b
UNIVERSITÄT
BERN

Search Interface Implementation

for a Learning Management System

Bachelor Thesis

Manuela Eschler
from
Bern BE, Switzerland

Philosophisch-naturwissenschaftlichen Fakultät
der Universität Bern

February 2019

Prof. Dr. Oscar Nierstrasz

Software Composition Group
Institut für Informatik
University of Bern, Switzerland

Abstract

The company Swissteach produces a Learning Management System (LMS) called Global Teach®, which lacks a good search engine. This project is about the implementation of a search engine for an LMS, focusing on iteratively designing and evaluating the search interface. Two search interface prototypes with different approaches were tested by users of Global Teach. The first prototype is focused on ease of use and mimics the Google search interface, while the second prototype concentrates on supporting the user if the search fails to produce useful results. Based on the feedback of the testers, both prototypes were combined into one. The resulting search interface is initially simple, and provides features which can be enabled to refine the search results. These features aim to give the user the impression of being in charge, which can lead to a positive search experience.

Contents

1 Introduction

A Learning Management System's (LMS) main purpose is to administrate learning content and the learner's progress. An LMS can contain hundreds or thousands of learning contents, which are made available to the users for learning. To find the desired piece of content, a search engine is necessary. System interfaces are very important for creating positive experiences, and bad interfaces can be very frustrating. The purpose of an interface should be to minimize negative situations, therefore a good search interface should deal with search failure.

In an LMS users search for different kinds of learning content. Some are simple to index, while others are stored in a complex data structure. In a Web Based Training (WBT), content is stored as slides, images, audio and videos. Despite the difficulties, it is important to extract and index information from WBT lessons, as they make up the majority of all content.

1.1 Preconditions and Motivation

The main product of the company Swissteach, where I am employed, is a Learning Management System called Global Teach®. Many of our customers are big companies and have a lot of learning content on the LMS. Most of those learning contents are available to all employees. Due to the amount of content, a search engine is very useful. Global Teach has a search engine, but some of the customers asked if the search engine could be made more like Google search. This implies that users do not trust the Global Teach search engine. Likely causes of that are bad performance, the lack of suggestions,

and some special cases where exact matches are not returned to the user.
For this project I have two main motivations. On the one hand, I want to implement a intuitive and easy to use search engine for Global Teach. On the other hand I am curious to find new and maybe better approaches for search interfaces. Two customers volunteered to test different prototypes for this project. Their feedback will be used to evaluate and improve the prototypes.

1.2 Project Description

While planning the project, I considered two approaches to designing the interface:

1. Survey the customers on how to improve the search and build a new search based on their feedback. Then, make a usability test, and use the results to refine the search interface.

2. Research what a good search interface is and propose different search interface prototypes. Evaluate the prototypes by testing with customers. Improve and change the prototypes to a final version based on the feedback from the tests.

I chose the second approach, because it gives me more room for new ideas and unknown features, while still relying on the experience of the customers.
Having decided on an approach, I determined the steps needed:

- Develop and implement search interface prototypes:
  Research about user search behavior and implement prototypes with different approaches based on that information.

- Choose and integrate a search back-end:
  The big part of this project will be to develop the user interface. As generating the search results is not part of the project, I use an existing search service.

- Prepare and index search data from customers for the tests:
  To create a more realistic testing environment, the search data will be sourced from the customers who are testing the prototypes. Since the data is not in the format needed for indexing, preparations are necessary. That includes extracting the necessary information from the learning contents, especially from WBT lessons.

- Testing prototypes:
  Test the prototypes of the search interface with actual users and improve them.

Although Google search served as inspiration for the first prototype, it will not be analyzed in this project. Moreover, no search engines from other LMSs are inspected for comparison. The content extraction from WBT lessons will improve the results of the search, but developing a tool to automate that process is not part of the project. A proposal on how this could be achieved is discussed in chapter 6.

2
Research about User Search Behavior

To build the prototypes, it was necessary to determine which search interface approaches are worth testing. For that I searched the internet about user search behavior with the following questions in mind.

1. How do people search?
   What do people have in mind when they search? Do they search by specific patterns that could help designing the prototypes?

2. What features are necessary to assure a positive search experience for the user?

3. How to avoid a bad search experience?
   Are there any common mistakes to avoid when designing a search interface?

4. What to do when a search fails? When a search fails to reveal the wanted result, this can lead to a negative search experience. Can the search interface support the user in this situation to mitigate the negative experience?

The answers to these questions will help me get a good impression about user search behavior. The research was focused on two areas. Reading about user search behavior from studies and search engine providers, and taking a look into psychological research. From psychological research I hoped to find out how a search interface can benefit from insights into user behavior, especially when users are confronted with negative experiences.

2.1 User Search Behavior Findings

A lot of what I learned was not directly about user search behavior but about SEO (Search engine optimization). SEO describes the process of increasing the online visibility of a web page. This isn't exactly what I was looking for, but the concept of SEO also reveals useful information about the search behaviour of users. Another interesting source was the statistical data on what was searched on the Google search engine. Google also published discussions with explanations about the search behaviour of their users. That information was very useful to get an idea of how users search today.

2.1.1 Typical Queries

According to a online article by Reputation X [4] about how people search, there are three typical approaches for users to search:

- Go-Queries
  When a user wants to reach a specific web page. Example: To find a specific online shop.
- Do-Queries
  When a user wants to achieve a certain action. Example: How to cook lasagne.
- Know-Queries
  When a user wants detailed information on a specific topic. Example: Information about the upcoming weather.

Relevance

The Go-Query is not relevant for an LMS, because the search interface only displays learning content and never links to an external web page. Learning contents are designed to enhance skills by containing instructions on how to achieve something, as well as information about the specific topic. While the user differentiates between Do- and Know-Queries, the results on an LMS should be the same or at least not differ much. Therefore, there is no need to enable the search interface to categorize the queries by the types Do and Know.

2.1.2 First three results

A study [1] from 2011 discovered that two thirds of the time the user chooses one of the top three results. The other results will get chosen significantly less often.

Relevance

Users seem to rather change the search query than looking at more than the first few results. A limitation of results could be implemented to meet this behaviour. Also the order of the results is quite important. A way to influence this order, like favoring selected words of the query, could improve the success rate of the searches.

2.1.3 Individual searching

The Blue Nile research study 'Psychology of the Searcher' [3] explains a lot about the difference in search behaviour of users, for instance the number of words the users use for searching. Although 29% of the test queries contained only two words, there are still 15% with five words. The study's conclusion states that user's search approach for the same search scenario can vary wildly.

Relevance

Different approaches of searching for the same should reveal the same results. The differences in the search approaches should be compensated by the search algorithm or the search settings. This leads to the idea that every user can set their own search settings individually to always get the best results. To automatically suggest such individual settings more information about user search behavior would be necessary.

2.1.4 Tailored Experience

Google published an article[1] in 2018 about how users expect their search results to be tailored to them. They found that searches which contained terms like 'should I', 'for me', 'do I need' and 'can I' increased about 65% over two years (2015-2017). They made a similar observation about searches with the terms 'near me'. Those increased by 300% over two years (2015-2017). Based on that information they state that users expect to receive a tailored search experience. It is not only important to provide good results, but good results based on the individual user. This is a huge challenge and can't be done without personal information about the user.

Relevance

To find the most suitable learning content for a given user, the relevant parameters are his current skills and the planned development of his career. Collecting this information about the users and using it to improve search results won't be part of this project, because it is not integrated in the LMS and therefore can't access user profile information.

[1] `https://www.thinkwithgoogle.com/consumer-insights/meetingconsumerexpectations/`

However, what can be taken from the insights Google provided, is that at least the interface should be adaptable and able to meet the user's need for a personalized search experience.

2.1.5 Quick and easy

The article 'Its all about me - how people are taking search personally' from Think with Google [2] is about people learning how to phrase the queries to get to the wanted result quickly. Users seem to accomplish that by phrasing the queries in a personal way with the words 'I' and 'me'.

Relevance

The gained information from the trend to use 'I' and 'me' for searching are discussed in the subsection 2.1.3.
Users expect to get their needs satisfied quickly, and that seems to be the expectation for search engines as well. One factor in getting to the desired result quickly is the simplicity of the search interface. If users are forced to apply settings or to read explanations before being able to use the search, they lose time they are most likely not happy to waste. To be quick to use, a search interface must be easy to use. An easy to use search interface could mean no settings that need explanations. Also it is not advisable to use new concepts that the user is unfamiliar with and that require an introduction.

2.2 Psychological Findings

Using an interface should be pleasant, which can be accomplished by minimizing bad or negative experiences. A negative search experience could be that the best matching result for the user does not show and he has to rephrase the search query. Situations where the user does not get the wanted result could be described as failures. What users need in those situations is the question I tried to answer.

2.2.1 What do people need when their request failed?

Although people handle failure differently, Guy Winch Ph.D. suggested a way to overcome failure in the online article '10 Surprising Facts About Failure' [2], namely to focus on the variables in your control. Taking control of these variables will provide a positive experience because the user does not feel as helpless as before and it can increase the likelihood of success for the next try.

[2] https://www.thinkwithgoogle.com/consumer-insights/personal-needs-search-trends/

Relevance

To experience failure when using a search interface may seem irrelevant. But if users connect negative feelings with a certain search engine, they will not enjoy using it, regardless of the quality of the search engine. For example if a search engine suggests results that do not exist, the user gets high hopes just to be disappointed when the desired result doesn't exist. If this happens repeatedly, the user might start associating bad feelings with the search engine.

To avoid user frustration, the search interface should deal with failure. The search interface could for example provide the opportunity to take control over the search parameters and settings. This could mean that the user can influence the order of the results or the weight of specific words in the query. It is not very important what the user can control, as long as he gets the impression of being in control.

3 Technical Details

3.1 Technologies

Since this project is for the company Swissteach, which is a Microsoft partner, the technologies were given. For the back-end I used the .NET Core framework by Microsoft with the programming language C#. For the front-end, the UI library React with the programming language Typescript were chosen. For version control and deployment I used the combination of Azure DevOps and Azure Web App Service. Azure DevOps is Microsoft's online development tool suite and includes Git repositories and Build pipelines among others, and Web App Service is a service on Azure to run web apps. Finally for the most important technology, namely the search engine, Azure provides the service Azure Search.

