






CONCEPTUAL DESIGN AND IMPLEMENTATION OF A PROTOTYPE SEARCH APPLICATION USING THE OPEN WEB SEARCH INDEX

A. Nussbaumer¹ , R. Kaushik^{1,2} , G. Hendriksen³ , S. Gürtl¹ , C. Gütl¹ 

¹Graz University of Technology, Graz, Austria

²University of Waterloo, Ontario, Canada

³Radboud University, Nijmegen, Netherlands

Abstract

The development of special-purpose search engines requires a crawling and indexing infrastructure, which needs technological knowledge and resources. This paper presents a concept and implementation of a prototype search application that enables creating own search applications using the OpenWebSearch.eu index. The concept consists of the integration of an index partition exported from the Open Web Index and a search service that builds on Apache Lucene and offers a REST API, which makes the index searchable. A prototype implementation has been created that applies the conceptual design and provides two demonstration applications. Concept and implementation should enable and encourage developers to create their own special-purpose search application.

INTRODUCTION

In contrast to general-purpose search engines like Google, vertical search engines enable focused search in specific domains and allow domain-specific search operations. Current popular vertical search solutions are mostly commercially focused or integrated into enterprises' business models, such as Amazon's product search, LinkedIn's people search, or Booking.com's hotel search.

Search engines are composed of basic components and processes, such as gathering web documents, indexing, metadata extraction, searching and ranking, and a user interface [1]. Also vertical search engines need a search index, which requires a lot of technological resources if newly created even for a fraction of the global web content. The OpenWebSearch.eu (OWS) project aims to provide unbiased, democratic, and free search across the internet through its open access to its Open Web Index (OWI). In particular, it allows downloading a portion or partition of the index, which can be used to create a search application [2].

This paper presents a conceptual design of a vertical search engine in the context of the OWS project and its integration with the OWI. Furthermore, it describes the implementation of a prototype search application based on this concept, as well as two demonstration applications. Thus this paper seeks to demonstrate and provide a technological basis how a search application can be developed based on the OWS infrastructure and the OWI.

CONCEPTUAL DESIGN

The overall concept of an OWS vertical search engine consists of two parts, the OWI and the search application (see Figure 1). The term *search application* is used for the stand-alone search component with imported index partition.

The OWI contains a vast corpus of websites collected from the World Wide Web, which is maintained in data centres distributed throughout Europe. It allows downloading partitions of the index that can be incorporated by search applications. An index partition is structured as Common Index File Format (CIFF) [3] and the corresponding metadata is shipped as Apache Parquet¹ file format.

CIFF aims to enable sharing and utilization of inverted indices across different search systems. It provides a standardized format for representing index data, allowing multiple search engines to access and utilize the same index files. This format is quite useful in academia for searching indices from various information retrieval systems and comparing their performance. However in an industrial setting, it cannot directly be used by most search libraries due to their own internal index formats. Therefore, in order to utilize the CIFF index, third party developers must convert it to the internal index format of the search library that they use. To resolve this issue, the CIFF-Lucene converter² developed by Radboud University can be used to generate an index that can be used by Apache Lucene³. Lucene has been chosen, since it is used as a search engine library by commonly used search engine systems, such as Elasticsearch and Apache Solr.

In addition to index data, the Open Web Index also provides metadata of web pages in Parquet format. When creating the OpenWebSearch.eu index, the original content of the websites goes through common pre-processing steps. However, snippets of original website data in conjunction with their links make for richer results in search applications. In order to preserve this original page, full text is stored in the metadata. Furthermore, the original metadata of the web pages and data stemming from the page analysis data are stored. In the future, further information, such as the language, the topics of the content, and geographic information (coordinates) are extracted from the pages and added to the metadata. These metadata are exported along with the index data in the Parquet format, which is a columnar storage file format widely used in big data processing and analytics

