Factual Knowledge Adaptive Learning System for Geography

Diploma Thesis

Vít Stanislav

Brno, spring 2015
Declaration

Hereby I declare, that this paper is my original authorial work, which I have worked out by my own. All sources, references and literature used or excerpted during elaboration of this work are properly cited and listed in complete reference to the due source.

Advisor:  doc. Mgr. Radek Pelánek, Ph.D.
Acknowledgement

I would like to thank my supervisor, doc. Mgr. Radek Pelánek, Ph.D., for his guidance, enthusiasm and valuable advices. I thank to my colleagues from Adaptive Learning Group (Jan Papoušek in particular) for their enjoyable cooperation on development and insightful discussions. I thank to all who tested the developed system and provided some feedback.
Abstract

This thesis deals with design, implementation, deployment and usage analysis of a factual knowledge adaptive learning system for geography – slepemapy.cz. The thesis puts the system into the context of systems for factual knowledge learning (flashcard software) and systems for learning geography. It also describes main system parts and explains their function. The system is implemented as a web application with client-server architecture. Django framework is used on server and AngularJS framework on client. The implemented web system is widely used and is a source of a big data set used for research.
Keywords

adaptive practice, online education, geography, outline maps, flashcard software, Python, JavaScript
Contents

1 Introduction ......................................................... 2
2 Similar Systems ...................................................... 4
  2.1 Flashcard Software ........................................... 4
  2.2 Geography Games ............................................. 8
3 System Functionality .............................................. 11
  3.1 Server-side Functionality ..................................... 11
  3.2 Client-side Functionality .................................... 15
4 Implementation ..................................................... 21
  4.1 Project Timeline ............................................... 21
  4.2 Used Technologies ............................................. 22
  4.3 Map-related Functionality .................................... 26
  4.4 Other Functionality ........................................... 28
5 System Usage Analysis ............................................ 31
  5.1 Used Technologies ............................................. 31
  5.2 Users of the System .......................................... 32
  5.3 User Registration A/B Experiments ......................... 36
  5.4 User Feedback .................................................. 38
6 Subsequent Related Work ......................................... 41
  6.1 Derived Systems .............................................. 41
  6.2 Research on the Collected Data .............................. 43
7 Conclusion .......................................................... 45
Bibliography ......................................................... 46
Appendices ............................................................ 49
A Source code ......................................................... 49
B Visualizations ....................................................... 50
Chapter 1

Introduction

In recent decades computers started to occupy more and more areas of human activities and education is no exception. There are many different forms of e-learning, such as learning management systems, (intelligent) tutoring systems, massive open online course platforms or educational games. Students can acquire knowledge in various forms and formats, ranging from text and images through audio and video to highly interactive applications or games.

In some cases students can also practice the knowledge and get feedback on their performance. Computers can easily give feedback on some tasks, such as learning elementary mathematics operations or learning foreign language vocabulary, whereas in the case of more creative tasks human feedback is still superior to feedback generated by computer (e.g. evaluation of an essay or an advanced mathematics proof). However, in the areas where computer can evaluate students’ knowledge, computer outperforms human in being more patient and available at any time. Computer can also recommend material for further study and provide learning that is customized to an individual student’s needs, as opposed to one-pace-fits-all learning style in traditional classroom.

Adapting to individual students is an important feature of the learning system developed in this thesis. The system allows students to practice association between a feature on a map (e.g. a country) and its name. The system is developed in collaboration with another student, Jan Papoušek, who is responsible for developing server-side features (i.e. student model and question selection algorithm), while I am responsible for user-facing features and preparing the maps.

The primary goal of this thesis is to develop part of the system as a web application. Other goals of the thesis are: to provide an overview of relevant similar systems and their underlying principles, to analyze the data col-
lected by the implemented system, and to outline modifications of the im-
plememented system for use in other areas.

The thesis consists of 7 chapters. Chapter 2 describes various features of
systems similar to the developed system. Chapter 3 describes functionality
of the whole system, that is regardless of who implemented it. Chapter 4
describes particular issues of author’s part of implementation. In Chapter
5 are described some experiments with the data collected by the system.
Chapter 6 mentions other research and development that was done thanks
to the implemented system.
Chapter 2

Similar Systems

The proposed system is essentially a combination of two common types of applications: flashcard software and geography games. This chapter contains a detailed description of the important features of both types of applications.

2.1 Flashcard Software

Flashcard software is a computerized version of physical (paper) flashcards. The basic idea of paper flashcards is that students write a small chunk of information that they want to learn on one side of a flashcard and a cue to recall the information on the other side (e.g. when learning Spanish vocabulary: “una mujer” / “a woman”). Then they learn by trying to recall one side while looking at the other and then checking if they were right by flipping the flashcard.

There are many different flashcard applications available; however they all share the following basic use case:

1. User chooses a topic to learn and creates or selects a set of flashcards covering the topic.

2. User practices the flashcards, i.e. user is asked one side of a flashcard and is supposed to recall the content of the other side.

3. After answering, the flashcard is scheduled for next practice. The interval between the current and next practice is based of the firmness of user’s knowledge of the flashcard content.

In the rest of this section are described common features of flashcard software illustrated on sample systems listed in Table 2.1.
### Table 2.1: Sample flashcard software

<table>
<thead>
<tr>
<th>Name</th>
<th>URL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Anki</td>
<td><a href="http://ankisrs.net/">http://ankisrs.net/</a></td>
</tr>
<tr>
<td>IS MUNI Dril</td>
<td><a href="https://is.muni.cz/auth/dril/">https://is.muni.cz/auth/dril/</a></td>
</tr>
<tr>
<td>Memrise</td>
<td><a href="http://www.memrise.com/">http://www.memrise.com/</a></td>
</tr>
<tr>
<td>Cerego</td>
<td><a href="http://cerego.com/">http://cerego.com/</a></td>
</tr>
</tbody>
</table>

#### Practicing Flashcards

The general workflow of practicing flashcards is shared among all the flashcard systems; however, the systems vary in some details of the practice. In some systems it is up to the users to evaluate their knowledge of the other side (Anki [8] and IS MUNI Dril [1]). Then they choose to see the other side of the flashcard and select one of the options (4 in Anki, 6 in Dril) describing their level of knowledge (ranging from “Very poor” to “Very good”). The system schedules next practice of the flashcard based on the selected option. This is a workflow that can also be used with paper flashcards, which is not the case of some other systems.

In other systems (Memrise and Cerego), the user is required to type the answer (or choose it from several options) and the system evaluates its correctness. The biggest disadvantage of this approach is that typing the answer takes more time than saying it (especially when typing on a smartphone). On the other hand, it is likely to measure knowledge more accurately as users tend to overestimate their knowledge when assessing themselves [6].

An important part of flashcard software is a scheduling algorithm. The algorithm determines when it is the best time to practice a flashcard again. It is usually based on well studied phenomena such as spacing and forgetting effects [27] and spaced repetition [14]. Considering these phenomena leads to increasing the interval between two consequent practices of a flashcard. How large is the increase depends on how firm is the user’s knowledge thought to be.

For example, learning algorithm of Cerego consists of learning module and review module [29]. The learning module asks students repeatedly questions about a small group of items for as many times as it is necessary to make the correct responses automatic (response time around 500 ms).
Then the review module schedules the spacing of reviews of the items to maximize spacing effect.

**Feedback for Users**

There are various approaches to provide users with information about their learning progress. Users might want to know “how many items do they know”, “how well do they know the items” or “how much time did they spend on studying”. This information can help users choose what to learn next and motivate them to continue learning.

