INSTITUT FÜR INFORMATIK
der Ludwig-Maximilians-Universität München

COMO: A LEXICOGRAPHICAL DATA STRUCTURING GAME WITH A PURPOSE

Maximilian Kristen

Bachelorarbeit

Aufgabensteller	Prof. Dr. François Bry
Betreuer	Prof. Dr. François Bry
Abgabe am	23.12.2020

Erklärung

Hiermit versichere ich, dass ich die vorliegende Arbeit selbständig verfasst habe und keine anderen als die angegebenen Hilfsmittel verwendet habe.

München, den 23.12.2020 Maximilian Kristen

Abstract

Access to lexicographical data is an important asset for modern linguistics and related practical applications. Wikidata, an open and collaborative Wikimedia project that hosts structured data, has recently started to integrate lexemes into its data model.

While the project has managed to accumulate a substantial amount of lexemes in a short time, it still struggles, for legal and other reasons, to collect a comparable amount of senses describing the already collected lexemes.

Como is an Android app which tries to gamify the creation of those lexeme senses, to attract new contributions. The application also checks the quality of the created senses by letting other players guess a lexeme from its sense.

An analysis based on a month-long testing points to both, the simplicity and the efficiency of Como as a mean to collaboratively collect lexemes' senses.

Zusammenfassung

Zugang zu lexikografischen Daten ist ein wichtiger Stützpfeiler der modernen Linguistik und im Zusammenhang dazu stehenden praktischen Applikationen. Wikidata, ein offenens und kollaboratives Wikimedia Projekt, das strukturierte Daten bereitstellt, hat vor Kurzem Lexeme in sein Datenmodel integriert.

Während das Projekt in der Lage war schnell größere Mengen von Lexemen zu akkumulieren, gab es Schwierigkeiten, legaler und anderer Natur, zu den Lexemen eine vergleichbare Menge an Wortsinnen zu sammeln.

Como ist eine Android App, die versucht die Erstellung von solchen Wortsinnen zu gamifizieren, um so neue Beiträge zu animieren. Die Applikation prüft die Qualität der produzierten Wortsinne, indem sie die anderen SpielerInnen ein Lexem anhand seines Wortsinns erraten lässt.

Eine Analyse einer einmonatigen Testperiode zeigt die Einfachheit und Effizienz von Como als eine Methode, die Wortsinne von Lexemen kollaborativ zu sammeln.

Acknowledgments

Thanks to Prof. Bry for his calm guidance, to all the testers of Como for their willingness to keep using the app, even with the initial beta issues and resets and to the Wikidata community for their amazing effort to advance free and open structured knowledge.

Contents

CHAPTER 1

Introduction

The possibilities of structured data to benefit both research and the public are the reason for the rising number of projects that target the generation or usage of said data. While institutions will often just convert their existing data sets into more structured ones, crowdsourced and open projects need volunteer help to accomplish this. Many projects have therefore developed a need for new approaches to volunteer engagement in creating structured data.

This work will focus specifically on the creation and evaluation of lexicographical structured data, by developing a "Game with a Purpose" (GWAP) prototype to demonstrate a possible method to engage a wider volunteer community.

To this end, this chapter will include a short overview of the history of Wikidata and its development, followed by a short explanation of "lexemes" and "senses" as they were included in Wikidata in 2017. Afterwards it will explain the problems with the Wiktionary project, that led to this development, and show details of the implemented lexeme data model. Then it will explore the noticeable discrepancies between the lexemes with and without senses in the existing data and how a GWAP could be useful to solve them.

1.1 Wikidata as a Hub of Structured Knowledge

Wikidata, the first new Wikimedia Foundation project since 2006 [36], was originally started in 2012 and initially deployed the first time in 2013. At the beginning its functionality was limited to link between different language versions of Wikipedia sites on the same topic, but it quickly developed into a more flexible, general purpose data modelling tool [17].

After its initial deployment Wikidata quickly became integral to other Wikimedia Foundation projects. As an example, over 70% of Wikipedia articles are using data from Wikidata in one form or another (excluding the aforementioned sitelinks) [34]. It also developed into a topic of interest for researchers, with a growing number of papers published every year since 2012, excluding the year 2020 [28]. Furthermore in 2014 Google decided to end its own structured knowledge project "freebase", and made efforts to migrate its contents to Wikidata [24].

In an effort to expand its data model, Wikimedia started developing lexicographical data structures as a new data type in 2017.

1.2 Lexemes and Senses

As part of the terminology of the following work, the terms "lexeme" and "sense" should be established, as they relate to their usage in Wikidata entities.

"Lexemes" are described by Wikipedia as "a unit of lexical meaning that underlies a set of words that are related through inflection.... For example, in English, run, runs, ran and running are forms of the same lexeme, which can be represented as RUN."[35]. In the user-facing parts of the app, the term "lexeme" is often replaced by the less scientific "word", to increase the apps ease-of-use.

"Senses" on the other hand are used by Wikidata to mark different meanings of a given lexeme. As an example, the lexeme "king" has three different senses in Wikidata.[32]

- piece from the board game chess
- class of male monarch
- playing card

In the application and this work "senses" will sometimes be replaced with "descriptions" or "definitions" depending on the context.

1.3 Improving Wiktionary with Structured Data

The English language Wiktionary strives to be "a collaborative project to produce a free-content multilingual dictionary. It aims to describe all words of all languages using definitions and descriptions in English." [10].

This goal is similar for the other language Wiktionaries, which already shows the monumental amounts of content each Wiktionary wants to create and maintain. Because of this most of the volunteers work would be spend on duplication of entries already done in other Wiktionaries. Additionally, most of the 176 different language Wiktionaries [33] are struggling to attract or keep active members. [25]

From a user perspective, there are multiple issues with the presentation of data in the current Wiktionary projects. They all use Mediawiki, the same software used to display Wikipedia pages, to present their lexeme data. This only allows lexicographical data entry as unstructured plain text, or loosely structured templates. As a result it is difficult to extract structural information about a Wiktionary project's data as it requires complex text processing to acquire structured information from the standard MediaWiki APIs. [25]

Considering these issues, a centralised multilingual data repository would be a worthwhile addition to all Wiktionary projects. On top of easing the work for Wiktionaryans, it would also be useful on its own merit, by providing a standardised Syntax (SPARQL) and Interface (Wikidata Query Service) for linguists and other researchers. These were the reasons for the aforementioned integration of lexemes in Wikidata.

1.4 Data Model for Lexemes

The realisation of the new data model for lexicographical data is handled by introducing a new extension to Wikidata's hosting software, Wikibase. [7]

The model, as shown in Figure 1.1, contains three distinct parts, all linked to the L(exeme)-id of the lexeme. L-ids start with a "L" in comparison to other items in Wikidata, which are identified with a "Q" number. A example of the model can be found in this lexeme for a noun in English: "book" with the L-id L536.

The first part of a lexeme includes:

- **Lemma**: the actual lemma, in this case: "book"
- **Language**: language of the lexeme, as a reference to a Q-numbered Wikidata item, here: Q1860 (English)
- **Lexical category**: lexical category of the lexeme, as a reference to a Q-numbered Wikidata item, here: Q1084 (Noun)
- **Lexeme Statements**: statements that relate to information not specific to the forms and senses, here: a reference to a homograph lexeme, the verb "to book"

This example also contains a number of forms, which are structured in three elements. When looking at the first form L536-F1, these elements are:

- **Representation**: the actual written form, here: "book"
- **Grammatical features**: these describe different attributes of the form, here it is "singular"
- **Statements**: state aspects of the form, in this example an IPA transcription of the form

The model also includes senses with two elements. In the example there is only one sense L536-S1 with these contents:

- **Gloss**: the actual description as a multilingual string, here the German version is: "fest gebundenes Druckwerk"
- **Statements**: these statements often range widely from possible synonyms to images. The example has many statements, including an Italian translation: "libro"

1.5 Status of Lexemes in Wikidata

Despite the introduction of the Lexeme datamodel in 2017 and the incremental deployment in 2018 and 2019 there is not yet a sufficient set of lexemes for any language in Wikidata. The most "complete" language in the dataset is Russian, although this is not because of an very active Russian Wikidata community effort, but because some of its content has been imported from the Russian Wiktionary with a bot [23].

Obviously this would be a promising strategy to increase the amount of lexemes, senses and forms in Wikidata. Wictionaries contain a vast pool of dictionary information, that could be imported to Wikidata, but this process is hindered by the fact, that they have differing licencing models.

Wikidata uses CC0 [8], which is mostly equivalent to the public domain while Wiktionary uses CC-BY-SA [10], a free culture license, but more restrictive. These licenses do not allow an automatic import of all Wiktionary content to Wikidata, except for factual statements, which can't be copyrighted in most legal systems.

This exception does not extend to all data points of the Wiktionary dataset, especially the descriptions of many lexemes, because they meet the threshold of a creative work [12].

While some languages, like the aforementioned Russian, are starting to populate Wikidata with lexemes, those lexemes often are lacking senses. In the top ten languages with the most lexemes, only Basque has at least one sense for over 50% of its lexemes. As demonstrated in Figure 1.2 there is a drastic difference between the amount of bare lexemes and the ones with one or more senses, which reduces the value of the data set. Every published dictionary and every major Wiktionary project contains at least one description of a lexeme, while Wikidata can't offer one in most cases.

1.6 Games with a Purpose

The goal of "Games with a purpose" (GWAPs) is to motivate human players to solve problems, while still being enjoyable. Most GWAPs rely on distributing computationally impossible or hard to solve tasks to multiple human players. These users should then solve the tasks in a playful manner. [31]

This technique is mostly applied for tasks, which humans (still) can solve more efficiently than software, like image recognition or comprehension of language. A part of those games is often dedicated to cross-checking results of other players to prevent errors.

As Wikidatas lexicographical data is incomplete and often lacks senses it would be useful to create a GWAP, that lets players fill in the missing senses while playing. As creating new sentences and guessing words are highly creative task and sharing new creations with others is social, this could potentially be the basis of an entertaining and effective GWAP.

Como, the application, which was developed in tandem with this work, is an effort to meet these characteristics, and has proven some effectiveness in creating new and useful senses for Wikidata.

Figure 1.1: Wikidata's data model for Lexemes [16]

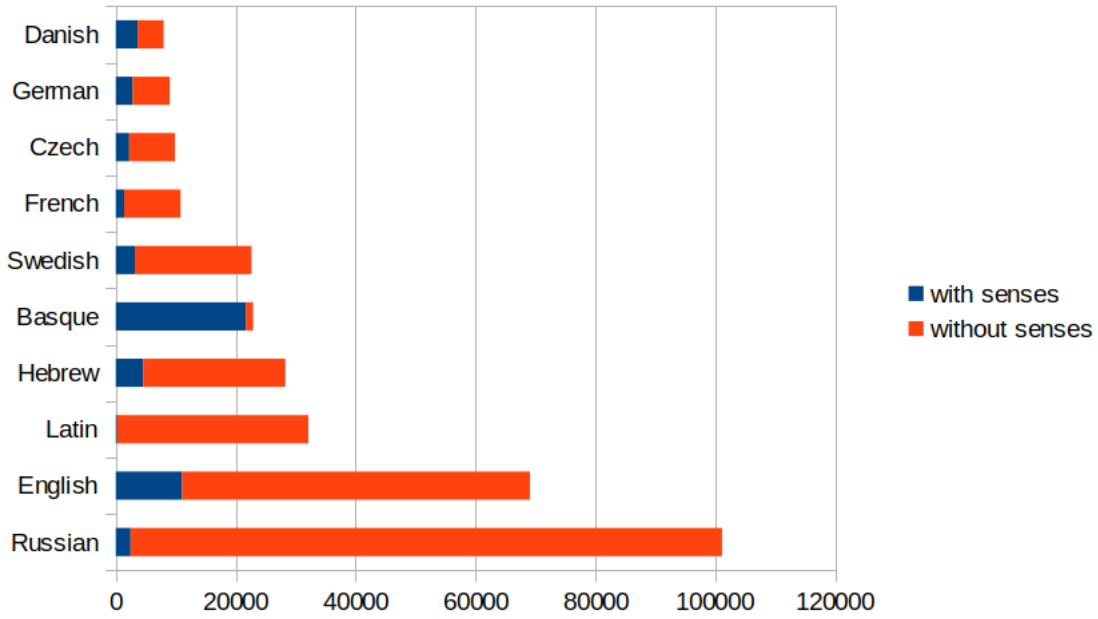

Figure 1.2: Number of lexemes with and without senses in the ten most populated languages on Wikidata [20]

CHAPTER 2

Related Work

The volunteer community around the Wikidata project has already created a number of GWAPs to encourage more contributions. As they are created based on the interests and skills of the volunteers they vary drastically in both design language and purpose. There is also some overlap between projects that would be better described as tools, gamified tools or real GWAPs. In the following, two examples of such Wikimedia related GWAPs will be presented and evaluated against the properties in the scope of Como.

