module were to build an app with at least 3 NLP tasks and an evaluation strategy.

NLP tasks For this project, we worked on three NLP tasks: syntactic tagging, lexical complexity evaluation, and text-to-speech generation. We started implementing four different algorithms for lexical complexity evaluation; see section 4.1 for more information. We used spaCy for syntactic tagging and gtts for text-to-speech generation. In addition, morphological tagging was included in the BNC corpus we used.

Evaluation The evaluation consists in: (1) the comparison of different algorithms to determine the most suitable and best performing to complete our lexical complexity analysis, and (2) the user experience evaluation. The lexical complexity assessment was carried out following a comparison of different tools and metrics. We considered three main methods to be compared: (1) cardinality of intersections between the corpus and CEFR word lists, (2) already existing and publicly available lexical complexity analysers, and (3) different neural network architectures trained with data retrieved from Cambridge English Assessments. A qualitative evaluation was also carried out and reported in Sections 4.1 and 4.2 as a description of the functioning and malfunctioning components of the features that have been worked on so far. The user experience evaluation was carried out by analysing quantitative and qualitative data from a survey and calculating an overall quality score according to the Quantitative Evaluation Frameword for educational games on mobile platforms.

5.2.2 Written Corpora

A small sample from the British National Corpus with POS annotation was downloaded[30] and processed via a script selecting relevant information from the xml file and storing it into three CSV files (one for individual words, one for sentences and one for texts found in the corpus). With the current data structure, we enable the usage of individual sentences, while still being able to access information about the single words within the sentence (such as POS tags) by means of their `word-index` value. We are also able to assemble a whole text by concatenating the single sentences labelled with the same `text-index` tag. Refer to subsection for more information on data handling.

he training corpus for the lexical complexity classifier was composed from several sources. The texts themselves were taken from the Readability Assessment dataset [7] and divided into several parts from 7 to 13 sentences each. Hence, 2480 texts were obtained from the 332 examples presented in the source dataset. After the dataset division, all the texts were processed through the TAALES framework and 462 various linguistic features were acquired. These features are stored in CSV files where each row corresponding to one file (which content is located in the `Filename` column). All these files were then processed and all the filenames were replaced with the corresponding texts so that it would be easier to treat the data during the training process. This resulted in 5 CSV files for each level of English storing all the texts and corresponding features.

[30] https://ota.bodleian.ox.ac.uk/repository/xmlui/handle/20.500.12024/2553

References

[1] Paula Escudeiro and Nuno Escudeiro. "Evaluating Educational Games in Mobile Platforms". In: *Int. J. Mob. Learn. Organ.* 7.1 (Jan. 2013), pp. 14–28. ISSN: 1746-725X. DOI: 10.1504/IJMLO.2013.051571. URL: https://doi.org/10.1504/IJMLO.2013.051571.

[2] Anders Johannsen, Dirk Hovy, and Anders Søgaard. "Cross-lingual syntactic variation over age and gender". In: *Proceedings of the nineteenth conference on computational natural language learning*. 2015, pp. 103–112.

[3] Gerold Lehmann Hans Martin & Schneider. "BNC Dependency Bank 1.0". In: *Oksefjell, S., Ebeling, J. & Hasselgard, H. (Eds.), Aspects of corpus linguistics: compilation, annotation, analysis. Helsinki: Research Unit for Variation, Contacts, and Change in English* (2012).

[4] Kristin Lemhöfer and Mirjam Broersma. "Introducing LexTALE: A quick and valid Lexical Test for Advanced Learners of English". In: *Behavior research methods* 44 (June 2012), pp. 325–343. DOI: 10.3758/s13428-011-0146-0.

[5] Kristin Lemhöfer and Mirjam Broersma. "Introducing LexTALE: A quick and valid Lexical Test for Advanced Learners of English". In: *Behavior Research Methods* 44.2 (June 2012), pp. 325–343. ISSN: 1554-3528. DOI: 10.3758/s13428-011-0146-0. URL: https://doi.org/10.3758/s13428-011-0146-0.

[6] XIAOFEI LU. "The Relationship of Lexical Richness to the Quality of ESL Learners' Oral Narratives". In: *The Modern Language Journal* 96.2 (2012), pp. 190–208. DOI: 10.1111/j.1540-4781.2011.01232_1.x. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/j.1540-4781.2011.01232_1.x. URL: https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1540-4781.2011.01232_1.x.

[7] Menglin Xia, Ekaterina Kochmar, and Ted Briscoe. "Text Readability Assessment for Second Language Learners". In: *CoRR* abs/1906.07580 (2019). arXiv: 1906.07580. URL: http://arxiv.org/abs/1906.07580.