Anki is capable of displaying various graphs of how many cards have been learned. Most of them are bar charts with bar per day over some period of time, where bars represent number of different states of cards or time spent on learning (e.g. see Figure 2.1). There also is a pie chart depicting the division of cards among the possible states (learn, young, mature or relearn, suspended, unseen).

![Figure 2.1: Anki Review Count: The number of answered questions](image)

Memrise visualizes users’ knowledge as plants (see Figure 2.2). At the beginning of learning a plant is planted; then the knowledge/plant has to be revised/watered in certain time intervals in order to bloom, otherwise the knowledge/plant withers. This metaphor allows Memrise to visualize the states of the plants; the same visualizations are used for groups of cards called levels (levels are parts of a course). Memrise also displays progress bars of learned words in a level or a course.

Cerego visualizes user’s knowledge by plotting the items on a 2D scatter plot (see Figure 2.3), where vertical axis roughly represents short-term memory and horizontal axis long-term memory. After practicing an item
2. Similar Systems

Figure 2.2: Memrise knowledge visualization

it moves to the top and slightly to the right. Over time (without practice) the item moves down. Horizontally every item begins as “unstarted” on the left; after each practice it moves to the right (through level 1 to 4); The goal is to reach the “Mastery zone” on the right.

Figure 2.3: Cerego knowledge visualization

Memory Encoding

In order to improve memory encoding of the learned items, various additional information can be provided with the learned items. It might be examples, interesting facts, amusing associations or mnemonics. Some systems provide very good support for memory encoding enhancement (e.g. Memrise), some systems allow this to be done with more general features (e.g. Anki) and other do not provide any tools for this at all (e.g. IS MUNI Dril).

For example, Memrise calls its memory encoding enhancement feature “mems” [19] and provides tools to crowd source them. More specifically,
every user can create mems and vote on the mems created by others. When practicing an item for the first time, user is assigned the highest voted mem (or can choose any other mem). This approach (as any other use of crowd sourcing) requires many users to be studying the same course, since only a small fraction of users is willing to spend time on creating mems. However, in my experience, high quality mems can be a great help for example when learning Chinese characters.

Even if a system does not support mems explicitly, there are often some ways around it. For example, Anki allows creation of more than two-sided cards. Three-sided cards can be defined such that the third side is never a part of the cue but always a part of the answer, while first and second side can be cue and answer in either direction. That is exactly what is needed for a mnemonic.

**Flashcards for Learning Maps**

Most flashcard systems support flashcards with images, which makes the systems capable of being used for learning locations of countries in maps. A Flashcard has to contain an image of an outline map with one item highlighted in a different color. For example, for Anki there is a shared deck “Countries of the World” and many others; Memrise also offers many map courses; last but not least, Cerego offers “Countries of Europe” and many others.

Static images are a reasonable way of practicing countries in a general purpose learning system; however, a system with native map support can offer better capabilities. In particular, the system can ask questions like “From all the countries on the map choose Mongolia” (as opposed to “Which of these four pictures represents Mongolia?”) and evaluate the answer.

### 2.2 Geography Games

There are many applications for practicing maps; however, they usually pay no attention to question selection and ask the questions either in fixed order or at random. These games also do not save answers and thus users’

---

1. [https://ankiweb.net/shared/info/2915332392](https://ankiweb.net/shared/info/2915332392)
3. [https://cerego.com/sets/721559](https://cerego.com/sets/721559)
knowledge cannot be tracked. This section describes three sample geography games listed in Table 2.2.

<table>
<thead>
<tr>
<th>Name</th>
<th>URL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sheppard Software</td>
<td><a href="http://www.sheppardsoftware.com/Geography.htm">http://www.sheppardsoftware.com/Geography.htm</a></td>
</tr>
<tr>
<td>geography games</td>
<td></td>
</tr>
<tr>
<td>World Geography Games</td>
<td><a href="http://world-geography-games.com/capitals_europe.swf">http://world-geography-games.com/capitals_europe.swf</a></td>
</tr>
<tr>
<td>geography games</td>
<td></td>
</tr>
</tbody>
</table>

Table 2.2: Sample geography games

A typical example is geography games on Sheppard Software. For most maps they offer three types of exercises:

- Level 1 – click on any item to learn its name,
- Level 2 – click on the item called X,
- Level 3 – drag the outlines from a stack and drop them to their locations on the map (easy when the coastline matches part of the shape).

Another example of geography game is World Geography Games. The questions are of “click on the item called X” type. After an incorrect answer the user has to try again as there is no option to skip the question, which can be very frustrating. The answered items are labeled on the map and the labels persist till the end of the practice. Users can practice countries, capitals, flags and 16 other map types (e.g. lakes, rivers or mountains).

A less common approach to practice is used in geography games at Digital Dialects. The type of questions is also “click on the item called X”, but the questions are asked in triplets. In each triplet the first question has three options (highlighted among all items); the other two options remain for the second question and the third question has only one option left (see Figure 2.4). After answering all tree questions the answers are evaluated and user can proceed to the next triplet of question. Another special feature of this system are buttons next to the items to click on instead of clicking on the items (e.g. countries) themselves. It is a reasonable solution for too small items, but it is used for all items, which seems as if it was chosen because it was easier to implement.
Figure 2.4: Digital Dialects geography games – Countries of Asia.
Chapter 3

System Functionality

This chapter describes system capabilities; it focuses on introducing the main ideas and avoids implementation details. The system developed in this thesis differs from the aforementioned flashcard systems mainly in two aspects:

- The system estimates global difficulty of each item across all users and uses it to estimate users’ prior knowledge of items.
- One side of a flashcard is an element in an SVG image (as a better alternative to pre-rendered raster images uploaded to a general purpose flashcard system).

The system is divided into several modules and uses client-server architecture (see Figure 3.1). The server-side module “Skills/difficulty estimation” estimates difficulty of items and skills of users based on previous answers. The estimates (together with other information about previous answers) are used by the “Question construction” module to choose appropriate questions for “Practice”. The estimates of users’ skills are also presented to users in “Knowledge map” and “Maps overview”.

3.1 Server-side Functionality

The description of the server-side modules, i.e. knowledge estimation and question selection, (see Figure 3.1) is given in EDM paper “Adaptive Practice of Facts in Domains with Varied Prior Knowledge” [25]; therefore, this thesis contains only main points and is based on the paper.

Skills/Difficulties Estimation

Difficulties and skills are estimated by a variation of Elo rating system [9], which was originally created for rating chess players. In Elo has each player
associated a number of rating points $\theta_i$ which describe their skill (higher $\theta_i$ means higher skill). After a player $i$ plays a match, $\theta_i$ is updated according to the result $R$ of the match ($R = 1/R = 0$) according to the following formula:

$$\theta_i := \theta_i + K(R - P(R = 1))$$

where $P(R = 1) = 1/(1 + e^{-(\theta_i - \theta_j)})$ is probability that the user wins the match and $K$ is a constant which determines how big influence has the latest match. $K$ can also be a function that decreases with increasing number of matches played by the player $i$ – the more data there is about the player the smaller adjustments player’s skill needs.

In the developed system is the Elo system used in the following way. When a student answers a question about an item it is considered to be a match between the student and the item and is evaluated as described in Elo. Skill/difficulty estimation module computes the following real values:

- **Difficulty** of an item – computed from the first answer of each student; decreased by correct answers and increased by incorrect.