2.1 The Distributed Game

"The Distributed Game" [18] is a web-platform developed by Magnus Manske [19] that features GWAPs related to editing Wikidata, as seen in Figure 2.1. It enables the creation of new games with a simple API, which allows less experienced community members to participate in the game creation process. The application can be considered highly successful, as the platform as a whole has recorded more than 1.3 million user interactions with its various games. [3]

The user experience changes with each different game, but some properties apply throughout the site. All games present the players with computer-generated suggestions that often aggregate data from multiple sources to create an additional statement for Wikidata. As can be observed in the example "Alias" (see Figure 2.2) most games require interaction with up to three buttons:

- **Confirm/Yes/True**: to confirm a matching suggestion
- **Deny/No/False**: to mark a suggestion as wrong
- **Skip**: to skip to the next entry

"Alias" presents the lemma of a Wikipedia page, here "Pat Skerry", and the first few lines of the article. Then it poses the question: 'Is "Patrick Joseph Skerry" a valid alias for the original lemma?' The user then has to choose between the already described three options.

Depending on the decision the alias will be added to the related Wikidata object, to make it searchable under both names. Therefore the task of "Alias" is to add more correct aliases to Wikidata objects. It generates suggestions mainly based on the first line of

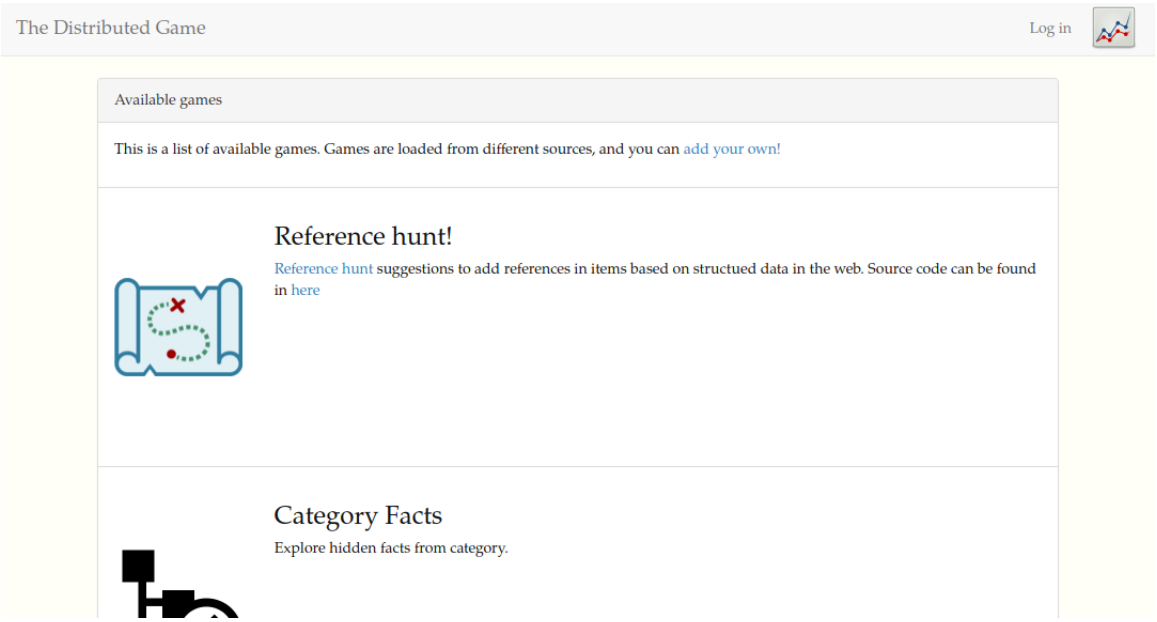

Figure 2.1: "The Distributed Game" homepage, listing some playable games

Wikipedia articles or redirect links to the same page. One possible effect of this game is to increase the searchability of married women, which often (traditionally) have at least two valid names in some European cultures.

"The Distributed Game" is manly oriented at people familiar with Wikidata's basics and is only playable in browsers. Users are offered a very reduced action set to complete games, that can be executed quickly. There are no classic gamification attributes like rewards or different tasks, most of the satisfaction is intrinsic to improving data statements with greater speed then manual entry.

2.2 Macht Sinn

"Macht Sinn" [22] (English translation: "makes sense") was created by Michael Schönitzer and is similar to "The Distributed Game" as it also suggests a possible edit, that then can be allowed, denied or skipped by the user (see Figure 2.3).

It diverges from the type of suggested edits, as it provides lexemes and relating senses, which is currently not possible with the other popular platform. Despite being created much later, it has already had nearly 30,000 user interactions [21] and is therefore a big contributor to the growing Wikidata lexicographical project.

The application has overlapping goals with the Como GWAP, as it also tries to add additional senses to lexemes. The main divergence is in the creation of those senses. "Macht Sinn!" pulls sense suggestions from the description of linked, or similar sounding, Wikidata items, whereby Como aims to let users create senses from scratch. This is of course a more demanding task than judging a previously generated suggestion, and therefore will slow down the speed of creation for new senses accordingly.

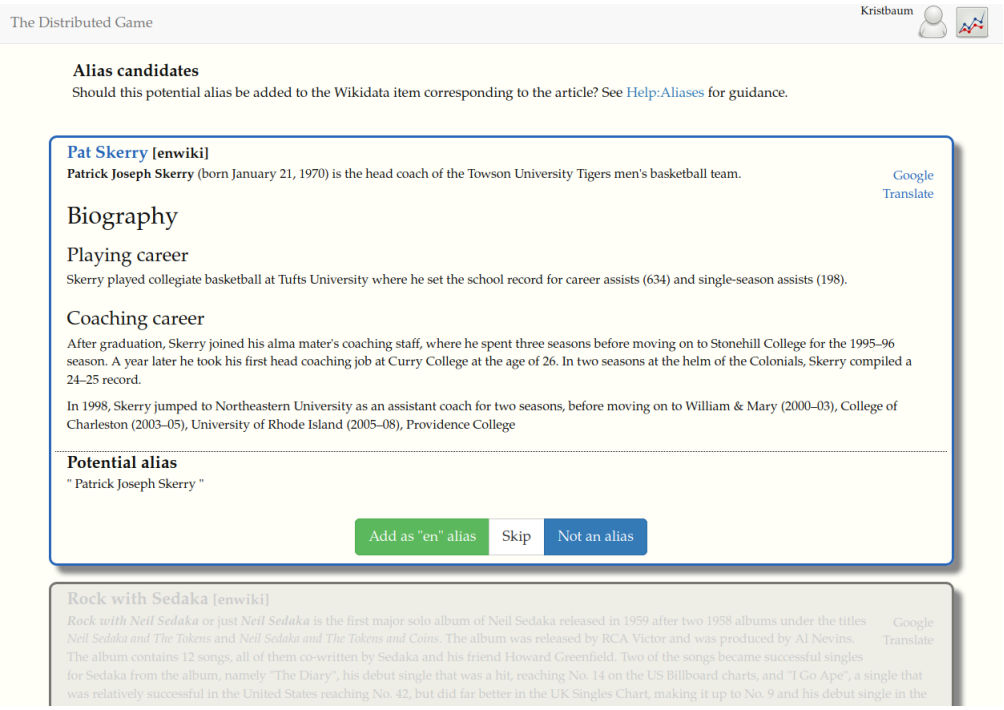

Figure 2.2: "Alias" is a simple matching tool, to add additional aliases to Wikidata entities

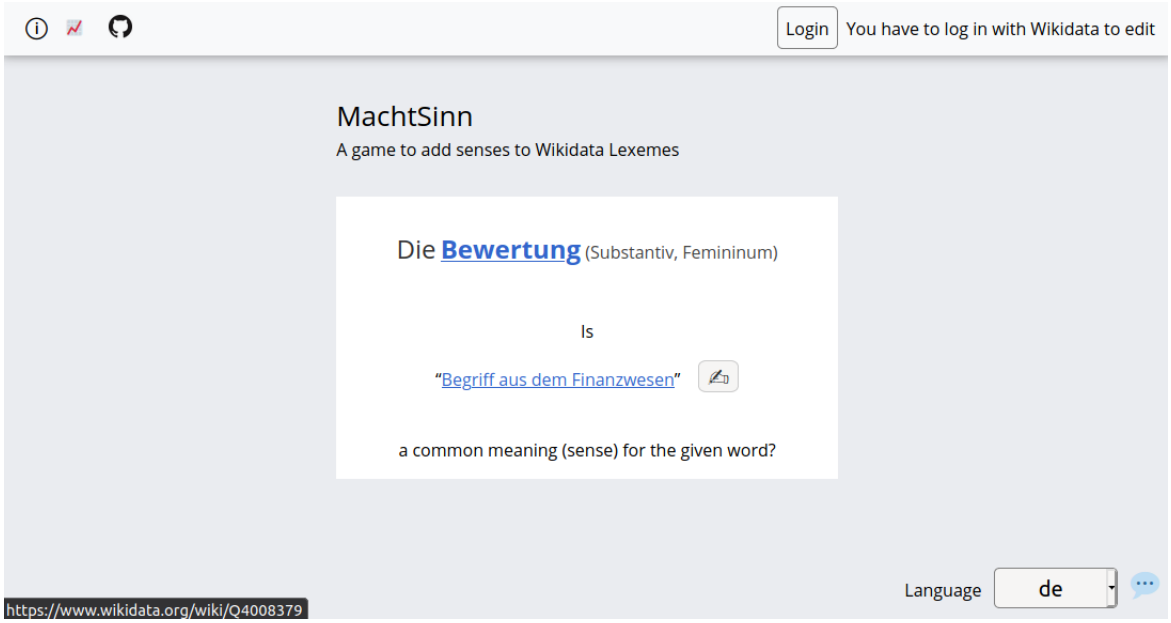

Figure 2.3: The "Macht Sinn" homepage, asking the user to evaluate if the suggested sense matches the lexeme

CHAPTER 3

Game Concept

The basic principle of Como is a word guessing game, similar to a crossword or the popular party games "Taboo" and "Catch Phrase". The concept mainly hinges on the assumption that some of the clues (crosswords) or tips ("Taboo" and "Catch Phrase") are sufficient as dictionary entries. This assumption is based on the frequent similarity between these clues and tips and a dictionary description of a lexeme.

In the following, there will be a presentation of other popular word guessing games and which aspects of them could fit into the concept of Como. Afterwards the different game modes will be explored.

