OPAL
OPEN DATA PORTAL

Deliverable D8.5
Dokumentation

Autoren: Adrian Wilke

Veröffentlichung	Öffentlich
Fälligkeitsdatum	30.06.2020 (vor Corona-Verlängerung)
Fertigstellung	Laufend aktualisiert
Arbeitspaket	AP8
Typ	Bericht
Status	Laufend aktualisiert
Version	-

Kurzfassung:
Dieses Deliverable enthält eine Dokumentation zu den entwickelten OPAL Komponenten.
Eine Online-Version kann unter folgender URL zugegriffen werden:
https://github.com/projekt-opal/doc

Schlagworte:
OPAL, Dokumentation

mFUND
Das Startkapital für die Mobilität 4.0

Bundesministerium
für Verkehr und
digitale Infrastruktur

Contents

OPAL Named Entity Disambiguation and indexing component

AGDISTIS (Agnostic Named Entity Disambiguation) aims at delivering a framework for disambiguating a priori annotated named entities.
It has been extended in the project **LIMBO** to integrate the search engine Elasticsearch.
In the project **OPAL**, the geographical database LauNuts was integrated.

Links to other versions:

- AGDISTIS
- AGDISTIS Elasticsearch

Installation

The underlying indexing system is Elasticsearch 6.6.
The installation with Docker is described at the Elasticsearch documentation.
The following commands can be used:

- `docker pull docker.elastic.co/elasticsearch/elasticsearch:6.6.0`
- `docker run -p 9200:9200 -p 9300:9300 -e "discovery.type=single-node"docker.elastic.co/elasticsearch/elasticsearch:6.6.0`

To import the required data, perform the following steps:

- Download and extract the LauNuts dataset
- Set the property *folderWithTTLFiles* in the configuration file *agdistis.properties*
- Run the *org.aksw.agdistis.indexWriter.TripleIndexCreator*
- Validate the data status on http://localhost:9200/_cat/indices?v using the Elasticsearch cat indices API

Start the webservices using org.aksw.agdistis.webapp.RunApp.

Usage

- Use the AGDISTIS webservice URLs to disambiguate places
- Disambiguation example: `curl --data-urlencode "text='The city of <entity>Paderborn</entity> has over 150,000 inhabitants.'"-d type='agdistis'http://localhost:8080/AGDISTIS`
- Candidates example: `curl --data-urlencode "text='The city of <entity>Hamburg</entity> is also a federal state.'"-d type='candidates'http://localhost:8080/AGDISTIS`

Catfish

Data cleaning component.

Usage with Apache Maven

Add the following lines to your `pom.xml` configuration file:

```
<dependencies>
    <dependency>
        <groupId>org.dice-research.opal</groupId>
        <artifactId>catfish</artifactId>
        <version>[1,2)</version>
    </dependency>
</dependencies>

<repositories>
    <repository>
        <id>maven.aksw.internal</id>
        <name>AKSW Repository</name>
        <url>http://maven.aksw.org/archiva/repository/internal</url>
    </repository>
    <repository>
        <id>maven.aksw.snapshots</id>
        <name>AKSW Snapshot Repository</name>
        <url>http://maven.aksw.org/archiva/repository/snapshots</url>
    </repository>
</repositories>
```

Available versions are listed at maven.aksw.org.

Example: Configuration

```
import java.io.File;
import org.apache.jena.rdf.model.Model;
import org.apache.jena.rdf.model.RDFNode;
import org.apache.jena.rdf.model.Resource;
import org.apache.jena.rdf.model.Statement;
import org.apache.jena.rdf.model.StmtIterator;
import org.apache.jena.vocabulary.DCAT;
import org.apache.jena.vocabulary.DCTerms;
import org.apache.jena.vocabulary.RDF;
import org.dice_research.opal.catfish.Catfish;
import org.dice_research.opal.common.utilities.FileHandler;
import org.dice_research.opal.common.vocabulary.Opal;

public class Example {

  /**
   * Cleans data.
```

```
18      *
19      * @param turtleInputFile  A TURTLE file to read
20      * @param turtleOutputFile A TURTLE file to write results
21      * @param datasetUri       A URI of a dcat:Dataset inside the TURTLE data
22      *
23      * @see https://www.w3.org/TR/turtle/
24      * @see https://www.w3.org/TR/vocab-dcat/
25      */
26     public void cleanMetadata(File turtleInputFile, File turtleOutputFile, String
            datasetUri) throws Exception {
27
28       // Load TURTLE file into model
29       Model model = FileHandler.importModel(turtleInputFile);
30
31       CleaningConfig cleaningConfig = new CleaningConfig();
32       // Remove blank nodes, which are not subject of triples
33       // (optional method call, default: true)
34       cleaningConfig.setCleanEmptyBlankNodes(true);
35
36       // Remove triples with literals as object, which contain no value or
             unreadable.
37       // And also extract Language Tag and DataType if it is mistakenly inside
             the string
38       // (optional method call, default: true)
39       cleaningConfig.setCleanLiterals(true);
40
41       // Check dct:format and dcat:mediaType for values and create new triples.
42       // (optional method call, default: true)
43       cleaningConfig.setCleanFormats(true);
44
45       Catfish catfish = new Catfish(cleaningConfig);
46
47       // Update model
48       catfish.processModel(model, datasetUri);
49
50       // Example for requesting formats
51       printFormats(model, datasetUri);
52
53       // Write updated model into TURTLE file
54       FileHandler.export(turtleOutputFile, model);
55     }
56   }
```

