## Objective

Create a generic framework for evaluation of alternate retrieval methods/softwares for the given set of documents.

## Background

I have a business need in my current work to choose either Oracle Free Text Search or Solr to index text data. This indexed data will be integrated and used for relevancy search. Thanks to the solid understanding I gained from this course “CS-410 Text Information Systems”, I was able to propose and got approval for this initiative.

Note: Also at my work place a third-party software Elsevier is currently used to index scientific grant related text like abstract, specific aims. The index score output is used to categorize research projects under various disease categories. In my original project proposal, I was considering to perform analysis of softwares that can be considered for a potential replacement for vendor provided software. But I didn’t get approval for this original proposal from team management. So I have evaluated information retrieval softwares.

## 1 Overview of Functions

### 1.1 Oracle Free Text Search Creation

* + Create a table with the required fields/columns.
  + Identify the column that will be used in the next for text search.
  + Create a context index on the column that will be used for text search.

### 1.2 Solr Index Creation

* + Create a schema with the required fields/columns.
  + Configure the query parser Lucene/Edismax.
  + Index the data.

### 1.3 Evaluation

* + Select query terms, phrases and other factors for evaluation.
  + Execute query and get results from evaluation software 1 – Oracle text search
  + Execute query and get results from evaluation software 2 – Solr
  + Get explicit feedback from users (Gold copy)
  + Calculate precision, recall and F-scores

## 2 Implementation

### 2.1 Oracle Free Text Search Creation

Oracle 12c is used for this evaluation. The assumption is the user who is creating the table already as required privileges to create tables and context index.

Reference: <https://docs.oracle.com/database/121/CCAPP/toc.htm>

create table extractions\_t(

extraction\_id NUMBER(10) not null,

appl\_id NUMBER(10) not null,

extracted\_text CLOB,

template\_section\_code VARCHAR2(3),

fy NUMBER

);

create index ARCH\_EXTRACTED\_TEXT on EXTRACTIONS\_T (EXTRACTED\_TEXT)

indextype is CTXSYS.CONTEXT;

### 2.2 Solr Index Creation

* Schema Creation (partial managed-schema.xml is given below)

Note extractText is configured as Text English general (text\_en). This means Porter stemmer and stopwords will be applied to index data.

<field name="applId" type="long" indexed="true" required="true" stored="true"/>

<field name="fy" type="long" indexed="true" required="true" stored="true"/>

<field name="extractText" type="text\_en" indexed="true" stored="true"/>

<field name="id" type="string" multiValued="false" indexed="true" required="true" stored="true"/>

<field name="templateSectionCode" type="string" indexed="true" required="true" stored="true"/>

* Method that creates and add data to Index using SolrJ client

private void addIndex() throws IOException, SolrServerException {

SolrClient client = new HttpSolrClient.Builder("http://localhost:8983/solr/archived\_extractions").build();

Collection<ArchivedExtraction> docs = new ArrayList<ArchivedExtraction>();

int i = 0;

List<ExtractionRecord> extRecords = null;

List<Integer> fys = new ArrayList<>();

fys.add(2013);

fys.add(2014);

fys.add(2015);

fys.add(2016);

fys.add(2017);

for (Integer fy :fys) {

List<Long> applIDs = textSearchService.getApplIDsByFy(fy);

BatchedList<Long> batchedList = new BatchedList<Long>(new ArrayList<Long>(applIDs), BatchedList.ORACLE\_IN\_CLAUSE\_SIZE);

while(batchedList.hasNextBatch())

{

List<Long> applIdList = batchedList.nextBatch();

extRecords = textSearchService.getFreeTextSearchResults(applIdList);

for (ExtractionRecord extRecord : extRecords) {

ArchivedExtraction doc = new ArchivedExtraction();

doc.setApplId(extRecord.getApplId().toString());

doc.setId(extRecord.getExtractionId()+"-"+extRecord.getApplId());

//doc.setDocCreatedDate(LocalDate.now());

doc.setExtractText(extRecord.getExtractedText());

doc.setTemplateSectionCode(extRecord.getTemplateSectionCode());

doc.setFy(extRecord.getFy());

client.addBean(doc);

i++;

if(i%100==0) client.commit();

}

}

client.commit();

}

}

### 2.3 Installing MyProject Web application

* Copy the war file MyProject.war to \Tomcat\8\webapps
* Start the tomcat server by \Tomcat\8\bin\startup.bat
* Edit the \Tomcat\8\wtpwebapps\MyProject\WEB-INF\MyProject-servlet.xml to configure datasource. Highlighted the values that need to be added for data access layer configuration.