3.1 Similarity to Crosswords

In a traditional crossword puzzle players need to guess matching words with the help of a clue and often some letters from other solutions. They also know the length of their solution, based on the number of missing fields.

Clues in crosswords can be of various types [27], but "straight" and "cryptic" clues are most commonly used. "Cryptic" clues include wordplays, anagrams, puns and other indirect references. "Straight" clues on the other hand can be:

- **generic**: type of boat = sailboat
- **synonymic**: synonym for king = monarch
- **antonymic**: opposite of rich = poor
- **definitory**: a quantity obtained by multiplication of two or more numbers = product
- **descriptive**: long, often yellow fruit = banana

There is a lot of overlap between generic, definitory and descriptive clues for the purpose of this work, so the terms will be used interchangeably.

As pertaining to Como, where the purpose of the GWAP is adding additional senses to Wikidata, only "straight" clues are of value to the app. Every "straight" crossword clue contains usable information, that could be integrated in Wikidata's lexeme model. In the scope of this work, only senses, which are similar to generic, definitory and descriptive

clues, will be used. While synonymic and antonymic clues are sometimes used in dictionaries, they would fit better in their respective statements as provided by Wikidata.

The problematic aspect of the similarity to crosswords is the possibility of players creating "cryptic" clues. As an example, during the test phase for Como, a player entered the following description for the lexeme "Philosophie" (English translation: "Philosophy"): "Studiengang für Denker und Taxifahrer der Moderne." (English translation: "Field of study for thinkers and modern cab drivers."). While this particular sense got solved two and skipped six times, which indicates a noticeable level of descriptivnes of the lexeme, it does however not fit an accurate dictionary definition. The player has in this case created a "cryptic" clue, which references cultural norms and stereotypes.

3.2 Similarity to Taboo and Catch Phrase

Both Taboo and Catch Phrase are two very similar multiplayer party games. They rely on team structures to create competition, use time restraints and are owned by the toy manufacturer Hasbro.

In Taboo a player from one team draws a card containing a word, which the other players on his or her team have to guess, and additionally some taboo words that the player cannot say.

An example would be a card with the word "apple" and the taboo words "tree, fruit, Adam, Jobs". The goal is to help the other players on your team to guess the word on your card correctly, while not using it yourself, the taboo words, or any derivation of them. Gestures and other means of communications are allowed. A acceptable solution would be to say: "A often red, round, healthy edible". The words must be guessed under time pressure and after the timer has run out the teams switch.

Catch Phrase has the same premise, but instead of a card with the word in question on it, the game uses a small handheld digital display, which can be thrown and catched. This device doubles as a timer. It also contains only one word at a time, so Catch Phrase has no taboo words.

Como could contain the same internal team structures with a person that describes a word and multiple others that try to guess it, represented as distinct modes. While most of the other mechanisms of those games would be interesting to implement, like giving players a time limit, or banning the use of certain words, it would be difficult to balance them with the purpose of creating senses in the first versions of Como.

3.3 Game Modes

There are two game modes, with tasks that emulate the team structures of the aforementioned party games (see Figure 3.1)

- **Creation mode**: In the creation mode players are able to create senses for words that are provided by the game. These senses should be descriptive as they pop up in other players guessing modes later, and the sense creator is only awarded points if other players guess correctly.

- **Guessing mode**: In the guessing mode players are asked to find the correct word, while only having the description of it. When they guess correct often enough, they get to create their own senses, in the aforementioned creation mode.

3.3.1 Creation Mode

Players of Como can create new senses for lexemes, by entering them into a text field. The game suggests a new sense by randomly choosing a downloaded lexeme from its database.

If a lexeme is hard to define or not comprehensible to the player, it is possible to skip to the next lexeme.

After the new sense for the lexeme is entered, the game uploads it in the background. This behaviour was implemented to make sure, that all user-created senses arrive in the server as fast as possible, to prioritize Como user-created senses over Wikidata generated ones.

This mode can only be used by spending a sense point, which is earned in the guessing mode.

3.3.2 Guessing Mode

The guessing mode is the most used screen of the app. It presents players a random sense, in their chosen language. The senses are downloaded beforehand from the server, and the users have no way of knowing if a given sense is coming directly from Wikidata, or from other users of the app. The given task for users is now to find the matching lexeme.

If a shown sense is unappealing to a player, or they can't guess the correct lexeme they have the option to skip to the next sense. When that happens, they are shown the correct answer for the last one on screen. This action is recoded for evaluation.

When a player guesses wrong for the first time, they unlock a hint screen with additional information. This information consists of the lexeme type (noun, verb, adjective or something else) and a hint for a letter. The first hint is always the first letter, as it helps players the most. With the unlocking of the hint box, it also displays all the other missing letters with an empty field, so players can now also see the length of the wanted lexeme. This is similar to clues in crosswords.

Every additional wrong guess leads to a another random unlocked letter, until half the letters of the wanted lexeme are displayed. The number of possible guesses is unlimited, but is recorded and transmitted as part of the player performance. Every entered guess is also recorded.

The game progresses to the next sense, if the correct answer is given. Guessing correctly also increases the sense point counter, which is indicated to the user. After four correct guesses the player receives a sense point. It can be used to create senses in the creation mode later. This design was chosen to make sure players only create senses, after having seen a few examples. It is also intended to make the more intellectually difficult process of creating a sense in the game a goal that can be unlocked, rather than just another task given to the player.

3.4 Point Reward Structure

There are two types of unlockable points in the game. The first one are sense points, which were already discussed earlier. The second type are the more important points, which are simply called points in the UI.

When players guess correctly with the sense of another user, instead of one from Wikidata, the creator of the sense is awarded a point in the server. These points are shown directly and prominently on the home screen of the app. They are intended as being not easy to attain, as they require playing the guessing mode often enough to attain a sense point and then creating a sense that helps another player guess the correct lexeme. If a sense helps multiple people to guess the right lexeme, the creating player earn multiple points.

As another incentive, the app includes an information display, that shoes how often users already played the guessing game, and how many times they solved the puzzle.

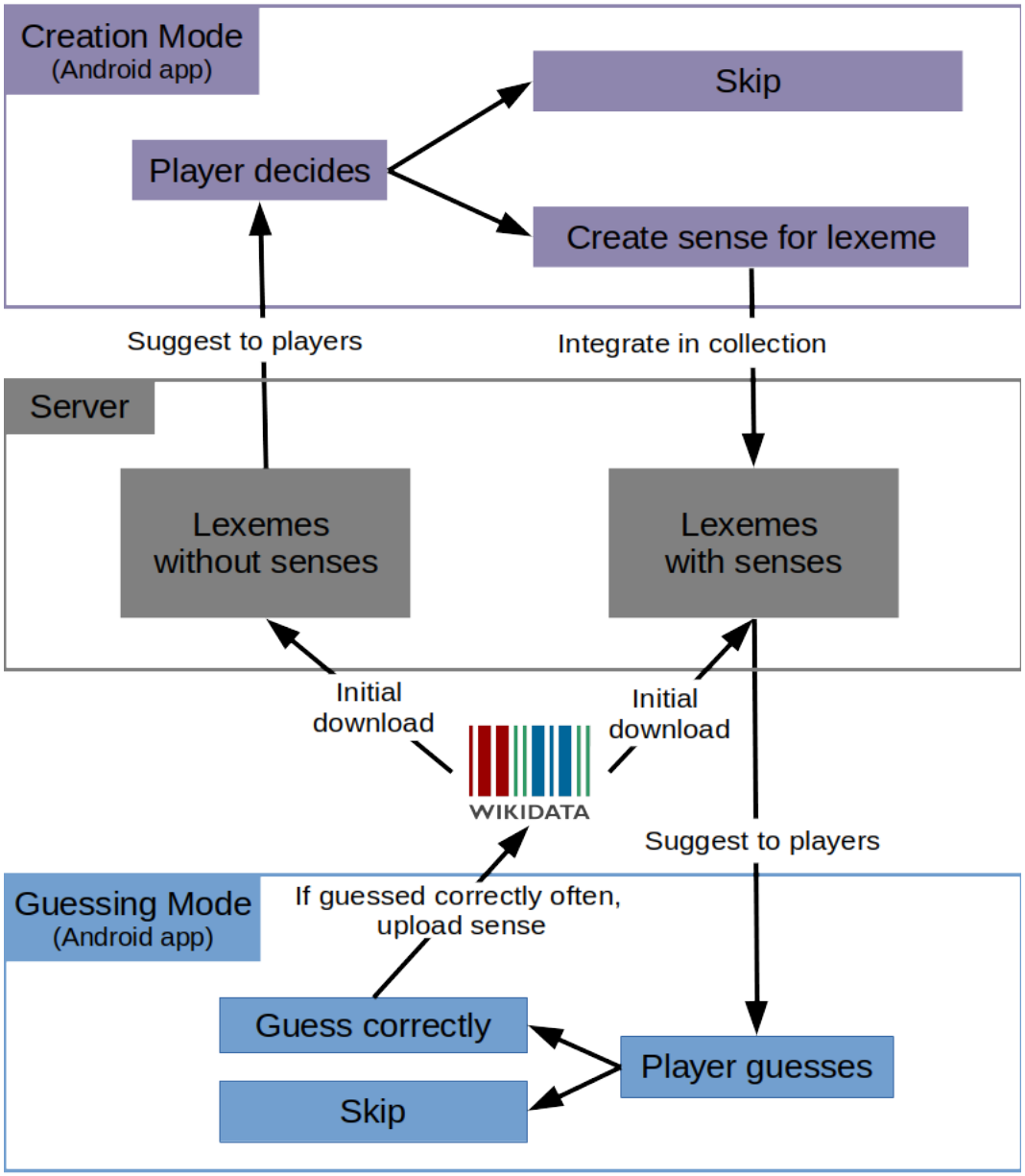

Figure 3.1: Function of the two interactive modes in Como

CHAPTER 4

Implementation

The GWAP is implemented as an Android application written in Java. It connects via a standard HTTP-JSON API to a web-service. The server is written in Python 3 and manages a Postgres Database which stores lexeme and user data. The server queries Wikidata via the Wikidata SPARQL endpoint.

An overview over the different technologies, that were used for this project, is given in Figure 4.1 and will be explored further in the following chapter, as well as the implementation of the UI.

4.1 Server

The server's main purpose is the creation and distribution of playable elements to clients. To this end it contains code to download lexemes (with and without pre-existing senses) from Wikidata.

These are then prepared to be delivered to clients, to be played in the app. The resulting user performance data is then synced back to the server, to be processed into new playable elements or to enhance data on existing games in the database. In the future high performing user-created senses will be uploaded to Wikidata.

4.1.1 Interaction with Wikidata

To download information form Wikidata, the SPARQLWrapper Python library [2] is used. This provides a simple abstraction for creating queries and helps with the parsing of returned information.

4.1.1.1 SPARQL

One of the main ways offered by Wikidata to interact with its data pool is a SPARQL endpoint at `https://query.wikidata.org`. This mechanism will be used to generate the initial data sets for the server (see Figure 4.4).

SPARQL is short for "SPARQL Protocol and RDF Query Language"[5] and references a standard by the World Wide Web Consortium [13], used to retrieve and manipulate RDF-formatted data from databases.