3.1.1 Azure Search

Azure is the online service portal from Microsoft. Azure itself describes the service Azure Search[1] this way: 'Microsoft Azure Search is a search-as-a-service solution that allows developers to embed a sophisticated search experience into web and mobile applications without having to worry about the complexities of full-text search and without having to deploy, maintain or manage any infrastructure'. The Azure Search service is ideal for this project because file content extraction, indexing and the whole search algorithm are provided and can be accessed using a web user interface. Therefore, there is no need to

[1] `https://docs.microsoft.com/en-us/azure/search`

implement anything for that, which allows me to concentrate on the implementation of the search interface.
Basically the search service takes care of the indexing and searching of the data. As illustrated in figure 3.1 the service can extract the content of different file types for indexing. The extracted information can be enriched by adding additional data derived from the input. For this the service introduces cognitive skills. Cognitive skills interpret the extracted content and add the extracted information to it. Another usage of skills is to divide the information and store it in different fields in the index. For example a cognitive skill could be the determination of the language of the extracted content. After enriching the extracted content it will be indexed by the service. For example, when searching the index, the indexed files can be filtered by the language that was added by the cognitive skill. This whole process can be done by the service, but it is also possible to write new skills.

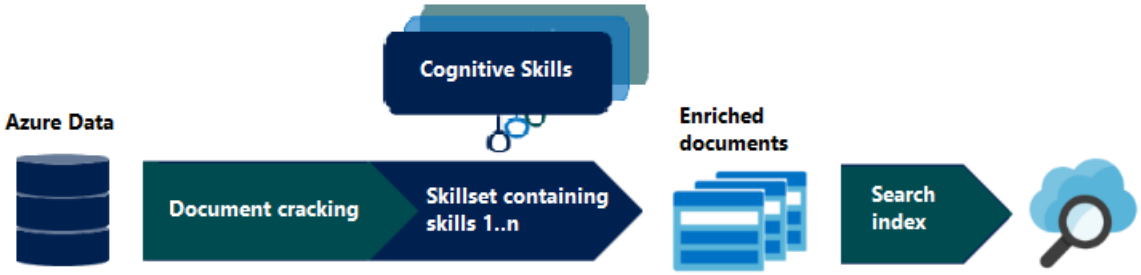

Figure 3.1: Azure Search architecture of enriching and indexing search data[2]

The searching of an index is shown in figure 3.2. Before searching the index the search query is analyzed. Also when choosing the full query parser, a lot of additional conditions can be passed. For example filtering by the language that was added by the cognitive skill. The analysed query is used to search the index and return the top 50 results.

[2] https://docs.microsoft.com/en-us/azure/search/cognitive-search-concept-intro

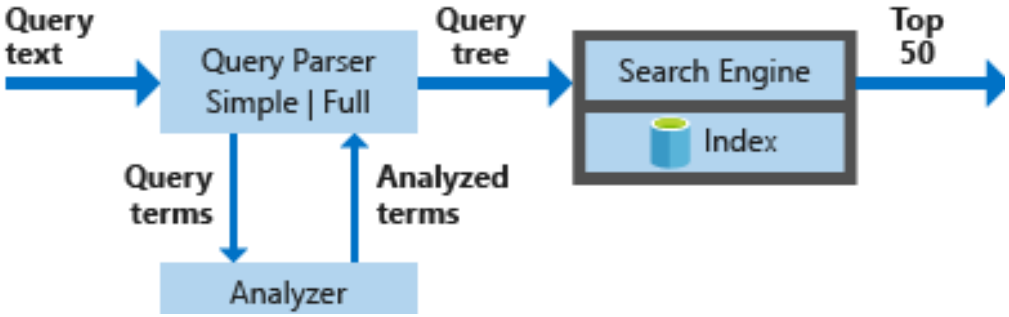

Figure 3.2: Search query architecture[3]

To realize the prototypes, I used the following of the many features of the service.

- HTML file cracking and enriching
- Lucene Query Syntax (Full Query Parser)
- Auto Complete

3.1.2 HTML Content Extraction

The service is able to strip HTML markup and extract the text from the file. Besides the content, it extracts certain metadata like the title, the description and the keywords. An HTML template to extract this information is shown in figure 3.3.

```
<!DOCTYPE html>
<html>
    <head>
        <title></title>
        <meta name="description" content="">
        <meta name="keywords" content="">
    </head>
    <body>
        Content
    </body>
</html>
```

Figure 3.3: Template HTML file for indexing

[3] https://docs.microsoft.com/en-us/azure/search/search-lucene-query-architecture

3.1.3 Lucene Query Syntax

The Lucene query syntax allows a lot of additional conditions for the search, like filtering, fuzzy search, scoring, AND/OR conditions, negations, regular expressions and more. In the prototypes the following features are implemented.

3.1.3.1 Fuzzy Search

The fuzzy search can only be used for words, not phrases. An integer defines by how many letters the matched word can differ from the search word.

- Syntax: $\sim$
- Example: The search query 'blue$\sim$1' returns 'blue', 'blues' and 'glue'.

3.1.3.2 Term Boosting

Every result gets a score, based on how often the search words appear in it. The term boosting feature changes the weight of a word for the score. The score is calculated by a formula called tf-idf[4]. With an integer value, the weight of a word can be increased relative to the others. The default value is one. Since the boosting is relative this option only makes sense if the query contains more than one word.

- Syntax: ^
- Example: With the query 'blue^1 rain^2' results with only 'rain' in it will have a higher score than results with only 'blue'.

3.1.3.3 Mandatory

This feature makes a word mandatory. This feature is only effective when the query is two or more words long.

- Syntax: +
- Example: The query 'blue +rain' returns only results that contain the word 'rain' and maybe 'blue'.

[4] https://en.wikipedia.org/wiki/Tf-idf

3.1.3.4 Negation

Negation is the exact opposite of mandatory. This feature will ban a word. Negation can only be used with more than one word, otherwise there will be no result at all. Also the search service does not allow a query with only banned words.

- Syntax: -
- Example: The results from the query 'sunshine hot -sunburn' will return results with the word 'sunshine' or 'hot' in it, but that do not contain the word 'sunburn'.

3.1.4 Auto Complete

The auto complete feature completes a search query based on the search data.

3.1.4.1 Auto Complete Modes

The search service provides three different auto complete modes:

- **One Term**
  The 'One Term' mode suggests a word for the last partial term of the query.
  Example: blue **ra** $\rightarrow$ blue **rain**, blue **raven**, blue **rail**
- **Two Terms**
  The 'Two Terms' mode suggests a word for the last partial term of the query and adds another.
  Example: blue **ra** $\rightarrow$ blue **rain shoes**, blue **rain man**, blue **rail tour**
- **One Term with Context**
  The 'One Term with Context' mode suggests a word for the last partial term of the query in context of the previous word. The suggested word and the previous word must exist in this combination in the index.
  Example: blue **ra** $\rightarrow$ blue **ray**

3.2 Data Preparation and Indexing

The search service indexes files, and when searching the index, returns the files as results. Since the extracted information from a file can be stored in different fields, the index defines which field can be searched for, retrieved and used for query suggestions. The service is able to read, extract and index information from several file formats like: PDF, Microsoft Office formats (DOCX/DOC, XLSX/XLS, PPTX/PPT, MSG), HTML, XML, ZIP, EML, RTF, Plain text files, JSON, CSV. The search data from the customers is not yet in the correct format to be indexed by the Azure search service.

3.2.1 Customer Data

To make the test with the customers more realistic, I prepared and indexed their data. For that I used their Web Based Training (WBT) and Face-to-Face courses because these are the most common learning contents on Global Teach.

3.2.1.1 Face-To-Face Course

A face-to-face course is a course where the participants have to be present, like a course for swimming. In Global Teach face-to-face courses contain information about the content of the course, the target group, the requirements and whatever more the organizer wants to reveal about the course that can be indexed. This information is stored in the database and has to be collected and converted in the chosen format for indexing.

3.2.1.2 Web Based Training

A WBT is an online lesson, like an enhanced PowerPoint presentation. The information is stored in multiple files, like videos, audio, images and text files. To index a WBT the information of all the relevant files has to be extracted and converted into one HTML file because the Azure search service returns a result per indexed file. How I did this is described in the section 6.3.

3.2.2 File Format for Indexing

For the prototypes I chose to rewrite the learning contents to an HTML file. The search service is able to extract title, description, content and metadata from the HTML file and store them as separate fields in an index. This diversity of information extraction of the service is the reason I chose HTML as the content type for indexing. With other file formats I would have had to implement cognitive skills to get the same result. To have the information in different fields allows me to decide per field if it is searchable, retrievable or used for suggestions. Every face-to-face course and WBT must be contained in a single file in order to be found as one result. An example HTML for indexing is shown in figure 3.4.

```
<!DOCTYPE html>
<html>
    <head>
        <title>Project Cycle Management (PCM)</title>
        <meta name="description" content="The training gives an overview on the result-based project cycle management (PCM) ...">
        <meta name="keywords" content="F2F">
    </head>
    <body>
        Prior to the training, assignments are made available to the participants.
        It is expected that each participant works through the assignments prior to the training (estimated work load: 3 hours).
        The participants have an overview on the phases of a results oriented project cycle management.
        They know the most important project steering instruments.
        They are familiar with logical frameworks and their use for planning and monitoring projects.
        The course uses concrete examples and short exercises to become familiar with the most important steps of project
        planning and the building up of monitoring and evaluation systems.
        The course focuses on the use of the logical framework to structure and describe result chains of projects.
        The training is focused on the use of the LogFrame approach as the standard management tool for planning,
        monitoring, reporting and evaluation. Short inputs and discussions around real-life cases,
        group work and exchange of experiences help to build up and strengthen the required competencies.
    </body>
</html>
```

Figure 3.4: Example HTML file for indexing

Explanation:

- Title: The title of the learning content
- Description: The description of the learning content
- Keywords: The type of the learning content (In the example it is a Face-To-Face course for project cycle management)
- Content: What is contained in the HTML body. All additional information of the learning content that is not in the description, for example the requirements.

Each piece of information will be a separate field in the index.

3.2.3 Indexing

The index for each customer was created with the settings shown in figure 3.5. Every line stands for a piece of information retrieved from the file (see figure 3.4) and stored as a field in the index. The attributes define what can be done with this information.

