ZeeGuu

A Platform for Second Language Acquisition Through Free Reading and Repetition

Bachelor's Thesis

Simon Marti

marti.simon@students.unibe.ch

Software Composition Group
Institute of Computer Science and Applied Mathematics
University of Berne

Supervisors:
Dr. Mircea F. Lungu
Prof. Dr. Oscar Nierstrasz

August 31. 2013

Acknowledgements

My thanks go to my supervisor *Dr. Mircea F. Lungu* who closely collaborated with me on this project.

Many other people helped me during the development of this project with their expert knowledge, support and feedback. I would like to thank all of them.

During the development of this project, I used some libraries of others to simplify my work. I'd specially like to thank the developers of the following openly licensed libraries:

1. flask, a microframework for Python-based web applications. [1]
2. Twitter Bootstrap, a sleek, intuitive, and powerful front-end framework for faster and easier web development. [2]
3. jQuery, a fast, small, and feature-rich JavaScript library. [3]

[1] `http://flask.pocoo.org/`
[2] `http://getbootstrap.com/`
[3] `http://jquery.org/`

Abstract

ZeeGuu aims to be the one and only online profile which monitors a user's vocabulary evolution as he is learning a new language. It seamlessly integrates into existing reading applications and provides distraction-free translations for a faster reading and learning experience.

ZeeGuu monitors a learner's progress and builds a model of what parts of the vocabulary are already mastered and what parts still need to be learned. The collected data is used to generate personalized exercises to repeat newly acquired knowledge and help retain it.

This thesis introduces the *ZeeGuu* ecosystem consisting of a central *Language Progress Model*, a *Language Gym* and *Augmented Readers*. It describes the overall architecture of the platform and provides proof-of-concept implementations for each component. It further shows how *ZeeGuu* can be brought onto other platforms and how it can otherwise be extended.

As a proof-of-concept implementation of an *Augmented Reader* this paper introduces an extension to the popular web browser *Google Chrome*.

Contents

Chapter 1

Introduction

This chapter introduces the background of this work. It gives a short overview of the ideas behind this project and shows some related work, mainly from existing applications.

1.1 Motivation

Learning a new language is a complex process and there are many resources, such as text books, exercises and vocabulary training systems, to aid in a better and more permanent language acquisition. In recent years many linguists such as *David Crystal* started to criticize the text books in particular for sentences which are often so far from reality, that they would never or rarely be used in real life situations. An example for such a sentence is the often used "The postilion has been struck by lightning". In a paper named after this exact sentence *David Crystal* notes that

> "Many sentences introduced in teaching seem to have little or no chance of ever being used in real life."[2]

Stephen Krashen among others concluded that one solution to this problem would be free voluntary reading. In an interview he noted:

> "Free voluntary reading during second-language acquisition, is the most powerful tool we have in language education."[5]

The obvious benefit of free reading is that reading material can be chosen according to personal interests and tastes. This can result in more focused reading and longer preservation of learned words. *Krashen* even argues that free reading alone can be enough to build a respectable vocabulary.[4]

Another more well-known fact about language acquisition is that repetition helps to retain newly acquired knowledge. It is however rather difficult to apply

this method to free reading as one might encounter many words only rarely and reading the same text multiple times defeats the purpose of voluntary reading altogether.

This thesis introduces a platform called *ZeeGuu*[1] which aims to help the learner to understand freely read texts and to retain acquired knowledge through repetition. Reading texts is simplified by providing fast and distraction free translations of unknown words to the reader, which allows him to read more complex texts without missing any of the meaning. *ZeeGuu* tracks the readers behaviour by recording read texts and the looked-up words in them. With this data *ZeeGuu* generates simple exercises for the reader to consolidate the acquired knowledge.

1.2 Related Work

There exist many applications providing translations on various platforms, including a wide range of browser extensions. However, to the authors knowledge none of them include any of the additional functionality provided by *ZeeGuu*. A few popular examples are listed here:

Google Dictionary (by Google)
: *Google Dictionary* provides definitions for double-clicked words and translates them into the user's preferred language if necessary.
`https://chrome.google.com/webstore/detail//mgijmajocgfcbeboacabfgobmjgjcoja`

Google Translate
: *Google Translate* translates entire websites into the user's preferred language.
`https://chrome.google.com/webstore/detail//aapbdbdomjkkjkaonfhkkikfgjllcleb`

Language Immersion for Chrome
: *Language Immersion* translates some words on pages in the user's preferred language into a foreign one. Language and skill level are adjustable.
`https://chrome.google.com/webstore/detail//bedbecnakfcpmkpddjfnfihogkaggkhl`

There also exist many vocabulary training tools, most of them use lists of frequently used words or let the user define his own. Some examples are listed here:

[1] The name refers to ziggurat[2] which refers to Babel which refersto the Babelfish[3] which was of a great inspiration for the project.
[2] Massive structures built by the Babylonians.
[3] A fictional universal translator in the form of a fish.[1]

PONS Vocabulary trainer
: The *PONS Vocabulary trainer* provides various methods for acquiring new vocabulary, including multiple-choice and character arrangement exercises
`http://trainer.pons.eu/`

babbel.com
: *babbel.com* offers multiple-choice exercises for many languages. They also offer the same functionality in free *iOS* applications.
`http://babbel.com/`

While *ZeeGuu* offers very similar functionality to some of these services, its defining factor is the seamless combination of vocabulary acquisition and repetition in one single platform.

Chapter 2

The ZeeGuu platform

This chapter gives a brief overview over *ZeeGuu*'s general architecture. The platform consists of three components: A central *Learning Progress Model*, a *Language Gym* and possibly multiple *Augmented Readers*. All of these components are further explained in the following sections.

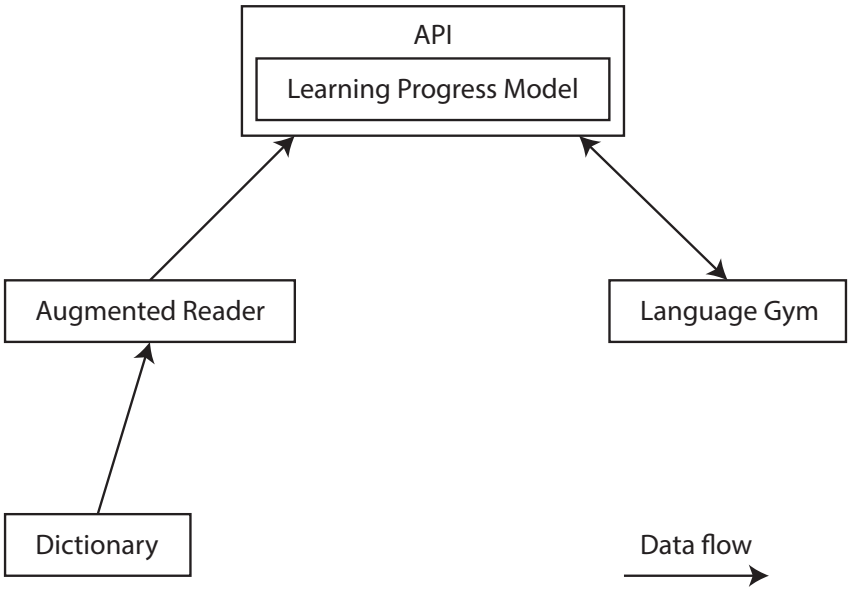

Figure 2.1: Architecture of *ZeeGuu*.

2.1 Augmented Reader

Augmented Readers are applications which allow the user to consume content in a foreign language and offer interruption-free translations for unknown words.

They further send usage data to the *Learning Progress Model* in order to form a *Learning Progress Profile* for the user.

An *Augmented Reader* can be a standalone application or extend an existing one. A few examples of applications which could be enriched this way are listed here:

Web Browser
Web Browsers are among the most used applications for many platforms and are mainly used to consume content. Many of them are in some way expandable, making them the perfect target for an *Augmented Reader*. Popular examples include *Google Chrome*[1], *Firefox*[2] and *Safari*[3].

eBook Reader
eBook Readers, such as *Apple*'s *iOS* application *iBooks*[4] or *Amazon*'s *Kindle*[5] provide access to thousands of books and offer a similar reading experience as an actual book.

Email Client
While many users access their emails though a *Web Browser*, many use a dedicated *Email Client*, such as *Thunderbird*[6] by *Mozilla* or *Outlook*[7] by *Microsoft*. Written conversations in a foreign language are a great way to learn and an *Augmented Reader* could be of benefit if the conversation partner occasionally uses complex words.

2.2 Learning Progress Model

The *Learning Progress Model* stores user data provided by *Augmented Readers* and grants the *Language Gym* access to said data. The latter can access the *Learning Progress Model* directly, the *Augmented Readers* on the other hand access it through an *Application Programming Interface* (API) to authenticate users and store their data. This allows *ZeeGuu* to be easily extended onto other platforms.

Each time a user looks up a word in an *Augmented Reader* the *Learning Progress Model* stores the looked up word, the time, the context and, if provided, the appropriate translation in the user's *Learning Progress Profile*.