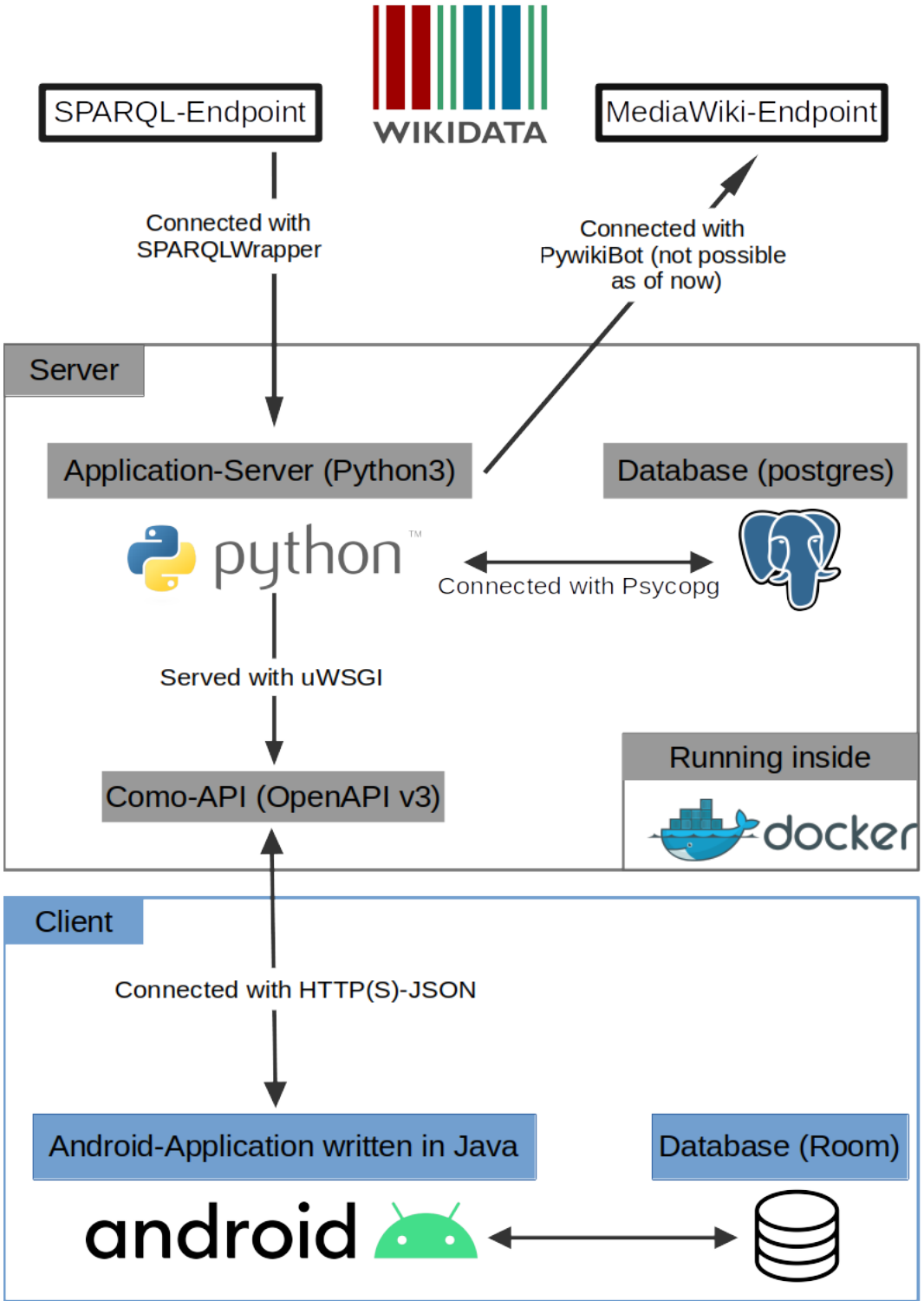

Figure 4.1: Usage of different technologies in Como's architecture

```
SELECT ?l ?lemma ?sense ?cat WHERE {
  ?l a ontolex:LexicalEntry;
  dct:language ?language;
  wikibase:lemma ?lemma;
  wikibase:lexicalCategory ?cat;
  ontolex:sense/skos:definition ?sense .
  ?language wdt:P218 'en' .
}
```

Figure 4.2: A SPARQL Query to download all lexemes and senses in the English language

```
SELECT ?l ?lemma ?cat WHERE {
  ?l a ontolex:LexicalEntry ;
  dct:language ?language ;
  wikibase:lexicalCategory ?cat ;
  wikibase:lemma ?lemma .
  ?language wdt:P218 'en'
  FILTER NOT EXISTS {?l ontolex:sense ?sense }
}
```

Figure 4.3: A SPARQL Query to download all lexemes with no senses in the English language

The structure of the two queries in the server implementation to download lexemes with and without senses is shown in Figures 4.2 and 4.3. They each return a set of over 58,000 (Lexemes without senses) an over 16,100 (Lexemes with senses) for the English language, 74,100 results total.

These results seem to be at odds with the previous results from Figure 1.2, where the English language contains around 70,000 lexemes on Wikidata, but only around 11,000 have at least one sense. The explanation for this mismatch is that query 4.2 counts every sense of a lexeme as a new result, which leads to an overshoot of around 4,000 results.

4.1.1.2 Content Generation

As described before the server has been equipped with functions to initially download two types of result tables from Wikidata.

First it downloads lexemes (in both supported languages, English and German), that don't possess any senses in Wikidata. They are saved in a database table called "sense creations" (Table A in Figure 4.4), as they will be used to create new senses. These will be suggested in the creator mode of the game, to animate players to enter an initial sense for them.

Additionally it can collect lexemes that are connected with senses and save them to the database as "games" (Table B in Figure 4.4). These are the originally provided playable lexemes, that users can interact with in the app. This creates a pool of games that the app can later pull from, while ensuring the existence of a sufficient number of games, even for very demanding app users.

The server does not need to, at this time, authenticate with Wikidata, as it only queries and stores data.

4.1.1.3 Result Upload

Game performance data is processed by the server, to track how many players can find the correct lexeme to a given sense. These results are the sorted and a cut-of-point can be determined, where certain senses are disregarded due to poor performance. The rest will

be played, until there is a sufficient sample of players, that have tried the sense in the game. When a sense is judged as descriptive enough, it is selected for upload to Wikidata.

As one goal of this project is contributing senses back to Wikidata, it would be appropriate to use Pywikibot [6]. It is a Python script, that can be used to interact with most of MediaWikis APIs. In this case it would handle the authentication with Wikidata and the upload of new lexemes. However this is not possible as of September 2020, because of the missing implementation of lexeme entities in Pywikibot. [11]

4.1.2 Database

All persistent information is stored in a Postgres database, that is exposed to the server by a dedicated user account. The database is deployed as a docker container and located on the same infrastructure as the server software. It consists of six tables:

1. **users**: A simple table to keep track of users and their points.
2. **games**: Contains all available riddles, both Wikidata and user-generated ones.
3. **games_to_users**: Keeps track of which games are sent to which user and records results.
4. **inputs**: Helper table for games_to_users. Records the different inputs for each game.
5. **sense_creations**: List of all lexemes on Wikidata without senses.
6. **sense_creations_to_users**: Keeps track of which sense-creations are sent to which users and records results.

A simplified view of the database can be seen in Figure 4.4, where 2, 3 and 4 are represented in Table A and points 5 and 6 are represented in Table B.

The actual database itself doesn't require very much space as all data consists of plain text. During testing it always stayed well below 100 MB in size.

The server connects to the database with a Python library called psycopg [1], that helps with preventing SQL injection attacks by sanitizing user inputs. As all inputs are directly sent to the server and the server doesn't require authentication it would be trivial to take control of the database without these measures.

4.1.3 API

The direct communication between the mobile app and the server is HTTP-based. It is implemented as an OpenAPI v3 interface in Python with uWSGI as the providing app service. The API specification itself is simple and handles only two request, see Figure 4.4.

The "GET /data/" request requires three parameters. The UUID, a "Universally Unique Identifier" (128 bit long random string) generated locally by the app and the requested lemma and sense languages. The UUID is relevant to prevent sending of duplicate game data, while the language settings are so far only used to set the language for both the displayed lexemes and their senses, not independently. This call returns a JSON object with two lists. One list contains randomly selected games for the guessing mode and the other a set of new lexemes without senses for sense creation. Additionally it sends the updated point value to the players.

The "POST /data/" request is used to upload player data to the server. It is also formatted as a JSON object with a list of all played games, their results and all newly created senses. The results contain information about the players behaviour with each new challenge, including if the actually found the correct lexeme or skipped it, as well as all failed guesses. The object concludes with the UUID.

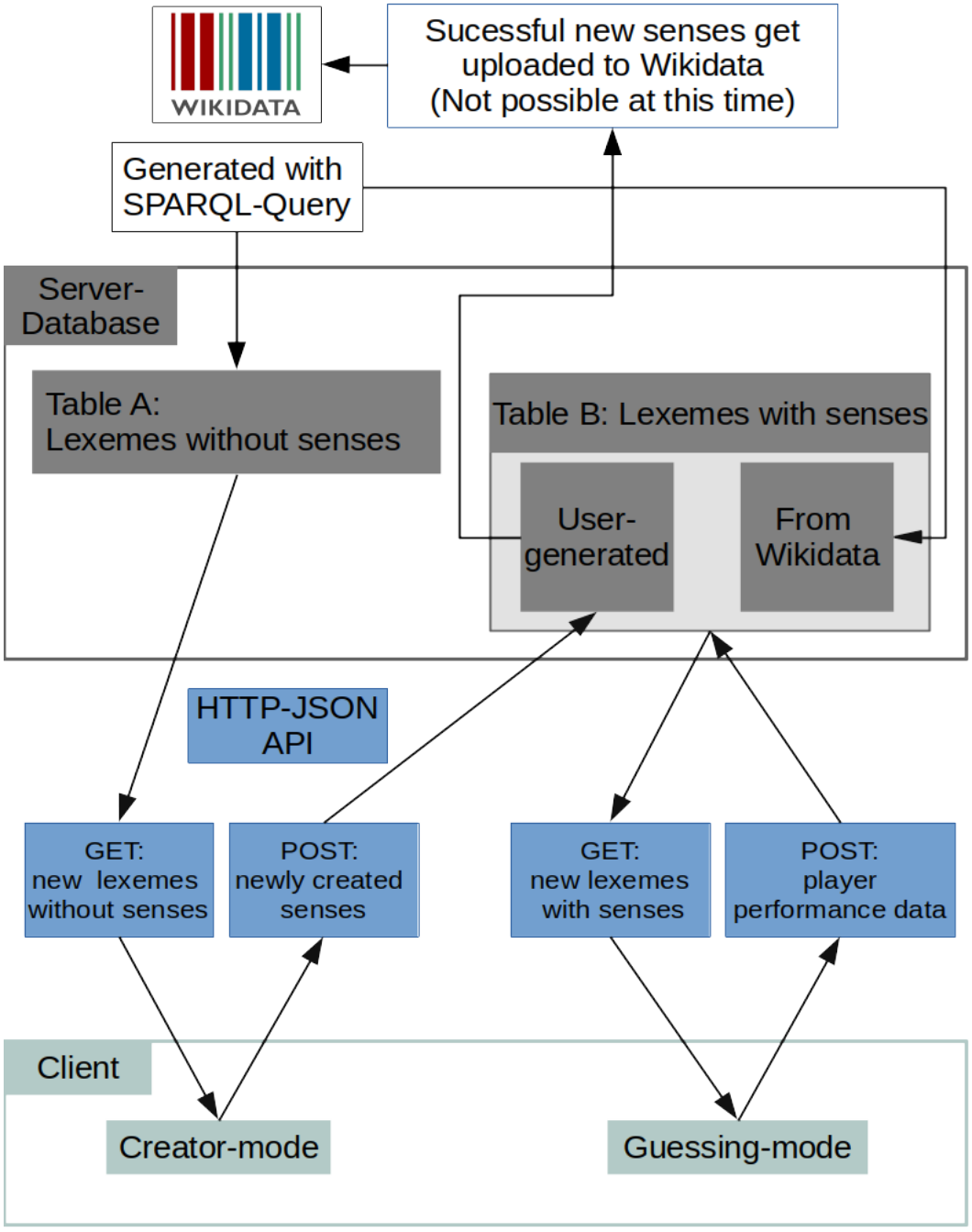

Figure 4.4: An Overview over the basic structure of the Como-API and the server data management

The API is provided unauthenticated, as it would require a more complete implementation to integrate it into the existing app. Nevertheless the communication is HTTPS-encrypted and it is not possible to request any player related data from the server, except for their points. The server also does not store any personal data and submitting wrong results for another player requires a hard-to-guess UUID. The API is vulnerable to DOS-attacks as it is not rate-limited, but could be protected with simple measures should the need arise.