Fields

FIELD NAME	TYPE	ATTRIBUTES
content	Edm.String	Searchable
metadata_storage_name	Edm.String	Key, Searchable, Retrievable
metadata_description	Edm.String	Searchable, Retrievable
metadata_keywords	Edm.String	Retrievable
metadata_title	Edm.String	Searchable, Retrievable

Figure 3.5: Settings for the index

- **Content:** The information is retrieved from the body of the HTML file. The field is only searchable because it isn't suited to be displayed in the result. The content is where most of the information is, and it has no order or format. For example the content of a WBT contains text from the slides in one block, as the user wouldn't be able to distinguish title and text. Besides, for the user to get the content of the lesson he is supposed to open it.
- **Metadata Storage Name:** The name of the file that is indexed. This field serves as a key and has to be unique.
- **Metadata Description:** The description of the learning content can be searched and retrieved.
- **Metadata Keywords:** The keywords from the HTML file represent the type of the learning content. They can be retrieved to show which learning content type the result represents. However, it shouldn't be searched, because then every learning content with the same type would be a result. The indexed learning contents for this project were WBT lessons, face-to-face courses (F2F).
- **Metadata Title:** The title of the learning content can be searched and retrieved.

3.2.3.1 Result

For every search, the service returns a list of the results with the information of the retrievable fields, namely title, name, description and keywords of the indexed file. Also

the score is added by the service. An example of the result response is shown in figure 3.6.


```
[
    {
        "score":0.5301155,
        "highlights":null,
        "document":{
            "metadata_storage_name":"R290dGhhcmQgUGFzcy5odG1s0",
            "metadata_description":"The Gotthard Pass or St. Gotthard Pass ...",
            "metadata_keywords":"Webpage",
            "metadata_title":"Gotthard Pass - Wikipedia"
        }
    },
    {
        "score":0.5020337,
        "highlights":null,
        "document":{
            "metadata_storage_name":"R290dGhhcmQgQmFzZSBUdW5uZWwuaHRtbA2",
            "metadata_description":"The Gotthard Base Tunnel ...",
            "metadata_keywords":"Webpage",
            "metadata_title":"Gotthard Base Tunnel - Wikipedia"
        }
    },
    ...
]
```


Figure 3.6: JSON response from the index by searching for ‘Switzerland’

With the additional information ‘highlights’ (figure 3.6) the service would highlight the text passages where a search word was found, but since this was not requested, it will always be null. With the retrieved information, the results can be displayed.

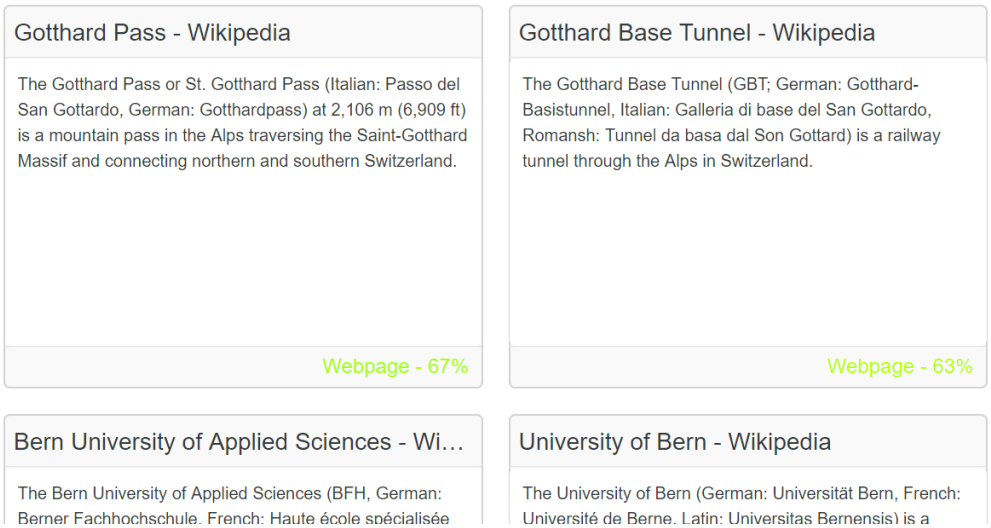

Figure 3.7: One result displayed as tile

Gotthard Pass - Wikipedia	The Gotthard Pass or St. Gotthard Pass (Italian: Passo del San Gottardo, German: Gotthardpass) at 2,106 m (6,909 ft) is a mountain pass in the Alps traversing the Saint-Gotthard Massif and connecting northern and southern Switzerland.	Webpage	67%
Gotthard Base Tunnel - Wikipedia	The Gotthard Base Tunnel (GBT; German: Gotthard-Basistunnel, Italian: Galleria di base del San Gottardo, Romansh: Tunnel da basa dal Son Gottard) is a railway tunnel through the Alps in Switzerland.	Webpage	63%
Bern University of Applied Sciences -	The Bern University of Applied Sciences (BFH, German: Berner Fachhochschule, French: Haute école spécialisée bernoise) is a	Webpage	14%

Figure 3.8: One result displayed as list entry

In the prototypes the user can choose to display the results as tiles or as a list as shown in figure 3.7 and 3.8. The results contain the title and the description and not the whole content, this way the user gets an overview of the result. For more information he is supposed to open the learning content.

3.3 Implementation

The system structure is simple, as shown in figure 3.9. The front-end sends the search requests through the back-end to the Azure search service via HTTP JSON APIs and gets the answer back in the same way. The back-end contains the keys to access the index, this way they are not part of the front-end application.

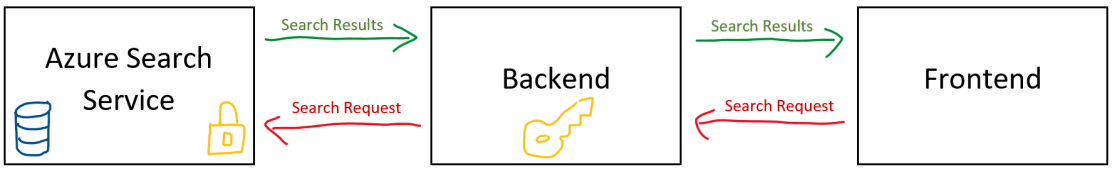

Figure 3.9: System structure

The back-end's main purpose is to forward the requests and answers and manage the keys to the Azure search service.

3.3.1 Front-end Architecture

The React library allows the implementation of single-page web apps with little effort, and follows the component pattern for breaking up the different interface components

into separate code entities. This way the many features are well-structured and the code stays neatly arranged. The search interface is constructed with the components shown in figure 3.10. It displays the architecture of the final search interface and may differ from the descriptions of the prototypes, as the structure has changed for the final prototype. The first of the initial prototypes did not have any of settings components.

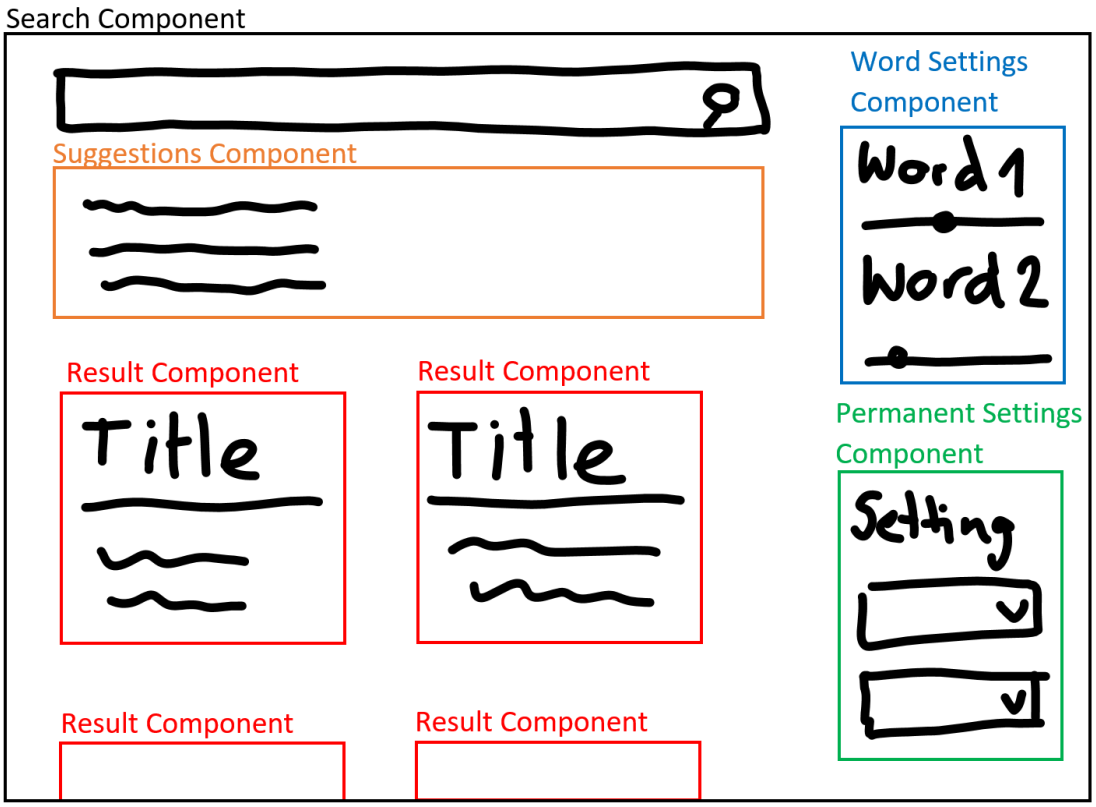

Figure 3.10: Front-end structure with components

- **Search Component**
  Where the search field is and the mechanism to communicate with the back-end. Includes all the other components. Most of the logic and the application state are contained in the search component.

- **Suggestions Component**
  Where the query suggestions are displayed. When typing in the search box, a request for auto complete is sent. The answer is displayed in the suggestions component.

- **Permanent Settings Component**

The permanent settings contain settings about the result display and the auto complete mode. If the advanced search mode is activated, they are always available.

- **Word Settings Component**
  Where the settings about the words of the sent query are located. If the advanced search mode is activated, the word settings appear as soon as a search request is sent. There, the user can add and change conditions for searching. The same settings are displayed for every word in the search query.

- **Result Component**
  Where the result is stored and displayed. When receiving the results, they are each stored in a result component and displayed.

3.4 Search Interface Prototypes

Based on what I learned from the research about user search behavior, I came up with two search prototypes. The prototypes are a standalone project each, therefore not integrated in Global Teach, nor imitating the design. The test with the customers will show which one is preferable and if their basic ideas work. It will be a choice between what users are used to and what could improve their user experience even more.

3.4.1 Prototype One

The first prototype's basic idea is to be easy to use. The details to this idea are discussed in the subsection 2.1.5. The Google search interface fulfills this criteria perfectly, so I decided to mimic it. The prototype can leverage the positive search experience a lot of people connect with Google search interface, because of their similarity. For the realization I used these features.

- **Query Suggestions**
  For the query suggestions, the first prototype uses the auto complete modes 'One Term' and 'Two Terms'. This will give the user a better range of query suggestions, than only one mode.