¹ <https://parquet.apache.org/>

² <https://github.com/informagi/lucene-ciff>

³ <https://lucene.apache.org/>

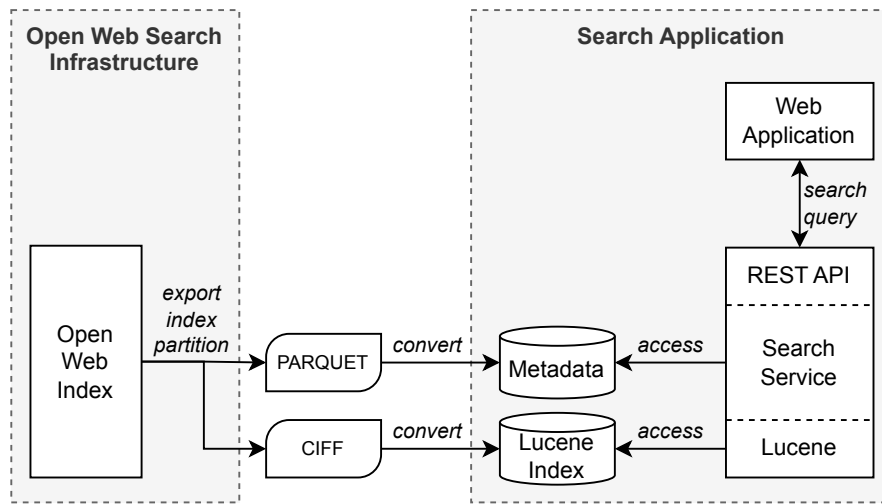


Figure 1: Conceptual design of a vertical search engine.

frameworks such as Apache Hadoop and Apache Spark. It is designed to optimize I/O performance for large-scale data processing. The columnar data storage has several benefits including columnar compression, schema evolution without dataset rewrites and efficient queries by reading columns.

The core of the search service is the search application that coordinates the search and retrieval process. It provides a REST API that accepts search queries and returns search results. In order to perform the actual search, it makes use of Lucene that accesses the partitioned index converted to the native Lucene format. Using the metadata information, the preliminary search result can be limited and ranked, as well as enriched with full-text snippets. The web application provides the user interface (front-end) where search queries are created and results are displayed. This can be done in classic style with a text field and link list, but also other forms are possible.

PROTOTYPE SEARCH APPLICATION

Based on the conceptual design of the OpenWebSearch.eu project, we created a prototype search application⁴ that can be used as a reference as well as the prototype can be extended by future search applications. This application includes a REST API that is intended for direct use by developers who wish to create search applications using the Open Web Index. A key aspect of the REST API resides in its capacity to operate across multiple indices rather than being limited to a single index. The application requires developers to place one or multiple CIFF files into the `ciff` folder and their corresponding Parquet files into the `parquet` folder.

To be able to use an index defined by a CIFF file in the prototype search application, it has to be converted by the CIFF-Lucene converter to a Lucene index beforehand. The application provides a script (`convert_index.sh`) that converts the CIFF file into an Apache Lucene index and stores

it in a folder named the same as the CIFF index. This also allows developers to use the converted Lucene index among other search libraries of their choice, such as Elasticsearch, Solr and Cassandra.

The compressed Parquet file corresponding to the index is accessed at runtime and read programmatically using the Java library `parquet-mr`⁵. The Parquet file is queried by the application through column pruning, enabling efficient large scale data retrieval. This allows rich metadata to be returned by the REST API along with the index search results. Additionally, metadata is used for ranking and filtering algorithms.

The search application is written in Java, and built using Apache Maven. The developer simply needs to run the `build.sh` script and then the start script (`start.sh`). By default the service listens to `localhost:8000` and Cross Origin Resource Sharing (CORS) is allowed from `localhost:80`. By this action, any front-end application hosted on the latter can access and send requests to the service. The CORS and hosting ports can be changed from these defaults as required.

At the application start, all indices in the `lucene` folder and their corresponding Parquet files in the `parquet` folder are loaded for later search request handling. The prototype search application provides a REST API with a single endpoint named `search` and multiple query parameters (see Table 1). Developers can use this endpoint to send an HTTP GET request with at least the parameter `q` that specifies the search query. While `q` is a required parameter, the remaining ones are optional. For each search request, a particular Lucene index can be defined by setting the parameter `index` to the desired value (i.e., the name of the desired Lucene index). Further, a filtering mechanism is realized by the parameter `lang` that restricts search results to a specified language. An additional parameter `ranking` is employed to determine whether the results should be