[1] `http://chrome.google.com/`
[2] `http://getfirefox.com/`
[3] `http://apple.com/safari`
[4] `https://itunes.apple.com/en/app/ibooks/id364709193`
[5] `https://kindle.amazon.com/`
[6] `http://www.mozilla.org/thunderbird/`
[7] `http://office.microsoft.com/outlook/`

2.3 Language Gym

In the *Language Gym* the user can solve a number of exercises based on his *Learning Progress Profile*. Its goal is to retain the knowledge acquired with free reading through repetition. Whether the user solves an exercise in the *Language Gym* successfully or not is recorded in his *Learning Progress Profile* and affects future exercises.

Possible exercise types include word translations, multiple choice questions, fill-the-gap sentences, picture-to-word association exercises, etc. The goal is to build *ZeeGuu* in such a way that new exercise types can easily be built onto the *Learning Progress Model* (see chapter 7).

Chapter 3

Google Chrome

Google Chrome[1] is a *Web Browser* developed by *Google*[2] and released to the public in 2008. It is based on the open-source project *Chromium*[3] which was created, and is still being maintained, by *Google*. *Chromium* uses *Blink*[4] as its rendering engine which is a fork of the popular *WebKit*[5] project.

Google Chrome is the most popular *Web Browser* among desktop users according to various statistics[6,7,8,9]. Like many other *Web Browsers Google Chrome* allows users to modify its behaviour through custom applications called *extensions*. These *extensions* can introduce new functionality in the form of JavaScript (an implementation of ECMAScript[10]) scripts which are injected into visited websites and which have access to various resources, such as HTML pages and images, bundled with the *extension*. *Extensions* can be uploaded to and installed from the *Chrome Web Store*[11] where thousands[12] of *extensions* are available.

[1] http://chrome.google.com/
[2] http://google.com/
[3] http://dev.chromium.org/Home
[4] http://www.chromium.org/blink
[5] http://www.webkit.org/
[6] http://gs.statcounter.com/
[7] http://clicky.com/marketshare/global/web-browsers/
[8] http://www.w3counter.com/globalstats.php
[9] http://stats.wikimedia.org/archive/squid_reports/2013-06/SquidReportClients.htm
[10] http://www.ecmascript.org/
[11] https://chrome.google.com/webstore
[12] http://blog.chromium.org/2010/03/google-chrome-developer-update-3000.html

Chapter 4

Goals

The goal of this thesis is to build a proof-of-concept implementation for every part of the *ZeeGuu* ecosystem. Specifically a *Learning Progress Model* capable of storing word searches for individual users, a *Language Gym* providing simple exercises based on the user's *Learning Progress Profile* and one example of an *Augmented Reader*.

The *Learning Progress Model* should be available on a central server and accessible through a simple *Application Programming Interface* (API) to *Augmented Readers*.

The *Language Gym* should be built alongside the *Learning Progress Model* and provide very simple exercises, based directly on word searches done by the user.

The *Augmented Reader* is an *extension* to *Google Chrome* and should allow the user to translate selected words into a configurable language on any website. Further it should update the *Learning Progress Profile* through the *Learning Progress Model*'s *API*.

Chapter 5

Realization

The following sections highlight more details of the specific realization of every part of *ZeeGuu*.

5.1 Learning Progress Model

The main task of the *Learning Progress Model* is to store data about every user. It is implemented using the *sqlalchemy*[1] library, which is an open *object-relational mapping* (ORM) framework for the programming language *Python*[2]. The *Learning Progress Profiles* can therefore be stored in a number[3] of different database systems, including *MySQL*[4], *SQLite*[5], *PostgreSQL*[6] and many others.

The *Learning Progress Model* stores, besides the user credentials and session information, word searches, contributed translations, read words and completed exercises. The database structure is included in the appendix.

5.2 Application Programming Interface

The *Application Programming Interface* (API) enables access to the *Learning Progress Model* through a RESTful[3] *HTTP* interface. The *API* must be capable of these tasks:

1. Add new users.
2. Authenticate existing users.

[1] `http://www.sqlalchemy.org/`
[2] `http://www.python.org/`
[3] `http://docs.sqlalchemy.org/en/rel_0_8/dialects/index.html`
[4] `http://www.mysql.com/`
[5] `http://www.sqlite.org/`
[6] `http://www.postgresql.org/`

3. Store word searches.

4. Contribute appropriate translations.

5.2.1 Request Format

Every request to the *API* is done as a `POST` request to one of the endpoints described below. If it was successful, the server responds with an *HTTP* status `200 OK`, otherwise an appropriate *HTTP* error status code is returned. Endpoints which require a user to be authenticated expect a valid *session ID* in the `GET` parameter `session`. Some endpoints allow or require some fields to be submitted as `POST` fields, those are expected to be included in the message body as a `application/x-www-form-urlencoded` string.