Example: Requesting formats

```
1  /**
2   * Example for requesting formats.
3   *
4   * Generated formats are of type http://projekt-opal.de/Format.
5   */
6  public void printFormats(Model model, String datasetUri) {
7
8    // Go through Distributions of current Dataset
```

```
9     StmtIterator distributionIterator = model.getResource(datasetUri).
          listProperties(DCAT.distribution);
10    while (distributionIterator.hasNext()) {
11      RDFNode rdfNode = distributionIterator.next().getObject();
12      if (rdfNode.isResource()) {
13        Resource distribution = rdfNode.asResource();
14
15        // Get formats of current Distribution
16        StmtIterator formatIterator = distribution.listProperties(DCTerms.format)
              ;
17        while (formatIterator.hasNext()) {
18          RDFNode format = formatIterator.next().getObject();
19
20          // Check if type is http://projekt-opal.de/Format
21          if (format.isResource()) {
22            Statement statement = format.asResource().getProperty(RDF.type);
23            if (statement != null
24                && statement.getObject().asResource().getURI().equals(Opal.
                      OPAL_FORMAT.getURI())) {
25
26              // Prints, e.g.
27              // http://projekt-opal.de/format/pdf
28              // http://projekt-opal.de/format/html
29              System.out.println(format);
30            }
31          }
32        }
33      }
34    }
35  }
```

OPAL Civet

OPAL metadata quality component.

Usage with Apache Maven

Add the following lines to your `pom.xml` configuration file:

```
<dependencies>
    <dependency>
        <groupId>org.dice-research.opal</groupId>
        <artifactId>civet</artifactId>
        <version>[2,3)</version>
    </dependency>
</dependencies>

<repositories>
    <repository>
        <id>maven.aksw.internal</id>
        <name>AKSW Repository</name>
        <url>http://maven.aksw.org/archiva/repository/internal</url>
    </repository>
    <repository>
        <id>maven.aksw.snapshots</id>
        <name>AKSW Snapshot Repository</name>
        <url>http://maven.aksw.org/archiva/repository/snapshots</url>
    </repository>
</repositories>
```

Available versions are listed at maven.aksw.org.

Example

```
import java.io.File;
import org.apache.jena.rdf.model.Model;
import org.dice_research.opal.civet.Civet;
import org.dice_research.opal.common.utilities.FileHandler;

public class Example {

    /**
     * Computes quality metric scores (measurements).
     *
     * @param turtleInputFile  A TURTLE file to read
     * @param turtleOutputFile A TURTLE file to write results
     * @param datasetUri       A URI of a dcat:Dataset inside the TURTLE data
     *
     * @see https://www.w3.org/TR/turtle/
     * @see https://www.w3.org/TR/vocab-dcat/
     */
```

```
18      public void evaluateMetadata(File turtleInputFile, File turtleOutputFile,
            String datasetUri) throws Exception {
19
20          // Load TURTLE file into model
21          Model model = FileHandler.importModel(turtleInputFile);
22
23          Civet civet = new Civet();
24
25          // If long running metrics should be included.
26          // (optional method call, default: false)
27          civet.setIncludeLongRunning(true);
28
29          // If it should be logged, if a measurement could not be computed
30          // (optional method call, default: true)
31          civet.setLogIfNotComputed(false);
32
33          // If existing measurements should be removed
34          // (optional method call, default: true)
35          civet.setRemoveMeasurements(true);
36
37          // Update model
38          civet.processModel(model, datasetUri);
39
40          // Write updated model into TURTLE file
41          FileHandler.export(turtleOutputFile, model);
42      }
43  }
```

