Desing of Full-text Search for Database and Linkedin Social Network in Electrophysiology

Jan Štěbeták, Roman Mouček and Jan Koreň
Department of Computer Science and Engineering, University of West Bohemia, Univerzitní 8, Pilsen, Czech Republic

Keywords: Neuroinformatics, Electroencephalography, Event-related Potentials, Information Retrieval, Full-text Search, Index Design.

Abstract: EEG/ERP (electroencephalography, event-related potential) laboratories produce experimental data and metadata. Authors' research group has contributed to the building of a neuroinformatics infrastructure by developing and integrating data management and analytic tools for EEG/ERP research - the EEG/ERP Portal. With the development of the Portal and the increasing amount of data/metadata, a proper full text search mechanism for efficient information retrieval is necessary to improve the user experience. The presented solution combines search over data/metadata stored in an electrophysiological database and in the LinkedIn social network. Open source search engines, criteria, suitable engine selection, and index design are presented. Integration of the full-text solution to the EEG/ERP Portal is described.

1 INTRODUCTION

Accessing required information from a large set of data in a quick and user-friendly manner is no longer an unachievable goal. Advancements in the field of information retrieval in the last few decades have made its applications very common. Full-text search, as one of such applications, has in fact become an essential part of everyday's life in a modern society.

Our research group specializes in the research of brain activity; especially attention of drivers is investigated. We widely use the methods of electroencephalography (EEG) and event-related potentials (ERP). EEG/ERP experiments usually take long time and produce a lot of data.

As the members of the XXX National Node of International Neuroinformatics Coordinating Facility (INCF, 2011) we participate in definition and development of standardized formats for electrophysiology research. Our efforts, following INCF recommendations (Pelt and Horn, 2007) resulted in the central custom solution - the EEG/ERP Portal. This portal serves as a managing application for storing and managing experimental data and metadata.

In this paper we briefly present the common architecture of full-text search engines. Specific open-source full-text search engines are also described. Then the basic requirements and selection of a suitable full-text search engine are introduced. The next section is focused on creating a document model for indexed data and metadata stored in EEG/ERP Portal and articles or discussion in the LinkedIn social network. Since a model of our electrophysiological database is specific, an identification of domain entities had to be made first. We combine indexing data from both sources (the database and the LinkedIn) into a common index. The presented solution of full-text search is integrated into the EEG/ERP Portal. This solution will enable users of the EEG/ERP Portal to retrieve information from database and the LinkedIn in one place.

2 STATE OF THE ART

This section briefly describes the architecture of full-text search engines, open source engines and the EEG/ERP Portal.

2.1 Architecture of Full-text Search Engines

A full-text search engines comprise several steps in order to provide a user with search results to a given

Štěbeták J., Mouček R. and Koreň J..
Desing of Full-text Search for Database and Linkedin Social Network in Electrophysiology.
DOI: 10.5220/0004748602380243
In *Proceedings of the International Conference on Health Informatics* (HEALTHINF-2014), pages 238-243
ISBN: 978-989-758-010-9

query. Query is in this case a text phrase, optionally enriched by special operators which serve for refining the query. It is a user of the search system who comes up with the query, expecting that the system will fulfill his/her information need by returning relevant search results (Koren, 2013).

2.2 Available search engines

Indri (Indri, 2013) is an academic C++ based text search engine developed at the University of Massachusetts and is a part of the Lemur Project. Its API is accessible also from other languages such as Java or C#. In the technical paper (Turtle, Hedge and Rowe, 2012) it was shown that Indri is achievers in comparison with Lucene even slightly better results in terms of retrieval effectiveness for short queries, index size and performance.

Lucene (Lucene, 2013) is an open source Java-based search library. It can enrich applications by its ability to index data and search over them. According to (Middleton and Baeza-Yates, 2007), it belongs to the fastest engines and its performance is high especially when querying one- or two-word phrases. Its small size of index it creates is also a plus when the lack of memory could be an issue.

Solr (Solr, 2013) is an open source search server built on top of Lucene which runs within a servlet container, such as Jetty or Tomcat. Because it can be considered as a Lucene extension, most of the Lucene terminology is used for Solr as well. Solr extends Lucene by providing many useful features related to full-text search, e.g. keyword highlighting, faceted search, or rich document. Since Solr runs as a separate process, it communicates with applications via HTTP requests, which represent query data, and HTTP responses, representing search results found in the index, by exposing its REST-like API.

Sphinx (Sphinx, 2013) is an open source search server distributed under GPL license. It consists of an indexing tool, which is also referred to as indexer, and a searching daemon. Sphinx is written in C++ and is especially designed for indexing database systems (it integrates well with MySQL). Among its strengths we can name quick installation and deployment and a perfect online documentation for an open source project.