5.2.2 Add User

Endpoint: `/adduser/<email>`

Adding a new user requires an email address and a password. The former is provided directly in the *URL*, the latter has to be sent as the `POST` field `password`, for security reasons[7]. To protect users in case the database ever gets compromised the passwords are not stored in clear-text, but salted[8] and hashed multiple times. This makes it close to impossible to derive the original password from the stored hash.

This endpoint returns a *session ID* for the new user which can be used to submit word searches and contributions to the user's *Learning Progress Profile*.

5.2.3 Authenticate User

Endpoint: `/session/<email>`

This endpoint works very similar to the one described above. It requires an email address and a password and the fields are submitted in the exact same way. However, instead of creating a new user this endpoint authenticates an existing one by comparing the email address and hashed password to the entries in the database. If no valid user can be found the *HTTP* status code `401 Unauthorized` is returned.

If successful this endpoint returns a new *session ID* for the requested user. Any *session IDs* requested earlier stay valid to allow the user to access his *Learning Progress Profile* from multiple clients.

[7] `GET` parameters can show up in web server logs, which defeats the point of hashing the password.
[8] A salted password is padded with random bytes to make decryption significantly harder. [7]

5.2.4 Word Lookup

Endpoint: `/lookup/<origin language>/<word>/<target language>`

Augmented Readers use this endpoint every time a user looks up a word. Note that the *API* does not translate the word, it merely registers the event. The actual translation is done by the *Augmented Reader* as described in section 5.3.

Languages are represented according to ISO 639-1:2002[9] as a two character wide, lower case string.

This endpoint accepts an optional `POST` field `text` which is the context of the looked-up word. The context should be limited to a paragraph or sentence as every word in it will be marked as being read by the *Learning Progress Model*.

5.2.5 Contribute Translation

Endpoint: `/contribute/<origin language>/<word>/<target language>/<word>`

Since the *Learning Progress Model* does not know the translations to any of the words submitted through the *Word Lookup* endpoint (see section 6.1) all *Augmented Readers* should allow the user to select the translation he personally believes is the most appropriate and submit through this endpoint. The submitted translations can later be used by the *Language Gym* to generate personalized exercises.

This endpoint does not accept any more fields than the ones present in the *URL*; the context should already have been submitted through the *Word Lookup* endpoint.

5.2.6 Implementation

The *API* is implemented as a *Python*[10] *flask*[11] application. *flask* is a microframework based on the *HTTP* library *Werkzeug*[12] and the templating engine *Jinja2*[13]. It allows fast development of small web applications and is therefore a good choice for this *API*.

At the time of this writing the *API* is available at `http://zeeguu.unibe.ch/`. The *Language Gym* and the *ZeeGuu Chrome Extension* are also available at this address. The source code of the *API* as well as the *Language Gym* is available on *GitHub*: `https://github.com/ZeeGuu/API`

[9] `http://www.iso.org/iso/home/standards/language_codes.htm`
[10] `http://python.org/`
[11] `http://flask.pocoo.org`
[12] `http://werkzeug.pocoo.org/`
[13] `http://jinja.pocoo.org/`

A *Step-by-Step Installation Guide* for the *ZeeGuu API* is provided as supplementary documentation to this thesis in [6].

5.3 Augmented Reader

The *Augmented Reader* provided by this thesis is an *extension* to *Google Chrome* with the following features:

1. Authentication of the user
2. Translating words on websites through the context menu
3. Translating words on websites by double-clicking (optional)
4. Translating words through a toolbar icon
5. Contributing appropriate translations
6. Disabling hyperlinks for easier selection
7. Synchronizing settings with the user's *Google* account.
8. Customize the used dictionary service

5.3.1 Translation

As hinted at in figure 2.1 the *ZeeGuu extension* uses a third-party dictionary service to provide translations to the user (see also section 6.1). By default the service *dict.cc*[14] is used which supports many European languages as well as English[15].

5.3.2 User Interface

To allow *ZeeGuu* to work as distraction-free as possible it displays the dictionary in an inline frame onto the website the user is translating a word from (fig. 5.1). In case the word would be covered up by said frame the page is scrolled up until the word becomes visible again. When the user uses *ZeeGuu* for the first time he is asked to authenticate or create a new account (fig. 5.2) and is shown the requested translations immediately afterwards.

The *ZeeGuu* extension works in three different modes:

[14] `http://dict.cc/`
[15] `http://browse.dict.cc/`

Default mode
In this mode the user has to activate *ZeeGuu* through an option in the context menu which becomes available if text is selected.

Fast Mode
In this mode double-clicking a word is enough to trigger *ZeeGuu*. To translate multiple words the context menu can still be used.