Example input:

```
1   <https://www.kreis-paderborn.de/kreis_paderborn/geoportal/opendata/>
2           a       <http://www.w3.org/ns/dcat#Dataset> ;
3           <http://purl.org/dc/terms/accrualPeriodicity>
4                   "IRREG" ;
5           <http://purl.org/dc/terms/license>
6                   "http://www.govdata.de/dl-de/by-2-0" ;
7           <http://purl.org/dc/terms/publisher>
8                   <https://www.kreis-paderborn.de/> ;
9           <http://www.w3.org/ns/dcat#distribution>
10                  <https://www.kreis-paderborn.de/opendata/alkis/> ;
11          <http://www.w3.org/ns/dcat#keyword>
12                  "map" , "Paderborn" .
13
14  <https://www.kreis-paderborn.de/opendata/alkis/>
15          a       <http://www.w3.org/ns/dcat#Distribution> ;
16          <http://www.w3.org/ns/dcat#accessURL>
17                  "https://www.kreis-paderborn.de/opendata/alkis/" .
18
19  <https://www.kreis-paderborn.de/>
20          a       <http://xmlns.com/foaf/0.1/Agent> .
```

Example output:

```
1   <https://www.kreis-paderborn.de/kreis_paderborn/geoportal/opendata/>
2           a       <http://www.w3.org/ns/dcat#Dataset> ;
```

```
        <http://purl.org/dc/terms/accrualPeriodicity>
                "IRREG" ;
        <http://purl.org/dc/terms/license>
                "http://www.govdata.de/dl-de/by-2-0" ;
        <http://purl.org/dc/terms/publisher>
                <https://www.kreis-paderborn.de/> ;
        <http://www.w3.org/ns/dcat#distribution>
                <https://www.kreis-paderborn.de/opendata/alkis/> ;
        <http://www.w3.org/ns/dcat#keyword>
                "map" , "Paderborn" ;
        <http://www.w3.org/ns/dqv#hasQualityMeasurement>
                [ a       <http://www.w3.org/ns/dqv#QualityMeasurement> ;
                  <http://www.w3.org/ns/dqv#isMeasurementOf>
                          <http://metric.projekt-opal.de/ProviderIdentity> ;
                  <http://www.w3.org/ns/dqv#value>
                          5
                ] ;
        <http://www.w3.org/ns/dqv#hasQualityMeasurement>
                [ a       <http://www.w3.org/ns/dqv#QualityMeasurement> ;
                  <http://www.w3.org/ns/dqv#isMeasurementOf>
                          <http://metric.projekt-opal.de/MultipleSerializations
                              > ;
                  <http://www.w3.org/ns/dqv#value>
                          0
                ] ;
        <http://www.w3.org/ns/dqv#hasQualityMeasurement>
                [ a       <http://www.w3.org/ns/dqv#QualityMeasurement> ;
                  <http://www.w3.org/ns/dqv#isMeasurementOf>
                          <http://metric.projekt-opal.de/LicenseAvailable> ;
                  <http://www.w3.org/ns/dqv#value>
                          5
                ] ;
        <http://www.w3.org/ns/dqv#hasQualityMeasurement>
                [ a       <http://www.w3.org/ns/dqv#QualityMeasurement> ;
                  <http://www.w3.org/ns/dqv#isMeasurementOf>
                          <http://metric.projekt-opal.de/MetadataQuality> ;
                  <http://www.w3.org/ns/dqv#value>
                          3
                ] ;
        <http://www.w3.org/ns/dqv#hasQualityMeasurement>
                [ a       <http://www.w3.org/ns/dqv#QualityMeasurement> ;
                  <http://www.w3.org/ns/dqv#isMeasurementOf>
                          <http://metric.projekt-opal.de/Retrievability> ;
                  <http://www.w3.org/ns/dqv#value>
                          5
                ] ;
        <http://www.w3.org/ns/dqv#hasQualityMeasurement>
                [ a       <http://www.w3.org/ns/dqv#QualityMeasurement> ;
                  <http://www.w3.org/ns/dqv#isMeasurementOf>
                          <http://metric.projekt-opal.de/Categorization> ;
                  <http://www.w3.org/ns/dqv#value>
                          3
                ] ;
        <http://www.w3.org/ns/dqv#hasQualityMeasurement>
                [ a       <http://www.w3.org/ns/dqv#QualityMeasurement> ;
                  <http://www.w3.org/ns/dqv#isMeasurementOf>
```

```
                        <http://metric.projekt-opal.de/DataFormat> ;
                  <http://www.w3.org/ns/dqv#value>
                        0
                ] ;
        <http://www.w3.org/ns/dqv#hasQualityMeasurement>
                [ a       <http://www.w3.org/ns/dqv#QualityMeasurement> ;
                  <http://www.w3.org/ns/dqv#isMeasurementOf>
                        <http://metric.projekt-opal.de/UpdateRate> ;
                  <http://www.w3.org/ns/dqv#value>
                        1
                ] .

<https://www.kreis-paderborn.de/opendata/alkis/>
        a       <http://www.w3.org/ns/dcat#Distribution> ;
        <http://www.w3.org/ns/dcat#accessURL>
                "https://www.kreis-paderborn.de/opendata/alkis/" .

<https://www.kreis-paderborn.de/>
        a       <http://xmlns.com/foaf/0.1/Agent> .
```

How to add additional metrics

Metrics have to implement the interface Metric.
To add an implemented metric, use the getMetrics() method.
Tests should be added to AllTests test suite.