<bean id="datasource" class="org.springframework.jdbc.datasource.DriverManagerDataSource">

<property name="driverClassName" value="oracle.jdbc.driver.OracleDriver" />

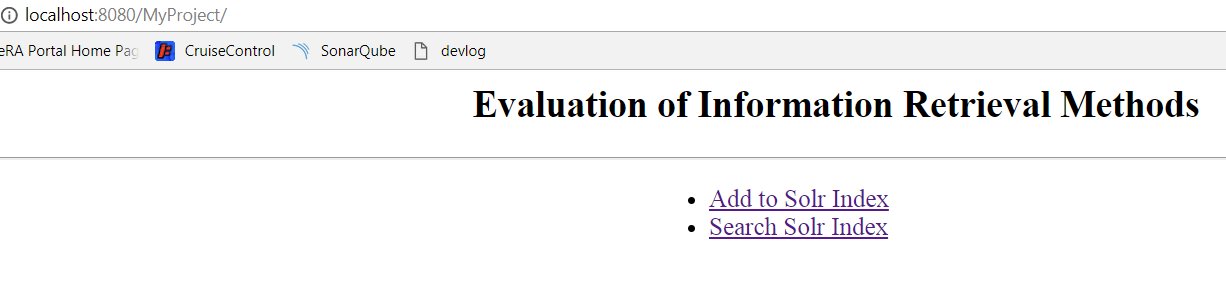
<property name="url" value="jdbc:oracle:thin:@(DESCRIPTION=(ENABLE=BROKEN)(ADDRESS\_LIST=(ADDRESS=(PROTOCOL=TCP)(HOST=localhost)(PORT=1530)))(CONNECT\_DATA=(SERVICE\_NAME=orcl)(SERVER=DEDICATED)))" />

<property name="username" value="scott" />

<property name="password" value="tiger" />

</bean>

* Stop the server by running by \Tomcat\8\bin\shutdown.bat
* Restart the tomcat server by \Tomcat\8\bin\startup.bat
* Access the application using URL <http://localhost:8080/MyProject/>



### 2.4 Evaluation

#### Table to store raw Statistics

Table to store appl\_id which is unique identifier for a research project and its relevance to the given query search\_term. If the given project is identified as relevant for a search term per explicit user feedback then the value of gold\_standard\_flag will be set to 1 else the  value will be 0. Similarly if oracle free text search (fts) identifies the project as relevant to a search term the value of oracle\_fts will be set to 1 else 0. If solr query identifies the project as relevant to a search term the value of will be set to 1 else 0.

CREATE TABLE stats\_calc (search\_term VARCHAR2(100),

                        appl\_id number,

                        gold\_standard\_flag number default 0,

                        oracle\_fts\_flag number default 0,

                        Solr\_idx\_flag  number default 0

                        );

ALTER TABLE  stats\_calc ADD CONSTRAINT stats\_calc\_pk PRIMARY KEY (appl\_id,search\_Term);

#### Oracle Query For Search Term

select appl\_id from

  ( select distinct arch.appl\_id appl\_id

  from archived\_extractions\_t arch

  where contains(arch.extracted\_text, 'Knee Osteoarthritis' , 1) > 0

    and latest\_code = 'Y'

    and arch.template\_section\_code is null

    and arch.fy= 2017

  );

#### Solr Query For Search Term

private Set<Long> selectData( String queryTerm) {

SolrClient client = new HttpSolrClient.Builder("http://localhost:8983/solr/archived\_extractions").build();

HashSet<Long> applSet = new HashSet();

try {

SolrQuery query = new SolrQuery();

Long fy = (long) 2017;

String searchquery ="fy:"+fy;

query.setQuery("extractText:"+queryTerm );

query.addFilterQuery(searchquery);

QueryResponse response = client.query(query);

SolrDocumentList results = response.getResults();

for (int i = 0; i < results.size(); ++i) {

applSet.add((Long) results.get(i).getFieldValue("applId"));

}

} catch (SolrServerException e) {

// TODO Auto-generated catch block

e.printStackTrace();