Selection Mode
To allow the translation of links this mode disables all of them until a selection is made. This mode automatically falls back to whatever mode was active before as having it active permanently would make little sense.

The user can switch between these modes in a pop-up available through an icon in *Chrome*'s toolbar (fig. 5.4). There the user can also translate words not currently present on the displayed website. To further configure the *ZeeGuu extension* the pop-up window offers access to an options page (fig. 5.3) which can also be accessed through *Chrome*'s preferences panel.

The source code of the *ZeeGuu extension* is available on *GitHub*:
`https://github.com/ZeeGuu/browser`

5.4 Language Gym

The *Language Gym* (fig. 5.5) developed for the purpose of this thesis is meant to be a simple example of the possibilities *ZeeGuu* offers. It generates very simple flashcard-based exercises from the contributions a user has made. These exercises always display a single word and ask a user for its translation. In which language the words are displayed can be specified by the user. Correctly solved exercises will be shown less often and after one has been solved successfully a number of times it won't ever be shown again.

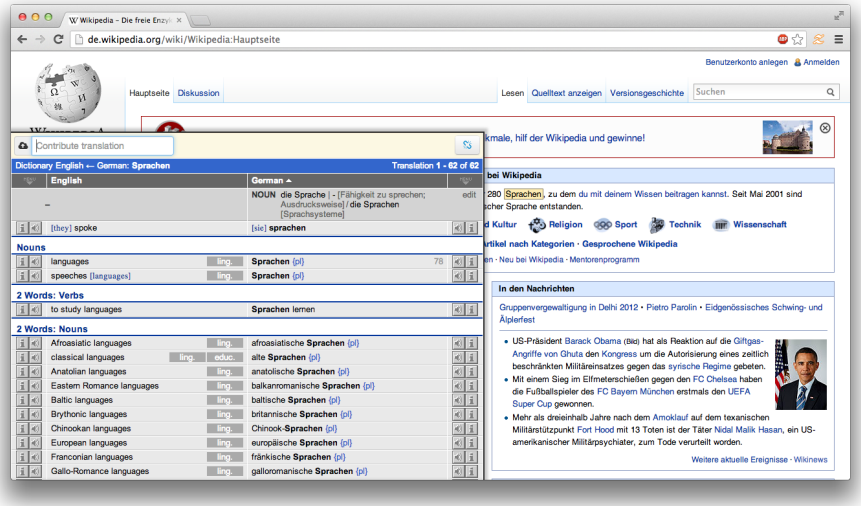

Figure 5.1: Displaying translations for the German word “Sprachen”.

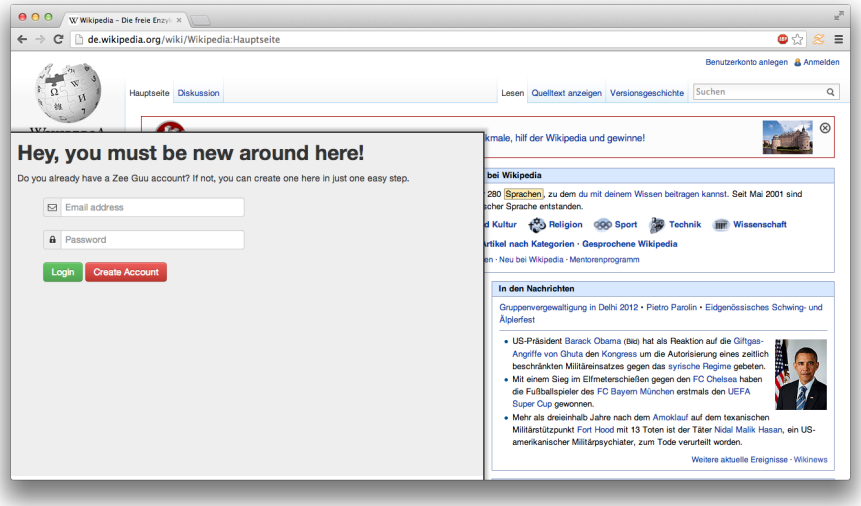

Figure 5.2: Asking the user to authenticate.