Note

Civet version 1 can be found at branch civet-version-1.

OPAL Classification

This component predicts objects of the dcat:theme predicate of dcat:Dataset subjects, based on the dc:description property. The implementation was done using WEKA https://github.com/Waikato/weka-3.8.
You can run the application with the default values with `mvn clean install` and `mvn exec:java -Dexec.mainClass="tools.Main"-Dexec.args="-c j48 -ngrams 1"-Dexec.cleanupDaemonThreads=false`
The result of the evaluation of the cross-validation of the training data and the evaluation of the test data is printed to console.

The following arguments can be provided:
-c {naive, j48}, default it j48
-ngrams {1,...,n}, default is 1
-query, sparql query default is

```
SELECT * WHERE { ?s a <http://www.w3.org/ns/dcat#Dataset> ; <http://www.w3.org/ns/dcat#theme> ?o ; <http://purl.org/dc/terms/description> ?d FILTER ( lang(?d)= "en" )} LIMIT 300
```

-endpoint, sparql endpoint, default is: https://www.europeandataportal.eu/sparql

Pre-processing

The following steps were taken:

1. Removed punctuation
2. Converted all text to lower case
3. Tokenization and Lemmatization
4. Removed the standard english stop words if either the lemma or the original word coincides

Word vectorization

The standard TF-IFD word vectors were computed.

Results

In the interest of time and since the approach is slow, the classifier was trained with 160 instances. That number might be too small to be representative.
The following accuracy was obtained for the cross-validation method with 4 folds:

Classifier	1-gram	2-gram	3-gram	4-gram
J48	75,625%	59,375%	59,375%	59,375%
NaiveBayes	47,5%	31,875%	36,875%	35%

The following accuracy was obtained for the evaluation of the test data.

Classifier	1-gram	2-gram	3-gram	4-gram
J48	62,07%	50%	59,09%	55,32%
NaiveBayes	28,09%	29,35%	27,59%	28,05%

Note

This component was developed by Ana Alexandra Morim da Silva during the OPAL hackathon. The component was on of the winners of the hackathon.

OPAL Common

Common utilities for OPAL components.

- constants for languages and data-portals
- interfaces for OPAL components
- utilities for handling files and data
- vocabulary for namespaces and uris (additionally, using org.apache.jena.vocabulary is recommended)

Usage with Apache Maven

Add the following lines to your `pom.xml` configuration file:

```
<dependencies>
    <dependency>
        <groupId>org.dice-research.opal</groupId>
        <artifactId>common</artifactId>
        <version>[1,2)</version>
    </dependency>
</dependencies>

<repositories>
    <repository>
        <id>maven.aksw.internal</id>
        <name>AKSW Repository</name>
        <url>http://maven.aksw.org/archiva/repository/internal</url>
    </repository>
    <repository>
        <id>maven.aksw.snapshots</id>
        <name>AKSW Snapshot Repository</name>
        <url>http://maven.aksw.org/archiva/repository/snapshots</url>
    </repository>
</repositories>
```

Available versions are listed at maven.aksw.org.

OPAL converter

The converter refines metadata and transforms it into 5-Star Linked Open Data. This is a publish-subscribe microservice design project via Spring Cloud and Java. The deployment of the project is based on docker-compose. Scaling can be configured manually.

The converter integrates the following OPAL components:

- Catfish (data-cleaner-service)
- Civet (quality-metrics-service)
- Metadata-Refinement (opal-confirm-conversion-service)

How to use

To build the project you must create a .env file and specify the required environment variables. Afterwards, by running the run.sh file it will build and setup the project.

```
# Triplestore to read data
CRAWLER_TRIPLESTORE_URL=
CRAWLER_TRIPLESTORE_USERNAME=
CRAWLER_TRIPLESTORE_PASSWORD=

# Triplestore to write data
OPAL_TRIPLESTORE_URL=
OPAL_TRIPLESTORE_USERNAME=admin
OPAL_TRIPLESTORE_PASSWORD=

# Free to choose
RABBITMQ_USERNAME=
RABBITMQ_PASSWORD=
H2_DB_PASSWORD=

# Elasticsearch configuration for logging
ELASTICSEARCH_JAVA_OPTS=-Xms512m -Xmx512m
```

Introduction

This repository provides the OPAL demonstrator, consisting of the Web UI and Webservices.

How to run

To be able to run you must provide a .env file in the root folder of the project that is similar to

```
BACKEND_ADDRESS=http://yourserver:8081/
ES_INDEX=opal_may
OPAL_ELASTICSEARCH_URL=opaldata.cs.upb.de
OPAL_ELASTICSEARCH_PORT=9200
```

Then, by running the command

```
docker-compose up -d
```

you have the demo containers running and the demo is available on port 3000 (you can set any port that you want in the docker-compose.yml) of your server.

