This deliverable provides a snapshot of the application software documentation, which has been written to accompany the community driven applications being developed as a part of WP9.
# Document Description

## D9.3 – Application Software Documentation – updated periodically

<table>
<thead>
<tr>
<th>WP9 – Community Driven Applications</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>WP participating organizations:</strong> ARC, University of Manchester, UKP-TUDA, INRA, EMBL, Agro-Know I.K.E., OU, EPFL, CNIO, USFD, GESIS, GRNET, Frontiers</td>
</tr>
<tr>
<td><strong>Contractual Delivery Date:</strong> 1/2017</td>
</tr>
<tr>
<td><strong>Nature:</strong> Report</td>
</tr>
<tr>
<td><strong>Public</strong> Deliverable</td>
</tr>
</tbody>
</table>

### Preparation slip

<table>
<thead>
<tr>
<th>Name</th>
<th>Organization</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>From</td>
<td>Matt Shardlow</td>
<td>UNIMAN</td>
</tr>
<tr>
<td>+ Communities</td>
<td></td>
<td>30/3/2017</td>
</tr>
<tr>
<td>Edited by</td>
<td>Matt Shardlow</td>
<td>UNIMAN</td>
</tr>
<tr>
<td></td>
<td></td>
<td>30/3/2017</td>
</tr>
<tr>
<td>Reviewed by</td>
<td>Andoniki Pavlidou</td>
<td>ARC</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3/5/2017</td>
</tr>
<tr>
<td></td>
<td>Gareth Owen</td>
<td>EBI</td>
</tr>
<tr>
<td></td>
<td></td>
<td>16/5/2017</td>
</tr>
<tr>
<td>Approved by</td>
<td>Andoniki Pavlidou</td>
<td>ARC</td>
</tr>
<tr>
<td></td>
<td></td>
<td>30/05/2017</td>
</tr>
<tr>
<td>For delivery</td>
<td>Mike Chatzopoulos</td>
<td>ARC</td>
</tr>
<tr>
<td></td>
<td></td>
<td>30/05/2017</td>
</tr>
</tbody>
</table>

### Document change record

<table>
<thead>
<tr>
<th>Issue</th>
<th>Item</th>
<th>Reason for Change</th>
<th>Author</th>
<th>Organization</th>
</tr>
</thead>
<tbody>
<tr>
<td>V0.1</td>
<td>Draft version</td>
<td>Incorporated documentation from communities</td>
<td>Matt Shardlow</td>
<td>UNIMAN</td>
</tr>
<tr>
<td>V1.0</td>
<td>Final version</td>
<td>Finalized deliverable</td>
<td>Matt Shardlow</td>
<td>UNIMAN</td>
</tr>
<tr>
<td>V1.1</td>
<td>For review</td>
<td>Incorporated Androniki’s feedback</td>
<td>Matt Shardlow</td>
<td>UNIMAN</td>
</tr>
<tr>
<td>V1.2</td>
<td>Post review</td>
<td>Incorporated Gareth’s feedback</td>
<td>Matt Shardlow</td>
<td>UNIMAN</td>
</tr>
</tbody>
</table>
Table of Contents

1. INTRODUCTION ................................................................................................................................. 6
   1.1 DOCUMENTATION SCOPE .................................................................................................................. 7
   1.2 METHODOLOGY .................................................................................................................................. 7
   1.3 STRUCTURE .......................................................................................................................................... 7

2. LS-A EXTRACT METABOLITES AND THEIR PROPERTIES AND MODES OF ACTIONS ........................................... 8


4. AS-C: MICROBIAL BIODIVERSITY ........................................................................................................... 47

5. AS-D: LINKING WHEAT DATA WITH LITERATURE .................................................................................. 62

6. AS-E: EXTRACTING GENE REGULATION NETWORKS INVOLVED IN SEED DEVELOPMENT (SEEDEV) ......................... 77

7. SS-A: FACILITATION OF COMPLEX INFORMATION LINKING AND RETRIEVAL FROM SOCIAL SCIENCES PUBLICATIONS ............................................................................................................. 89

8. SC-A: RESEARCH ANALYTICS – FUNDING MINING SERVICES ...................................................................... 99
Disclaimer

This document contains description of the OpenMinTeD project findings, work and products. Certain parts of it might be under partner Intellectual Property Right (IPR) rules so, prior to using its content please contact the consortium head for approval.

In case you believe that this document harms in any way IPR held by you as a person or as a representative of an entity, please do notify us immediately.

The authors of this document have taken any available measure in order for its content to be accurate, consistent and lawful. However, neither the project consortium as a whole nor the individual partners that implicitly or explicitly participated in the creation and publication of this document hold any sort of responsibility that might occur as a result of using its content.

This publication has been produced with the assistance of the European Union. The content of this publication is the sole responsibility of the OpenMinTeD consortium and can in no way be taken to reflect the views of the European Union.