4.1.4 Hosting

The whole application server is at the time hosted on private infrastructure. It runs as two docker containers (application and database) on an Ubuntu OS. To continue the project long-term it would be necessary to move it to Wikimedia infrastructure, which does require some code adjustments. As the database and application are very lightweight hosting them in the future should not require many financial and technical resources.

4.2 Client

The client is implemented as an Android application that connects to the server via a HTTP-JSON API to download new content and upload user input and performance data.

The application's source code can be found online in Como's Gitlab repository. It is licenced under GNU GPL v3.

4.2.1 Platform

The Android platform was chosen for the client implementation, as it is under-represented in the Wikidata tool collection. So far, most of the successful GWAPs around Wikiprojects (e.g Section 2.1 or Section 2.2) are build as web applications with sometimes limited mobile and no offline usability. This praxis is understandable, as most human editors of these projects use desktop web-browsers, due to habits and lacking or less functional mobile editing interfaces.

This is however a huge loss of potential users/players, as desktop usage shrinks and mobile platform usage rises [30]. Furthermore the implementation of Como as a native Android app enables offline usability. This matters not only in regions with a lack of mobile internet access, but also allows usage of Como while commuting or flying.

The application requires a minimal Android API-Level of 16 (Android 4.1) which should make the app compatible with over 99% of sold Android devices [4].

4.2.2 Software Architecture

The code implementation follows a Model-View-ViewModel [29] pattern. This style is the default, when interacting with a local database in Android. In this architecture pattern the "views", called "activities" in Android, only contain the visible portions of the software. The "model"-logic is separated from the "view" with the "viewModel". The "viewModel" is responsible for converting and exposing all relevant data from the "model" to the "view" and thereby drives most of the display logic as well. The "view" then only observes the "viewModel" as can be seen in Figure 4.5

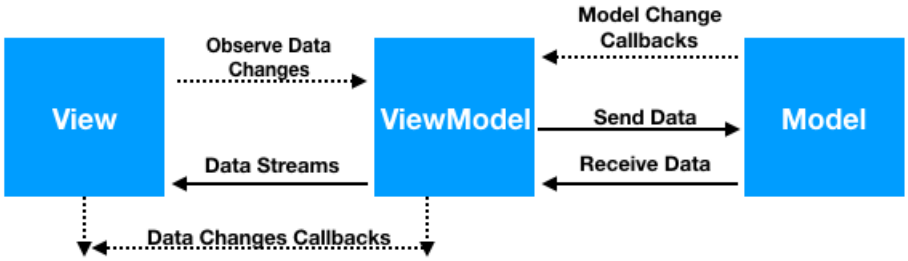

Figure 4.5: An Overview of the relations in a MVVM-Pattern [14]

4.2.3 Interaction with Server

The client connects to the Como API with asynchronous executor threads. As there is no implemented authentication, aside from the transmitted UUID, there are no further steps to take to establish a communication channel.

The application can be forced to manually connect to the sever, but mostly auto-connects when a new sense is created or there are fewer than ten lexemes left to play. It always sends the GET /data/ and POST /data/ requests together, to reduce the frequency of requests overall. Therefore every time a connection is made the app syncs completely, to improve offline capabilities. In conclusion, during every sync event these data points are transmitted:

- **From the server (GET /data/)**:
  - New lexemes without senses (sense creations)
  - New lexemes with senses (games)
  - The updated point total
- **From the client (POST /data/)**:
  - Player performance data on games
  - Newly created senses
  - Language and platform information

4.2.4 User Interface

Como's UI is simplistic and features no styled animations. The color scheme consists of three distinct colors (red, green and blue), which are toned to equal the Wikidata logo.

The whole application consists of four Android activities (Android's implementation of views).

The main activity and the settings activity are used to concisely explain the purpose and functionality of the app. Additionally they offer access to the players statistics and the other activities.

The guessing mode activity is the one most used by players and only consists of a text entry field and two buttons, "check" and "skip".

The creation mode activity needs to be unlocked by use of the guessing mode and only features the same set of interactive elements, except the "check" button is now a "save" button.

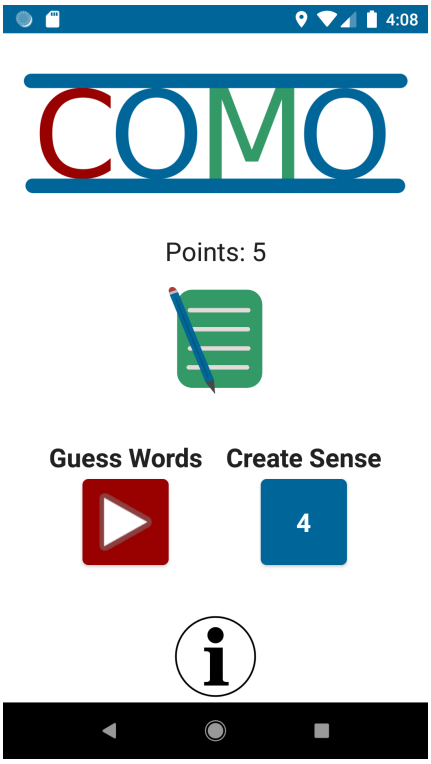

Figure 4.6: Como's main activity

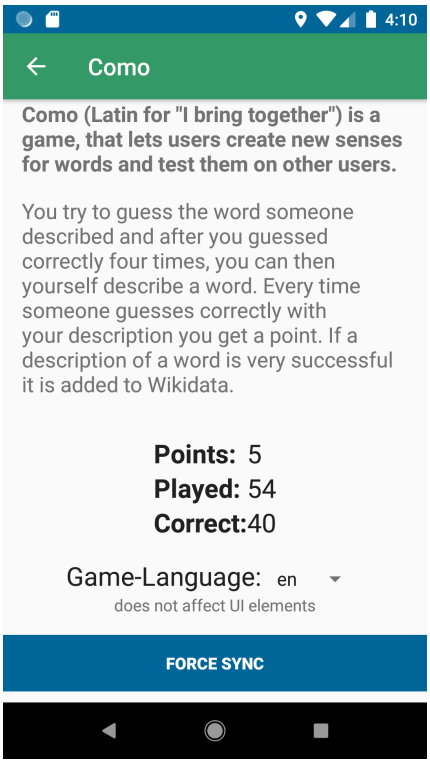

Figure 4.7: Como's settings activity

4.2.4.1 Main Activity

Como's main activity, as displayed in Figure 4.6, is the starting point for user interaction with the application. It tries to distill the essence of the game to "Guess Words" and "Create Sense", to negate the need for long tutorials and explanations. If players still want to learn more about Como, it links to the settings page with information icon.

All in all the activity contains three interactive buttons and two counters. The red "play" button leads to the guessing mode activity, while the blue one leads to the creation mode activity. The "I" button, as mentioned, brings up the settings.

The upper counter, labeled "Points", displays the times that a senses created by the player helped another player to guess correctly. The lower counter in white on the blue button shows how many senses a player is allowed to create. A sensepoint is added every fourth time a player guesses the right lexeme in the guessing mode. The blue creation mode button is only clickable, if the number of creatable senses is greater than zero.

4.2.4.2 Settings Activity

The settings activity is a combination of a short description of the game, a personal statistics page and a language setting, as displayed in Figure 4.7.

The page displays short explanation of the name, then continues to provide a quick tutorial. Below that, players can see a summary of their points and their performance in the guessing mode.

The game allows to change the language of the used lexemes and senses, which does not change the UI language of the app. This process triggers a sync request with the server.

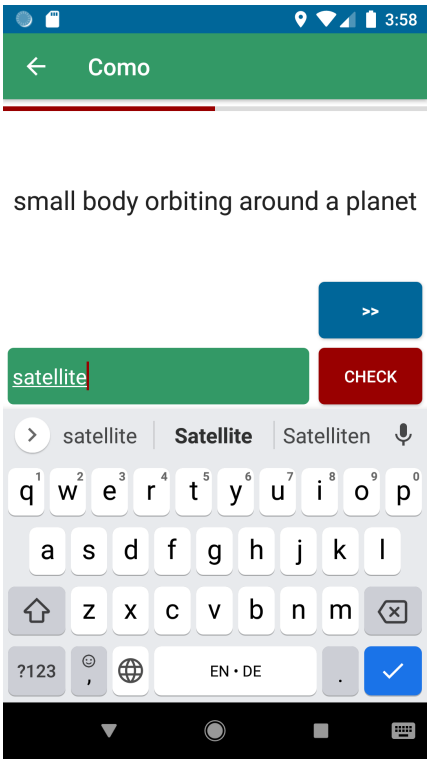

Figure 4.8: Como's guessing mode activity

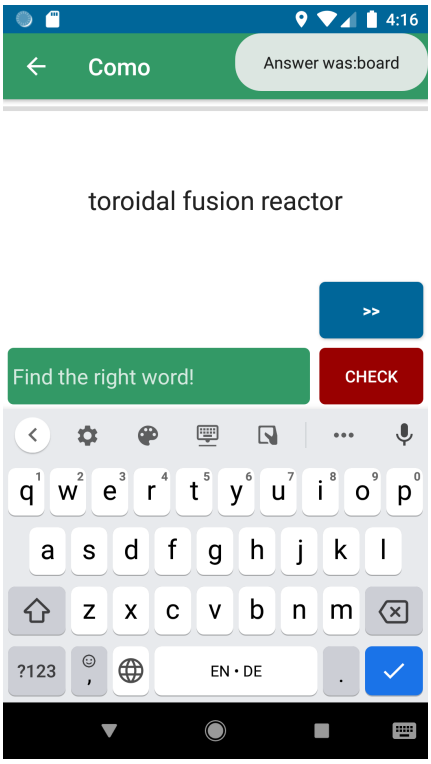

Figure 4.9: App feedback on skipping

For debugging purposes the menu allows to force synchronisation should any errors have occurred during communication or on the server side.

4.2.4.3 Guessing Mode Activity

Most of the player's time is spent in the guessing mode activity. Its main purpose is to direct the players through the game and to provide feedback and hints on the way.

The UI, as presented in Figure 4.8, consists of three interactive and two informative elements. The informative displays are:

- **Sense point progress bar:** A red bar at the top indicating the progress of the player towards a new sense point (which allows the creation of new senses). A sense point is awarded, when the players guess the right lexeme four-times.
- **Sense display:** Presents the sense for the current lexeme.

Below these, there are the three interactive elements:

- **Input textbox**: A simple textbox, with the Android keyboard spellchecker enabled. Players should enter their guesses here.
- **"Check" button:** Triggers comparison between the entered text and the actual solution.
- **"Skip" button** (represented by ">>"): Allows players to skip to the next entry, with a short message that displays the right answer to the last riddle, as shown in Figure 4.9.