2.3 EEG/ERP Portal

The purpose of the EEG/ERP Portal (Jezek and Moucek, 2010) (Fig. 1) is to serve as a managing tool for EEG/ERP experiments that enables to share and interchange stored experiments (data, metadata, experimental scenarios, etc.) among interested laboratories.The EEG/ERP Portal is developed as a standalone product running on servers in our department. The usage of the Portal does not require any software installation, only a web browser.

Data and metadata are stored in the Oracle relational database. Access to the database is ensured with the Hibernate framework (Hibernate, 2013). Each database entity is presented by the Java POJO class. The model of the database will be described in Section 4.

The EEG/ERP Portal is registered in the LinkedIn social network. Articles, news, and discussion are stored in the database and also in the LinkedIn, where we created a research group. Posts in the LinkedIn are not stored in our database. Currently, we design a full-text search solution, which allows searching information in the database and also in the LinkedIn. That enables users of the EEG/ERP Portal to retrieve information from both sources (database and LinkedIn) in one place.

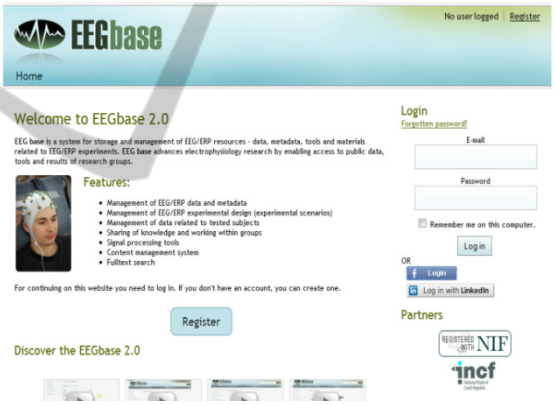

Figure 1: EEG/ERP Portal.

3 SELECTION OF SUITABLE FULL-TEXT SEARCH ENGINE

Based on our expectations (such as speed or independence of data sources) and used technologies by developing the EEG/ERP Portal, the selection of full-text search engine is restricted by the following criteria (Koren, 2013):

- *Speed* - the speeds of compared search engines differ only slightly and the observable differences depend on a specific use case. Therefore, all listed search engines can be considered as a suitable solution.

- *Integration with the EEG/ERP Portal* - The EEG/ERP Portal is based on Java technologies and this is why search engines providing Java API are easier to be integrated to the working infrastructure and therefore preferred.
- *Other Features and Extension* – Since we offer comfort to users of the EEG/ERP Portal, it is necessary to have a set of built-in features such as result highlighting, faceted search, synonym search, etc.
- *Independence of Data Sources* - The chosen search engine must be able to accept data from various sources and not to be limited only to one specific data source, such as relational database. The reason behind this is the mentioned need to index LinkedIn articles as well as to enable further possible indexing scenarios in the future, such as indexing *.pdf* or *XML* files.
- *Independence of other Technologies* - This criterion means that the search engine should not rely on a specific technology to be used. Dependence of Hibernate Search on Hibernate or heavy orientation of Sphinx on MySQL may serve as the examples of the search engines which perform well if certain conditions are met, but cannot run or do not perform well if not.
- *Community* - Numerous and active developer community also plays a big role in the final choice. The bigger community around the search engine is, the higher is the chance that the engine development will not stop early, new features will be introduced and found bugs will be resolved quickly.

The search engines were evaluated based on these criteria. Since it was stated for the speed criterion that its differences for the discussed search engines were not significant, it is not included in the final evaluation.

Note that the last criterion, *community* cannot be evaluated by an exact manner. This criterion involves the following four points: the size of mailing lists, the number of search results about the search engines found on Google, the number of related blog posts as well as the number of posts on specialized websites.

According to the evaluation, the Lucene based full-text search *Solr* was chosen.

4 INDEX DESIGN

The full-text search engines create an index document that ensures faster search through source texts or databases. This section is focused on identifying domain entities and design of a common index structure for the electrophysiological database that the EEG/ERP Portal uses and the LinkedIn social network.

4.1 Identification of Domain Entities

Data and metadata are logically stored into tables. Unfortunately, our database contains 71 tables. Therefore, we do not attach our data model as a figure. Instead of the data model, we describe core data that we store. There are tables containing core data. These tables represent domain entities. Other tables extend core data. In relation to POJO classes, the domain entities are represented by parent classes in the full-text search context. There exist one-to-one or one-to-many associations between parent POJO classes and child classes.