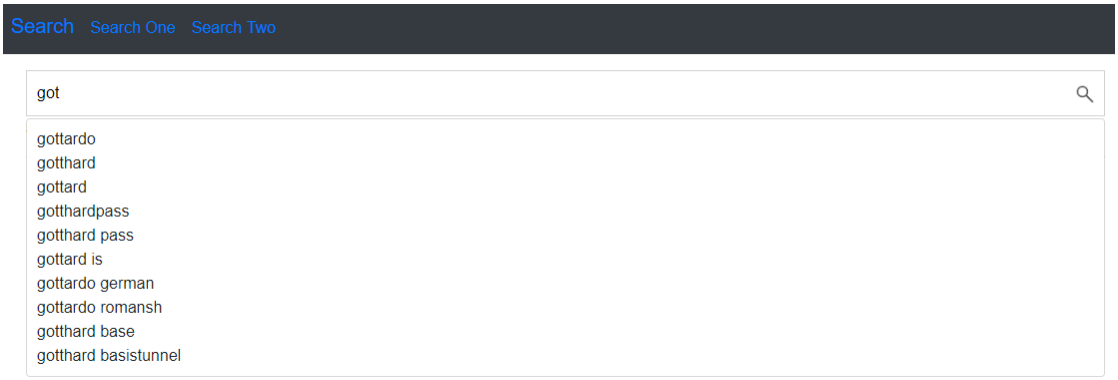

Figure 3.11: Prototype One suggestions

- **Score display**
  With every result comes its score. To clarify the order of the results and give information about the relevance of a result, the score will be displayed as percentage and in color. The color is derived from the score. The score zero leads to the color red, 0.5 to yellow and 1 to green. The colors for values in between are calculated with the color gradient. But at this point I didn't realize that the score can be higher than one which makes the idea to display the score as percentage senseless. That's why I removed it later.

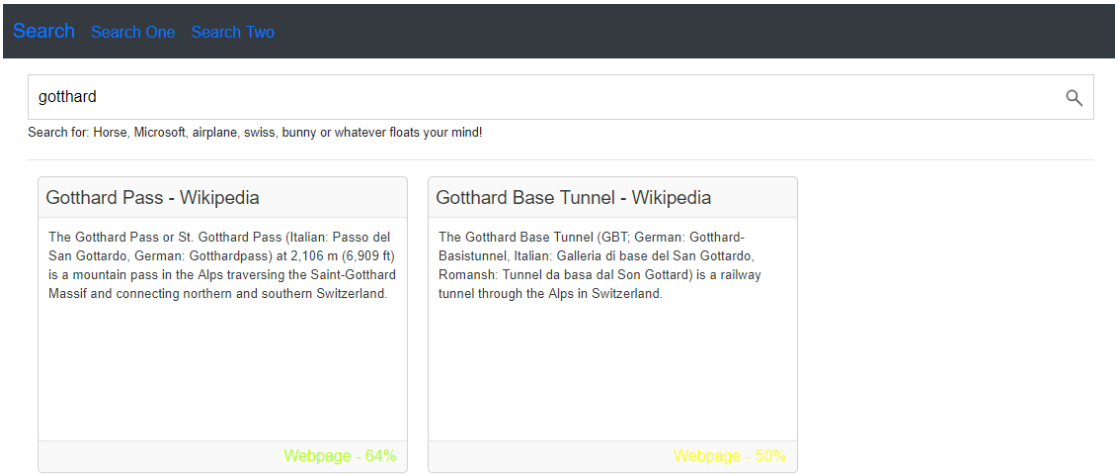

Figure 3.12: Prototype One results

3.4.2 Prototype Two

As a counterpart to the first prototype, the second prototype should deal with search failure when a search doesn't reveal the best result for the user. Psychology teaches us that a way to overcome failure is to focus on the variables in control. The details of this idea are discussed in the subsection 2.2.1. This interface should give the user control over the search settings by providing different features to influence the search. I supposed it wouldn't matter too much what kind of features the search interface provides as long as the user gets the impression of being in charge. With this in mind I looked at every feature of the search service and selected the most promising ones for the interface.

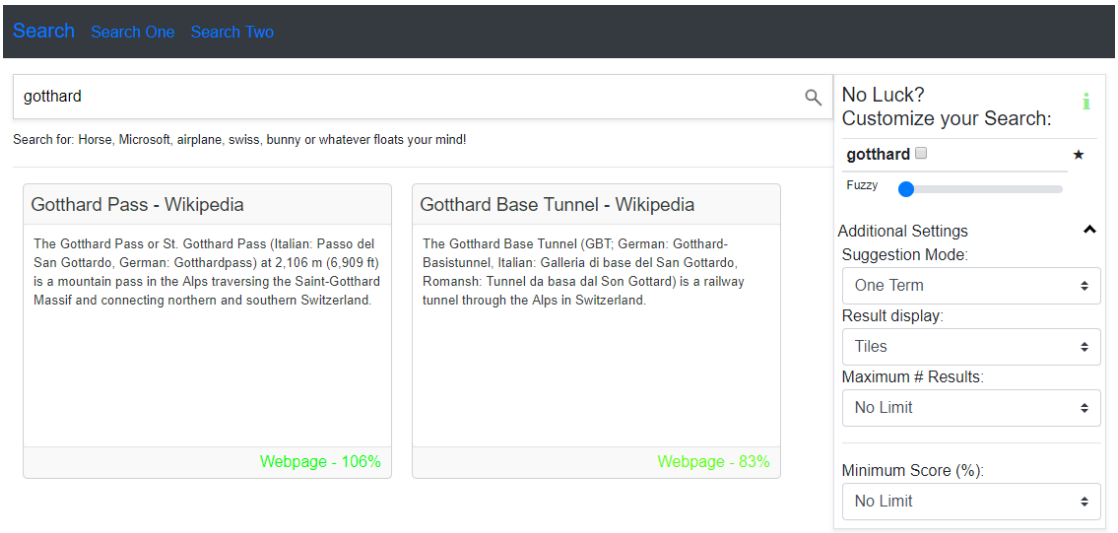

Figure 3.13: Prototype Two

For this, I introduced two different settings that complement the basic search.

3.4.2.1 Permanent Settings

The permanent settings are about displaying the results and the query suggestions. They do not depend on what was searched. In the actual interface the permanent settings are named 'Additional Settings'.

- **Query Suggestions**
  By default the 'One Term' auto complete mode is set. The user can change this and choose another mode.

- **Result Display**
  The user can choose if the results are displayed in tiles or in a list. When displaying

the results as tiles, the score will be multiplied by 100 to display as percentage and shown in color (0=red, 50=yellow, 100=green), to highlight the relevance of the result. But since the score can be higher than 1, the score won't be displayed as a percentage anymore after the first test. The change is applied immediately.

- **Maximum results**
  The user can choose the maximum number of results that should be displayed. This only has an effect on the future searches. Because two thirds of the time the user chooses one of the top three results, as discussed in subsection 2.1.2, users might like to limit the number of results although it doesn't do much.

- **Minimum score**
  The user can set a minimum score for a result to be displayed. The results with a lower score will not be shown. This feature can assure the user that all the results have a certain relevance. The change is applied immediately.

3.4.2.2 Word Settings

The word settings appear when a search is started. For every word in the query the same settings are shown. With these settings the search can be influenced in case the initial search failed.

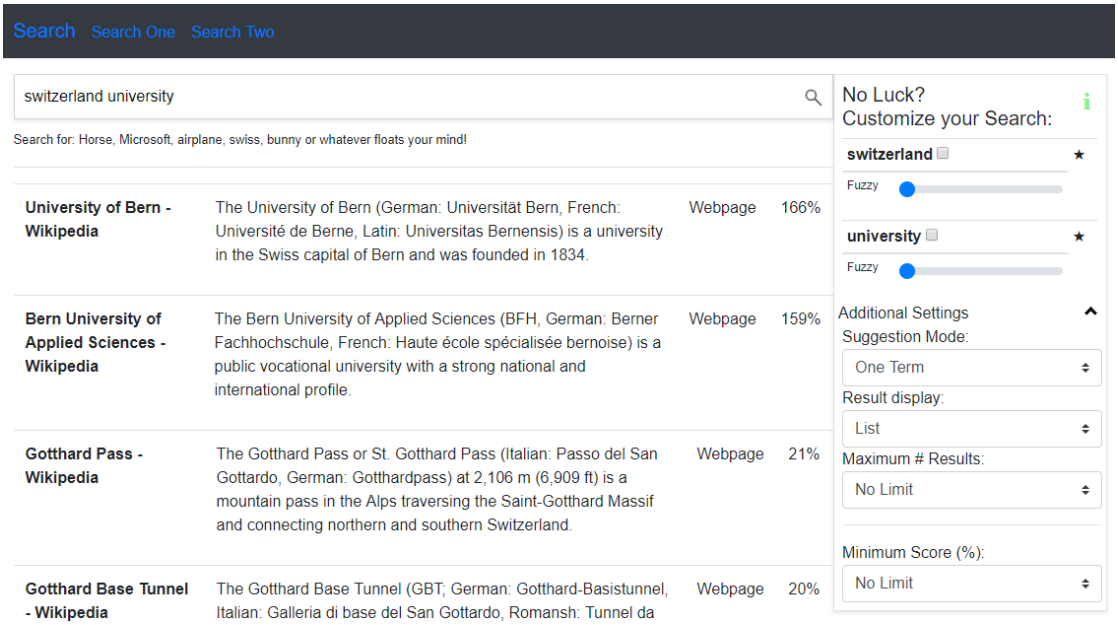

Figure 3.14: Prototype two with the results displayed as list

- **Fuzzy Search**
  The user can choose the fuzzy value of the word between zero and four. For a fuzzy value higher than four a lot of the results could be misleading and useless therefore it is limited. The change triggers a new search with the new setting. By default the value is set to zero.

- **Term boosting**
  The user can favor a word by clicking on the star. This will change the scoring value for this word from one to two and the star will turn yellow. Since the scoring value for this word is now higher than for the others, results that contain this word will have a higher score. The change triggers a new search with the new setting.

- **Mandatory**
  The user can make a word mandatory, otherwise the results have to contain only one of the words of the query, to be a match. The change triggers a new search with the new setting.

A short test with colleagues of the company Swissteach showed that explanations for the settings are needed. Hovering over a feature will reveal the explanation as a tooltip.

4 Testing

To determine if one of the approaches for a search interface is working, two customers who are using Global Teach in their company are testing the two prototypes. With each test I improved the prototypes and at the end the customers were supposed to decide which of the prototypes is the best.