- **Global skill** of a student – computed from the first answer on each item; increased by correct answers and decreased by incorrect.
3. System Functionality

- **Local skill** of a given student on a given item. It is initialized with the difference between student’s global skill and item difficulty and modified by subsequent answers.

  Local skill can be transformed via a sigmoid function to probability of correct answer, which is used in the questions construction module.

**Question Construction**

Questions construction module has to decide the following aspects of constructing a question.

- Which item is the most useful to be practiced at the moment.
- Whether to make the question multiple choice or open answer (i.e. all the items on the current map are possible answers)
- The direction of the question, i.e. “Choose item on the map.” or “What is the name of the highlighted item?”.
- How many choices does the question have (2 to 6) in the case of multiple choice question.
- Which items are the alternatives in the case of multiple-choice question.

The item to be practiced is selected among the available items by a scoring function. The scoring function has three parts:

- **predicted probability** of correct answers – the system has a target probability (e.g. 75%) and items far away from the target probability are penalized the most,
- **time since last question** about the same item – to avoid practicing the item when still in short-term memory,
- **number of answers** for the given user and item – to ensure that each item will be practiced eventually.

The choice of target probability has been subject to some research based on the developed system [24]. Too low target probability is suboptimal as it demotivates users. On the other hand, too high target probability leaves
fewer space for learning. The probability of correct answer can be increased by decreasing number of choices, which is used when there is no item with target probability.

**Alternative Options Selection**

Another important part of the question construction is the selection of the choices of a multiple-choice question. The naive way is to select the choices at random. That is suboptimal as such questions have highly variable difficulty. It might result in questions that can be answered correctly without the knowledge of the asked item, e.g. a question about Angola with alternatives Peru, Kazakhstan and Moldova.

The choices that do not require specific information about an item to be retrieved are called non-competitive (as opposed to competitive). Impact of these two types of choices on retention have been studied in controlled experiments [18], which conclude that competitive choices result in same retention of the asked item as non-competitive; however, they result in higher retention of the item in a competitive alternative choice.

To create more competitive choices the system can prefer closer countries as choices. A possible way to determine closeness of two countries is to use a list countries and their bordering countries. Using only direct neighbors results in very easy to guess questions as the correct answers is usually in the middle. A method with decent results is to also use neighbors of neighbors. Using neighbors, however, cannot be applied to cities or mountains.

A more general approach (currently used in the system) is to prefer items that users used as incorrect answers to the item in open questions. For each pair of items a confusion factor is defined as the number of times the first item has been confused with the other one. The options are chosen randomly among 8 items with highest confusion factor; the randomness is weighted by the confusion factor (items with higher confusion factor are more likely to be chosen). For example, Cameroon is most often confused with Niger (38%), Nigeria (27%), Central African Republic (10%), Republic of the Congo (9%), Gabon (6%), Ivory Coast (5%), Uganda (3%), and Guinea (2%) [25].

A disadvantage of the currently used approach is that by seeing a multiple-choice question with the same alternatives several times stu-
dents can create an association between the alternatives and later be more likely to confuse the items \(^{[20]}\). This disadvantage could be prevented by choosing non-competitive alternatives when the skill of the asked item is low. I would also suggest to avoid asking questions with two (maybe even three) alternatives as they have high probability of guess and thus are less useful for skill estimation. The decreased probability of correct answer can be compensated by choosing alternatives with lower difficulty.

3.2 Client-side Functionality

The essential parts of the client are the practice page, knowledge map and maps overview (see Figure 3.1). Other features described in this section include mnemonics, personal goals and difficulty rating poll.

Practice Page

![Practice screenshot.](image)

Figure 3.2: Practice screenshot.

Practice is the main use case of the whole system. User selects a map and an item type to practice (e.g. countries of Europe – Figure 3.2). The system asks user questions about position of the selected type of items. After answering a question user gets immediate feedback on the correctness of the answer. If the answer is correct, the next question follows automatically in about half a second; whereas after incorrect answer users are given time
to reflect on their mistake and have to explicitly click a button to continue (when they are ready for the next question).

After a set of ten questions a summary page is displayed. It contains correctly answered items highlighted in green and incorrectly answered items in red. Summary page also presents success rate progress bar and buttons for further action (view knowledge map or practice again).

**Knowledge Map**

Knowledge map page provides users with information about their knowledge of a given map. An essential part of the page is a map (see Figure 3.3) with the items colored in different colors from green-to-red color scale (intuitively, green stands for high level of knowledge, red for poor). The scale is defined by five colors (red and green on the far ends and yellow and two shades of orange in the middle). In order to express the change of estimated knowledge, the color can be changed either continuously or in discrete steps (e.g. in 5 or 10 steps).

![Knowledge map of African countries.](image)

It is not clear which of the two possibilities provides users with better feedback on their knowledge. In my opinion, in the continuous case there is too much information and it is impossible for user to distinguish slight changes in colors after practicing a map, thus it is better to use discrete
colors, which show only bigger changes in estimated knowledge. Discrete colors also have a “final” green color, which can express achieving mastery (as in mastery learning [3]).

Another aspect of the knowledge map to be determined is its initial state (prior to any practice). On one hand, each item can be initially colored according to user’s prior knowledge of the item. This way, it is likely that user begins with most items in red, which is, in my opinion, demotivating. On the other hand, items can begin as unpracticed (white) and only after being practiced for the first time obtain a color according to user’s local skill on the given item. A disadvantage of the second approach is that items with high prior skill are unlikely to be practiced and therefore can remain white even when the most of the items are already practiced or even learned, which again can be frustrating to the user who wants to make the whole map green.

One way to overcome the aforementioned disadvantages is to color the high prior skill items from the beginning and the rest of the items only after they have been practiced. This can also be confusing to users, but it is, in my opinion, the best options with the given question selection algorithm. Another option is to change the question selection algorithm so that it selects the high prior skill items to be practiced at least once early in the practice (e.g. one third of questions can be about unpracticed high prior skill items for as long as there are any such items).

On the knowledge map page, user can switch between three main feature categories (Political map, Water and Surface). In each of these categories there are several feature types (e.g. in Political map there are States and Cities), each of which can be hidden. The three main categories have been chosen so as to equally distribute feature types among them. A more equal category to “Political map” would probably be “Physical map”, but that would result in too many feature types in “Physical map”, which would be too cluttered.

Maps Overview

With a higher number of maps in the system it was necessary to create a page summarizing all the available content. It has been created as a list of all maps with links to view user’s knowledge maps and buttons to practice them (see Figure 3.4). Below each button there is a progress bar show-
3. System Functionality

...ing how many of the items have been practiced and how many have been learned by the user.

Figure 3.4: Maps overview.

Mnemonics for Places

Rate of learning of facts can be increased with mnemonics [23]. A relevant mnemonic can be shown to user after an incorrectly answered question. It is relatively easy to implement the feature; however, it is rather difficult to create the content of the mnemonics. It requires either an expert in Geography with interest in mnemonics, or some kind of crowd sourcing.

It is useful to have some data about which items were most often answered incorrectly. Those items have the biggest room for improvement by mnemonics; therefore, choosing them is most likely to result in highest students’ performance improvement at lowest cost. We attempted to crowdsource mnemonics for the system and got them for 20 most often confused items (e.g. Estonia, Latvia and Romania).
3. System Functionality

Personal Goals

The purpose of personal goals is to incentive users to practice regularly in order to get the items into their long-term memory. The feature works as follows: User chooses a map and a completion date of the goal. In order to complete the goal, user has to achieve certain minimum probability of correct answer (0.9 in this case) of all items on the map. The system visually informs user about the goal completion progress and compares the actual progress to an ideal progress (linear increase of average probability of correct answer in time).