The European Union is established in accordance with the Treaty on European Union (Maastricht). There are currently 28 Member States of the Union. It is based on the European Communities and the member states cooperation in the fields of Common Foreign and Security Policy and Justice and Home Affairs. The five main institutions of the European Union are the European Parliament, the Council of Ministers, the European Commission, the Court of Justice and the Court of Auditors. (http://europa.eu.int/)

OpenMinTeD is a project funded by the European Union (Grant Agreement No 654021).
## Acronyms

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TDM</td>
<td>Text and Data Mining</td>
</tr>
<tr>
<td>LS</td>
<td>Life Sciences</td>
</tr>
<tr>
<td>SC</td>
<td>Scholarly Communications</td>
</tr>
<tr>
<td>AS</td>
<td>Agriculture / Biodiversity</td>
</tr>
<tr>
<td>SS</td>
<td>Social Sciences</td>
</tr>
<tr>
<td>NLP</td>
<td>Natural Language Processing</td>
</tr>
</tbody>
</table>
Publishable Summary

The OpenMinTeD project is producing a set of applications driven by the needs of the communities at the heart of our project. These applications will both serve a functional need for each community, as well as showcasing the functionality of the platform. In this report we provide the first set of documentation for each application. Each application development team has worked together with the users to document the functionality of their application.
1. **Introduction**

This deliverable gives a snapshot of the community driven applications documentation after the first application release, as documented in MS37. This documentation is updated periodically and as such will be subject to change from the content seen herein. We have provided links to the permanent locations of each set of documentation, where up-to-date versions can be found, in Table 1 below:

<table>
<thead>
<tr>
<th>Application Name</th>
<th>URL</th>
</tr>
</thead>
</table>
1.1 Documentation Scope

There are many different forms of documentation. These may be broadly split into ‘technical documentation’ and ‘user-facing documentation’. Technical documentation should give an overview of the technologies and resources used in the construction of an application along with system architectures and APIs. It should be useful from a technical perspective to understand how the system was built, for reimplementing and also for refactoring or implementing new features. Technical documentation for the applications is contained in D9.1 – Community Driven Application Design.

Conversely, user-facing documentation is aimed at the eventual user of an application. It should act as a user guide, which gives clear instructions on how to use the application. The instructions should be written from a non-technical perspective, especially as in OpenMinTeD, our users have very little, or no, text-mining expertise. The user-facing documentation should act as a whole unit, containing all the information a user needs to run the application and get results.

1.2 Methodology

To generate the documentation, we asked each application development team to write the following pages of documentation:

1. **Introduction**: A single page to describe the application and its intended functionality. This page acts as a ‘shop window’ for a potential user, allowing them to decide whether the application is for them or not.

2. **Data Processing Scenarios**: In Deliverable 9.1, Community Driven Applications Design, we asked each application to specify a set of data processing scenarios, which explained how the application would interact with data (i.e. loading data in, data processing, data export). We asked developers to write one page of documentation per data processing scenario explaining how their application currently handles that scenario.

3. **Technical Information**: If the developers felt it was necessary to also include some technical information in the documentation, we asked for this to be contained in a separate page. This information may be of use for more technical users. We will incorporate any technical information back into the forthcoming update to Deliverable 9.1.

We decided to use Asciidoc format to generate our documentation. Asciidoc is a markup language that allows a user to quickly generate structured text. Asciidoc is being used for documentation elsewhere in the project and hence we felt it was appropriate to use it for our application documentation also. In order to

1.3 Structure

The rest of this deliverable contains the documentation that was generated from Asciidoc. The documentation is presented in the order shown in Table 1. Up to date versions of the documentation can be found by following the links in Table 1.
2. **LS-A Extract metabolites and their properties and modes of actions**
Introduction

LS-A is designed to aid in the curation of the ChEBI database. At the first release of this application we have developed a web application built using the Argo text mining workflow system (http://argo.nactem.ac.uk/). The web application allows a user to enter a piece of text from a biological article and see the results of the annotation displayed in the brat annotation visualisation tool (http://brat.nlplab.org/).
Data Processing Scenarios

In LS-A, we have outlined the following data processing scenarios:

- Adding documents to the database
- Showing annotations of a given document
- Retrieving relations from the database

The first release of the application has focused on the second scenario - i.e., showing annotations to the user. This allows us to get feedback from the users on the types of annotations that are of interest to them and helps us to focus and improve our future efforts.

Some technical information for the application can be found at: Technical Information

Last updated 2017-02-23 18:51:38 EET
Adding documents to the database

This data processing scenario is yet to be addressed in the current LS-A application.

Last updated 2017-02-23 18:51:38 EET
Showing annotations of a given document

The application is currently focussed around this data processing scenario.

Currently, a user can navigate to the web form at: http://hactem.ac.uk/apps/openminted/chebi/ where they can enter their biological text for processing. The text is sent via a REST interface to a running Argo workflow, which processes the text for the entities of interest and returns the file path of the document that has been created. This is then processed to give the link to the file, visualised in brat.