The following list shows domain entities. It also captures relations between parent and child classes (Koren, 2013):

- *Article* - Articles are represented by the Article class. It contains the title and text string fields that hold values of article title and its text, respectively. Articles and news are stored in the database and also in the LinkedIn social network. Articles can be commented on, so each article may have one or more article comments associated. Text of the comments themselves is contained in the *ArticleComment* instances.
- *Experiment* - This class has the field *environmentNote* to describe the experiment environment. Apart from this field, it also refers to related *Weather*, *Disease*, *DataFile*, *Hardware* and *Software* objects that include information about weather, diseases of a tested subject, related data files, and used hardware and software during an experiment, respectively.
- *Person* - This class contains a person's name, surname and a note about the person. Although there are also associations to other classes, it is not necessary to include them for indexing and searching purposes.
- *Research group* - This class keeps a name, title and description of a research group and as in the case of the *Person* class, its structure

matches the structure of its domain entity, so no other referenced class instances are needed.
- *Scenario* - This class contains its name, title and description. The class itself corresponds to its domain entity.

The full-text search functionality is oriented on searching and displaying information about the domain entities mentioned above.

Many POJO classes possess the *title* and *description* fields. It is also worth noticing that the fields *description*, *text*, and *note* contain a text information and have a similar meaning. On the other hand, there are several class fields that are distinctive only for a few or for a single domain entity such as the *temperature* field in the *Experiment* class.

4.2 Ensuring Document Uniqueness

When storing different types of documents (entities) under a single index, a problem of document uniqueness arises. Although a relational database record can be uniquely identified by its primary key and a source table, its information about the source table is lost after its corresponding Solr document is created. Hence, the original uniqueness is lost as there might be documents created from records coming from different tables and having the same id.

The following fields related to unique identification of documents were added in the Solr schema file:
- *Id* - The purpose of this field is to keep an original ID value.
- *Class* - Identifies a type of indexed document (entity). In combination with *id* value it provides an identification of a specific object.
- *Uuid* - This field serves as a unique identifier for each document in the index. It consists of two concatenated parts. The first part is the class name of an indexed object; the second part is the ID value of the object. For example, the *uuid* value of an article with ID 20 is *cz.zcu.kiv.eegdatabase.data.pojo.Article20*.

4.3 Proposed Index Structure

Based on the structure of the database, object hierarchy, and the fields contained in these objects, the following index fields were configured:
- *Title* - It is a title of a parent object.
- *Text* - It is a longer textual sequence such as description or a note of parent object. It also includes text of articles or discussion stored in the database or the LinkedIn within our research group.
- *Name* - It contains a person's name.
- *Source* - It determines if the object's source is the database or LinkedIn (in case of LinkedIn articles).
- *Temperature* - It stores a temperature during an experiment
- *Param_datatype* – A model of our database allows adding additional information about core tables. This field represents a data type or value of an additional parameter of a parent POJO class.
- *File_mimetype* - It stores *mimetype* values of stored data file of parent POJOs. In this case, it concerns the *Scenario* class.
- *Child_title* - It is a multi-valued field which stores all titles of child objects.
- *Child_text* - This field is analogous to the *text* field. The difference is that it is a multi-valued field and stores all textual values of child objects.
- *Child_File_mimetype* - It is used for the same reasons as the *File_mimetype* field, except that it is used for child object values (e.g. *DataFile* class).

5 INTEGRATION INTO EEG/ERP PORTAL

This section describes necessary implementation steps to integrate the Solr based full-text search into the EEG/ERP Portal.

To separate common functionality of indexing from the specialized step during which a Solr document is built, the *Template Method* design pattern (Gamma, et al. 1994) was used. By applying this design pattern, the class hierarchy depicted in Figure 2 was implemented.

The parent abstract class *Indexer* takes care of performing the generic steps only by providing an *HttpSolrServer* instance.

PojoIndexer class is responsible for indexing data from the database. We created a set of annotations, which POJO classes are marked with. The annotation *@Indexed* means that the marked class should be indexed. The annotation *@SolrField* marks an attribute of a class and set the relationship with belonging field in the index schema. The annotation *@SolrId* marks an attribute representing unique identifier of indexed class. Indexing using annotation interface works in cooperation with Java reflection.

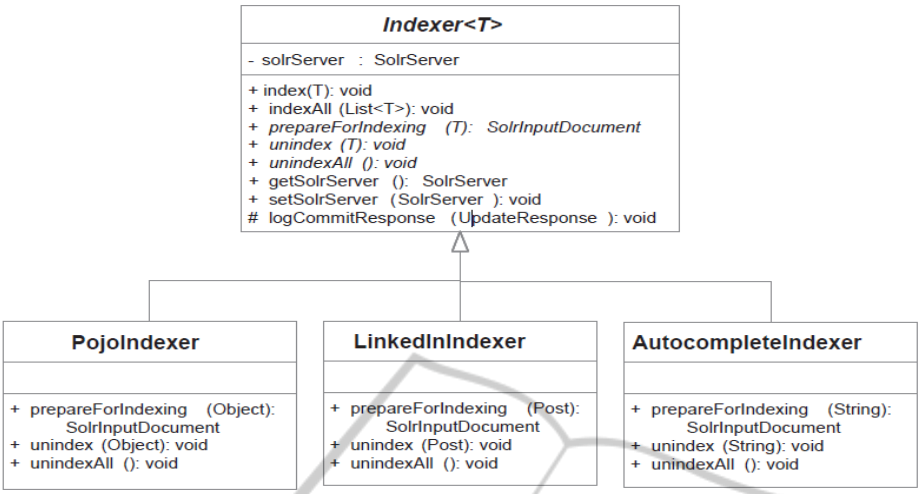

Figure 2: UML diagram of Indexer classs (Koren, 2013).