4.1 Customers

Two customers have volunteered to test the prototypes. Let's call their companies Eagle and Purple Berry. They have both been using Global Teach for several years. The twelve testers from both companies are a mix of administrators and users of the LMS.

4.2 Test One

The first test was to get a first impression if the two prototypes work and if there are major or minor issues to improve. The testers were asked to perform some tasks that highlight the differences between the prototypes and then answer some related questions. Both customers started with the same prototypes and the same tasks and questions.

4.2.1 Tasks

Below the tasks are listed as they were presented to the testers, together with brief explanations of their purpose. These explanations were not provided to the testers.

1. Search for "**** *****" (Hint Search Two: you can change the suggestion mode under "Additional Settings" if the suggestions are not satisfying).
   Explanation: This is the tester's first interaction with the search interface. By searching for two words they get a good impression of the suggestions.

2. Search for "*****". Only Search Two: Increase the Fuzziness of the word (on the right side) and see how the results change.
   Explanation: The change of the fuzzy value of the single word will reveal more and other results and will show the influence of this feature.

3. Find a certain learning content without using the title to search (assume you do not know the title).
   Explanation: This task encourages the tester to use the features and shows the main difference of the prototypes.

4. Use the searches exploratively, as you would on Global Teach.
   Explanation: This task ensures that the tester has the opportunity to use all features, if he has not already done so. Also they could use the search interface in a more natural way than with given tasks.

4.2.2 Questions

The questions how they were presented to the testers are specific for the prototypes.

Questions for Prototype One

- Did you find everything you searched for?
- This search has only default settings you can't change. Is there a default setting you would like to change? Which and change to what?
- Add a comment about this search

Questions for Prototype Two

- Did you use any of the settings? Which?
- Was the purpose or use of a setting unclear? Which and what exactly was not clear?

- Is there a specific feature you would like to use that isn't available?
- Add a comment about this search

General Questions

- Is there a feature missing you required?
- Did something bother you?
- Which prototype did you prefer?

4.2.3 Results

The evaluation of the testers' answers showed that the main concept of both prototypes worked. Basically, the testers appreciated the improvement compared to the Global Teach search. Both customers made several suggestions.
Suggestions from customer Eagle:

Suggestion / Problem	Improvement	Prototype	# Testers
The same suggestions were shown repeatedly	Bug fix	One	2
What does score mean?	Don't give the number, only show color with explanation	Both	1
Set default minimum score for a result to be displayed	Default minimum score 0.2	One	2
Order results by 'most viewed' or 'newest'	No need to evaluate this feature, because it is set for the actual implementation	Both	1
Filter by lesson type (WBT, F2F Course)	No need to evaluate this feature, because it is set for the actual implementation	Both	1
Unclear meaning of auto complete mode 'One Term with Context'	Improve explanations	Two	1
Exclude results with certain words	Implement feature to ban a word	Two	1

Suggestions from customer Purple Berry:

Suggestion / Problem	Improvement	Prototype	# Testers
The same suggestions were shown repeatedly	Bug fix	One	3
Display results as list as well	Won't do, because prototype one should not have any settings	One	1
Set default minimum score for a result to be displayed	Default minimum score 0.2	One	1
Add feature sort by	No need to evaluate this feature, because it is set for the actual implementation	Both	2
Add filter (lesson type, location, topics)	No need to evaluate this feature, because it is set for the actual implementation	Both	3
Feature fuzziness was not clear	Give the user better help	Two	1
Feature to choose multiple auto complete mode	Implement multiple choice for the auto complete mode	Two	1
Maximum # results setting is not necessary	No change, because other testers liked the setting	Two	1

Finally 10 out of 12 testers preferred the prototype two. Reasons to prefer the first prototype were that the features of the second prototype were not clear. Generally, if the testers had trouble understanding the features of prototype two, they did not find the explanations or weren't willing to look them up. An approach to deal with that is to make the features more intuitive by using known standards of web design, so that no explanations are necessary, or give easier access to the explanations. It was difficult to make all the features more intuitive, because some introduced an idea, like the fuzzy search, which has to be explained. So it was important to improve the access to the explanations as well.

4.3 Test Two

Based on the result of the first test, I improved the two prototypes individually for each of the customers. That involved bug fixes, improvements and new features. The second test concentrated on the evaluation of these changes and their contribution to the search prototypes.

4.3.1 Test for Customer Eagle

All of the tasks from test one were also part of this test because I wanted the testers to experience the improvements in the same situation as they requested them. This way the tester would have a better chance to evaluate whether the changes were an improvement or not. For the additional features to ban a word I formulated an additional task. The explanation of the task was not part of the test.

- Only prototype two: Search for courses that are not about a certain topic (use the new feature for banning a word: set - in front of the word that should be banned or search normally for the word and use the setting).
  Explanation: To make sure that everyone uses the new feature, I added a hint to use it and how to do so. This way everyone was able to evaluate it.

After completing the tasks, the tester answered these questions.

Questions for Prototype One

- What do you think about the things that changed (minimum score set to 0.2, fixed suggestions bug, no score display)? An improvement or not?
- Add a comment about this search

Questions for Prototype Two

- Is the score display (list: number and tiles: color) helpful? If not, why not?
- Was it understandable how to ban a word? What should be changed to make this feature more user friendly?
- Add a comment about this search

General Questions

- Is there a feature missing you required?
- Did something bother you?
- Which prototype did you prefer?

4.3.2 Test for Customer Purple Berry

All of the tasks from test one are also part of this test, because I wanted the tester to experience the improvements in the same situation, as they requested them. This way the tester would have a better chance to evaluate whether the changes are an improvement or not. For the additional features to combine the auto complete modes I formulated an additional task.

- Only prototype two: Use different combinations of the suggestion modes (Additional Settings).

After completing the tasks, the tester answered these questions.

Questions for Prototype One

- What do you think about the things that changed (minimum score set to 0.2, fixed suggestions bug)? An improvement or not?
- Add a comment about this search

Questions for Prototype Two

- Did you find the explanations of the features if needed? (Hover over the feature or the help icon in the upper right corner)
- Would you categorize the feature to combine the suggestion modes helpful or confusing? How should this feature look like to be more helpful than confusing?
- Add a comment about this search

General Questions

- Is there a feature missing you required?
- Did something bother you?
- Which prototype did you prefer?

4.3.3 Results

The second test revealed that users preferred a simple search without additional settings, as long as the results are as expected. Only when the results are not satisfactory, the user was looking for additional settings and was willing to invest time to actually understand how they work. Three out of nine testers preferred the first prototype.

Evaluation customer Eagle:

Improvement	Positive Feedback	Negative Feedback	Prototype	Solution
0.2 default value for minimum score	Better overview: 3	No further result: 2	One	Show results with less than 0.2 score, but mark them as less important.
No score displayed		2	One	No change
Score displayed as color	Clearer: 4		Two	No change
Ban a word	Helpful: 4	Create separate search box for word banning: 2	Two	No change

Evaluation customer Purple Berry:

Improvement	Positive Feedback	Negative Feedback	Prototype	Solution
0.2 default value for minimum score	2	No further result: 3	One	Show results with less than 0.2 score, but mark them as less important.
Explanation	Easy to find: 2	1	Two	No change
Combine auto complete modes		Confusing: 3	Two	Mark suggestion from which mode it comes. Add synonyms.

After removing the score displayed as percentage, some testers missed the information, although it doesn't provide much.

The first intention was to optimize the two prototypes and then let the testers decide between them. But this test led to a new approach. Each prototype has advantages the other can't compete with. The first prototype is simple, intuitive and user friendly. In contrast, the second prototype is more advanced and gives the possibility to personalize the search and the result display. The testers appreciated both for different situations. Prototype one is ideal to begin with and the features from prototype two may help after an unsuccessful search. Both advantages should be combined in one prototype and that's what I did for the third and final test.

4.4 Test Three

For the third test I combined the improvements from the two different customers. Both customers will perform the same tasks and answer the questions again. After evaluating the answers of the second user test, it became clear that the best option was to combine the advantages of both prototypes into one final prototype. The search starts with the features from prototype one and with a button that enables the features from prototype two, called advanced search mode. But now the final test can't be about what prototype is to be preferred. So I let the testers use every single feature and give feedback whether the feature is useful and if they would use it, and if not, why not. It was a challenge to formulate this test, because the previous tests showed that testers did a better job when they really wanted to use the feature. If that was the case, their answers were more meaningful and more comprehensive than the others. To overcome this problem, the test is divided into two parts. Each part is about testing the features from one of the modes, and the tester can choose the preferred mode and only test that.

4.4.1 Results

Seven out of eight testers preferred the advanced search mode and the remaining tester didn't choose one. Despite the instruction to only test the preferred search mode, six out of eight testers tested both modes.

Simple Search Mode

Feature	Description	Is useful	Why not
Information about result accuracy	The accuracy of every result is different and the results are ordered by it. Since this accuracy is not displayed, there will be an information if the result is beneath a certain accuracy (called score in the advanced search).	6	
Suggestions	By default, if two or more letters are typed into the search box, suggestions appear.	6	

Advanced Search Mode

Feature	Description	Is useful	Why not
Suggestion Modes	When typing into the search box, suggestions are displayed. What kind of suggestions are shown is defined by the chosen suggestion modes. By default, the suggestion modes 'One Word' and 'Two Words' are selected. But they can be combined freely (the suggestion modes 'One Word' and 'One Word with Context' return similar suggestions, it doesn't make sense to select them both).	7	1: synonyms not helpful (bad source)
Ban a word	When a word is banned, then the results do not contain this particular word.	5	3: afraid to exclude useful results, checkbox not useful, just leave the word out
Blur a word	To blur a word means the opposite of an exact match of a word. The chosen number indicates how many letters the matched word can differ from the search word. By default, the blur number is set to one or two (depends on the length of the search word).	7	1: doesn't make sense

Favor a word	The results are ordered by their accuracy (score). To influence the accuracy of the results, it is possible to favor a word. Results that contain this particular word will get a higher accuracy and will show up earlier in the results. If all words are favored then they equalize each other and no word is favored.	7	1: not relevant for single words
Make a word mandatory	When a word is marked mandatory, then every result must contain that particular word.	8	
Display results as tiles (with color coding)	The results are displayed as tiles (by default). The border color of the tiles shows the accuracy of the result. A caption explains which color stands for what accuracy.	7	1: personal preference
Display results as list (with score)	By choosing the display mode 'List', all results are displayed in a list. The accuracy of the results is now shown by a number (score). The list title 'Score' is a link to the theory about how the score is calculated.	2	6: prefer tiles, round score, too much information, unattractive illustration
Minimum Score / Minimum Accuracy Color	The value of 'Minimum Score' (result display: 'Tiles') or 'Minimum Ac-curacy Color' (result display: 'List') defines the minimum score a result must have to be displayed.	6	2: most relevant are shown anyway

Maximum # Results	The number of 'Maximum # Results' defines the maximum numbers of results that are display. This is the only setting that won't take place immediately, it will only affect the following searches.	2	6: not necessary

Overall the new approach to combine the two prototype was a success. Two features clearly stood out as unpopular. Displaying the results in a list was less preferable than as tiles, and the setting to choose the maximum number of results was disliked. But users can just avoid using these features, therefore I decided against changing anything. The rest of the features were generally appreciated. All testers were provided with descriptions of the features to avoid them being unable to perform the tests, so the evaluation does not show how well the interface would work with new users.