Measuring Users’ Perceived Difficulty of Questions

Different versions of the question construction algorithm have been compared in live experiments (outside of the scope of this thesis). It turned out to be rather difficult to find a solid measure for comparing which of the algorithms is “better”.

Essentially we want to measure how much has a user learned, but that is difficult to measure objectively as users’ skills are affected by questions they have been asked, which are selected adaptively. To avoid this feedback loop it would require to introduce pre-test and post-test questions (either fixed or selected at random). Pre-test/post-test is problematic as the average amount of answers by user is around 30 and we do not know this number for a given user in advance so it is possible that the user leaves the system before the post-test.

Given the difficulty of measuring learning effect, we instead measure the number of questions answered by a user assuming that a user who spends more time in the system is likely to learn more (which does not have to be always true). Although it is easy to count number of answers it also has its limitations as it is affected not only by how well the questions fit user’s individual knowledge but also by other (external) factors (e.g. how much time is a student in a school classroom instructed to spend on learning).

As a particular thing we want to know is if the algorithm selects questions of appropriate difficulty, a difficulty rating poll has been implemented. A modal dialog asking “How hard are the questions?” with options “Too easy”, “Just right” and “Too hard” (see Figure 3.5) is displayed to
users after finishing 3rd, 7th, 12th and 20th set of questions. Although perceived difficulty might depend on personal preferences and expectations of a particular user, these level out over a large data set. It also gives us a valuable insight into users perception of the system.

How hard are the questions?
Your answer helps us adjust difficulty of questions.

Too easy  Just right  Too hard

Don't know / Don't want to rate

Figure 3.5: Difficulty rating poll.
Chapter 4

Implementation

In the previous chapter the system functionality was described in a way which is independent of any particular programming language, framework or library. Since one of the goals of this thesis is to implement the system described in the previous chapter, specific technologies have to be chosen and used.

This chapter describes which technologies were used and some of the interesting issues that have been encountered during the development. The source code of the implemented web system is publicly available in its Github repository (see Appendix A). Implementation of some of the features described in the previous chapter was a straightforward use of the technologies and thus such features are omitted in this chapter.

4.1 Project Timeline

The system has been developed over almost two years. That influenced many aspects of the system; its features went through several versions in order to experiment with various approaches. The functionality has been added incrementally and iteratively as well as the content. To provide the reader with an overview of the project timeline a list of major milestones follows:

- **Project Start** (June 2013) – an initial vision of an adaptive practice system for map outlines.

- **Alpha version** (July 2013) – practice with naive question selection (skills and difficulties computed as correct to incorrect answers ratio) and knowledge map; testing on Adaptive Learning group members; only one map map available – states of the U.S.
• **Beta version** (Fall 2013) – content extended with countries of the world and continents; testing on dozens of beta testers; polishing user interactions.

• **FI MUNI students’ research and development project** (December 2013 to May 2014) – implemented support for other types of items (e.g. cities, rivers or mountains); maps overview implemented; SEO to reach broader audience; Elo for skill/difficulty estimation and comparing it with a naive algorithm in A/B experiment.

• **CTT project** (August to December 2014) – Implemented internationalization, personal goals, mnemonics, difficulty rating poll, derived driving license system; user registration A/B experiments launched.

### 4.2 Used Technologies

Various technologies were used to implement the system. The server-side part of the web application is written in Python web framework Django extended by several Django packages. The client is build in AngularJS framework and uses some other JavaScript libraries, particularly Kartograph.js. The client source code deployment is managed with Grunt.

**Django**

The main server building block is a well-known Python web framework Django [10]. Django has been chosen as it is the most popular web framework written in Python. Python has been chosen because it is used for data analysis and experiments by members of the Adaptive Learning Group and thus it is easier to migrate an algorithm evaluated in an off-line experiment to become a part of the web system.

The most important part of Django used by the system is Django ORM (object-relational mapping). All the system data is stored in a relational database (MySQL) and the ORM handles creating the database schema and converting database rows to Python objects and vice versa. Very convenient is also the Django admin – it provides a web interface to manage the database content.

Since Django is very popular, there are many packages (2,719 as of March 31, 2015 [5]) that extend Django with some functionality and can be included in a Django project. The developed geography system uses
four Django packages: south, django-social-auth, django-lazysignup and django-modeltranslation.

**South** is a package for database schema migration. It is such an essential part of developing a web application that it has been incorporated into the core Django package since Django 1.7.

**django-social-auth** is a wrapper around many different social networks Authentication APIs. It supports dozens of authentication back-ends out of which are two used in the system (Google and Facebook).

**django-lazysignup** automatically creates user profiles when users enter a part of the system that needs to save some data about their behavior (e.g. answer a question). It also allows users to register and thus migrate their lazy user profile to a regular user profile.

**django-modeltranslation** is for keeping multiple language versions of model object fields in the database. For each translated Django model field, django-modeltranslation creates multiple database fields (with language code suffixes). Then it decides which of the database fields is returned as Django model field according to the current language.

Django also offers a template language, which extends HTML with tools like cycles and conditions. The template language is, however, used in the system only for the main menu and footer as other UIs are handled by AngularJS templates.

The system uses Django version 1.5 as it was the latest version at the time the project started. However, version 1.5 is no longer supported so the system has to be migrated to a newer version. As it is planned to also migrate the system to a common back-end with other applications described in Section 6.1, these two migrations can happen at the same time.

**Internationalization**

The first implementation was in Czech only. In order to reach broader audience, the application has been internationalized and translated to English. The internationalization later allowed a Mexican volunteer to translate the system also into Spanish and other volunteers are currently translating the system to German.
Internationalization of the UI is done by standard Django translation functions (such as \texttt{ugettext} or \texttt{ungettext}), which rely on GNU \texttt{gettext} toolset \cite{11}. Translation of the practiced items is managed by \texttt{django-modeltranslation} package, which creates additional database fields for each Django model \texttt{TextField} registered for translation (e.g. for field \texttt{name} are created additional fields \texttt{name cs} and \texttt{name en}).

\textbf{AngularJS}

The client of the developed system is a single page application build in JavaScript framework AngularJS \cite{15}. AngularJS has been chosen because it is a Model View Controller (MVC) framework, which makes it significantly easier to develop an interactive application, such as this one, compared to traditional JavaScript frameworks like jQuery. When comparing AngularJS to other JavaScript MVC frameworks (Backbone.js, Amber.js), AngularJS has been chosen because it is backed by Google and has the largest community whether it is measured by Stars on Github, StackOverflow questions or YouTube results \cite{30}.

AngularJS uses templates that are an extension of HTML by \texttt{ng-} prefixed attributes (such as \texttt{ng-if} or \texttt{ng-repeat}) and Angular-specific elements. Templates can be further extended by custom elements – directives. The template system can also evaluate expressions in double curly brackets notation (\texttt{{{ expression}}}). These expressions are evaluated on a \texttt{scope}, which is the “Model” part of the MVC.

The “Controller” part of MVC is a function with its dependencies (AngularJS uses Dependency Injection) as arguments. The dependencies of a controller are services. Services are supposed to be reusable pieces of code with a given purpose (e.g. to communicate with the server).

\textbf{Kartograph.js}

Kartograph.js \cite{1} has been chosen for displaying the SVG maps generated by its Python counterpart, Kartograph.py. It is rather a convenience library because internally it uses Raphaël for the DOM manipulation.