LinkedInIndexer class is responsible for indexing articles, news and, discussion from the LinkedIn social network within our research group. We use LinkedIn REST api via Spring Social.
The following example shows the usage of field selectors in the REST call. The REST call in the listing gets full information about twenty latest LinkedIn articles published in the EEG/ERP Portal group. Note the *group-id* placeholder which is later substituted by the real value of the EEG/ERP Portal group ID.

```
http://api.linkedin.com/v1/groups/{group-id}/ posts: (creation -timestamp, title, summary, id, creator:(first -name ,last -name))? count=20& start =0& order=recency"
```

Data stored in the database are indexed on change. However, LinkedIn articles are independent of the EEG/ERP Portal and cannot be indexed on change. Therefore, a periodical indexing was created. In Figure 3, the created user interface is shown.

5.1 Addition of a New Entity

The presented index schema is dependent on the database structure. It usually happens that the database is modified, new entities are added. When a new entity is added, two situations arise:

- A new entity includes metadata that are already covered by the index schema (e.g. titles, texts, notes, or descriptions). In this case, no change of the index schema is required.
- A new entity includes new type of metadata. In this case, it is necessary to add this attribute as a field into the index schema.

6 RESULTS

We implemented and successfully integrated a Solr based solution of full-text search into the EEG/ERP Portal. The Solr server runs at the address `147.228.63.134/solr`.

We created several JUnit tests. These tests covers use cases such as searching a string in the database or LinkedIn and returning whole record, indexing and unindexing (when data is removed from the database) of records, or verification of proper indexing (Koren, 2013).

Currently, the database includes approximately 200 experiments. Time requested to display full-text search results is 1s.

7 CONCLUSIONS

We created a research group in LinkedIn. This group is connected to the EEG/ERP Portal. Users of our Portal can contribute to the Portal or the LinkedIn. Therefore, we presented solution combines full-text search data/metadata from an electrophysiological database and the LinkedIn social network. Based on requirements described in Section 3, the full-text search engine Solr was chosen.

The index schema presented in Section 4 is designed according to our database. However, new entities can be easily added. If a new entity includes type of metadata that are already included in the index schema, no change of the index schema is

Figure 3: Full-text search user interface.

required. Otherwise, the new attribute can be simply added into the schema.

Our solution puts together search over data and metadata from the database and news and discussion from the LinkedIn within our research group. Integration to the EEG/ERP Portal allows users searching from both sources in one place.

Our future work will focus on improvement of the full-text search solution. It includes defining synonyms of electrophysiology research and integrating them into the full-text search solution in the EEG/ERP Portal.

ACKNOWLEDGEMENTS

The work was supported by the UWB grant SGS-2013-039 Methods and Applications of Bio- and Medical Informatics.

REFERENCES

International Neuroinformatics Coordinating Facility (INCF, 2011), http://www.incf.org/about/what-is-neuroinformatics

Pelt, J. van, Horn J. van, 2007, Workshop report. *1st INCF Workshop on Sustainability of Neuroscience Databases*. Stockholm.

Middleton, C., Baeza-Yates, R., 2007, "A comparison of open source search engines," tech. rep., 10. http://wrg.upf.edu/WRG/dctos/Middleton-Baeza.pdf

Indri homepage (2013), http:// www.lemurproject.org/indri.php

Turtle, H., Hegde, Y., Rowe, S. A., 2012, "Yet another comparison of lucene and indri performance", in *Proceedings of the SIGIR 2012 Workshop on Open Source Information Retrieval*, vol. 46, pp. 64–67.

Lucene homepage (2013), http://lucene.apache.org/

Sphinx homepage (2013), http://sphinxsearch.com

Solr homepage (2013), http://lucene.apache.org/solr

Gamma E., Helm, R., Johnson, R., Vlissides, J., 1994, *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley Professional, 1 ed.

Hibernate (2013), http://www.hibernate.org/

Jezek, P., Moucek, R., 2010, "System for storage and management of EEG/ERP experiments – generation on ontology" *Databases and Information System Integration* vol. 1, Madeira: *SciTePress, ICEIS*.

Koren, J., 2013 *"Fulltext search in the database and in text of social netwoks"* Master Thesis, Pilsen.