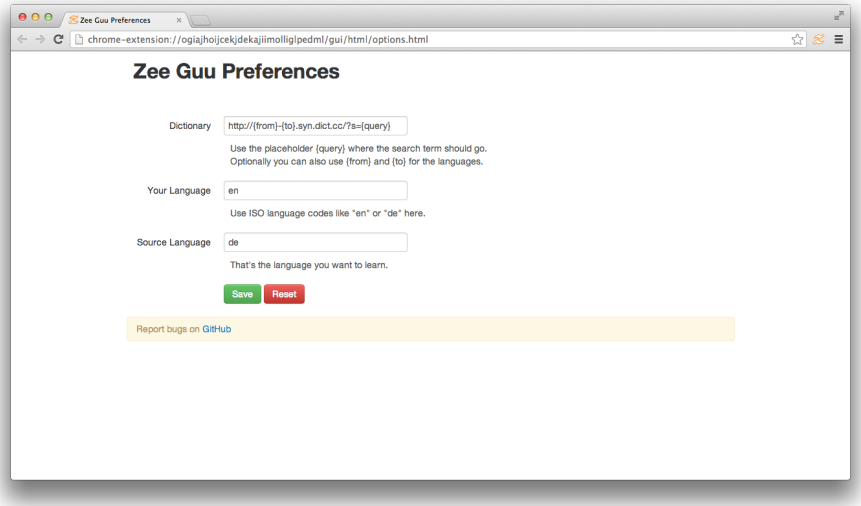

Figure 5.3: *ZeeGuu*'s options

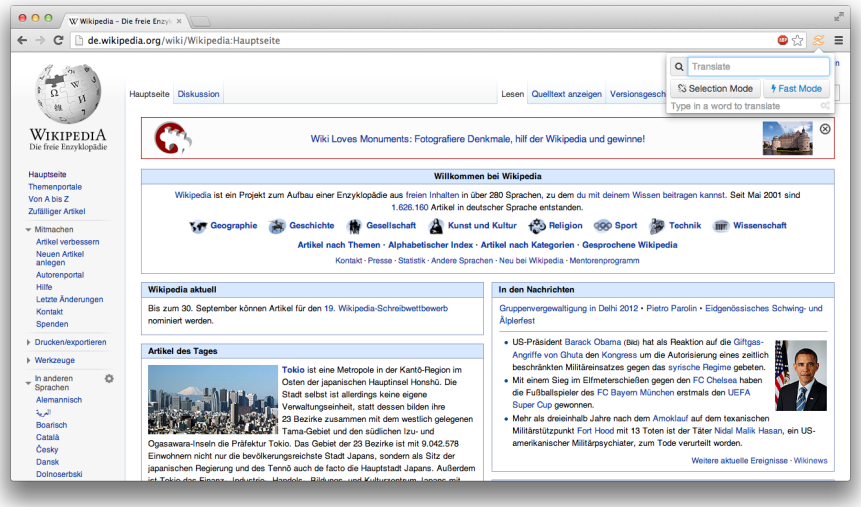

Figure 5.4: *ZeeGuu*'s toolbar pop-up window

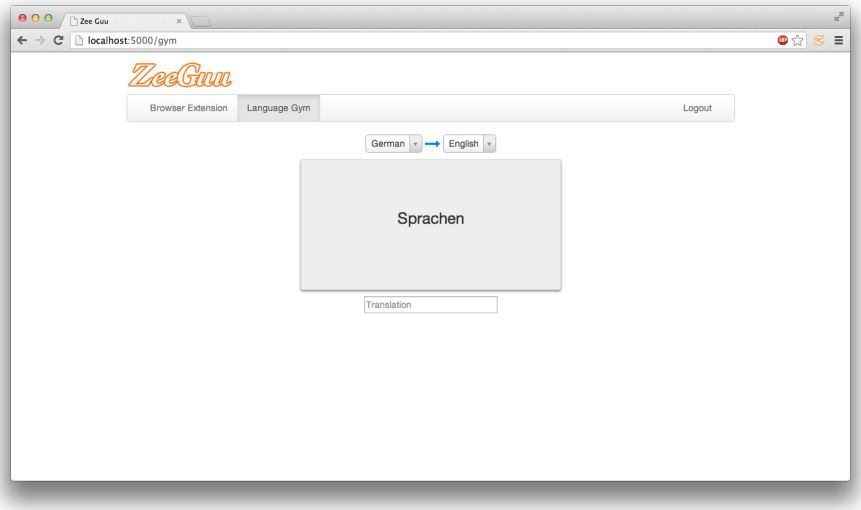

Figure 5.5: The *Language Gym*

Chapter 6

Challenges

The following sections highlight a few of the challenges faced during the development of the *ZeeGuu* platform and its individual components.

6.1 Dictionary

The initial intention for *ZeeGuu*'s overall design was that the *Learning Progress Model* included a dictionary and could provide the *Augmented Readers* with translations. This has proven to be very difficult as almost no dicitonaries are freely available for this use-case. Some seemed viable at first, however all of them failed to translate compound words or complex conjugations of verbs. Another problem was the huge impact on the database these dictionaries had. The *Oxford English Dictionary*[1] contains 171,476 words in current use[2], not including derivative words. Since translations are not a relationship between words but meanings, the number of needed entries could be close to three quarters of a million.