The Web interface is shown in the image below:

![Web interface screenshot](image-url)

The output of the workflow is displayed in brat as shown:

```
1. In the 70s serine protease 5 (P) was identified in normal lactating secretions, it was believed that it was only expressed in the glandular tissues.
2. At the present time, the nucleocapsid of the P-containing protein has been shown to be a unique and specific anti-inflammatory protein.
3. In addition, the biological samples of the P-containing proteins are expressed in many tissues and cell lines from numerous species.
4. In addition, the P-containing proteins have been identified in a variety of tissues and cell lines from numerous species.
5. Rather than providing diverse pathways for modulating the expression of P in a variety of tissues and cell lines, it is proposed that the enzyme is acting directly on P to produce the desired outcomes.
6. The enzymes bind to novel, unique, and specific plasma membrane receptors and modulate opposing actions on mitosis, apoptosis, and cytoskeletal and cell adhesion molecules via cell signaling pathways.
7. The enzyme is involved in the modulation of P by regulating cell proliferation and detachment, whereas Saptaph promotes mitogenesis and reexpression by stimulating cell proliferation and detachment.
8. The enzyme is involved in the modulation of P by regulating cell proliferation and detachment, whereas Saptaph promotes mitogenesis and reexpression by stimulating cell proliferation and detachment.
9. The enzyme is involved in the modulation of P by regulating cell proliferation and detachment, whereas Saptaph promotes mitogenesis and reexpression by stimulating cell proliferation and detachment.
10. The enzyme is involved in the modulation of P by regulating cell proliferation and detachment, whereas Saptaph promotes mitogenesis and reexpression by stimulating cell proliferation and detachment.
11. The enzyme is involved in the modulation of P by regulating cell proliferation and detachment, whereas Saptaph promotes mitogenesis and reexpression by stimulating cell proliferation and detachment.
12. The enzyme is involved in the modulation of P by regulating cell proliferation and detachment, whereas Saptaph promotes mitogenesis and reexpression by stimulating cell proliferation and detachment.
```
The REST interface is accessible at:

http://argo.nactem.ac.uk/dev/proxy/webservice/<ServiceID>

Where <ServiceID> is a unique ID which locates the service. Currently, this is set at "384", although this is liable to change.

Users should submit a request via POST with the header: Content-type:text/plain

The function will return a filepath, the last element of which is the filename of the annotated document. This can be accessed in Brat by appending the filename to the URL: http://nactem0.ml.b.man.ac.uk/brat-v1.3/#/OpenMnt-ed-Chebi App/

A minimum working example in javascript is presented below:

```javascript
// set up the HTTP Request
var xhr = new XMLHttpRequest();
xhr.open("POST", "http://argo.nactem.ac.uk/dev/proxy/webservice/<ServiceID>", true);

// Send the proper header information along with the request
xhr.setRequestHeader("Content-type", "text/plain");

// Callback function for when the state changes (i.e. response is received
xhr.onreadystatechange = function() {
    if(xhr.readyState == XMLHttpRequest.DONE && xhr.status == 200) {
        // Request success
        var splitStr = xhr.responseText.split("/");
        var docName = splitStr[splitStr.length - 1];

        link = "http://nactem0.ml.b.man.ac.uk/brat-v1.3/#/OpenMnt-ed-Chebi App/" + docName;
    } else
        console.log("Processing failed");
};

// Send the request
xhr.send(document.getElementById('inputText').value);
```

Last updated 2017-03-13 14:38:15 EET
Retrieving relations from the database

This data processing scenario is yet to be addressed in the current LS-A application.

Last updated 2017-02-23 18:51:38 EET
Technical information for ChEBI application

Argo is a text-mining workbench that can be used to combine UIMA components into text-mining workflows. Argo provides components for a number of text-mining purposes. In this application, we use Argo to annotate the following entities:

- Chemicals
- Metabolites
- Proteins
- Species

The Workflow we have created is shown below.

The Functionality of each component is described below:
TextWebServiceReader: Reads in text provided by a user through the REST API

LingPipe Sentence Splitter: Splits the input text into sentences using a model targeted at biological texts.

OSCAR 4 Tokeniser: Takes the input text and splits it into tokens (i.e. continuous text units broken on whitespace, punctuation, etc.) Specifically tuned for Chemical texts

GENIA Tagger: Used to annotate the text with tokens, parts-of-speech, etc.

Chemical Entity Recogniser (Chemicals): Recognises Chemical names in the text according to a pre-trained model.

SPARQL Annotation Editor (Chemicals): Converts the output format (i.e. type) of the previous tool to a format which is required for output.

Species Tagger: Identifies Species names in the text

Duplicate Annotation Remover: Removes duplicated Species annotations

NERsuite Custom Tagger (Proteins): Tags the text with protein names using a custom CRF-based model.

Chemical Entity Recogniser (Metabolites): Identifies metabolite named entities in the text.

BioNLP ST Type System Loader (2): Makes sure that the correct Types are loaded in the Underlying Common Annotation Structure (CAS).

BioNLP ST WebService Writer: writes the annotations to brat and returns the link to the file to the user.

Last updated 2017-02-23 18:51:38 EET
3. **AS-A**: Text mining over bibliographic data: AGRIS & CORE & AS-B: Text-mining over RSS Feeds: Food Safety & Water Health
Introduction

The text-mining problem to be addressed in the context of the AS-A Use Case is the extraction of structured information (domain specific topics, images/figures, captions etc.) from unstructured bibliographic resources (i.e. PDF documents).

More specifically, Research Articles from Viticulture are automatically annotated by text-mining pipeline, using relevant open resources (e.g. AgroVoc).

The text-mining problem to be addressed in the context of the AS-B Use Case is the extraction of geolocation information from unstructured RSS feeds that target the Food Safety and the Water Health Community.