} catch (IOException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

return applSet;

}

## 3. Usage

### Step 1: Store explicit feedback

Creating Gold Standard(user feedback) Upload for query term Knee Osteoarthritis provided by users

|  |
| --- |
| insert into stats\_calc(search\_term,appl\_id,gold\_standard\_flag) values ('Knee Osteoarthritis',9435379,1); |
| insert into stats\_calc(search\_term,appl\_id,gold\_standard\_flag) values ('Knee Osteoarthritis',9413126,1); |
| insert into stats\_calc(search\_term,appl\_id,gold\_standard\_flag) values ('Knee Osteoarthritis',9386212,1); |
| insert into stats\_calc(search\_term,appl\_id,gold\_standard\_flag) values ('Knee Osteoarthritis',9385849,1); |
| insert into stats\_calc(search\_term,appl\_id,gold\_standard\_flag) values ('Knee Osteoarthritis',9375095,1); |
| insert into stats\_calc(search\_term,appl\_id,gold\_standard\_flag) values ('Knee Osteoarthritis',9371389,1); |
| insert into stats\_calc(search\_term,appl\_id,gold\_standard\_flag) values ('Knee Osteoarthritis',9364179,1); |
| insert into stats\_calc(search\_term,appl\_id,gold\_standard\_flag) values ('Knee Osteoarthritis',9353269,1); |
| insert into stats\_calc(search\_term,appl\_id,gold\_standard\_flag) values ('Knee Osteoarthritis',9310337,1); |
| insert into stats\_calc(search\_term,appl\_id,gold\_standard\_flag) values ('Knee Osteoarthritis',9197607,1); |

### Step 2: Store Oracle search results for query term Knee Osteoarthritis

MERGE INTO stats\_Calc sc USING ( select distinct arch.appl\_id appl\_id ,'Knee Osteoarthritis' search\_term

from extractions\_t arch

where contains(arch.extracted\_text, 'Knee Osteoarthritis' , 1) > 0

and arch.template\_section\_code is null

and arch.fy= 2017

) ex

ON (ex.appl\_id = sc.appl\_id AND ex.search\_term = sc.search\_Term)

WHEN MATCHED THEN UPDATE SET sc.oracle\_fts\_flag =1

WHEN NOT MATCHED THEN INSERT(appl\_id,search\_Term,oracle\_fts\_Flag) VALUES (ex.appl\_id,ex.search\_term,1);

;

### Step 3: Store Solr search results for query term Knee Osteoarthritis

public void setStatsCalc(Long applId, String searchTerm , String flag) {

MapSqlParameterSource params = new MapSqlParameterSource();

StringBuilder nativeQL = new StringBuilder("MERGE INTO stats\_Calc sc USING (SELECT :appl appl\_id,:st search\_term FROM dual) ex " );

nativeQL.append(" ON (ex.appl\_id = sc.appl\_id AND ex.search\_term = sc.search\_Term) " );

nativeQL.append(" WHEN MATCHED THEN UPDATE SET sc." + flag + " =1 " );

nativeQL.append(" WHEN NOT MATCHED THEN INSERT(appl\_id,search\_Term,"+flag+" ) VALUES (ex.appl\_id,ex.search\_term,1) " );

params.addValue("appl",applId);

params.addValue("st", searchTerm);

this.namedParameterJdbcTemplate.update(nativeQL.toString(), params);

}

### Step 4: Calculate Stats

Stats calculation for one Software (Oracle FTS) is given here. Same steps can be repeated for another software evaluated.

|  |  |  |
| --- | --- | --- |
|  | User Y | User N |
| System + | TP | FP |
| System - | FN | TN |

Precision  = TP/(TP+FP)

Recall  =   TP/(TP+FN)

Oracle free text search  TP = Count(records) where oracle\_fts\_flag = 1 and gold\_standard\_flag = 1

Oracle free text search FP = Count(records) where oracle\_fts\_flag = 1 and gold\_standard\_flag = 0

Oracle free text search FN= Count(records) where oracle\_fts\_flag = 0 and gold\_standard\_flag = 1