Raphaël is a well-documented \cite{2} vector graphics library for the web. It does all the necessary DOM manipulation inside an HTML5 SVG element.
Raphaël also uses VML as a fallback in Microsoft Internet Explorer (MSIE) version 8 and older, which does not support SVG.

Kartograph.js parses an SVG file to JavaScript objects, where it can be resized and sanitized. Then it converts the objects back to SVG path strings and feeds them into Raphaël, which displays it in an SVG element inside the HTML page. This process of converting the image happens to be the main bottleneck of the client-side performance.

The slowest part is parsing the \texttt{d} attribute of SVG \texttt{path}, which is basically a string representation of an array of coordinates. This string is parsed to a JavaScript Array, which is more suitable for resizing the image, although in order to pass it to Raphaël \texttt{Paper.path()} \texttt{[2]}, it has to be converted to a string again.

On desktop, this bottleneck causes a delay in order of hundreds of milliseconds when loading a map, which is not a serious problem as it is preceded by fetching the map from the server, which often takes longer. The problem is worse on mobile devices with lower performance, but since majority of traffic comes from desktop, this issue has not been addressed further.

\textbf{Grunt}

Homepage initialization performance was negatively influenced by the large amount of HTTP requests made to the server to collect all JavaScript files. Grunt (The JavaScript Task Runner) is used to tackle this issue. Firstly, it is used to minify and concatenate JavaScript files and CSS styles on file modification while development and on deploy. There are about 15 JavaScript libraries files reduced to only one. The amount of project specific JavaScript files is reduced in a similar way.

Furthermore, Grunt is used to compile AngularJS templates to JavaScript files, which are also included in the aforementioned concatenation, thus again reducing the amount of HTTP requests. Last but not least, Grunt is used to run JSHint (a JavaScript code quality tool) in a Git pre-commit hook to prevent any potentially problematic code from being added to the project. JSHint in Grunt was at first also used to check files in real time while editing, but this was later replaced by use of a Vim plugin, which is better integrated to the editor.
4.3 Map-related Functionality

There is a lot of source code that deals with maps. Firstly, SVG images are generated from geospatial data files; then important information from the images is loaded to the database; finally, parts of the SVG images are displayed in a browser.

Generating Maps

Maps are generated by a Python script which is stored in its own Git repository (see Appendix A). The main responsibility of the script is to pass right parameters to Kartograph.py generate function. There are different sets of layers for different maps. In some cases the script does some ad hoc modifications to the generated maps, such as replacing state codes or fixing UTF-8 encoding.

Kartograph.py provides a command line interface for generating maps based on a map file (shapefile format) and a configuration file (JSON); however, after trying it out I decided to use the Kartograph.py programming API. The main reason was that a Python script is more versatile, thus the script can generate only a specified map, a group of maps or all the maps at once. Using a Python script allows avoiding code duplication by use of inheritance, and text replacements in the output.

Adding Generated Maps to the Production System

The output of the script described in the previous subsection is a folder with SVG images (maps). These files also contain names and types of the items for practice, which have to be extracted and loaded to the production system database.

At first, Django migrations were used to add new Places to the database. Eventually, I realized that adding new maps will probably never stop and it should be automated, since all the information is already in the SVG maps and it only needs to be parsed and read.

A custom Django command (called update_maps) has been implemented to automate adding maps to the system. On input it takes a folder with SVG files; each file is parsed in order to find SVG groups which contain SVG paths or circles. These SVG elements contain all attributes
necessary to create a Place object with its name, code and place_type. All the elements being part of a specific SVG file also allows creation of PlaceRelation objects, which carry the information about which Places are on which map (map is a Place, too).

Maps and Layers

Most maps contain multiple layers (e.g. background, countries, cities and rivers). Although all of the layers could be displayed at the same time (since an SVG file contains all of the layers), for the sake of clarity it is better to display only some of the layers (as discussed in Section 3.2). The layers are managed by addLayer() and getLayer() methods of Kortograph.js map object [1]. The hiding of layers is done by calling Raphaël Element.hide() method on each element of the given layer.

Each layer consists of SVG elements, which differ based on what they represent (see Figure 4.1). The basic element, which is used for states, islands and mountains, is an SVG path with a white fill and a black stroke. The element slightly changes color on hover (by decreasing opacity) and a callback function can be bound to the element click event. Other element types differ in some aspects.

A city is an SVG circle. Its radius depends on city population and zoom level. More specifically, there are five population thresholds that increase the size of the circle (e.g. 100 thousand, 5 million). Some cities are very close to each other (e.g. Bratislava and Vienna), which makes it very difficult to select the one underneath the other. In order to overcome this
issue, circles decrease their relative size to other elements when zooming
the map (but they still increase their absolute size).

A **river** is a blue *stroke*, which increases its *stroke-width* on hover.
Since there is no *fill*, it is the *stroke* that changes the color to signal cor-
rectness of an answers or student knowledge. A special property of a **lake**
is its bright blue *fill* and a dark blue *stroke*. **Background** has gray *fill*
and does not react to hover.

**Boundary Box Cache**

A boundary box (bbox) of a path is the smallest rectangle such that all
the points of the path are inside it. Bbox is a useful tool to summarize how
large a path is and where it is located, which is rather difficult to see from
a list of points. Bbox is used in the system when highlighting an item (its
path). The highlight is an animation which scales up the size of the path.
Bbox is needed to find the center of the scaling and to determine the scale
ratio (smaller paths are scaled more).

The complexity of computing the bbox is linear to the length of the path
points list because each of the points has to be checked as it can enlarge
the bbox. In the worst case (which is the Canadian territory Nunavut with
its many islands with complex coastlines), the bbox computation in browser
takes more than a second.

To prevent these delays bboxes are computed on the server on deploy.
They are saved in JSON format and loaded by the client to a global object
called *bboxCache*. The implementation of *getBBox* function in Raphaël
library, has been altered to look for a bbox into the *bboxCache* before com-
puting the bbox. Only if there is no bbox for the given path in *bboxCache*,
it is computed and saved to the cache for next time. *bboxCache* also takes
care of resizing the bboxes to fit the resized map.

**4.4 Other Functionality**

Apart from the map-related functionality, most of the implementation was
rather a straightforward use of the used technologies. One interesting detail
is requesting questions from the server. Another one is the implementation
of logging data, which is mentioned in order give an overview of what data
is logged as there have been done some analyses of the data (see Section 5.2 and Section 6.2).

**Requesting Questions from the Server**

In the implementation of the practice it had to be determined how often to request questions from the server to the client. Questions can be either requested all at once at the beginning of a set or one at a time. A downside of the former option is that it reduces adaptivity of the system as the answers earlier in the set cannot be used to select the questions later in the set. The latter option, on the other hand, creates bad user experience on real-world network as user often has to wait for the next question.

A synthesis of these approaches has been used. The whole set is fetched at the beginning; however, after each answer the list of questions to be asked is updated. Since a correct answer is followed by the next question automatically in 0.6 seconds, the next question is often selected from the already stored list as the request for new questions has not finished yet. However, when the user has already well estimated skill, the stored question and the newly fetched one are likely to be the same.

This point proved to be a source of serious problems. Firstly, the newly fetched question has been set as the next question, which results in the same question being asked twice in a row. Another issue was that sometimes the request after \( n \)th answer can finish earlier than the request after \( n+1 \)th question. If this situation is not handled, it results in newer questions being replaced by older questions. And as the older questions might contain a question that has been already asked at any position this can again lead to the same question being asked twice in a row. All these issues have been solved.