5 Conclusion and Future Work

5.1 Final Prototype

At the end the final prototype is a combination of the two initial prototypes. The user starts with a simple search. The suggestions contain the two auto complete modes ‘One Term’ and ‘Two Terms’. As sample data 40 Wikipedia pages were indexed.

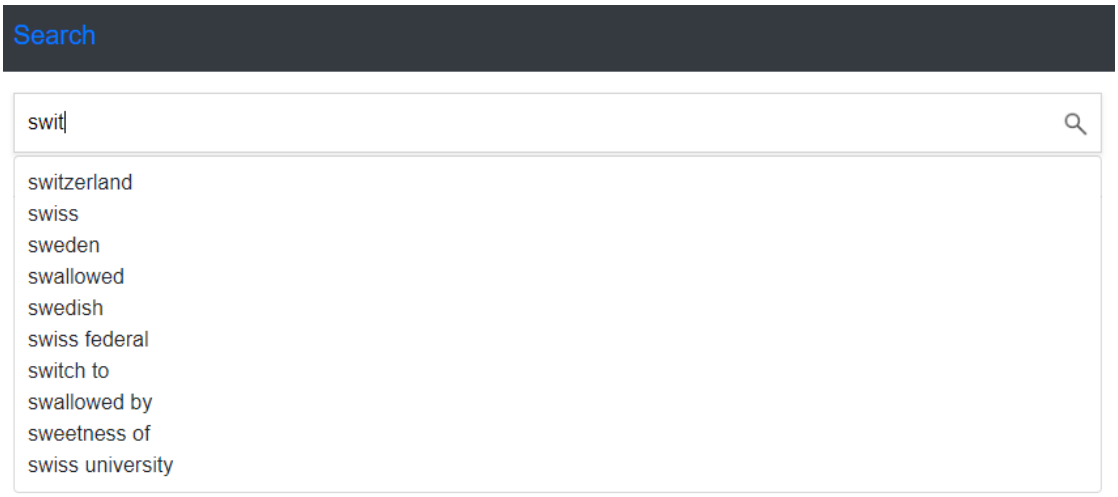

Figure 5.1: Final prototype in simple search mode suggestions

The results are displayed as tiles and do not contain any information about the score.

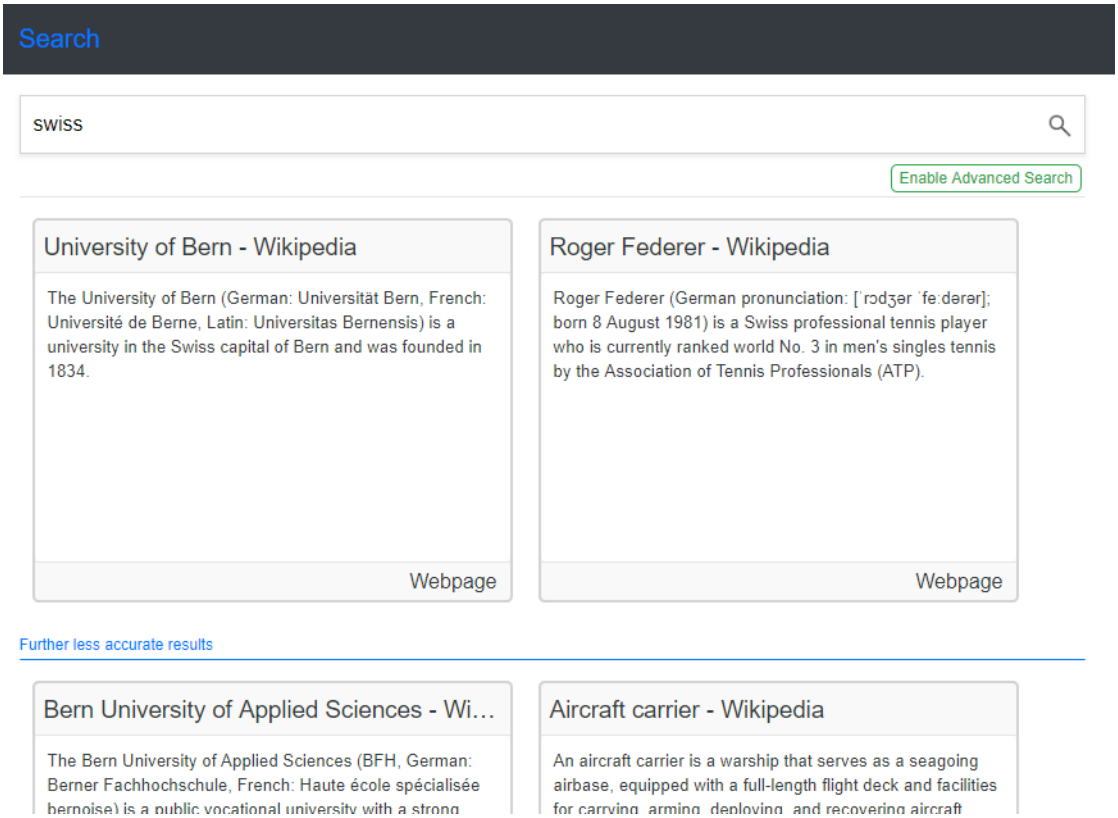

Figure 5.2: Final prototype in simple search mode results

In essence, the search in simple mode works the same as search in advanced mode with default settings.

At any time the user can enable the advanced search mode. The auto complete modes can be combined freely. The suggestions are grouped and labelled by auto complete mode.

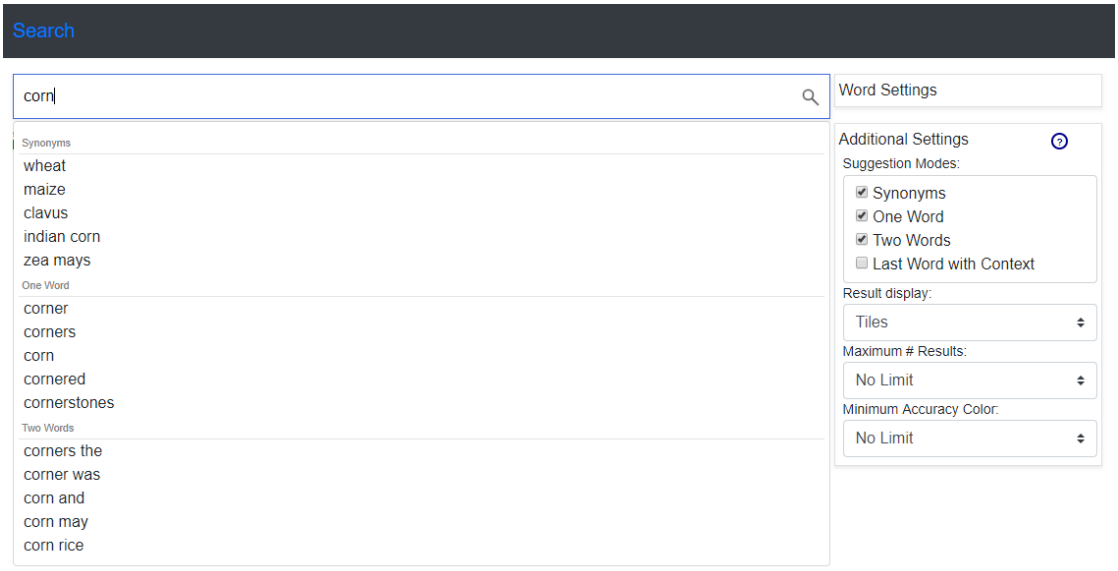

Figure 5.3: Final prototype in advanced search mode suggestions

The results can be displayed in tiles or as a list. The score is shown as color in the tiles display mode.

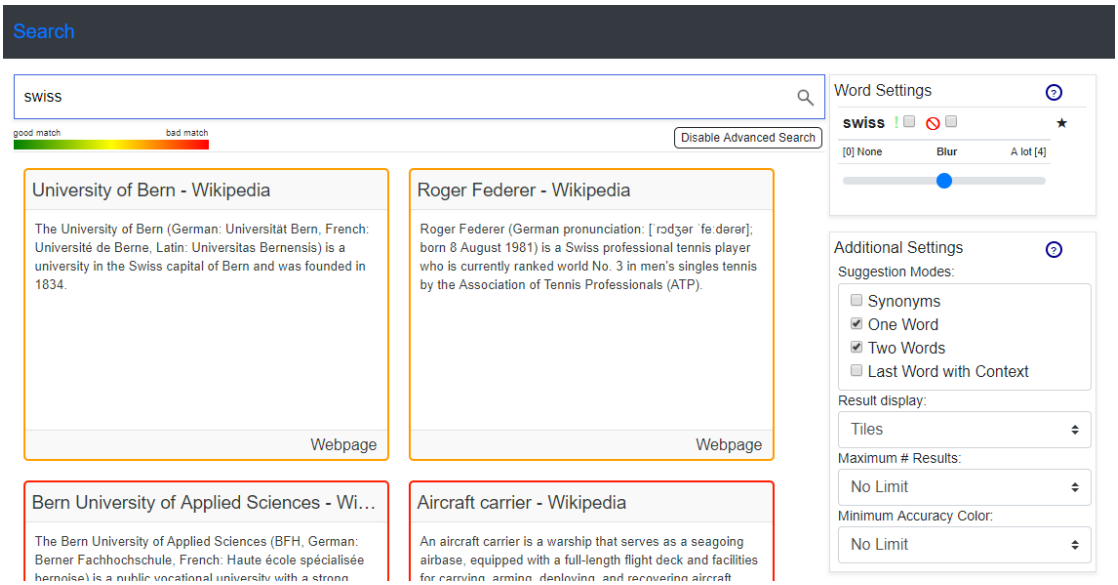

Figure 5.4: Final prototype in advanced search mode results displayed as tiles

The score is displayed as value in the list display mode.