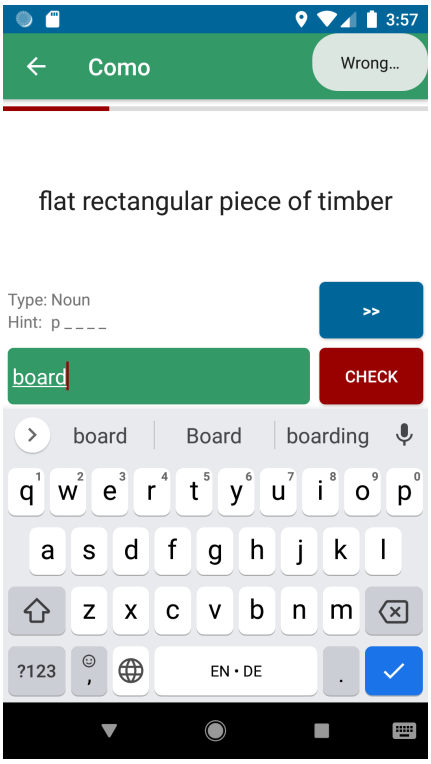

Figure 4.10: App feedback on wrong input

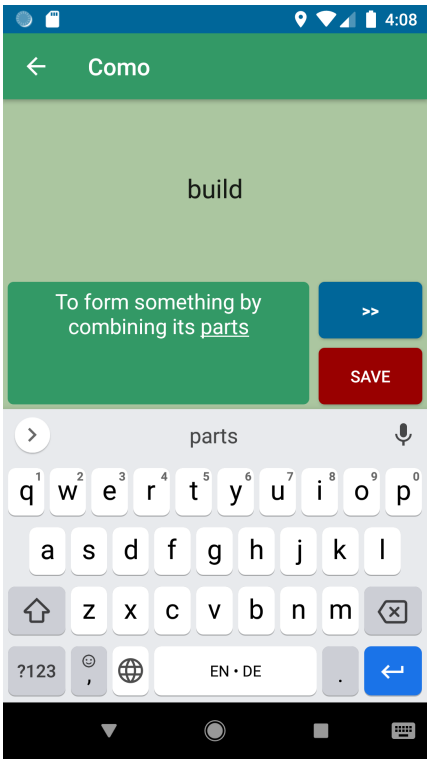

Figure 4.11: Como's creation mode activity

After players have entered there first guess and hit check, the UI reacts based on the correctness of the suggested answer.

When players guess wrong, the game gives feedback in the form of a short message, as can be observed in Figure 4.10. Additionally it creates a small hint box. The hint box shows the word type and the length of the lexeme in question with underscores. It also displays the first letter of the lexeme, as it is often the most helpful. With every consecutive try, another random letter from the answer is revealed, until half of the complete answer can be read.

4.2.4.4 Creation Mode Activity

In the creation mode (see Figure 4.11) players are tasked with finding a fitting description for a displayed lexeme. For this reason the interface presents a random lexeme from the selected language and offers the user a multiline textbox with the Android spellchecker enabled.

Furthermore players can press the "skip" or "save" buttons. "Skip" brings up the next possible lexeme, while "Save" writes the entered description to the database and tries to sync with the server.

4.2.5 Distribution

The game was made freely available on the Google Playstore during the evaluation phase. After the client source code was additionally published on Gitlab.com the game was submitted as a request for packaging to the FOSS repository F-Droid, but wasn't released to F-Droid users during the evaluation period.

As a replacement for those users, the app was distributed as an .apk (Android app packages) file. As the game registered 30 users during the period, and the playstore registered 12 installations, more than half the users were using the distributed apk version.

CHAPTER 5

Evaluation

The main reason for Comos existence is to determine if it could be possible to encourage mobile users to contribute to open knowledge projects. Como testers could use the app for a 30 day period, between 23.09.2020 to 23.10.2020, during which all useful interactions were recorded by the server. These contributions can be measured in type, quantity and quality and will be evaluated in the following.

The raw data tabulation from the test period can be found online in the Como Gitlab repository.

5.1 Data Collection Format

All data was collected through the Android application and saved in the server SQL database. Users are identified through UUIDs that are generated in the Android app. All data in the database was fed into a Grafana [15] instance to visualize the information. This was especially helpful during early development as a way to verify data integrity during the addition of new SQL statements.

5.2 Test Group Selection

The actual composition of the test group cannot be determined, as the application was freely available to download during the evaluation period. Of all 30 testers, most were probably acquired by a post on the Wikidata Weekly Newsletter [26] as well as through personal contact. Most of the people targeted through the Wikidata newsletter are highly likely to share an interest in open knowledge projects from the start, while most of the personal contacts were people without such a background.

As the number of testers is too little to draw any conclusions from other attributes of the group like age, occupation, gender or location, the single measurable user metric available was language.

5.2.1 User Language

As the language settings in Como can be changed on the fly, the server can't really assign a permanent language value to a user. For the purpose of a rough overview over language separation it does however save the language flag from the last request in the user table. When looking at the chart in Figure 5.1, there is often a visible constant slim majority of German users.

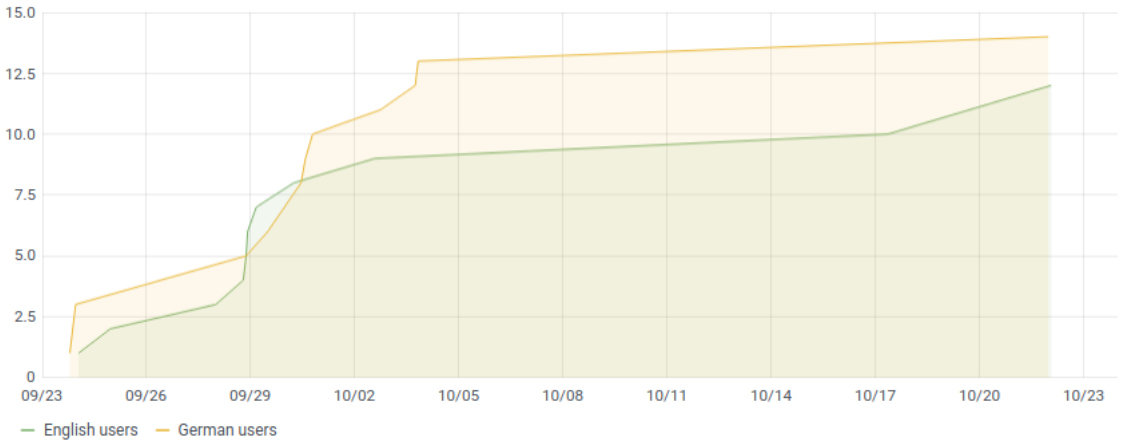

Figure 5.1: Users grouped by language over time

5.2.2 User Activity

In the 30 day test period the server recorded a total of 2596 interactions (not including downloads only result uploads). These interactions are concentrated at the beginning (see Figure 5.2), with a noticeable spike after the announcement on the Wikidata weekly newsletter (28.09.2020) [26].

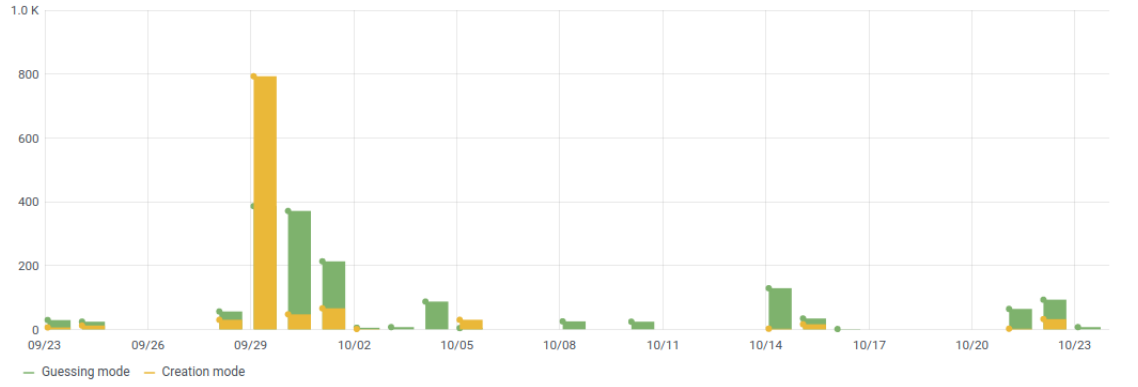

Figure 5.2: Recorded interactions per day

These interactions can be categorized in the following way:

	Guessing mode	**Creation mode**	**Both**
Sum	1559	1037	**2596**
Daily average	∼52	∼35	∼**87**

5.3 Results

The server initially downloaded over 18,000 lexemes (23% German, the rest English) with senses for the guessing mode and over 34,000 lexemes (10% German, the rest English) without senses for the creation mode from Wikidata. The discrepancy between lexemes with and without senses accurately reflects the lack of senses on Wikidata, as the SPARQL queries were written to collect all existing lexemes for German and English (see Figure 1.2 for an overview).

After a few initial players uploaded their own senses, they were prioritized as games so most of the guessing mode feedback would be on them. This was done to increase the amount of quality evaluations for the possibly soon to-be-uploaded senses to Wikidata. If user-created senses were not prioritized in the guessing mode they would have had a very low chance of actually getting played by users, as the amount of Wikidata-generated games is significantly larger.

5.3.1 Creation Mode

As the access to the creation mode was limited by the performance in the guessing mode to only allow more experienced players to create senses, there were only relatively few recorded interactions.

While users actively saw 1037 lexemes in the creation mode, they decided to only create senses for 59 (6%), see Figure 5.3.

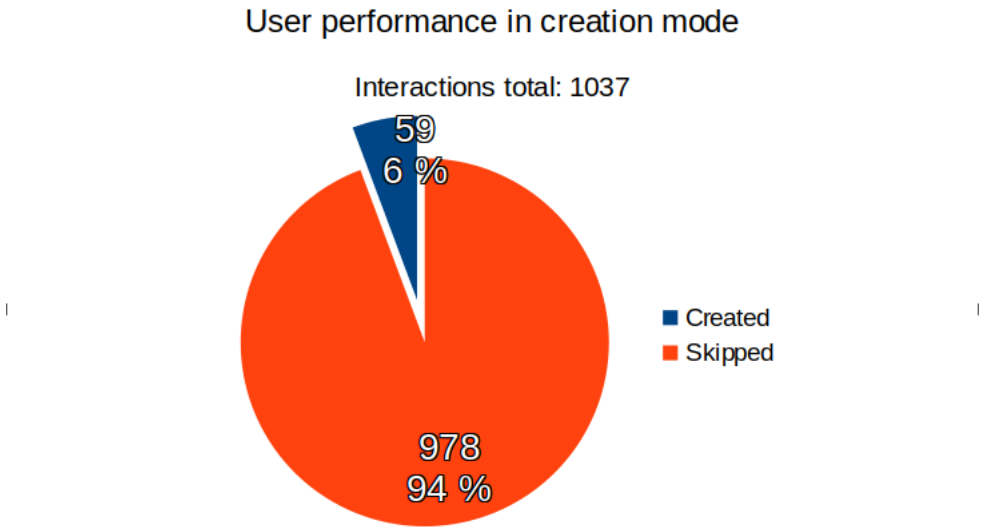

Figure 5.3: The comparison between created senses and skipped entries in the creation mode

5.3.2 Guessing Mode

As visible in Figure 5.4, during the test period there is a disparity in requested and completed games in the guessing mode.

This behaviour is a result of the apps design, whereby it tries to download a number of games for offline use every time a synchronisation happens. As this happens quite frequently, for example, when players upload a new sense from the creation mode, the number of requested games vs completed games naturally diverges.