ElasticTriples

Elasticsearch powered triple storage.

Preparation: Elasticsearch installation

To use ElasticTriples, you have to install Elasticsearch.
That can by done directly or via Docker:

- Installing Elasticsearch
- Install Elasticsearch with Docker

Preparation: Big data

If you want to process big files, the data should be available in N-Triples format (instead of e.g. Turtle).
A software for that is RDF2RDF.
After installing it, use `{GOPATH}/bin/rdf2rdf -in=in.ttl -out=out.nt` to transform you data.

Import data

An import of around 90 million triples can be performed in around 77 minutes
(89,902,895 triples; 10.3 GB in Turtle format; 16.3 GB in N-Triples format; import time: 4642.451 seconds).
A code example is given in OpalImport.java.

Query data

A search query takes around 2-3 seconds. E.g. extracting one (out of a million) DCAT-dataset with 206 triples uses 2,281 queries inside 3 multi-queries.
A code example is given in OpalQuery.java.

Split data

Splitting data is done by requesting single dataset graphs (each 2-3 seconds) and writing the resulting data no files in N-Triples format.
A code example is given in OpalSplitter.java.

Filter data

Data can be filtered based on language tags of title literals.
A code example is given in OpalSplitter.java.

OPAL Open Data Hackathon

Website

The OPAL hackathon website is available in hacker style and on GitHub.

Results

Title	Origin	Copy on branch
Dataset Locator	DatasetLocator	dataset-locator
OSM Bounding Box	HackathonBoundingBox	osm-bounding-box
Show Geo	opal-hackathon	show-geo
Theme Classify	opal-theme-classify	theme-classify
UPB Geo Data	UPBGeoData	upb-geo-data

LauNuts

Download the LauNuts knowledge graph at the Hobbit server.

Statistics

```
Countries:       1 (Germany)
NUTS-1:         16 (Federal states)
NUTS-2:         38 (Government regions)
NUTS-3:        401 (Districts)
LAU:        11,087 (Municipalities)
Total:      11,543

GeoData:    10,695
No GeoData:    848

Triples:   100,360
```

LauNuts generator

This code creates a knowledge graph for German regions and cities consisting of

- Local Administrative Units (LAU)
- Nomenclature of Territorial Units for Statistics (NUTS)
- GeoData from DBpedia

Usage of the data generator

- Download the required data (backup available at the Hobbit server)
  - nuts.rdf at data.europa.eu
  - EU-28-LAU-2019-NUTS-2016-DE.csv included in XLSX at ec.europa.eu
- Execute the main method in org.dice_research.opal.launuts.Main
- Edit the created configuration file “launuts-configuration.xml”
- Execute the main method in org.dice_research.opal.launuts.Main again

Note

This is not related to the musical instrument launut.

OPAL Licenses

This repository contains code to generate a list of compatible licenses based on multiple input licenses to be checked.
The following main steps are executed:

- Creation of a KnowledgeBase which contains the licenses and attributes.
- License Attribute values are mapped according to their type.
- The Operator computes composite attributes of all input licenses.
- Finally, the BackMapping creates a list of compatible licenses.
- To add additional Knowledge Bases, the AttributeFactory can be utilized and afterwards the Execution methods help to run an experiment.

Evaluation and experiments

For the evaluation of the approach, experiments based on two license datasets are provided:
Creative Commons and the European Data Portal (EDP) License Compatibility Matrix.

Creative Commons experiments

To run the evaluations based on Creative Commons, you first have to download the underlying dataset.
Therefore, download or clone the
cc.licenserdf
repository.
Afterwards, the directory of the repository can be set by the system property `cc.licenserdf`.
Example commands to run the experiments are listed below.

Creative Commons License Compatibility Chart

```
1  java -Dcc.licenserdf=../../cc.licenserdf/cc/licenserdf/licenses/ -jar licenses-
       jar-with-dependencies.jar cc1
```

This will run the CcExperiment.

Creative Commons cc.licenserdf with two input licenses

```
1  java -Dcc.licenserdf=../../cc.licenserdf/cc/licenserdf/licenses/ -jar licenses-
       jar-with-dependencies.jar cc2
```

This will run the CcExperimentTuples.

Creative Commons cc.licenserdf with three input licenses

```
java -Dcc.licenserdf=../../cc.licenserdf/cc/licenserdf/licenses/ -jar licenses-
    jar-with-dependencies.jar cc3
```

This will run the CcExperimentTriples.

European Data Portal experiments

To run the evaluation of the European Data Portal (EDP) License Compatibility Matrix, run the following maven command:

```
mvn clean test -Dtest=EdpLcmEvaluationTest -Drun.edp.lcm.tests=true
```

This will run the EdpLcmEvaluationTest.