Oracle Precision free text =  oracle fts  TP/(oracle fts TP +oracle fts FP)

select count(distinct appl\_id) from stats\_Calc where search\_Term='Knee Osteoarthritis'

and oracle\_fts\_flag = 1 and gold\_standard\_flag = 1; *-- TP*

select count(distinct appl\_id) from stats\_Calc where search\_Term='Knee Osteoarthritis'

and oracle\_fts\_flag = 1 and gold\_standard\_flag = 0; *--FP*

select count(distinct appl\_id) from stats\_Calc where search\_Term='Knee Osteoarthritis'

and oracle\_fts\_flag = 0 and gold\_standard\_flag = 1; *--FN*

select 31/(31+129) from dual; *-- Precision*

select 31/(31+32) from dual; *-- Recal*

select count(distinct appl\_id) from stats\_Calc where search\_Term='Knee Osteoarthritis'

and solr\_idx\_flag = 1 and gold\_standard\_flag = 1;

select count(distinct appl\_id) from stats\_Calc where search\_Term='Knee Osteoarthritis'

and solr\_idx\_flag = 1 and gold\_standard\_flag = 0;

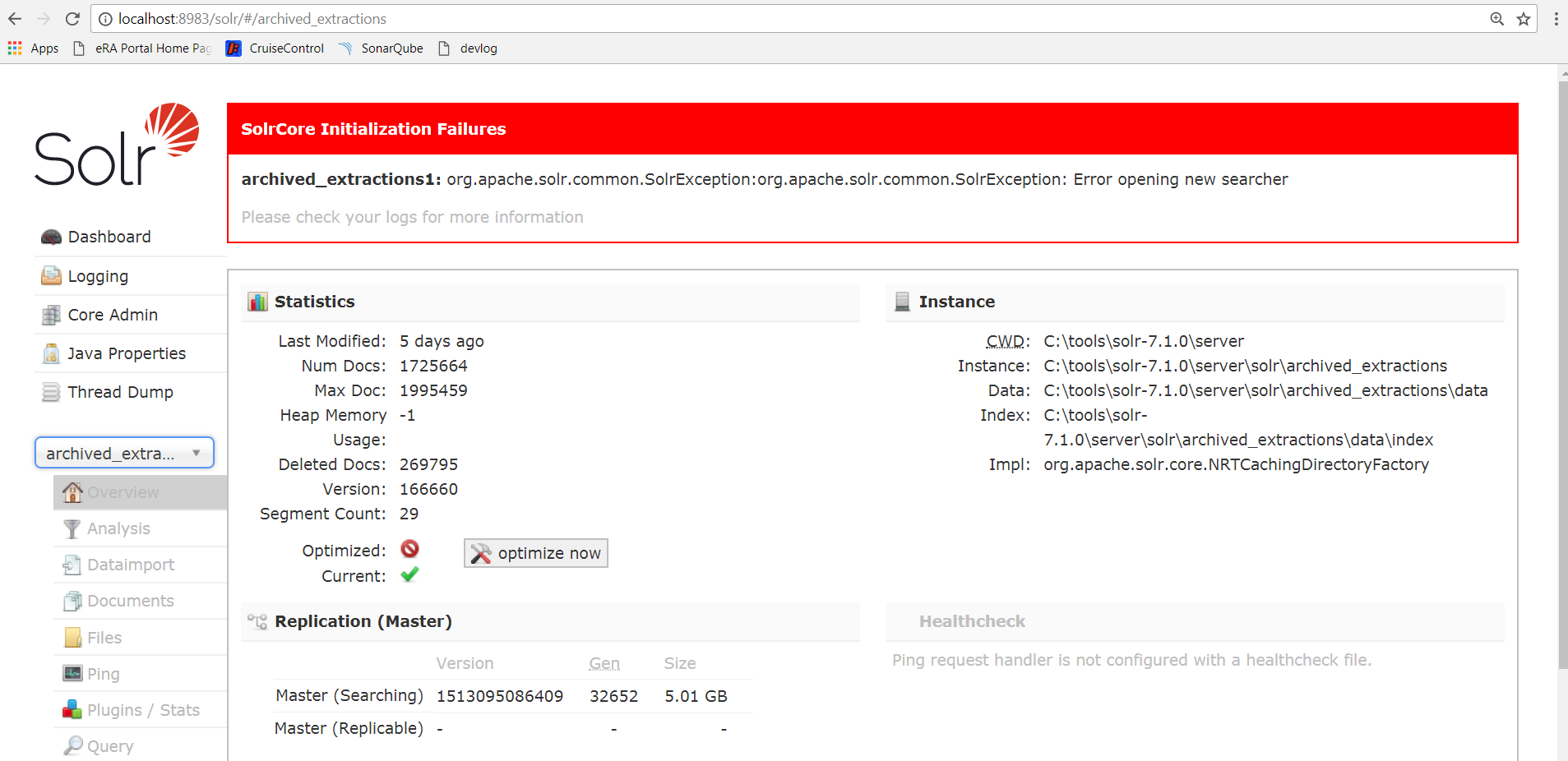
select count(distinct appl\_id) from stats\_Calc where search\_Term='Knee Osteoarthritis'