More specifically, RSS Feeds from FoodSafety and WaterWorld are automatically annotated by text-mining pipeline, using relevant open resources (e.g. GeoNames, FAO Geopolitical Ontology).
Data Processing Scenarios

In AS-A, we have the following data processing scenario:

Data Processing Scenario:

- This Component extracts PDF Artifacts such as outline, main text, figures, tables, captions and references
- This Component extracts Grape Variety terms (based on the official OIV grape variety list) from the main text of a PDF Document
- This Component extracts AgroVoc terms (using the FAO AgroVoc Ontology) from the main text of a PDF Document

In AS-B, we have the following data processing scenario:

Data Processing Scenario:

- This Component retrieves RSS Feeds from FoodSafety and WaterWorld
- This Component extracts Geolocations from retrieved RSS feeds, using the Geonames Ontology
- This Component extracts Geolocations from retrieved RSS feeds, using the FAO Geopolitical Ontology

Last updated 2017-03-21 13:43:54 EET
1. Component Description

The OpenMinted PDF Extractor is a stand-alone component used for extracting textual, pictorial and tabular information from documents following the Portable Document Format standard [1].

The component is installed and run locally over Linux systems. It accepts as input a directory path, where the PDF files to be processed are located.

The component creates a directory for each input document, which contains the following items:

- A plain text file containing the document’s metadata (e.g. author(s), creation date, title, etc.);
- A plain text file containing the raw text retrieved from the document;
- A plain text file containing the document’s outline;
- An XML file describing the references (if any) found in the document;
- A BibTex document describing those references that include a DOI link, if such references exist;
- Any figure found in the document as image files, in the format found within the document;
- A JSON document containing information on the aforementioned figures, as well as, the tables (if any) found in the input document. More specifically, the document defines Figure and Table objects. Each Figure object declares the caption and size and positioning metadata for the respective figure. Similarly, each Table object declares the caption and positioning of the respective table within the PDF document. Additionally, it presents the content’s of the table’s cells in separate brackets.
OpenMinted PDF Extractor
2. Component Installation

2.1. Component Dependencies

The following frameworks and tools are required before installing and running the component:

- A distribution of Python v2. The component was tested with the latest v2.7 version. Python is normally pre-installed in most Linux distributions;
- The PDF Miner Python library [2].
- The Poppler Python library [3].
- The pdf-extract tool [4].
- The PDF Figures tool [5].

2.2. Installation of Dependencies

2.2.1. Python v2

Python is normally pre-installed in most major Linux distributions. The component was tested with the latest (v2.7) version at the time of development. For checking if your Linux distribution comes with Python run the following command:

```bash
python -V
```

In case no Python version is found, it can be installed by giving the following commands:

```bash
* wget http://www.python.org/ftp/python/2.7.3/Python-2.7.3.tgz
* tar -xzf Python-2.7.3.tgz
* cd Python-2.7.3
* ./configure --prefix=/usr --enable-shared
* make
* make install
```

2.2.2. PDF Miner

The following actions are required to install PDF Miner:
Download the source code from https://github.com/euske/pdfminer/ and unpack it a system directory.

Run `setup.py`:

```
$ python setup.py install
```

Do the following test:

```
$ pdf2txt.py samples/simple1.pdf
```

### 2.2.3. Poppler

For debian-based Linux systems, run the following command:

```
sudo apt-get install python-poppler
```

For RHEL-based Linux systems, run the following command:

```
yum install poppler-utils
```

### 2.2.4. pdf-extract

In debian-based Linux systems, install the following packages:

```
sudo apt-get install build-essential
sudo apt-get install ruby-full
sudo apt-get install zlib1g-dev
sudo gem install nokogiri
sudo apt-get install sqlite3-dev
sudo gem install specific_install
```

```
sudo gem specific_install https://github.com/EbookGlue/libsvm-ruby-swig.git
```
In RHEL-based Linux systems, install the following packages:

```bash
sudo yum groupinstall "Development Tools"
sudo yum install ruby
sudo yum install zlib-devel
sudo gem install nokogiri
sudo yum install sqlite-devel
sudo gem install specific_install
sudo gem specific_install https://github.com/EbookGlue/bsvm-ruby-swig.git
sudo gem install pdf-reader -v 1.1.1
sudo gem install prawn -v 0.12.0
```

### 2.2.5. PDF Figures

- Download the source code from https://github.com/allenai/pdffigures and unpack it a system directory.
- Execute the following command:

  ```bash
  make DEBUG=0
  ```

### 2.3. PDF Extractor Installation

Download the PDF_Extractor.py2 script (https://github.com/openminted/uc-tdm-agriculture/tree/master/PDF%20Extractor) into a local directory. No further steps are required for using the component.
3. Relevant Data Processing Scenarios

3.1. Retrieve information from a collection of PDF documents

The user aims to analyze a collection of publications as PDF documents. To this end, she creates a directory in the local system’s filesystem:

```
mkdir <dirname>
```

She copies the relevant PDF documents in the created directory:

```
cp <filename>.pdf <dirname>/.
```

Finally, she executes the PDF Extractor script:

```
sudo python PDF_Extractor.py2 <dirname>
```

Upon completion, the script produces the following items:

- A file `<filename>.txt` containing the raw text of the `<filename>.pdf` document;
- A `<filename>.metadata` text file containing metadata information for the respective PDF document;
- A `<filename>.outline` text file containing the outline of the respective PDF document;
- The figures found in the PDF document as image files. The naming convention for the images follows the pattern `<filename>-<page number>-<increment>`.
- A `<filename>.references` XML file, containing the references found in the respective PDF document.
- A `<filename>.refs.bib` BibTex file, containing the references associated with a DOI.
- A `<filename>.json` file, containing information on the figures and tables discovered in the respective PDF document.