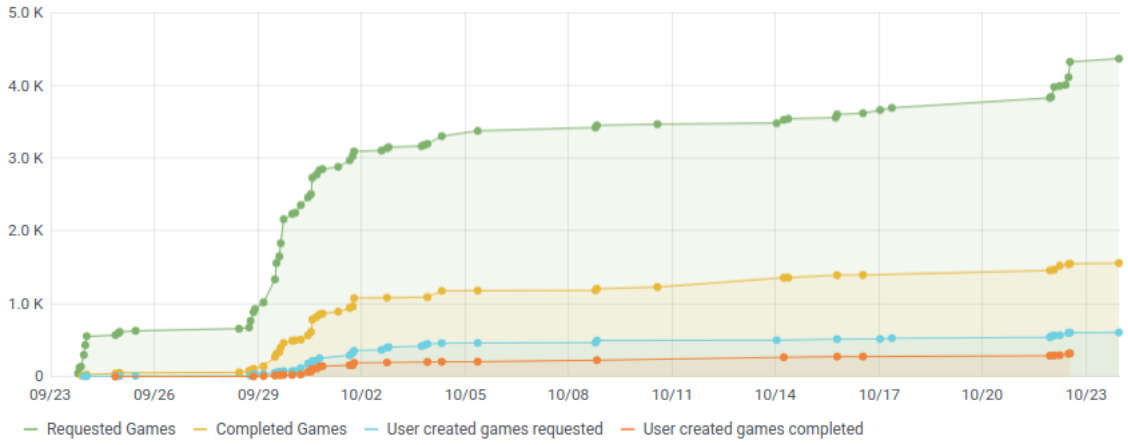

Figure 5.4: Downloads and upload of games to the server

As mentioned in Figure 4.4 there are two types of games for the guessing mode in the database:

- Games generated from lexemes with senses from Wikidata
- Games created with lexemes from Wikidata and senses from the users of Como

In total the app delivered 4365 games to players, of which only 14% were created by the apps users directly, see Figure 5.5.

But of those delivered games, the players only interacted/completed 1676 (38%). The breakdown of those interactions can be found in Figure 5.6.

5.3.2.1 Wikidata Senses

The performance of senses from Wikidata in the guessing mode can be measured in the amount of solved vs skipped games. As observable in Figure 5.7 most (69%) of the Wikidata senses were skipped.

This is most likely due to some Wikidata senses being very general. In the available list of played senses there are six occurrences alone of descriptions being some variation on: “Heraldic symbol” which were exclusively skipped and never solved by players.

5.3.2.2 User-created Senses

User-created senses are quite more important to analyse, as their creation and evaluation is the main purpose of Como as a project.

These senses have a remarkable difference in their relationship between skipped and solved games. As shown in Figure 5.8 around 54% of those senses were enough deduce the right lexeme.

This demonstrates a measurable increase in the solvability rate and indicates a higher overall descriptiveness of the user-created senses.

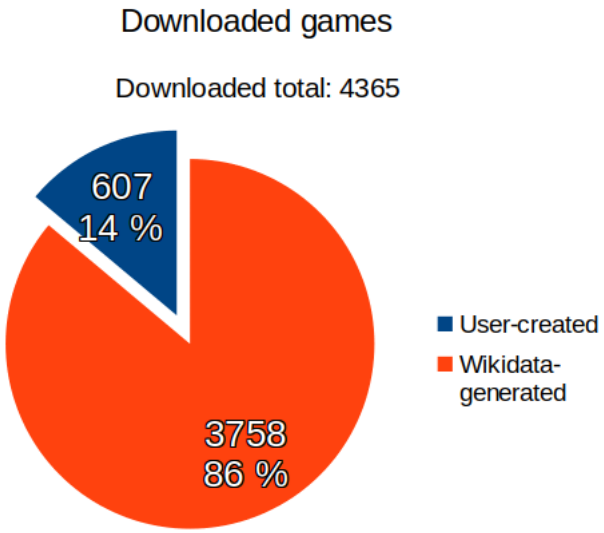

Figure 5.5: Downloaded games during test phase

While already being on average easier to solve, to ensure the quality of the senses uploaded to Wikidata, some restrictions should to be applied on the user-created senses. This process was not implemented in the application, because of the missing features mentioned in section 4.1.1.3. The following upload restrictions suggestion is only presented and evaluated as a possible implementation design for the future.

There are many available selection factors to choose from, but a check on the number of times a sense was played in the guessing mode is a straightforward and simple indication of reliability. An adequate cut-off point for the minimum number of interactions would be the average of all interactions, which is 6.25.

So using this example value the server would only upload user-created senses if they have been played at least six times. As visible in the frequency distribution in Figure 5.9, this reduces the number of possible uploads by a third.

Another reasonable assumptions would be that user-created senses need to have at least more players that solved them than skipped them. When applying this selector only 25 senses to upload remain. But the solved-skipped distribution changes even more in the favour of solved games, as visible in Figure 5.10.

5.3.3 Comparison

These findings are encouraging for the methods Como uses, as they suggest a higher quality of the new descriptions coming from Como compared to the already existing ones in Wikidata. This effect also happens before the selection of higher quality senses from the user-created pool. There already is a 23% difference in the solved-rate between Figures 5.7 and 5.8.

Even more impressive is the fact, that this difference is visible before the game removes hardly or unsolvable senses from the suggested Wikidata uploads. When applying some simple restrictions on the upload, the difference expands by additional 12%.

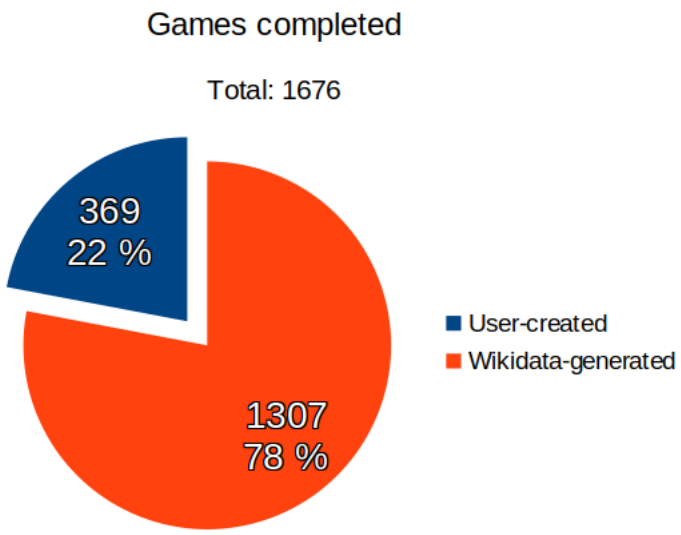

Figure 5.6: Completed games during test phase

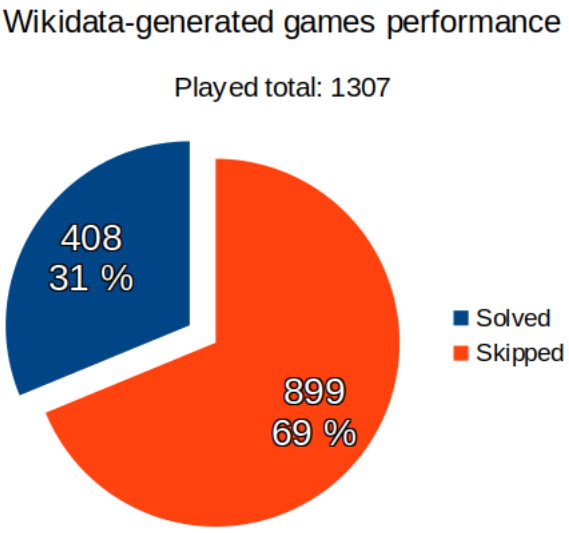

Figure 5.7: Player feedback on Wikidata-generated games

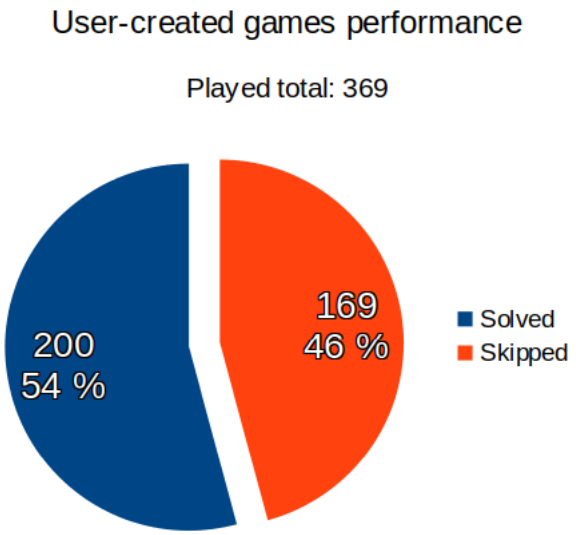

Figure 5.8: Player feedback on user-created games

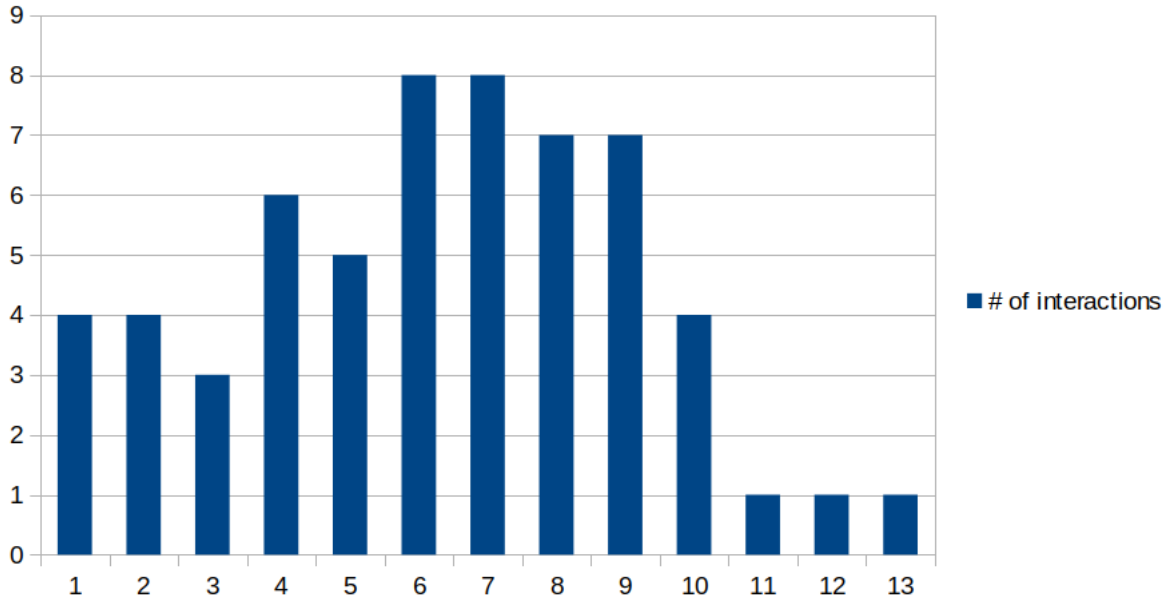

Figure 5.9: Frequency of interactions with user-created senses

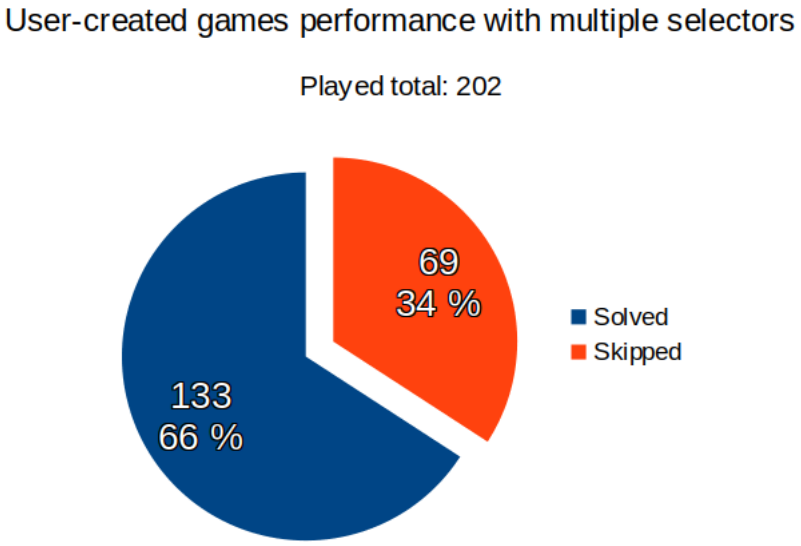

Figure 5.10: Performance of user-created games with restrictions

CHAPTER 6

Conclusion and Future Work

The amount of testers and the duration of the test period are not sufficient to draw any definitive conclusions.