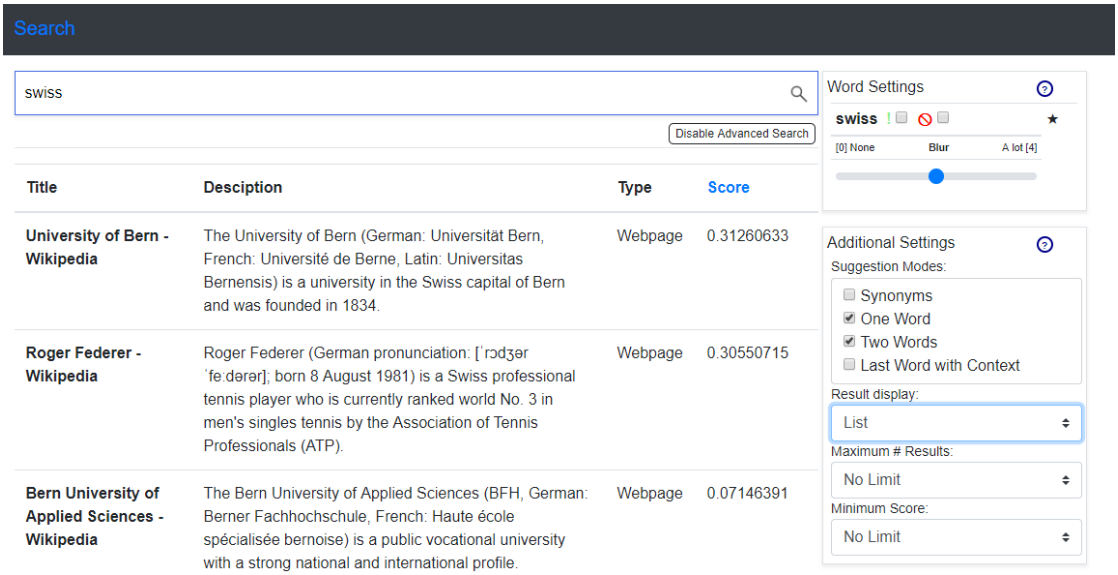

Figure 5.5: Final prototype in advanced search mode results displayed as a list

The search mode can be changed any time.

5.2 Challenges

This project contained several challenges, some of which were anticipated and others not.

5.2.1 New Technologies

To realize this project, I had to use several technologies. I was acquainted with only one of them. It took a lot of time to get to know them well enough to be able to use them for the project.

5.2.2 Working with Customers

To work with customers for such a project can yield a lot of interesting insights and valuable information. But it also brings a uncertainty to the planning. Naturally the project is not their top priority and this may lead to delays, since their contribution is voluntary.

Because the whole project is in English I decided to implement the search interface as well as the tests in English. Now I realize, that since the testers' and my native language is German, it may have contributed to the difficulties in understanding the features and

questions. Also I didn't get the first two user tests reviewed by anyone. When I did so with the third user test, the quality of the answers improved. Despite these challenges, the insights the testers gave me contributed a lot to the successful outcome of this project. But the next time I would choose interviews over written tests.

When evaluating the answers of the second test, I made an interesting observation. Although the score in the first prototype provides nothing useful, testers complained about losing the information. It seems that users would rather have useless information than lose a feature.

5.2.3 WBT Content Extraction

Beforehand I did some research about how WBT lessons are built and how to extract the important information. It promised to be a challenge, because there are not only different file types in which the information stored, but it is not clear in which of them the relevant information is. To evaluate this automatically would be beyond the scope of this project. To do this manually required a lot of time too. Due to this, the quantity of the indexed learning contents per customer was a compromise, and limited to about 50. Most of the customers do have between 200 and several thousand pieces of learning content on their LMS. This difference between the test environment and the reality influences the significance of the tests. The results are significant enough to do a proof-of-concept, but not to validate the features with any scientific certainty. For that, tests in more accurate environments will have to follow.

5.3 Conclusion

The project didn't lead to the decision to choose one of the two optimized prototypes as planned, but the third test showed that the combination of the prototypes was the best solution. The users would use almost every feature. Based on the research I claimed that giving the users control over the single components would help to overcome unsuccessful searches. To demonstrate that this theory worked, we can look at the feature to blur a word. This feature does two things: first it allows users to get results that only contain other word forms (e.g. plural) than what was search for. Second, at the same time it searches for words that have absolutely nothing to do with the original word (e.g. blue $\rightarrow$ glue). Even though this feature can worsen the results, seven out of eight testers would use this feature, in case their search was unsuccessful. This leads to the conclusion that the chosen approach worked and will be successful. Still one problem remains. The features are in need of an explanation and users do not like to read. So the usability of the features has to be improved.

5.4 Future Work

The next step will be to improve the usability. When the testers were asked how to improve the usability, they mostly did not know. The last test revealed where the usability should be improved, but not how. For that I will have to do a comprehensive usability test.

Then the main part will be to integrate the search into the LMS. A good way to steadily improve the search interface could be to collect live data, to reveal which features were used. Or integrate a feedback feature for the user to help improve the search.

For automatically extracting information from WBT a new tool has to be developed. Probably other LMS do not search in the content of WBT. It will take some additional effort to develop this tool, but it will really improve the results.

The search queries tell what the users would like to learn. One can take advantage of this and analyze the queries. By looking at the queries that do not have a lot of results or only those of little relevance, new learning content can be created, based on the needs. Another idea would be to display a wish list, if no result was found, where users can enter suggestions for new learning content.

Not tomorrow, but maybe sooner than we think, searches may be not needed anymore. If a LMS contains all relevant information about a user in its profile, the search could be replaced by profile-based suggestions. If a new interest arises, the user just enhances his profile with it and fitting learning contents are suggested. For now the search could be extended by suggestions, based on the user profiles.

The new search interface, together with the new method to extract content from WBT lessons, will be a benefit for the customers and improve the competitiveness of the product.

6 Anleitung zu wissenschaftlichen Arbeiten

A big challenge was to index the Web Based Training files. WBT lessons cover a great part of the learning content on LMS in general, because they enable an appealing way to teach without a teacher. In Global Teach WBT lessons are uploaded as SCORM. SCORM stands for Sharable Content Object Reference Model and defines a specific way of building learning content in a way that can be read by the LMS. But SCORM is not a standard, meaning the contents are not always constructed in the same way. A WBT can contain slides, audio, video, images and questions made of a various number of file formats: html, json, swf, mp3, mp4, xml, jpg, png, js. There is no sure way to know in which file the relevant information lies. The audio can be very informative, but it could be just music with no information at all or worse music with lyrics. Therefore it was necessary to find a way to extract only the relevant information.

6.1 Web Based Training Structure

One thing every WBT in the SCORM format has in common is the start file, called imsmanifest.xml, shown in figure 6.1. The most important part of the manifest file to extract information is the tag resource. There, all content files are referenced. Now the brute force variant would be to index them all. But that would lead to a lot of unnecessary and even misleading information. For example an image could be an arrow to use as a button for the next slide, which has nothing to do with the topic of the learning content. The next step is to determine where the relevant information lies.

```
<manifest xmlns="http://www.imsglobal.org/xsd/imscp_v1p1" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xmlns:adlcp="http://www.adlnet.org/xsd/adlcp_v1p3" xmlns:adlseq="http://www.adlnet.org/xsd/adlseq_v1p3"
 xmlns:adlnav="http://www.adlnet.org/xsd/adlnav_v1p3" xmlns:imsss="http://www.imsglobal.org/xsd/imsss"
 identifier="com.scorm.manifesttemplates.scorm2004.3rdEd.nometadata" version="1"
 xsi:schemaLocation="http://www.imsglobal.org/xsd/imscp_v1p1 imscp_v1p1.xsd http://www.adlnet.org/xsd/adlcp_v1p3 adlcp_v1p3.xsd
 http://www.adlnet.org/xsd/adlseq_v1p3 adlseq_v1p3.xsd http://www.adlnet.org/xsd/adlnav_v1p3 adlnav_v1p3.xsd
 http://www.imsglobal.org/xsd/imsss imsss_v1p0.xsd">
  <metadata>
    <schema>ADL SCORM</schema>
    <schemaversion>2004 3rd Edition</schemaversion>
  </metadata>
  <organizations default="B0">
    <organization identifier="B0" adlseq:objectivesGlobalToSystem="false">
      <title>Title</title>
      <item identifier="i1" identifierref="r1" isvisible="true">
        <title>Title</title>
      </item>
    </organization>
  </organizations>
  <resources>
    <resource identifier="r1" type="webcontent" adlcp:scormType="sco" href="index.html">
      <file href="index.html"/>
    </resource>
  </resources>
</manifest>
```

Figure 6.1: Template imsmanifest.xml[1]

6.2 Locating relevant Information

The relevant information can be in four different places.

1. **Slides Text**
   The main information is often on the slides. A challenge is to avoid extracting useless or misleading information. A lot of WBT lessons do have questions included. On many slides, there are sentences like ‘This is correct’ or ‘This is wrong. Please try again’. Sentences like this should not be indexed.

2. **Video**
   A lot of the WBT lessons do only contain a video. To extract any useful information, it is unavoidable to analyze the content of the video. A video has two components, the images and the audio. The image analytic tools give interesting results, but the description is often too general. But it is possible to extract the text from a video, which could be useful. To analyse the audio can be very rewarding, but only if there is spoken text.

3. **Audio** Often slides have additional audio, and it could make more sense to use this information, rather than the slides, because often the audio is more thorough or even describes images on the slides. But there can be audio that is just music, in

[1] https://scorm.com/wp-content/assets/SchemaDefinitionFiles/SCORM%202004%203rd%20Edition%20Schema%20Definition%20(no%20metadata)/imsmanifest.xml

the worst case it is music with lyrics, that has nothing to do with the content of the WBT.

4. **Images**
   On many slides, there are images with important details. But results of image analysis are not specific enough to get the relevant information from the image. The image analysis reveals the object and some action on the image, but the most relevant information lies in the details, which the analyzer is often not able to recognize. For example, an image about how to tie a safety knot for climbing, with a red rope in front of a wooden wall, would return tags like 'indoor', 'table', 'wooden', 'red', 'sitting', 'white', 'different', 'old' and 'colorful' from the analysis. Most of these tags are true but they do not capture the main information of the picture. Also most of the images are brand or company logos or navigation elements. The chance to get misleading information is very high.

For this project I went through a lot of different WBT lessons and found many different combinations of where the relevant information is to be found. The most interesting are described here.