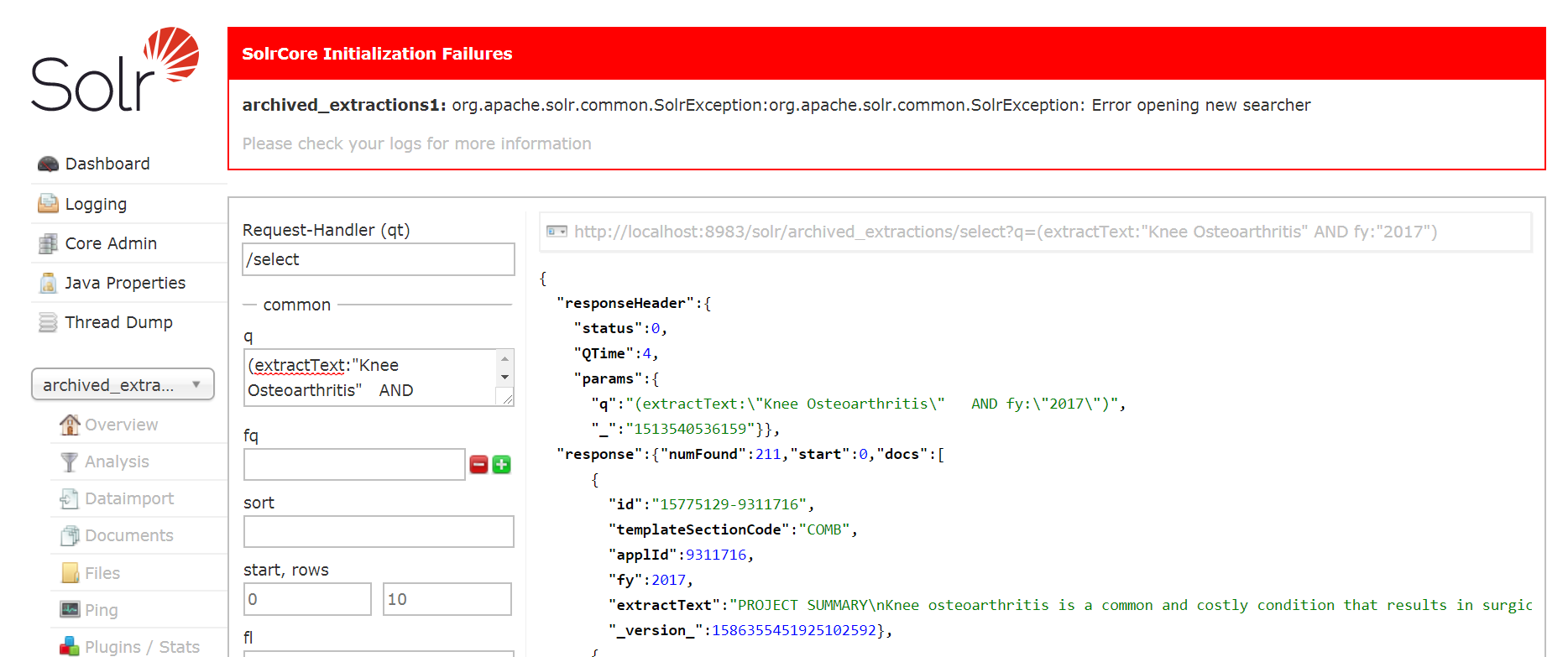
and solr\_idx\_flag = 0 and gold\_standard\_flag = 1;

select 34/(142+34) from dual;

select 34/(34+29) from dual;

# 4. Solr Admin Screenshots





# 5. Evaluation Framework Webpage Screenshots