**Logging Data**

Users generate a lot of data while using the system. Majority of the data is users’ answers and smaller part is the difficulty rating poll results. The rating poll answer consists of user, inserted date time, value (one of “Too easy”, “Appropriate”, “Too difficult”).

The answer record consists of the following fields:

- user ID,
4. IMPLEMENTATION

- asked item (e.g. Estonia),
- answered item (if same as item asked, then the answer is correct),
- map (e.g. Europe – to distinguish from Estonia on the map of the World),
- date and time when the answer was answered,
- response time (in milliseconds),
- question type (“Choose on the map” or “What is the name of highlighted item”),
- number of options (2 to 7 in the case of multiple-choice question or none in the case of open question).
- language (Czech, English or Spanish)
- IP address (Can be used to identify group of students from the same IP address and label them as a school class.)

The total number of answers gathered since launching the system is in order of millions (8,728,802 as of April 14, 2015) and growing by about 1 million per month. This big data set has been used in many off-line experiments conducted by members of Adaptive Learning Group at Faculty of Informatics Masaryk University (see Section 6.2). The data is also examined in the following chapter of this thesis.
Chapter 5
System Usage Analysis

The system has been deployed to [http://slepemapy.cz](http://slepemapy.cz) in late 2013. Ever since then it attracts increasing numbers of users. This chapter examines who those users are, what they have in common and in what aspects they differ.

Some analyses of users were done in bachelor thesis of Dionýz Lazar [17], but there are still many other aspect of user behavior to be examined. Also, they were done a year ago and the data set consisted of only about one eighth of the current amount of answers (1 vs. 8 million).

5.1 Used Technologies

System traffic has been continuously monitored by Google Analytics [12], the most widely used website traffic analysis tool [33]. From its wide variety of reports, has been especially useful the number of users in time and where do the users come from.

Google Analytics also offers capability to conduct A/B experiments. Usage of this feature is simple – the system has to fetch A/B value from an API, change the system according to the A/B value, and send the A/B value with all Google Analytics requests. To make it even simpler, there is an AngularJS module angular-google-experiments [22], which provides an AngularJS directive to display the appropriate variation (after setting an experiment ID).

Off-line analyses have been done in Python data analysis library Pandas [31], which internally uses matplotlib [13] for plotting the data. The source code of off-line analyses is available in a separate Github repository (see Appendix A).
5. System Usage Analysis

5.2 Users of the System

Number of users is gradually rising, as can be seen in Figure 5.1. Similar trends have also numbers of sessions and page views. Different trend show the last three plots in Figure 5.1. The changes at the beginning of those plots are caused by some changes of implementation details and do not carry any important information, so the important parts of these plots are relatively constant. The bounce rate keeps on being rather low (11%), which is caused by how is the system implemented – most users arrive to the homepage, where they are immediately faced with several buttons to start a practice; therefore, they make a least this interaction with the system. The average time in the system is about 11 minutes during which the average user visits 11 pages, which, in my opinion, signifies that the practice offered by the system is useful to the users.

![Figure 5.1: Numbers of sessions per month.](image)

Although there are 23 maps in the system, most of the traffic is concentrated to just a few of them (World, Europe, Asia and Czech Republic – see Figure B.1 in Appendix B). The figure also shows that after reaching certain amount of traffic and adding most of the maps into the system (around week 11 of 2014) users’ preference for different maps does not change much over time.
Since vast majority of the users are from the same time zone (Central European Timezone), fluctuations in traffic during a day can be observed (see Figure 5.2). Obviously, there is low traffic at night, then significant increase in the morning and the highest traffic in the evening around 7PM. An interesting phenomenon are six peaks between 8AM and 2PM. At first they might look like random noise, but after a more careful examination they match timings of Czech school lessons and breaks, thus confirm that the system is used in schools. However, most of the users use the system in the evening in their free time, which is, in my opinion, a good thing, since users can use the system at their convenience and spend in the system different amount of time according to their individual needs.

Another indicator of the school usage of the system is the decrease in number of sessions during weekends (see Figure B.2 in Appendix B) and holidays (see number of sessions in July and August in Figure 5.1). The usage in schools is also confirmed by written feedback from users via the feedback form in the system. Some of the feedback contains explicit reference to school and in many other cases it contains signs of being written by pupils (e.g. serious grammar issues or expressed annoyance of using the system unwillingly).
Dividing Users

If there are significant differences between some groups of users (e.g. school users differ from other users), we can classify them into those groups and adjust system parameters for each group to enhance adaptivity of the system. I examined five possible division of users into two groups:

- morning vs. evening users,
- non-returning vs. returning users,
- school vs. out-of-school users,
- low vs. high skill users,
- low vs. high response time users (i.e. fast vs. slow).

In the following paragraphs are the aforementioned division methods specified more precisely, e.g. where is the boundary between morning and evening.

Evening users are defined as those whose average time of an answer is greater than 4PM. The exact time has been chosen ad hoc, based on B.3(a) in Appendix B so as to make the two groups of similar size.

School users are detected by their IP address. More specifically, if there are more than or equal to 20 users whose first answer comes from a given IP address, they are all considered to come from a school classroom. Again, the threshold of 20 users has been chosen ad hoc. In B.3(b) in Appendix B can be seen that this threshold results in about 10% of users being detected as classroom users.

Returning users are determined by difference between the date and time of their first and last answer. Returning users are considered those whose first and last answer are not in the same day. About half of users are being considered returning by this threshold (see B.3(c) in Appendix B). The same figure also shows comparison with other possible division thresholds (e.g. hour or month).

High skill users are considered those with skill 0 and higher. Skill is a real number $\theta$, described in Section 3.1, however, most of users' skills are between -4 and 4 (as can be seen in B.3(d) in Appendix B). 0 has been chosen to be the threshold as it is the default skill of a new user.
Low response time users are those with the median of response time lower than 3.5 seconds. The threshold has been chosen to divide the users into two groups of similar size (see B.3(e) in Appendix B). Median has been chosen instead of average because average response time is way higher because about half of the users have at least one very high response time (above 30 seconds). These very high response times are likely to be caused by users doing other things in the middle of practice. Users do not care about the response time as the system does not suggest in any way that it is measured and users should pay attention to it.

Divisions Correlation

It has been examined how the divisions of users correlate with each other (see Table 5.1). A vector of booleans (one value for each user) is computed for each division. For each pair of divisions are their vectors compared by Pearson correlation coefficient.

The first observation is that none of the correlations is very strong. The strongest correlation is between school and evening users (-0.27), which is not surprising since schools operate mostly in the morning. In-school users also have lower prior skill, which is, again, not very surprising.