OPAL linguistic-ambiguities

This OPAL component handles linguistic ambiguities.
It is designed as a batch process to enhance ElasticSearch.

- Extraction of synonyms of German nouns is based on DBnary

Usage

- Typical usages are described in the class
  `org.dice_research.opal.linguistic_ambiguities.Main`
- Single test cases can be found in the test folder
  `org.dice_research.opal.linguistic_ambiguities`

Notes

This components is part of *OPAL Deliverable D7.1: Suchkomponente*.
The document (in German) is available at the OPAL deliverables website.

OPAL Datenintegration

Dieses Repositorium enthält Daten zu den Bereichen Datenverknüpfung / Linkspezifikationen. Dazu werden Link Discovery Framework for Metric Spaces (LIMES) sowie Decision Tree Learning for Link Discovery (DRAGON) verwendet.

OPAL metadata refinement

- **Language Detection** based on Apache OpenNLP updates language tags of title and description literals of 4 languages.
- **Geographic data** based on LauNuts adds geo data of 8,495 places in Germany.

Usage with Apache Maven

Add the following lines to your `pom.xml` configuration file:

```
<dependencies>
    <dependency>
        <groupId>org.dice-research.opal</groupId>
        <artifactId>metadata-refinement</artifactId>
        <version>[1,2)</version>
    </dependency>
</dependencies>

<repositories>
    <repository>
        <id>maven.aksw.internal</id>
        <name>AKSW Repository</name>
        <url>http://maven.aksw.org/archiva/repository/internal</url>
    </repository>
    <repository>
        <id>maven.aksw.snapshots</id>
        <name>AKSW Snapshot Repository</name>
        <url>http://maven.aksw.org/archiva/repository/snapshots</url>
    </repository>
</repositories>
```

Available versions are listed at maven.aksw.org.

Examples

Language tags

```
import java.io.File;
import org.apache.jena.rdf.model.Model;
import org.dice_research.opal.common.utilities.FileHandler;
import org.dice_research.opal.metadata.GeoData;
import org.dice_research.opal.metadata.LanguageDetection;
public class Example {

    /**
     * Updates language tags of title and description literals.
     *
     * @param turtleInputFile  A TURTLE file to read
```

```
     * @param turtleOutputFile A TURTLE file to write results
     * @param datasetUri       A URI of a dcat:Dataset inside the TURTLE data
     *
     * @see https://www.w3.org/TR/turtle/
     * @see https://www.w3.org/TR/vocab-dcat/
     */
    public void updateLanguageTags(File turtleInputFile, File turtleOutputFile,
        String datasetUri) throws Exception {

        // Load TURTLE file into model
        Model model = FileHandler.importModel(turtleInputFile);

        // The call of initialize() is optional. It can be used to trigger the
            download
        // of the required language model (10 MB).
        LanguageDetection languageDetection = new LanguageDetection();
        languageDetection.initialize();

        // Update model
        languageDetection.processModel(model, datasetUri);

        // Write updated model into TURTLE file
        FileHandler.export(turtleOutputFile, model);
    }
```

Example input:

```
<http://example.org/>
        a       <http://www.w3.org/ns/dcat#Dataset> ;
        <http://purl.org/dc/terms/title>
                "Places in Berlin" .
```

Example output:

```
<http://example.org/>
        a       <http://www.w3.org/ns/dcat#Dataset> ;
        <http://purl.org/dc/terms/title>
                "Places in Berlin"@en .
```

Geographic data

```
import java.io.File;
import org.apache.jena.rdf.model.Model;
import org.dice_research.opal.common.utilities.FileHandler;
import org.dice_research.opal.metadata.GeoData;
import org.dice_research.opal.metadata.LanguageDetection;
public class Example {

    /**
     * Creates geo data based on names of places that are found in the title
         and
     * description of the specified dataset.
```

```
     *
     * @param turtleInputFile  A TURTLE file to read
     * @param turtleOutputFile A TURTLE file to write results
     * @param datasetUri       A URI of a dcat:Dataset inside the TURTLE data
     *
     * @see https://www.w3.org/TR/turtle/
     * @see https://www.w3.org/TR/vocab-dcat/
     */
    public void createGeoData(File turtleInputFile, File turtleOutputFile,
        String datasetUri) throws Exception {

        // Load TURTLE file into model
        Model model = FileHandler.importModel(turtleInputFile);

        // Update model
        new GeoData().processModel(model, datasetUri);

        // Write updated model into TURTLE file
        FileHandler.export(turtleOutputFile, model);
    }
```

Example input:

```
<http://example.org/>
        a       <http://www.w3.org/ns/dcat#Dataset> ;
        <http://purl.org/dc/terms/title>
                "Places in Berlin" .
```