2. https://pypi.python.org/pypi/pdfminer/
4. https://github.com/CrossRef/pdfextract
5. https://github.com/allenai/pdffigures

Last updated 2017-03-21 12:37:32 EET
Grape Varieties Extractor
1. Component Description

The Grape Varieties Extractor is a stand-alone component responsible for discovering grape variety names, as defined by the OIV specification [1]. The OIV specification has been embedded programmatically into the component.

The component is distributed as an executable JAR file, thus it can be used in any platform.

Grape Varieties Extractor operates over a locally stored text file that contains the text to be analyzed for the presence of grape variety names.

Upon completion of its execution, it produces an XML document following the schema defined by a simple XSD file, included with the distribution package of the component. The XML presents the grape variety names found in the document ranked according to a modification of the classic tf-idf metric, along with their OIV ID.
2. Component Installation

2.1. Using the JAR Distribution

As an executable JAR file, the AgroVoc Extractor requires solely the presence of a compatible Java Runtime Environment distribution in the host system. AgroVoc Extractor is compatible with JRE 7 [2].

2.2. Using the Source code

Download the source code of the component from the OpenMinted GitHub repository [3] and run from the command line:

```bash
ant -buildfile build.xml
```

2.3. Testing the installation

Grape Varieties Extractor can be called without any arguments, using a default exemplary text input included in the distribution package.
3. Relevant Data Processing Scenarios

Extract Grape Variety names found in a document. The user aims to retrieve the references to grape varieties found in a text document, named <filename>, found in the local <path> directory. To this end, she executes the AgroVoc Extractor component from the command line:

```
java -jar AK_Agrovoc.jar <path> <filename>
```

Upon completion, the component produces the <filename>.GrapeVine.xml file, which contains information on the presence and frequency of any grape variety name discovered, namely:

- The grape variety that was found in the input document;
- The Lucene score (a variation of the tf-idf metric) of the term in the input document;
- The OIV ID for the term.

**Exemplary XML record in the output document**

```
<term>
  <text>Grape Variety</text>
  <oiv_id>50</oiv_id>
  <score>0.39380478858947754</score>
</term>
```

---

   /international-list-of-vine-varieties-and-their-synonyms

Last updated 2017-03-21 12:37:32 EET
AgroVoc Extractor
1. Component Description

The AgroVoc Extractor is a stand-alone component responsible for discovering AgroVoc terms within text segments.

The component is distributed as an executable JAR file, thus it can be used in any platform.

AgroVoc Extractor operates over a locally stored text file that contains the text to be analyzed for the presence of AgroVoc terms.

Upon completion of its execution, it produces an XML document following the schema defined by a simple XSD file, included with the distribution package of the component. The XML presents the AgroVoc terms found in the document ranked according to a modification of the classic tf-idf metric.
2. Component Installation

2.1. Using the JAR Distribution

As an executable JAR file, the AgroVoc Extractor requires solely the presence of a compatible Java Runtime Environment distribution in the host system. AgroVoc Extractor is compatible with JRE 7 [1].

2.2. Using the Source code

Download the source code of the component from the OpenMinted GitHub repository [2] and run from the command line:

```
ant -buildfile build.xml
```

2.3. Testing the installation

AgroVoc Extractor can be called without any arguments, using a default exemplary text input included in the distribution package.
3. Relevant Data Processing Scenarios

3.1. Extract AgroVoc terms found in a document

The user aims to retrieve the AgroVoc terms found in a text document, named `<filename>`, found in the local `<path>` directory. To this end, she executes the AgroVoc Extractor component from the command line:

```
java -jar AK_Agrovoc.jar <path> <filename> <lang> (0|1) [ontology_path]
[ontology_filename]
```

The `<lang>` parameter defines the language of the input text document, expressed as an ISO 639-1 Alpha-2 code [3].

The value of the last parameter specifies if the user wants to use the FAO service (http://web.archive.org/web/20160407175423/https://bitbucket.org/aims-fao/agrovoc-web-services/) for searching a given term (1), or if she wants to use a local copy of AgroVoc (0). The local AgroVoc copy should be located in the [ontology_path] directory and should be named [ontology_filename].

Upon completion, the component produces the `<filename>.AgroVoc.xml` file, which contains information on the presence and frequency of any AgroVoc term discovered, namely:

- The term that was found in the input document;
- The Lucene score (a variation of the tf-idf metric) of the term in the input document;
- The AgroVoc code for the term.

**Exemplary XML record in the output document**

```
<term>
  <text>agriculture</text>
  <code>203</code>
  <score>0.2628660500049591</score>
</term>
```


Last updated 2017-03-21 12:37:32 EET
RSS Feed Reader
1. Component Description

The RSS Feed Reader is a stand-alone component responsible for accessing, retrieving and downloading RSS Feeds from specific sources.