<table>
<thead>
<tr>
<th></th>
<th>evening</th>
<th>low RT</th>
<th>high skill</th>
<th>school</th>
<th>returning</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>evening</strong></td>
<td>1</td>
<td>-0.009335</td>
<td>0.018608</td>
<td>-0.269656</td>
<td>-0.016726</td>
</tr>
<tr>
<td><strong>low RT</strong></td>
<td>-0.009335</td>
<td>1</td>
<td>0.07515</td>
<td>0.014363</td>
<td>0.113427</td>
</tr>
<tr>
<td><strong>high skill</strong></td>
<td>0.018608</td>
<td>0.07515</td>
<td>1</td>
<td>-0.116768</td>
<td>0.072943</td>
</tr>
<tr>
<td><strong>school</strong></td>
<td>-0.269656</td>
<td>0.014363</td>
<td>-0.116768</td>
<td>1</td>
<td>-0.068281</td>
</tr>
<tr>
<td><strong>returning</strong></td>
<td>-0.016726</td>
<td>0.113427</td>
<td>0.072943</td>
<td>-0.068281</td>
<td>1</td>
</tr>
</tbody>
</table>

Table 5.1: Pearson correlation coefficient between divisions

The third strongest correlation shows that returning users have lower response times. I can see two reasons for this. Firstly, returning users are more familiar with the system thus respond faster. Secondly, returning users are likely to have higher skill and thus do not need to think for so long to recall an answer. I think that both have an effect to some extent and it should be examined more closely and taken into account when conducting research on response times.
Differences of Question Difficulty between the Divisions

I have examined the differences between the two groups of each division. More specifically, were examined results of difficulty rating poll (see Figure B.4 in Appendix B) and success rate (see Figure B.5 in Appendix B). Several next paragraphs describe some observations from the figures.

School users have lower success rate (73% vs. 80%) and more often rate the questions as “Too difficult” (27% vs. 10%) compared to out-of-school users. The difference is, in my opinion, caused partly by lower skills of school users but mainly by their preference of easier tasks as they use the system only because they were told to do so. The differences between morning and evening users are very slight and probably can be explained by the correlation of morning users with school users.

Low response time users and high prior skill users have very similar trends. They both have higher success rate (83% vs. 77%, resp. 84% vs. 74%) and ratings shifted towards “Too easy”, which is, especially in case of the high prior skill, rather by definition.

Returning users have slightly higher success rate (81% vs. 78%), but unlike any other division they have slightly less of both “Too easy” and “Too difficult” ratings (25% vs. 29%, resp. 10% vs. 11%) compared to non-returning users. This can be explained in various ways. In my opinion, users who are content with the difficulty are more likely to return. However, the reason might be also that the system estimates the difficulty more precisely for returning users as there is more data about them.

5.3 User Registration A/B Experiments

Google Analytics Experiments feature has been used to conduct two online experiments with the aim of increasing the portion of registered users by altering some parts of the UI. While users do not have to register to use the system, the main reason to increase registration rate is to decrease the probability that a single person uses the system as multiple users. That can happen either when a person uses the system from multiple computers or when a person returns after more than two weeks and no longer has the lazysignup session active.
The First Experiment

The first experiment has been conducted because of the assumption that users are not aware of the possibility to register. The assumption stemmed from the fact that the option for registration in the main toolbar was labeled “Log in”. It was labeled that way because the only two authentication options were via Facebook and via Google, which from the implementation point of view do not make any difference between registration and login.

An alternative version with the option labeled “Log in/Sign in” has been created and deployed. After 20 days of running the experiment, the data did not show any significant difference between the two versions (see Table 5.2), so the experiment was replaced by the second experiment.

<table>
<thead>
<tr>
<th>Variation</th>
<th>Sessions</th>
<th>Convs</th>
<th>Conv Rate</th>
<th>CtO</th>
<th>PoOO</th>
</tr>
</thead>
<tbody>
<tr>
<td>Log in</td>
<td>3,794</td>
<td>91</td>
<td>2.40%</td>
<td>0%</td>
<td>0.0%</td>
</tr>
<tr>
<td>Log in /</td>
<td>3,486</td>
<td>83</td>
<td>2.38%</td>
<td>-1%</td>
<td>49.3%</td>
</tr>
<tr>
<td>Sign up</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 5.2: Results of the first experiment (Abbreviations: Conv = Conversion; CtO = Compare to Original; PoOO = Probability of Outperforming Original)

The Second Experiment

The second experiment tested how promoting the possibility to register affects the ratio of registered users. More specifically, in addition to the original site with no promotion, there were deployed four alternative variations with a pop-up box (similar to the one for difficulty rating Figure 3.5) displayed to user after completing 2nd, 4th, 6th, or 8th set of questions. The pop-up box informed users about features for registered users and asked them to register.

I assumed that promoting registration would have positive effect the on ratio of registered users; however, none of the alternatives outperformed the original, as can be seen in Table 5.3. The results neither prove that the original outperforms any of the alternatives. The experiment run for 3 months, which is the maximum length that Google Experiments allow.

The results of neither experiment showed any way to increase ratio of registered users. In theory, there are two possible explanations for this. Ei-
5. System Usage Analysis

<table>
<thead>
<tr>
<th>Variation</th>
<th>Sessions</th>
<th>Convs</th>
<th>Conv Rate</th>
<th>CtO</th>
<th>PoOO</th>
</tr>
</thead>
<tbody>
<tr>
<td>Original</td>
<td>8,888</td>
<td>135</td>
<td>1.52%</td>
<td>0%</td>
<td>0.0%</td>
</tr>
<tr>
<td>After 8th</td>
<td>6,127</td>
<td>92</td>
<td>1.50%</td>
<td>-1%</td>
<td>44.8%</td>
</tr>
<tr>
<td>After 4th</td>
<td>7,173</td>
<td>102</td>
<td>1.42%</td>
<td>-6%</td>
<td>40.2%</td>
</tr>
<tr>
<td>After 6th</td>
<td>5,251</td>
<td>66</td>
<td>1.26%</td>
<td>-17%</td>
<td>15.9%</td>
</tr>
<tr>
<td>After 2nd</td>
<td>1,603</td>
<td>19</td>
<td>1.19%</td>
<td>-22%</td>
<td>19.2%</td>
</tr>
</tbody>
</table>

Table 5.3: Results of the second experiment (Abbreviations: Conv = Conversion; CtO = Compare to Original; PoOO = Probability of Outperforming Original)

ther there was not enough traffic to show the difference or the variations used in the experiments do not have significant impact on registration rate.

In case of the first experiment the variation was, in my opinion, too small to have an impact. However, in the second experiment I think that more data would show statistically significant difference, but it would still be at most a few percent increase of registration rate. My conclusion of both experiments is that users have their own motivations and goals and therefore cannot be easily forced to do something they do not want to do.

With this experience in mind I suggest another A/B experiment with the aim to increase registration rate. There is some functionality for registered users only (e.g. personal goals) but all the links to this functionality are hidden for unauthenticated users. I suggest not to hide the links but instead let the users click the links and then tell them that they need to be registered for this functionality.

5.4 User Feedback

A great amount of user feedback has been gathered through the feedback form on the site. Even though most users do not send us any feedback (less than 1% does), these are often very valuable insights.

To gain some general insight about the feedback all received feedback (162 messages as of March 29, 2015) has been manually categorized. Exactly one type has been assigned to each message to simplify the situation, although in some cases a message could be of multiple types. Messages shorter than 50 characters are filtered by the system and have to be ap-
5. System Usage Analysis

proved to be in the data set as they usually contain no relevant information. Figure 5.3 shows how many messages of each type are there. Most of the feedback is one of the following types (sorted from most often to least):

- **Content request** – users would like like to practice another map (Oceans and seas are often requested).
- **Functionality request** – e.g. countries capitals practice, flags practice or exclude some items from practice.
- **Error in functionality** – e.g. some item cannot be clicked.
- **Praise** – users only express their liking of the application.
- **Report of error in content** – incorrectly named or located item.
- **No information value** – usually send by children who were forced to use the system at school.

Figure 5.3: Feedback messages by type.