Example output:

```
<http://example.org/>
        a       <http://www.w3.org/ns/dcat#Dataset> ;
        <http://purl.org/dc/terms/spatial>
                <http://data.europa.eu/nuts/code/DE3> ;
        <http://purl.org/dc/terms/title>
                "Places in Berlin" .
<http://data.europa.eu/nuts/code/DE3>
        a       <http://purl.org/dc/terms/Location> ;
        <http://www.w3.org/ns/dcat#centroid>
                "POINT(52.5167 13.3833)"^^<http://www.opengis.net/ont/geosparql
                    #wktLiteral> .
```

Note

Version alpha can be found at branch metadata-alpha.
It includes
Language Detection based on Apache OpenNLP,
Named Entity Recognition based on FOX, and
a JavaScript word picker
as well as configurations for Docker usage and webservices.

OPAL data

Setup file(s) for OPAL data server with one Elasticsearch and one Apache Fuseki instance.
This component is used by the OPAL converter to store data and by OPAL web-service to read data.

Configuration and start

- Install Docker Compose
- On your machine, set max_map_count=262144 (source: Elasticsearch guide)
- Download or clone the GitHub repository opaldata
- Edit the configuration file `.env`.
  Settings:
  - `FUSEKI_ADMIN_PASSWORD`
    Fuseki admin password for the frontend. If not set, a random password will be created.
  - `FUSEKI_JVM_ARGS`
    Java configuration for Fuseki, e.g. the maximum heap size (Xmx).
    Default value: `-Xmx16g`
  - `ELASTICSEARCH_JAVA_OPTS`
    Java memory configuration for Elasticsearch.
    Default value: `-Xms8g -Xmx8g`
- Run `docker-compose up -d`

Access

- **Fuseki**
  URL: http://localhost:3030/
  Username: `admin`
  Password: As set in .env file or randomly created.
- **Elasticsearch**
  An example URL to list the available indexes:
  http://localhost:9200/_cat/indices?v
  An example to count datasets:
  ```
  curl -XGET http://localhost:9200/opal/dataset/_count
  ```
  An example to add a dataset:
  ```
  curl -H "Content-Type: application/json"-XPOST "http://localhost:9200/opal/
  dataset/test123"-d "{ \"title\" : \"Hello World\"}"
  ```

An example to get a dataset:

```
curl -XGET http://localhost:9200/opal/dataset/test123
```

Notes

- Some documentation is in the GitHub wiki: https://github.com/projekt-opal/opaldata/wiki

OpalSpark

Spark Job for dataset processing

The goal of this project is to extract dcat:dataset graphs from a large RDF file and store each one on it's own file.

To build the project, certify that you have scala installed and run:

```
$ mvn clean install
```

under /target, you will have the builded Jar and the lib folder, containing all the necessary libs to run this job.

Before running, certify that you have ***Apache Spark 2.4.X*** installed on your system.
Check this tutorial for a Standalone installation:
https://www.tutorialspoint.com/apache_spark/apache_spark_installation.htm

Or you can use this docker image:
https://github.com/big-data-europe/docker-spark

To run the Job, use the command through command line:

```
spark-submit --class org.dice_research.opal.processing.DatasetPartitioner --
    jars $(echo lib/*.jar | tr ' ' ',') --driver-memory 100G --master local[*]
    OpalSpark-1.0.jar <path_to_*.ttl_file> <destination_folder>
```

The driver-memory parameter represents the ammount of physicial memory that will be allocated to run the job. Larger the file, larger the ammount that should be allocated.

In the given command line example, the masteris set to local, considering running the job locally. For other configurations like running on a cluster, check Spark documentation:

https://spark.apache.org/docs/latest/submitting-applications.html

Search-Component-Benchmark

This repository contains the implementation for Deliverable 7.2 Search component benchmark.

Installation

First clone the repository:

```
git clone https://github.com/XXX/XXX.git.
```

Then obtain the required libraries:

```
conda env create -f environment.yml
source activate benchmarking
```

The code is compatible with Python 3.6.4.

Reproducing reported results

- python evaluation.py

OPAL Slicer

The OPAL data selection component uses patterns in SPARQL format to extract subsets of knowledge graphs.

Usage example

To extract all dataset statements, use the Slicer class and the following parameters:

```
-source input.ttl
-patterns "Select * where {?d a <http://www.w3.org/ns/dcat#Dataset>}"
-out datasets.ttl
```

Notes

- This component is mainly based on RDFSlice
- See also the documentation wiki

OPAL statistics

Generation of RDF data statistics.

This code is obsolete and not intended to be reused.
It was used to perform SPARQL queries and create statistics.
The state of code for D3.3 to create licence statistics is also available for documentation purposes.