The component is distributed as an executable JAR file, thus it can be used in any platform.


Upon completion of its execution, it produces an XML document for each source feed, containing the RSS entries of the source as they were returned by its RSS service.
2. Component Installation

2.1. Using the JAR Distribution

As an executable JAR file, the RSS Feed Reader requires solely the presence of a compatible Java Runtime Environment distribution in the host system. RSS Feed Reader is compatible with JRE 7 [3].

2.2. Using the Source code

Download the source code of the component from the OpenMinted GitHub repository [4] and run from the command line:

```shell
ant -file build.xml
```
3. Relevant Data Processing Scenarios

3.1. Obtain RSS Feeds from relevant sources

The user aims to retrieve and locally store the RSS feeds from the supported sources. To this end, she executes the RSS Feed Reader component from the command line:

```java
java -jar AK_RSS_Reader.jar <path>
```

Upon completion, the component produces a set of XML files, following the `<source>`, `<datetime>.xml` naming pattern, denoting the source site (`<source>`) and the timestamp on which the retrieval of the feed took place (`<datetime>`).

---


Last updated 2017-03-21 12:37:32 EET
GeoNames Extractor
1. Component Description

The GeoNames Extractor is a stand-alone component responsible for discovering GeoNames entities within text segments.

The component is distributed as an executable JAR file, thus it can be used in any platform.

GeoNames Extractor operates over a locally stored text file that contains the text to be analyzed for the presence of terms corresponding to entities defined within GeoNames.

Upon completion of its execution, it produces an XML document following the schema defined by a simple XSD schema, included with the distribution package of the component. Furthermore, it returns the RDF description as defined in GeoNames for each of the discovered entities.
2. Component Installation

2.1. Using the JAR Distribution

As an executable JAR file, the GeoNames Extractor requires solely the presence of a compatible Java Runtime Environment distribution in the host system. AgroVoc Extractor is compatible with JRE 7 [1].

2.2. Using the Source code

Download the source code of the component from the OpenMinted GitHub repository [2] and run from the command line:

```bash
ant -buildfile build.xml
```

2.3. Testing the installation

GeoNames Extractor can be called without any arguments, using a default exemplary text input included in the distribution package.
3. Relevant Data Processing Scenarios

Extract GeoNames entities found in a document. The user aims to retrieve the GeoNames entities found in a text document, named `<filename>`, found in the local `path` directory. To this end, she executes the GeoNames Extractor component from the command line:

```
java -jar AK_GeoNames.jar <path> <filename> <username> <maxrows>
```

The `<username>` parameter corresponds to the GeoNames account of the user. A registration for a GeoNames account is carried out via the GeoNames site [3].

The `<maxrows>` parameter defines the amount of lines that will be included in the response RDF delivered by the GeoNames Search web service used by the component [4].

Upon completion, the component produces the `<filename>.GeoNames.xml` file, which contains information on the presence and frequency of any GeoNames entity discovered, namely:

- The lexicalization of the entity that was found in the input document;
- The Lucene score (a variation of the tf-idf metric) of the term in the input document;
- The GeoNames ID for the entity.

Exemplary XML record in the output document

```
<term>
  <text>Greece</text>
  <geoname_id>390903</geoname_id>
  <score>0.262866050049591</score>
</term>
```

Furthermore, it produces a `<filename>.entity_name.rdf` file for each GeoNames entity discovered in the input text, containing the `<maxrows>` first lines of its RDF description in the GeoNames knowledge base.


Last updated 2017-03-21 12:37:32 EET
Geopolitical Extractor
1. Component Description

The Geopolitical Extractor is a stand-alone component responsible for discovering terms defined in the FAO Geopolitical ontology [1].

The component is distributed as an executable JAR file, thus it can be used in any platform.

Geopolitical Extractor operates over a locally stored text file that contains the text to be analyzed for the presence of FAO Geopolitical terms.

Upon completion of its execution, it produces an XML document following the schema defined by a simple XSD file, included with the distribution package of the component. The XML presents the terms found in the document ranked according to a modification of the classic tf-idf metric.
2. Component Installation

2.1. Using the JAR Distribution

As an executable JAR file, the GeoPolitical Extractor requires solely the presence of a compatible Java Runtime Environment distribution in the host system. GeoPolitical Extractor is compatible with JRE 7 \(^2\).

2.2. Using the Source code

Download the source code of the component from the OpenMinted GitHub repository \(^3\) and run from the command line:

\texttt{ant -f \texttt{build.xml}}
3. Relevant Data Processing Scenarios

3.1. Extract Geopolitical terms found in a document

The user aims to retrieve the references Geopolitical terms found in a text document, named <filename>, found in the local <path> directory. To this end, she executes the GeoPolitical Extractor component from the command line:

```
java -jar AK_Geopolitical.jar <path> <filename> <ontology_path> <ontology_filename>
```

The components require the existence of a local copy of the FAO Geopolitical ontology, named <ontology_filename> and located in the <ontology_path> directory of the host system.