- **Only video with relevant text and audio**
  A WBT lesson with only a video is common. In this case the only information can be derived from the video's images and audio. Both have to be taken into account, because both could contain relevant information. Only text should be extracted from the images, because the description of the image from the image analysis is not specific enough.

- **Slides with relevant audio**
  If the audio of a WBT lesson contains relevant information, there is no sure way to tell if extracting the text on the slide is still necessary. If the text only contains what was spoken in the audio, there is a risk to also extract useless sentences, like 'This is correct' or 'Try again' from question feedback. But if the audio does not cover all information, the text is important. The best way would be to find a way to exclude the useless sentences. For example create a black list with the most common question feedback.

- **Slides with video**
  In addition to the slides the WBT lesson can contain a video that should also be analyzed for information.

The idea is to handle all the different WBT lessons in the same tool, therefore some extraction rules are necessary.

6.2.1 Extraction Rules

From what was established above about the different formats, the following extraction rules can be derived.

1. **Image: No indexing**
   Images should not be indexed, because image analysis does not reveal the relevant details but a general description of the image.

2. **Audio: Only when relevant**
   Audio can contain a lot of information, but it has to verified before using. Therefore the audio always has to be converted to text and then analyzed. If the audio does not contain any text, it is just music. When some text is found, the first few words could be compared with lyrics to ensure that it is not a song.

3. **Video: Audio only when relevant and text from images**
   Every video is worth indexing, but not all of it. From the images only the written text is of interest and the audio files must be checked for lyrics before using the transcription.

4. **Slides: exclude useless sentences**
   Find a way to exclude the irrelevant sentences from the text.

6.3 Manual WBT Content Extraction

For extracting information from WBT lessons I analysed them and categorized them by their structure. Then I chose one structure per customer data that would only request one content extraction method.

- **Only Video**
  The WBT lessons of the customer Eagle all contained only a single video. To extract the information for indexing, I converted the video to audio and used an Azure service to convert the audio to text.

- **Only Slides**
  The WBT lessons of the customer Purple Berry contained only slides. Since the slides were HTML files, they just could have been indexed. But a WBT lesson has several slides, therefore several files. For indexing there can be only one file per WBT. With regular expressions I stripped the files of the HTML tags and combined them into one file.

To realize this, I wrote a small tool that did the main work for me.

```
public static void ConvertVideoToWav(string filename)
{
    var infile = new MediaFile { Filename = @"resources/video/"+filename+".mp4" };
    var outfile = new MediaFile { Filename = @"resources/audio/"+filename+".wav" };
    var conversionOptions = new ConversionOptions
    {
        AudioSampleRate = AudioSampleRate.Hz44100
    };

    using (var engine = new Engine(@"resources/FFmpegLib/bin/ffmpeg.exe"))
    {
        engine.GetMetadata(infile);
        engine.Convert(infile, outfile);

    }
}
```

Figure 6.2: Code sample that converts the video to audio

6.3.1 Method to extract text from audio of a video

Azure provides a service called Speech[2], that converts audio to text. But for that to happen, the audio must have a certain format: 'Audio samples in PCM format, one channel, 16000 samples per second, 32000 bytes per second, two block align (16 bit including padding for a sample), 16 bits per sample'. Although the documentation suggested a sample rate of 16000 samples per second, experience showed that 44100 got better results and it worked for transcription. To get the spoken text of a video two steps were necessary. First convert the video to the requested audio format; the associated code sample is shown in figure 6.2.The second step was to use the speech service from Azure to convert the audio to text.

The audio was in a very good quality so the transcription of the audio and therefore the content extraction worked quite well. The results showed only few misinterpretations.

6.3.2 Method to extract information of HTML files

After stripping all the HTML files of the tags, they have to be combined into one file. The stripping was done with regular expressions, shown in figure 6.3.

[2] https://docs.microsoft.com/en-us/azure/cognitive-services/speech-service/

```
public static string RemoveAllTags(string input)
{
    Regex rRemScript = new Regex(@"<script[^>]*>[\s\S]*?</script>");
    var output = rRemScript.Replace(input, "");

    Regex rRemScript2 = new Regex(@"<!--[\s\S]*?-->");
    var output2 = rRemScript2.Replace(output, "");
    Regex rRemScript3 = new Regex(@"<[\s\S]*?>");
    var output3 = rRemScript3.Replace(output2, "");
    Regex trimmer = new Regex(@"\s\s+");

    var s = trimmer.Replace(output3, " ");
    return s;
}
```

Figure 6.3: Code sample strips the HTML file of all tags

The text of the HTML was extracted in four steps, with the help of four regular expressions that matched the markup to be removed.

1. `<script[^>]*>[\s\S]*?</script>`: The HTML file deletes the script section.
2. `<!--[\s\S]*?-->`: All comments are removed
3. `<[\s\S]*?>`: All opening and closing tags are removed.
4. `\s\s+`: By compacting white spaces, the text becomes more readable. The search service would not care about white spaces but it is useful to take a look a the text to control if this method worked correctly.

The method worked, all HTML specific syntax was removed and only the plain text was returned.

6.4 Automated WBT Content Extraction

To automate the content extraction from WBT, I suggest certain steps, as illustrated in figure 6.4.

1. Read the resources from the imsmanifest.xml file
   (a) Extract resources from Shockwave Flash files (.swf)

2. For every file, determine the file type: image, video, audio or slide
3. Extract the data as defined in the extraction rules in subsection 6.2.1
4. Combine all the extracted information from the several files

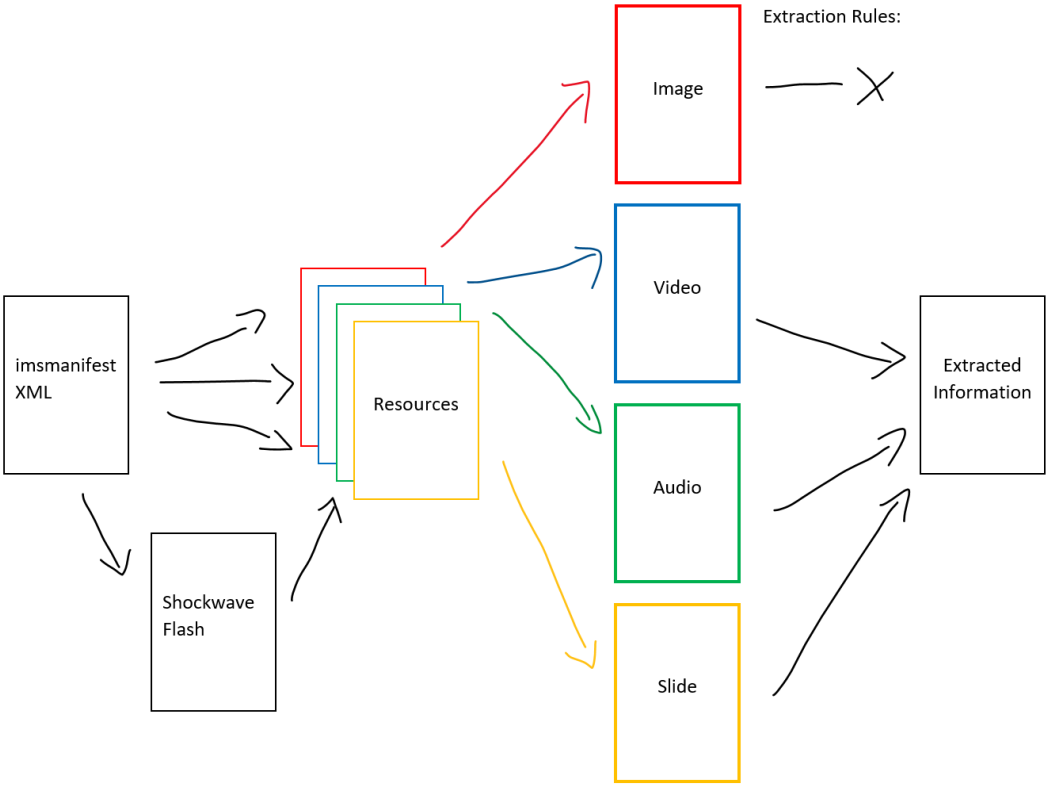

Figure 6.4: Illustration of concept for automated WBT cracking

6.4.1 Read imsmanifes.xml File

At the top level, every WBT lesson has an imsmanifest.xml file. The structure of this file, shown in figure 6.1, is given allowing all the resources to be extracted.

6.4.2 Crack .swf Files

To extract the files from a Shockwave Flash file, a external service or library is necessary.

6.4.3 Determine File Type

With the file extensions it is a simple task to determine the file type, because there are only four different ones: image, video, audio, slide. From experience the file extensions of the slides are .htm or .json.

6.4.4 Apply Extraction Rules

As defined in the extraction rules, all images can be ignored.

Audio files can be converted to text by using the Azure service Speech to Text [3]. If the audio only contains music, the transcription will give no text, but if the audio contains a song with lyrics, they will be converted to text. To avoid indexing lyrics, the extracted text should be checked. For that a service could be used, which needs more evaluation. Another option is to analyze the converted lyrics for patterns and use this to detect lyrics without a service.

The video indexer from Azure[4] can convert the audio to text and does visual text recognition (OCR) to extract text from the images. Again, the text from the audio must be checked for lyrics.

The extraction from the slides is simple, the challenge being to exclude the useless sentences. The main problem is the feedback of questions that tells if the user answered correctly or not. These sentences all being very similar, an approach could be to create a blacklist of forbidden sentences. Another approach is to count how many times the sentence appears, because normally the feedback for every question is the same.

6.4.5 Combine extracted Information

The last step is to combine all the extracted information and store it in the body of an HTML file that contains the title, description and learning content type. Then the file is ready for indexing.

[3] https://azure.microsoft.com/en-us/services/cognitive-services/speech-to-text/

[4] https://azure.microsoft.com/en-us/services/media-services/video-indexer/

Bibliography

[1] David Towers. PPC accounts for just 6% of total search clicks [infographic]. https://econsultancy.com/ppc-accounts-for-just-6-of-total-search-clicks-infographic/.

[2] Guy Winch Ph.D. 10 surprising facts about failure – Psychology Today. https://www.psychologytoday.com/us/blog/the-squeaky-wheel/201501/10-surprising-facts-about-failure, 2015. Online; accessed: 2018-08-28.

[3] Nathan Safran. Psychology of the Searcher. https://www.immagic.com/eLibrary/ARCHIVES/GENERAL/BNILE_US/B150428S.pdf, 2015. Online; accessed: 2018-08-27.

[4] Reputation X. How People Search: Understanding the Landscape. https://blog.reputationx.com/how-people-search, 2017. Online; accessed: 2018-08-27.