There exist however a number of web-based dictionaries which can be used for free by private persons. This led to the solution of displaying a third-party dictionary to the user directly from the *Augmented Reader*. This has the added benefit that users can use whatever dictionary they like, allowing them to use *ZeeGuu* in languages which would otherwise not be supported. As a default *dict.cc*[3] was chosen as it supports many of the languages spoken in Europe and even offers a version explicitly designed to be used in a frame (fig. 5.1).

[1] `http://oxforddictionaries.com/`
[2] `http://oxforddictionaries.com/words/how-many-words-are-there-in-the-english-language`
[3] `http://dict.cc/`

6.2 Security

Many *APIs* of popular online services such as *Facebook*[4], *Twitter*[5] and *GitHub*[6] use *OAuth*[7] to authenticate users. If implemented properly *OAuth* provides protection against a number of known exploit vectors, such as *Man-in-the-middle* attacks, *replay* attacks and *timing* attacks.

However, the *OAuth* protocol was under heavy development until 2010 which resulted in many libraries not being compatible with each other, lacking features or being vulnerable to certain kinds of attacks. Many client-side libraries are built specifically to work with one of the popular *APIs*, making them incompatible with server-side libraries which implement the latest *RFC* specification. Some libraries also expect the user to implement some sensitive methods himself, in an attempt to make integration into an existing application easier. This can however result in vulnerabilities to *timing* attacks if not done with extreme care.

Since the data exchanged by *ZeeGuu* is not very sensitive this challenge was solved by using a much simpler solution based on unique *session IDs* as outlined in section 5.2.

6.3 Same-Origin Policy

As described in chapter 3, extensions to *Google Chrome* inject custom *JavaScript* scripts into visited websites. These scripts run in the context of the website and are therefore subject to the *Same-Origin Policy*[8]. This means a script can inject an inline frame into a website, but if the domain name of the page displayed inside the frame does not match the one of the parent website, the script won't be able to access the created frame. This is an issue as the *ZeeGuu extension* displays its user interface in an inline frame which in turn displays the dictionary in another one. This is further complicated if the visited website already includes frames from different domains. The pop-up window displayed by *ZeeGuu*'s toolbar icon runs yet in another context, making it hard to trigger events, such as a mode change, in the displayed website.

Google Chrome offers an elaborate solution to this problem: A special page which runs persistently in the background at all times. These pages are called *background pages*[9] and allow *extension* to keep a shared state across all open windows and tabs. The background page can exchange information in the form

[4] `https://facebook.com/`
[5] `https://twitter.com/`
[6] `https://github.com/`
[7] RFC specification for OAuth 1.0: `http://tools.ietf.org/html/rfc5849`
[8] `http://www.w3.org/Security/wiki/Same_Origin_Policy`
[9] `http://developer.chrome.com/extensions/background_pages.html`

of messages[10] with injected scripts, the pop-up window and the toolbar icon. These messages can contain *JavaScript* objects which allows the *ZeeGuu extension* to share the complete application state with every injected script whenever it changes.

[10] `http://developer.chrome.com/extensions/messaging.html`

Chapter 7

Future Work

This thesis provides an easily accessible and expandable *Learning Progress Model* and sample implementations of both the *Language Gym* and an *Augmented Reader*. It is meant to be improved and expanded upon by new contributors as well as the original author. This chapter discusses a few possible improvements and expansions.

7.1 Possible Improvements

7.1.1 Improve the Learning Progress Model

All the *Learning Progress Model* currently does is storing data and making it directly available to the *Language Gym*. It could be improved by processing the gathered data into an overall language skill level, interests and more. This data could in turn be used by the *Language Gym* to generate more elaborate exercises and possibly introduce new words the user might need in the future.

7.1.2 Improve the ZeeGuu Extension

The *ZeeGuu extension* for *Google Chrome* already offers many features, it could however still be improved in many ways. One feature suggested by early users is the ability to translate words in-place; that is replacing the selected word directly with a translation instead of showing a window. The user could cycle through multiple possible translations by simple double-clicking the word again. This feature would allow *ZeeGuu* to work with even less interruption of the reading process.

Another possible feature is a predefined list of dictionaries of which a user can chose. This would make it much easier for inexperienced users to change the used dictionary. It could also make the search for dictionary services in certain languages obsolete.

7.1.3 Improve the Language Gym

The *Language Gym* currently only offers one single type of exercises and a very simple way of processing the results. It could easily be extended to generate a wider range of exercises (see section 2.3) and in combination with an improved *Learning Progress Model* could even introduce new words to the user. The generated exercises could also include the context a word was encountered in, for example as a fill-the-gap exercise.