Content request are the most common type of feedback probably because we lack a person whose main responsibility would be to create new content and thus the requests are often not met and reoccurring. To fix this we should be looking for someone with knowledge of geography and experience with using a GIS software (e.g. QGIS) to edit maps.
We are also getting many functionality requests that are not implemented as they do not fit our research directions (e.g. practice of flags). On the other hand, we do our best to fix all reported errors, which is probably the reason why error reports are fewer than requests for something new.
Chapter 6

Subsequent Related Work

Since the project started almost two years ago there has already been a significant amount of work done based on the system developed in this thesis. Most of the consequent work fits into one of two categories: development of similar systems (with potential code reuse) and research on the data collected by the system.

6.1 Derived Systems

After implementing the geography practice application I have participated on development of similar applications that reuse some source code and also build on the experience gained with designing and developing the geography application.

Driving License Tests

A system for practicing Czech driving license multiple-choice test questions has been implemented and deployed to [http://autoskolachytre.cz](http://autoskolachytre.cz). It is similar to the geography application as both applications ask users questions and estimate users knowledge based on the answers.

One difference is that questions in autoskolachytre.cz are created by the Czech Ministry of Traffic and are all multiple-choice with two or three static choices, thus there is no need for adaptive choices selection.

There also is no need for a map, which saves a lot of space and makes the application more suitable for use on mobile devices. I am working on converting the web application client into an Android mobile application with Phonegap; however, there are some performance issues with AngularJS to be solved.
The driving license application has been created to be general enough to allow use in another domain with available multiple-choice test questions. The configuration of the system consists of a JSON file with the questions and another file with UI texts. All the domain specific texts in the UI are managed by django-flatblocks and thus saved in the database as opposed to being hard-coded.

One particular domain it could be used in is TSP (Learning Potential Test – in Czech “Testy Studijních Přepokladů”), which are used as admission tests at Masaryk University. Previous years questions of TSP are publicly available; however, the university distributes them only in PDF, which is the main reason it has not been used yet.

**Anatomy for High School Students**

Our system was a starting point for development of an adaptive learning system for human anatomy, which was created as a diploma thesis by Jan Kučera. The anatomy system reuses some of the client-side code of our system, whereas its server-side reuses only the ideas as it is written in a different programming language (Java). The system has been deployed to [http://www.slepaanatomie.cz/](http://www.slepaanatomie.cz/) and its content covers the extent of anatomy in high school biology.

**Anatomy for Medical Students**

Currently we collaborate with Memorix team (authors of an anatomy textbook for medical students called “Memorix of Anatomy”) on an adaptive practice of anatomy for medical students. The practice is very similar to geography (students want to learn names of areas on a picture); however, it differs in other challenging aspects.

Anatomy encompasses significantly larger amount of items and pictures to be learned. The textbook contains about 2000 items and about 600 images and our target group users have to eventually know all of them (and even more) for their anatomy exam. We need to find a way to display them an overview of all their knowledge. We plan to use the topics hierarchy from the textbook and aggregate the knowledge in certain categories/chapters. We also want to motivate students to use the system continuously during their three semesters of anatomy and better incorporate forgetting effects into the student model.
Another challenge is to label all the parts of the images from the textbook in a format supported by the practice application. The book is created in a software that allows the images to be exported to SVG. The SVG paths in the images do not contain names of the items (body parts) they represent. Therefore, I developed a web application which is used by medical students with knowledge of anatomy for labeling the items.

6.2 Research on the Collected Data

The implemented system has produced over 8 million answers and currently produces about 1 million answers per month. This data set is subject to research conducted in Adaptive Learning Group.

There is a paper that describes the adaptivity of the system and evaluates it on the collected data [25]. The paper divides the adaptivity of the system into three main parts: estimation of prior knowledge, estimation of current knowledge and questions selection.

Some experiments focused on comparing different variations of the Elo system to estimate student prior knowledge and item difficulty [21]. These experiments used only correctness of first answer of each student-item pair as the other answers are affected by student’s learning. Results of these experiments show that some are slightly more accurate than others to predict answer correctness; however, the differences are not very high and their impact on user’s learning is to be examined.

Other experiments examined how to improve student current skill estimation. One way to do this is to use time spacing and response time, i.e. how much time elapsed since last answer the user was asked the same item and how long the user spent on answering it. The assumptions are that users forget what they have learned over time and that faster answer indicates better knowledge. The goal of these experiments is to describe the assumed effect more accurately and use it to improve predictions of the current skill. Various models of student’s memory have been evaluated on the data [28].

Research on response times [26] showed (among others) the following findings. When response times are longer, success rate of the answers de-
creases (see Figure 6.1). The same relation is between response times and success rate of the next answer about the same item. However, the relation of response time of an incorrect answer and success rate of the following answer is reverse. These findings can be incorporated to the production system to increase the accuracy of the estimations.

Figure 6.1: Response times and probability that the (next) answer is correct. [26]

There has also been some research on question selection algorithms. It took form of A/B on-line experiments that compared multiple versions of the adaptive algorithms [24]. Firstly, the adaptive question selection has been compared to random selection. It has been proved that the adaptive version causes users to spend more time in the system. Secondly, the question selection with given target success rate has been compared to question selection with target success rate adjusted according to the observed success rate of the past ten answers. The adjustment was done to choose more difficult questions when observed success rate was too high and vice versa. The target success rate has been also manipulated for different users (ranging from 55% to 95%).
Chapter 7

Conclusion

The proposed system has been implemented and deployed to \url{slepemapy.cz}. There are many indicators that suggest the quality and usefulness of the system. Firstly, there is the high amount of users (hundreds of users per day and about 20 thousand sessions per month) and the time they have spent in the system (in total over 3,000 hours per month as the average is 10 minutes per session). Furthermore, there is the positive verbal feedback send by the users. Last but not least, the system took 8th place in the competition called “Společně otevíráme data 2014” (a competition of applications that use open data).

Because of its large user base, the system is not only useful to the users, but also provides opportunities for research. The answers and other collected data have been subject to numerous off-line experiments. Moreover, on-line A/B experiments have been conducted to explore different versions of question selection algorithms and their impact on users’ usage of the system.

There are many opportunities to enhance the functionality of the system. For example, the system can be extended by a teacher’s mode. That is to allow teachers to create classes of students and to display statistics about individuals or the class as a whole. Another useful extension might be allowing users to exclude certain items from practice (e.g. to practice only cities above a given population threshold). According to the feedback, many users would also appreciate more content to practice (e.g. seas, lowlands or deserts).

Another way of building on top of this project is to reuse the developed functionality in a similar system focused on different area than geography. A system for practicing driving license exams questions has been developed in this way and currently we develop a system for practicing human anatomy for medical students.
Bibliography


7. CONCLUSION


7. CONCLUSION


Appendix A

Source code

Source code implemented in this thesis is in one of the following Git repositories and clones of the repositories are in the Archive of Thesis in Information System of Masaryk University:

The geography web application
https://github.com/adaptive-learning/geography

The Python script for generating the maps
https://github.com/adaptive-learning/geography-map-generator

Data analyses
https://github.com/slaweet/thesis-analysis
Appendix B

Visualizations

Figure B.1: Portion of users on most practiced maps in time
Figure B.2: Number of answers during a week
B. Visualizations

(a) Morning vs. evening

(b) In-school usage

(c) non-returning vs. returning

(d) low vs. high skill

(e) Response time (seconds)

Figure B.3: Different thresholds for dividing users
B. VISUALIZATIONS

Figure B.4: Difficulty ratings by divisions

Figure B.5: Success rate by divisions