⁴ <https://opencode.it4i.eu/openwebsearcheu-public/prototype-search-application>

⁵ <https://github.com/apache/parquet-mr>

Table 1: URL query parameters

Parameter	Necessity	Description
q	Required	Search term(s) to be searched for in the Lucene index.
index	Optional	Specifies the Lucene index to be searched in. The passed value must match the folder name of the Lucene index. If no index is specified, the default index passed as argument at the application start is used.
lang	Optional	Restricts the search result to only consider pages in the specified language (e.g., en). If no language is specified, the search results are language in dependent.
ranking	Optional	Specifies the order of the search result based on the number of words a page has. Can be either <i>asc</i> or <i>desc</i> . If no ranking is specified, the order of the search result yielded by Lucene's similarity search is used.
limit	Optional	Sets the maximum number of results to be returned. If no limit is specified, a maximum of 20 results are returned by default.

organized in ascending or descending order based on the word count within an individual web page. The filtering and ranking of results made possible by these query parameters demonstrates the handling of index partitions and the associated metadata. A representative URL format for a GET request may resemble the following:

```
http://localhost:8000/search?q=tower&index=
websites-graz&lang=en&ranking=asc&limit=10
```

With this request to the REST API, the term *tower* is searched in the index *websites-graz* and only web pages with English language in ascending order in terms of word count of the page limited to a maximum of 10 results are returned.

The search service sends the results back in the form of JSON data as an array of objects. Each object comprises the *url*, the *title*, a *textSnippet* consisting of the longest sequence within the text without a line break, the *language*,

Table 2: Fields of the search result

Key	Description
id	the id of the result item or web page
url	the URL of the result item or web page
title	the title of the result item or web page
textSnippet	a piece of text in context of the search term
language	the language of the result item or web page
warcDate	the date of the WARC file where the page is found, which is crawling date
wordCount	the number of words of the result item or web page

the *warcDate* and the *wordCount* of the respective web page. An overview of the object fields is given in Table 2.

DEMONSTRATION

The Prototype Search Application can be used to make any search application on the internet. To demonstrate this, we created two web applications (front-ends) that demonstrate the search applications and its REST API. These front-ends are part of the Prototype applications.

Basic search

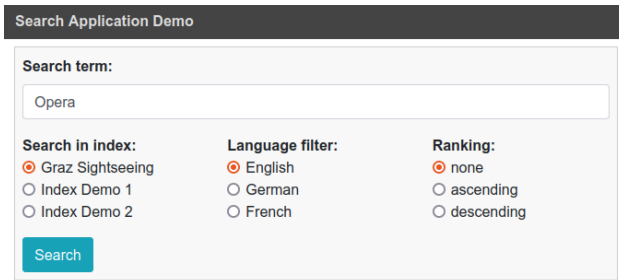
The basic search application demonstrates the features and API of the prototype search service. A specific index (CIFF and Parquet file) has been used that contains sightseeing information of Graz, Austria. Furthermore, a demonstration index has been included. This application demonstrates that prototype service can handle multiple indices, as well as the features of the REST API.

The web interface (see Figure 2) consists of a field for entering a search term, as well as radio boxes for selecting the index, the language, and the ranking method. The selected information is translated to a REST call using the API of the service. The result as specified in Table 2 is displayed below in commonly used form.

Sightseeing Search

This application uses the Graz sightseeing index that consists of popular attractions in Graz. It consists of a CIFF and Parquet file that were converted and imported to the search service. The application demonstrates that search applications with a different kind of user interface can be created easily using the concept described in this paper.

The front-end (Figure 3) consists of a graphical map on which predefined attractions are placed. By clicking on an attraction a popup displays the name and a description of the respective item. In addition, a search request is performed using the title as the search term. The search result is displayed below the graphical map. Hence, each time a user



Search result for term: "Opera"

Opera House Graz

Something fascinating about Graz - the interplay of modernism and tradition - is illustrated by the sculpture "light sword" next to the opera house. It was originally made for the festival "steirischer herbst" in 1992. To celebrate the 500th annivers https://www.graztourismus.at/en/sightseeing-culture/sights/opera-house_shg_1472

Graz Opera

Past general music directors (GMD) of the company have included Niksa Bareza (1981-1990), Philippe Jordan (2001-2004), Johannes Fritzs (2006-2013), and Dirk Kaftan (2013-2017).[7] In the autumn of 2016, Oksana Lyrin made her first guest-conducting https://en.wikipedia.org/wiki/Graz_Opera

Best attractions in Graz

The cafe, opened in a hotel with the same name, can be called a full-fledged Austrian attraction, because it is here that they serve the real Sacher cake, cooked according to the original ancient recipe. Keep in mind that there are always a lot of vi <https://www.tripzaza.com/en/destinations/top-attractions-in-graz>