Usage

Create a file `src/main/resources/private.properties` and set your preferred SPARQL endpoints.
The required keys for configuration can be found in `src/main/resources/config.properties`.
Run the preferred main() methods.

OPAL JUnit test cases

This component provides test cases. The single tests are separated into sets. Take a view at the available test cases.

Test case sources

- OpalGraph 2019-06-24
- EDP 2019-12-17 (data.10.nt in edp-de-2019-12-17.tar.gz)

Request for additional test case

If you need an additional dataset, please create a new issue.

Examples

```java
import org.dice_research.opal.test_cases.OpalTestCases;
import org.dice_research.opal.test_cases.TestCase;
// ...

// Print all available test-sets.
SortedSet<String> testSets = OpalTestCases.listTestSets();
System.out.println("Sets: " + testSets);

// Print all available test-cases for a test-set
SortedSet<String> testCases = OpalTestCases.listTestCases(testSets.first());
System.out.println("Test-cases for " + testSets.first() + ": " + testCases);

// Get a test-case. Print the model-size and dataset-URI.
TestCase testCase = OpalTestCases.getTestCase(testSets.first(), testCases.first
    ());
System.out.println("Test case model: " + testCase.getModel().size());
System.out.println("Test case dataset-URI: " + testCase.getDatasetUri());
```

Usage with Apache Maven

Add the following lines to your `pom.xml` configuration file:

```xml
<dependencies>
    <dependency>
        <groupId>org.dice-research.opal</groupId>
        <artifactId>test-cases</artifactId>
        <version>[1,2)</version>
        <scope>test</scope>
```

```
    </dependency>
</dependencies>

<repositories>
    <repository>
        <id>maven.aksw.internal</id>
        <name>AKSW Repository</name>
        <url>http://maven.aksw.org/archiva/repository/internal</url>
    </repository>
    <repository>
        <id>maven.aksw.snapshots</id>
        <name>AKSW Snapshot Repository</name>
        <url>http://maven.aksw.org/archiva/repository/snapshots</url>
    </repository>
</repositories>
```

Available versions are listed at maven.aksw.org.

Topic-Extraction

OPAL component to extract entities such as places, keywords from dataset descriptions to improve relevant dataset seraching. This repository contains all data and utilities to train and test a topic extraction model.

Requirements

python version 3.x, rasa_nlu

Usage Instructions

1. Generation of training and testing data (rasa_nlu format).

The component assumes the training and testing data (annotated manually or automatically) is contained in .txt file where each training example is a single line consisting of annotated text. In topic extraction, we focus on 5 entity types, namely: place, person, date and keyword. An example annotation is as shown below

```
This is a Housing Benefit dataset for all new claims and change of
    circumstances received by the <entity type=place uri=null>London</entity>
    Borough of Barnet in the second half of <entity type=date uri=null>2015</
    entity>.
```

To generate training and testing data, run the following command by adjusting paths to input training and testing (.txt) files and output files accordingly.

```
python generateOpalTrainingData.py
```

Once the command is finished, the files are generated at the desired output location.

2. Train the topic extraction model.

Once the training file is generated from the above step, create a model by adjusting the paths to map to the training and configuration file and running the command

```
python opalPersister.py
```

The above command generates a persistent model which could be found under the base directory of the project. Note that, one can create several models which could be found in the model folder's default directory.

3: Test the generated model.

Once a model is generated, it can be tested using the test data. To test the model, run the following command

```
python -m rasa_nlu.evaluate \
    --data path/to/test.json \
    --model path/to/model/default/model_20180323-145833
```

Note that, in the above command, model specifies the model to evaluate on the test data specified with data.

OPAL Web-Service

This component is part of the demo project.
It requires opaldata to access data.
It provides data for the web-ui.

Notes

- API REST methods can be found in RestAPIController.
- Development notes are in the wiki.

OPAL Web User Interface

This component provides the main user interface.
It mainly builds on React and Next.js - a JavaScript library and a framework for building user interfaces.

The OPAL Web User Interface is part of the OPAL demo component.
It requires a running OPAL webservice, which provides the data to display.

Integrated: City App Demonstrator

For OPAL deliverable D7.3, a mobile app was integrated to the OPAL user interface. It supports users in identifying data at their current locations.

The demonstrator was implemented by a web-app using Responsive web design (RWD).
The required user geo information is requested via the `navigator.geolocation.getCurrentPosition` function specified by W3C [GeoApi, Position].
The functionality was added using an integrated table sorting [OrderBy].

- [GeoApi] [W3C Geolocation API Specification](https://w3c.github.io/geolocation-api/)
- [Position] [OrderBy.js:96](src/components/report/datasets/dataset/OrderBy.js#L96)
- [OrderBy] [OrderBy.js:79](src/components/report/datasets/dataset/OrderBy.js#L79)

Notes

- Development notes are in the wiki.