Upon completion, the component produces the <filename>.GeoPolitical.xml file, which contains information on the presence and frequency of any geopolitical terms discovered, namely:

- The lexicalization of the term (in English) that was found in the input document;
- The Lucene score (a variation of the tf-idf metric) of the term in the input document;

Exemplary XML record in the output document

```
<term>
  <text>Agiourgikos</text>
  <oivid>50</oivid>
  <score>0.39380478658947754</score>
</term>
```

---


Last updated 2017-03-21 12:37:32 EET
4. **AS-C: Microbial Biodiversity**
Introduction

At the first release of this application, we have developed a text-mining pipeline using the AlvisNLP/ML platform, as well as a web application (semantic search engine) which makes use of the text-mining results and allows the user to search information about microbial biodiversity (available at http://bibliome.jouy.inra.fr/demo/pubmed-ontobiotope/alvisir/webapi/search?).
Data Processing Scenarios

In AS-D, we have outlined the following data processing scenarios:

- **Application design** (corpus building, resources gathering, etc.)
- **Production processing** (automatic annotation of a corpus by a text-mining pipeline and integration of the results into a parent application)
- **Resource update** (corpus increment, external resource update)

The web application of this first release is part of the "Production processing" scenario and is fully documented [here](#). As the results of the text-mining pipeline could not be integrated into the parent application targeted by this use case at the time of the first release, we integrated them into the above-mentioned web application instead. This semantic search engine is close to what will be available in the final application since it provides an easier access to information about microbial biodiversity from the scientific literature.

Last updated 2017-03-13 16:25:48 EET
Automatic annotation of a corpus

The AS-C text-mining pipeline automatically extracts the following entities:

- Microbial taxa (normalized according to the NCBI taxonomy)
- Microbial habitats (normalized according to the OntoBiotope ontology)
- Relationships between taxa and their habitats ("lives_in" relation)
Integration into a parent application

As the results of the text-mining pipeline could not be integrated into the parent application targeted by this use case at the time of the first release, we integrated them into a semantic search engine instead. This semantic search engine is close to what will be available in the final application since it provides an easier access to information about microbial biodiversity from the scientific literature. It is a web application, available at: http://bibliome.jouy.inra.fr/demo/pubmed-ontobiotope/alvisir/webapi/search?.

Detailed documentation on how to use the search engine can be found in the AS-C web application documentation.

Last updated 2017-03-15 18:16:11 EET
Application goal

One of the main goal of the microbial biodiversity TDM use case is the completion of the knowledge described in databases with knowledge from the literature and other databases on microbial biodiversity, as well as their comparison for further analysis. A key knowledge in microbiology is the microbial biotope, its properties and the phenotypes of the microorganism. Most of this information is expressed in free text. Information content analysis and standardization calls for information extraction tools that can automatically analyze descriptions of microorganism biotopes so that biotope descriptions originating from different experiments can be compared at a large scale. Here analysis means not only the extraction of the relevant spans of text, but also the normalization or categorization with reference resources (e.g. taxonomy of organisms, ontology of habitat, ontology of phenotypes, ontology of physico-chemical properties, etc.).
Application description

This version of the application consists of the automatic recognition and normalization of 1.6 million microorganism taxa and their 3.6 millions habitats and the automatic identification of 3.63 millions relationships between them. The corpus is composed of 1.16 million PubMed references. The taxa are normalized according to NCBI taxonomy. The habitats are normalized according to OntoBiotope Ontology. The TDM components and the workflow application are deployed on Alvis platform at INRA MaIAGE lab. The end-user can use two front-end tools to search the extracted information a semantic search engine or a treemap builder. Alvis Semantic Information Retrieval engine is web application, publicly available usable with any web browser.

Reference

Microbial Biodiversity Search

A separate page for each ‘data processing scenario’ from the Application Design report for that application, outlining the execution of the scenario from the user’s perspective.
User profile

The user of the microbial biodiversity search engine is a microbiologist that want to collect information about the type of environment where a bacteria or a class of bacteria live according to the literature. Conversely the microbiologist may want to collect the types of bacteria that live in a given environment.
Architecture of the application

The relevant corpus has been fully pre-processed by AlvisNLP pipeline and indexed for query through AlvisIR engine available from any web browser.
User manual, How-to

Launching the application

The URL to run AlvisIR search engine on Microbial Biodiversity is: http://bibliome.jouy.inra.fr/demo/pubmed-ontobiope/alvisir/webapi/search?

![AlvisIR search engine](image)

A click on the i at the left of the query field display the query language.

alvisIR home Aclick

Query

The query is typed in the query field.

![AlvisIR search engine](image)

Results
The results of the query are displayed in four frames:

- the query interpretation (blue frame)
- the list of hits (pink frame)
- the facets (yellow frame)
- the page number (green frame).

**Query interpretation**

The query is interpreted according to the syntax of the query language. The terms of the query that designate bacteria taxa with respect to the NCBI taxonomy or habitats with respect to the OntoBiotope ontology are interpreted as concepts of these hierarchies. -loc is interpreted as the "localization relation between the taxon term at left and the habitat term at right. The query interpretation frame displays the interpretation of query terms as concepts, the list of their synonyms and subconcepts (more specific). The number of occurrences for each term is also displayed. All documents that are indexed by a synonym or a more specific term of the query term are retrieved if they respect the relationship constraint if ever.

**Hits**
Each hit represent a document that verify the query constraints. It is composed of

- the title of the document, (click on the title display the full reference on PubMed site.)
- the publication date,
- the name of the journal,
- the text of the abstract and
- the list of the indexes that is displayed by a click on the >> symbol.