Generallhof in Graz

What would the old town of Graz be without the wonderful inner courtyards? What would a summer in Graz be without jazz concerts in the Generallhof? Every year in summer, the Generallhof is a meeting place for connoisseurs: every evening, 100 guests h https://www.graztourismus.at/en/sightseeing-culture/sights/generallhof_shg_6958

Figure 2: Demonstration Search Application

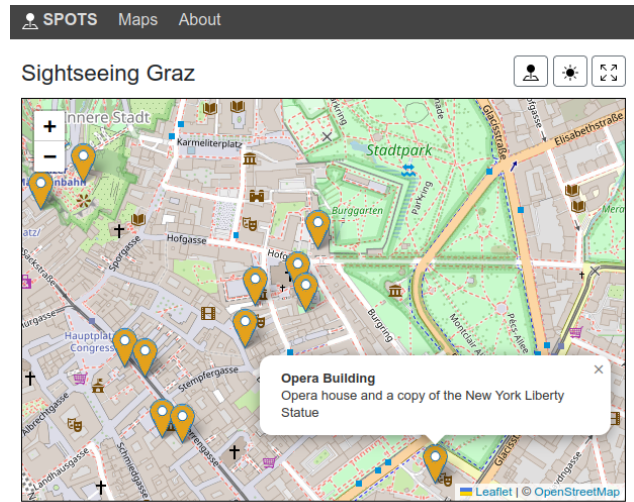
clicks on an item on the map, a search is automatically performed and the result is displayed. This allows the user to get more information on demand.

CONCLUSION

This paper presents a concept and implementation of a search application in the context of the OpenWebSearch.eu project. The design is deliberately kept simple, as it serves as a blueprint for other search applications and vertical search engines. A key aim of this paper is to provide a link between index partitions created by OpenWebSearch.eu and search applications for one’s own purposes. In the near future it will be possible to download an index partition restricted to certain characteristics, such as the topic, region, or time frame.

The prototype search application should encourage the development of various search applications using the OpenWebSearch.eu technology and index data. The source code is available in a public GitLab repository under an open software licence and includes enough documentation so that the technology can be taken up and used for new search applications. The prototype can be used in two ways. First, the service can be used out of the box and a new front-end can be added. Second, the service can be altered and updated to add specific search features.

Future work will include the handling of multiple and large index files. While the search is currently only possible in a single index that has to be specified, in the future a combined search over multiple indices should be enabled.



Search result for term: Opera Building

Best attractions in Graz

The cafe, opened in a hotel with the same name, can be called a full-fledged Austrian attraction, because it is here that they serve the real Sacher cake, cooked according to the original ancient recipe. Keep in mind that there are always a lot of vi <https://www.tripzaza.com/en/destinations/top-attractions-in-graz>

Opera House Graz

Something fascinating about Graz - the interplay of modernism and tradition - is illustrated by the sculpture "light sword" next to the opera house. It was originally made for the festival "steirischer herbst" in 1992. To celebrate the 500th annivers https://www.graztourismus.at/en/sightseeing-culture/sights/opera-house_shg_1472

Graz Opera

Past general music directors (GMD) of the company have included Niksa Bareza (1981-1990), Philippe Jordan (2001-2004), Johannes Fritzs (2006-2013), and Dirk Kaftan (2013-2017).[7] In the autumn of 2016, Oksana Lyrin made her first guest-conducting https://en.wikipedia.org/wiki/Graz_Opera

Figure 3: Graz Sightseeing Search Application

Current demonstration indices are rather small, which requires tests and efficiency checks over large indices. More requirements will be collected when it is taken up by others for new types of search applications.

ACKNOWLEDGEMENTS

This work has received funding from the European Union’s Horizon Europe research and innovation programme under grant agreement No 101070014 (OpenWebSearch.EU, <https://doi.org/10.3030/101070014>).

REFERENCES

- [1] R. Baeza-Yates and B. Ribeiro-Neto, *Modern Information Retrieval*. Addison Wesley, 1999.
- [2] M. Granitzer *et al.*, “Impact and development of an open web index for open web search,” *Journal of the Association for Information Science and Technology*, in press. 10.1002/as.1.24818
- [3] J. Lin *et al.*, “Supporting interoperability between open-source search engines with the common index file format,” in *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2020, pp. 2149–2152. 10.1145/3397271.3401404