7.2 Extensions

ZeeGuu is meant to be available on as many platforms as possible. This thesis introduces an extension to the popular *Web Browser Google Chrome* which already targets a relatively large portion of content consumption. There are however many other platforms which would benefit from an *Augmented Reader*; a few are listed here:

Other Web Browsers
: *Google Chrome* might be the most used *Web Browser*, but others, such as *Firefox*, have a respectable user base as well. The *ZeeGuu extension* is built with this in mind and can easily be ported to *Web Browsers* with a similar extension system as *Google Chrome*.

iOS and Android Application
: *Apple*'s *iOS* and *Google*'s *Android* operating system are dominating the mobile platform and are a good target for *Augmented Readers*. As these operating systems generally don't allow applications to be modified by a third party these *Augmented Readers* would be required to implement the management and display of content themselves, for example in the form of an *eBook Reader*.

When *Augmented Readers* are available on mobile platforms it would make sense to build stand-alone applications for the *Language Gym* or integrate it directly into an *Augmented Reader* application. This would however require the *API* to be extended.

Chapter 8

Conclusion

The *ZeeGuu* ecosystem provides a number of tools, helping users to expand their vocabulary through free reading and retaining acquired knowledge through repetition exercises. It is built to be easily expandable by additional *Augmented Readers* to collect user data from as many platforms as possible.

The *ZeeGuu extension* to *Google Chrome* targets not only one of the most used content-consuming applications but also one with the widest range of available content. It simplifies reading texts in foreign languages significantly, especially if they contain complex and unknown words.

In the *Language Gym* the user can train a vocabulary that is personalized to his interest through the central *Language Progress Profile*. It can easily be extended to offer a wider variety of exercises to make the learning process more interesting and rewarding.

As a whole the *ZeeGuu* platform can help to a faster and longer lasting language acquisition.

Bibliography

[1] Douglas Adams. *The Hitchhiker's Guide to the Galaxy.* Pan Books, 1979.

[2] David Crystal. Postilion sentences. *Journal of Clinical Speech and Language Studies*, 1995.

[3] Roy Thomas Fielding. *Architectural Styles and the Design of Network-based Software Architectures.* PhD thesis, University OF California, 2000.

[4] Stephen D. Krashen Jeff McQuillan. Can free reading take you all the way? a response to cobb. `http://llt.msu.edu/vol12num1/mcquillan/default.html`, 2007.

[5] Stephen D. Krashen. Achievement profile. `http://www.eslminiconf.net/september/krashen.html`, 2002.

[6] Simon Marti. Zeeguu - step-by-step installation guide. Supplementary documentation, University of Berne, August 2013.

[7] Ken Thompson Robert Morris. Password security: A case history, 1978.

Appendix A

Appendix Chapter

A.1 Database Structure

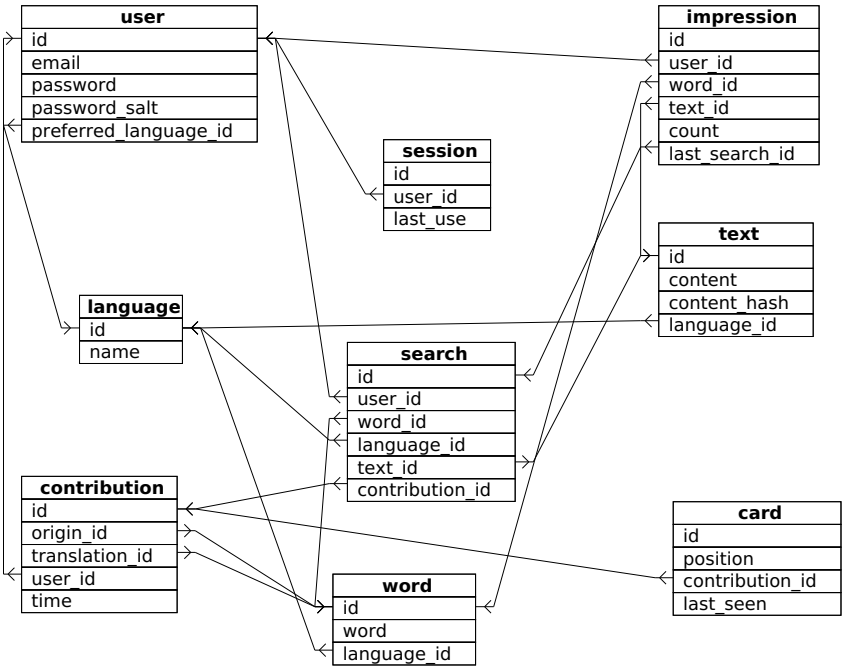

Figure A.1: Database structure of the *Learning Progress Model*