Additionally there are many caveats to using the solved-skipped rate as an indication of descriptiveness of a sense. For example users could, when confronted with a descriptive and accurate sense just decide to skip the lexeme because it came form a unfamiliar subject field. But since the game, in an effort to be simplistic, only offers these two choices, they were used to evaluate the performance of the senses. Aside from these caveats, this approach of comparing sense performance detects clear differences between senses from the app and Wikidata. Following this, there is evidence that Como and its underlying methods could be helpful in creating and collecting structured lexicographical data through collaborative gameplay.

The amount of created senses per interaction is small in comparison to other similar tools (see section 2.1). This can be expected, due to Como demanding a great amount of creative work from its users, while most of the other tools only let players choose between pre-generated options. Which is also the reason why Como can be used more flexibly, as it doesn't have any codependent data sources it needs to provide suggestions to players.

6.1 Futureproofing Como

One major concern with the application is its hosting location, as it was primarily hosted privately during the test phase. The server part of Como should be moved into a proper hosting environment, preferable even the Toolforge infrastructure used by other Wikimedia related tools and games [9].

This move would require the use of a different database binding, as Toolforge only supports MySQL. Aside from that the project is technologically compatible with the hosting environment and should be moved there.

Apart from this change the server should include a way to upload created senses to Wikidata as soon as Pywikibot has included this feature. [11]

Como already measures the performance of lexemes with senses on Wikidata without acting on that data. It could be possible to replace badly performing Wikidata lexeme senses with better Como created ones, if such a feature was implemented.

One of the outstanding features of Como is the ability to play instantly, without the hassle of account creation or logins. To offer this API without authentication could be a great asset for the popularity of the game, but should include other measures to prevent spam or malicious edits. Possibly this could be archived through manual human approval of Comos sense uploads by one or multiple registered Wikidata editors. This approval process could also be implemented as a GWAP on the "The Distributed Game" platform.

Additionally the design and offline functionality of the client app could be improved and the application could be made available to other platforms, like iOS or even Ubuntu Touch.

6.2 Additional Data Harvesting

Due to time constraints it wasn't possible to expand Comos functionality to more than senses. The game could be adapted further to fully use the complete Wikidata lexeme model (as seen in Figure 1.1), by allowing users to enter more data on certain lexemes, like synonyms, antonyms, word types, translations etc.

Especially the last point could aid a new set of users, namely people interested in learning other languages. There are many possible implementations of this, including additional modes or fields. Those should, however, be integrated with care, to keep the simplistic interface and gameplay intact.

Many players, while personally giving feedback on the app remarked that they wished to correct their own or other players spelling or grammar, which could be an interesting feature.

In addition to this, Wikidata also offers support for more exotic data types like IPA symbols or pronunciation audio, which could be used to enhance the variation of the gameplay.

Particularly interesting would be to allow users to create, describe and categorize their own lexemes. As languages are a creative and dynamic target, the application could offer a way to see the changing nature of new words as they are created and tested by the players of Como.

Bibliography

[1] *Psycopg, a PostgreSQL driver for Python Website*, https://www.psycopg.org, Dec 2020, [Online; accessed 13. Dec. 2020].

[2] *SPARQL Endpoint interface to Python*, https://rdflib.dev/sparqlwrapper, May 2020, [Online; accessed 22. Sep. 2020].

[3] *Wikidata - The Distributed Game*, https://wikidata-game.toolforge.org/distributed/#mode=stats, Jun 2020, [Online; accessed 3. Dec. 2020].

[4] Android Authority, *Android version distribution: Are Google's faster roll-out initiatives working?*, https://www.androidauthority.com/android-version-distribution-748439, May 2020, [Online; accessed 11. Dec. 2020].

[5] Dave Beckett, *Re: What does SPARQL stand for?*, https://lists.w3.org/Archives/Public/semantic-web/2011Oct/0041.html, Oct 2011, [Online; accessed 22. Sep. 2020].

[6] MediaWiki Contributors, *Pywikibot manual*, https://www.mediawiki.org/wiki/Manual:Pywikibot/Overview, Sep 2020, [Online; accessed 22. Sep. 2020].

[7] ———, *Wikibase Lexeme Data Model*, https://www.mediawiki.org/wiki/Extension:WikibaseLexeme/Data_Model, Aug 2020, [Online; accessed 14. Aug. 2020].

[8] Wikidata Contributors, *Wikidata:Introduction*, https://www.wikidata.org/wiki/Wikidata:Introduction, Aug 2020, [Online; accessed 21. Aug. 2020].

[9] Wikitech Contributors, *About toolforge*, https://wikitech.wikimedia.org/w/index.php?title=Portal:Toolforge/About_Toolforge&oldid=1872592, 2020, [Online; accessed 15-December-2020].

[10] Wiktionary Contributors, *Wiktionary Main Page*, https://en.wiktionary.org/wiki/Wiktionary:Main_Page, Aug 2020, [Online; accessed 21. Aug. 2020].

[11] Wikimedia Foundation, *T189321 Support for Lexicographical data/WikibaseLexeme in Pywikibot*, https://phabricator.wikimedia.org/T189321, Sep 2020, [Online; accessed 22. Sep. 2020].

[12] ———, *Wikilegal:Lexicographical Data*, https://meta.wikimedia.org/wiki/Wikilegal/Lexicographical_Data, Aug 2020, [Online; accessed 21. Aug. 2020].

[13] Steve Harris, Andy Seaborne, and Eric Prud'hommeaux, *Sparql 1.1 query language*, Recommendation, W3C, Mar 2013, https://www.w3.org/TR/sparql11-query/.

[14] JournalDev, *Android MVVM Design Pattern*, https://www.journaldev.com/20292/android-mvvm-design-pattern, Mar 2019, [Online; accessed 10. Dec. 2020].

[15] Grafana Labs, *Grafana Product Website*, https://grafana.com, Dec 2020, [Online; accessed 13. Dec. 2020].

[16] Lea Lacroix, *Lexeme data model - Wikimedia Commons*, https://commons.wikimedia.org/wiki/File:Lexeme_data_model.svg, Aug 2020, [Online; accessed 21. Aug. 2020, License CC BY-SA 4.0].

[17] Lydia Pintscher, *First steps of Wikidata in the Hungarian Wikipedia*, https://blog.wikimedia.de/2013/01/14/first-steps-of-wikidata-in-the-hungarian-wikipedia, Jan 2013, [Online; accessed 20. Dec. 2020].

[18] Magnus Manske, *The Distributed Game*, https://wikidata-game.toolforge.org/distributed/, Jun 2020, [Online; accessed 6. Nov. 2020].

[19] ———, *Wikidata-game's source code repository on Bitbucket.org*, https://bitbucket.org/magnusmanske/wikidata-game/src/master, Dec 2020, [Online; accessed 3. Dec. 2020].

[20] Maximilian Kristen, *SPARQL Query to determine the amount of lexemes and missing senses in the top ten languages*, https://w.wiki/r74, Dec 2020.

[21] Michael Schönitzer, *MachtSinn Statistics*, https://machtsinn.toolforge.org/statistics, Dec 2020, [Online; accessed 4. Dec. 2020].

[22] ———, *MachtSinn Website*, https://machtsinn.toolforge.org, Dec 2020, [Online; accessed 4. Dec. 2020].

[23] Finn Nielsen, *Lexemes in Wikidata: 2020 status*, Proceedings of the 7th Workshop on Linked Data in Linguistics (LDL-2020) (Marseille, France), European Language Resources Association, May 2020, pp. 82–86 (English).

[24] Thomas Pellissier Tanon, Denny Vrandečić, Sebastian Schaffert, Thomas Steiner, and Lydia Pintscher, *From freebase to wikidata: The great migration*, Proceedings of the 25th International Conference on World Wide Web (Republic and Canton of Geneva, CHE), WWW '16, International World Wide Web Conferences Steering Committee, 2016, p. 1419–1428.

[25] Lydia Pintscher, *Presentation Wikimania 2017: Bringing Lexicographical Data to Wikidata - Supporting Wiktionary and Beyond*, https://wikimania2017.wikimedia.org/w/index.php?title=File:Wikimania_2017_-_Bringing_Lexicographical_Data_to_Wikidata_-_Supporting_Wiktionary_and_Beyond.pdf, Aug 2020, [Online; accessed 20. Aug. 2020].

[26] Mohammed Sadat, *Wikidata Weekly Newsletter #435*, https://de.wikipedia.org/wiki/Wikipedia:Wikidata/Newsletter/Archiv/2020#Wikidata_weekly_summary_#435, Dec 2020, [Online; accessed 14. Dec. 2020].

[27] Corina Sandiuc and Alina Balagiu, *The use of crossword puzzles as a strategy to teach maritime english vocabulary*, Scientific Bulletin" Mircea cel Batran" Naval Academy **23** (2020), no. 1, 236A–242.

[28] Scholia, *Scholia information on Wikidata*, https://scholia.toolforge.org/topic/Q2013, Dec 2020, [Online; accessed 14. Dec. 2020].

[29] Josh Smith, *Patterns - WPF Apps With The Model-View-ViewModel Design Pattern*, https://docs.microsoft.com/en-us/archive/msdn-magazine/2009/february/patterns-wpf-apps-with-the-model-view-viewmodel-design-pattern, Feb 2009.

[30] StatCounter Global Stats, *Mobile and tablet internet usage exceeds desktop for first time worldwide*, https://gs.statcounter.com/press/mobile-and-tablet-internet-usage-exceeds-desktop-for-first-time-worldwide, Nov 2016, [Online; accessed 11. Dec. 2020].

[31] Luis von Ahn, *Games with a purpose*, IEEE Computer Magazine (2006), 96–98.

[32] Wikidata, *Lexeme:L9670 Wikidata*, https://www.wikidata.org/w/index.php?title=Lexeme:L9670&oldid=1054119918, 2019, [Online; accessed 1-October-2020].

[33] Wikimedia Contributors, *Wiktionary statistics*, https://meta.wikimedia.org/wiki/Wiktionary, May 2020.

[34] Wikimedia Foundation Inc., *Wikidata Usage and Coverage in WMF Projects*, https://wdcm.wmflabs.org/WD_percentUsageDashboard, Dec 2020, [Online; accessed 20. Dec. 2020].

[35] Wikipedia Contributors, *Lexeme*, https://en.wikipedia.org/w/index.php?title=Lexeme&oldid=946515735, 2020, [Online; accessed 1-October-2020].

[36] _____, *List of wikimedia foundation projects*, https://en.wikipedia.org/wiki/Wikimedia_Foundation#Wikimedia_projects, Oct 2020, [Online; accessed 20-December-2020].