The taxa and the habitat names are highlighted. The colour for each entity type corresponds to the colour of the query interpretation frame. A fine green line links the taxon to its habitats.

**Facets**

There is one facet table per index type: Microorganisms, Habitats and Journals. They give information about the set of retrieved hits. Each table gives the list of values, the total frequency and the number of documents. 10 (by default), 20 or all of them can be displayed. Click on any line refines the query: it adds to the query the term in the facet that has been selected.

**Page number**

The number of hits displayed per page can be 10, 20, 30 or 50.

**Habitat selection in the ontology**

Click on the tree icon at the right of the Search button displays the OntoBiotope ontology that indexes the habitat and a query composer.

The ontology window is composed of two parts:

- ontology navigation (pink frame)
- the query composer (yellow frame)

Click on the main window in the back closes the ontology window without query composition.
Ontology navigation

The levels of the ontology are displayed from the most general to the specific, from left to right. The triangle at the right of a concept name represents the size of the subtree it is the root of. The surface of the triangle is proportional to the depth and the branching factor. Click on the triangle to open the subtree.

Zoom is done by using the sliding button on the top of the frame.
Query composer

Click on the name of the concept adds it to the query that is displayed as a term stack. The two operators Or or And are available.

Click on Refine button adds the terms to the current query. Click on the Search button replace the current query by the new one.

Further information

Liens vers autres docs et info Alvis

Last updated 2017-03-13 15:09:15 EET
5. **AS-D: Linking Wheat Data With Literature**
Introduction

At the first release of this application, we have developed a text-mining pipeline using the AlvisNLP/ML platform, as well as a web application (semantic search engine) which makes use of the text-mining results and allows the user to search information about wheat phenotypes (available at http://bibliome.jouy.inra.fr/demo/wheat/alvisir/webapi/search).
Data Processing Scenarios

In AS-C, we have outlined the following data processing scenarios:

- **Application design** (corpus building, resources gathering, etc.)
- **Production processing** (automatic annotation of a corpus by a text-mining pipeline and integration of the results into a parent application)
- **Resource update** (corpus increment, external resource update)

The web application of this first release is part of the "Production processing" scenario and is fully documented [here](#). As the results of the text-mining pipeline could not be integrated into the parent applications targeted by this use case at the time of the first release, we integrated them into the above-mentioned web application instead. This semantic search engine is close to what will be available in the final applications since it provides an easier access to information about wheat phenotypes from the scientific literature.

Last updated 2017-03-13 16:13:34 EET
Automatic annotation of a corpus

The AS-D text-mining pipeline automatically extracts the following entities:

- Genes
- Markers
- Wheat phenotypes
- Wheat-related species (taxa)

All configuration files, AlvisNLP workflow plans, scripts, and resources used to create the AS-D text-mining pipeline are provided in the uc-tdm-AS-D repository in the openminted github (https://github.com/openminted/uc-tdm-AS-D). Documentation for running the pipeline to annotate and index a corpus is also included in the repository.
Integration into a parent application

As the results of the text-mining pipeline could not be integrated into the parent applications targeted by this use case at the time of the first release, we integrated them into a semantic search engine instead. This semantic search engine is close to what will be available in the final application since it provides an easier access to information about wheat phenotypes from the scientific literature. It is a web application, available at: http://bibliome.jouy.inra.fr/demo/wheat/alvisir/webapi/search.

Detailed documentation on how to use the search engine can be found in the AS-D web application documentation.

Last updated 2017-03-15 17:41:11 EET
Application goal

This application offers access to highly relevant articles on wheat in the field of phenomics, i.e. the study of physical and biological traits of organisms in their relation to genetic mutations and environmental change. The user can browse and search papers mentioning phenotypes, but also genes, markers, and taxa that have been identified and normalized beforehand by a text-mining workflow. Unlike classical bibliographic databases that provide information on the general scope of the article, the Wheat Phenotypes application states the presence of the very objects that the user is interested in. In addition, the semantic indexing relieves the user from language variation, providing results in a harmonized way based on common vocabularies and nomenclatures.
Application description

This version of the application consists in the automatic recognition and normalization of phenotypes, genes, markers, and wheat-related taxa from scientific abstracts.

The corpus is composed of 3,881 abstracts from 414 journals downloaded from the Web of Science website. 8,996 phenotypes, 10,537 genes, 2,749 markers and 13,015 taxa have been annotated and indexed by the TDM workflow. Taxa and markers are normalized according to identifiers from the URGI database. Phenotypes are normalized according to the Wheat Phenotype Ontology. Genes are normalized according to an in-house lexicon.

The TDM components and the workflow application are deployed on Alvis platform at INRA MaIAGE lab. The end-user can use a semantic search engine to search the extracted information. The Alvis Semantic Information Retrieval engine is a web application, publicly available and usable with any web browser (available at http://bibliome.jouy.inra.fr/demo/wheat/alvisir/webapi/search).
Wheat Phenotype Search
User profile

The targeted user of the wheat phenotype search engine is a breeder who wants to access the literature relating to genes that may be involved in a given trait of a specific wheat subspecies. Conversely, the breeder may be interested in any phenotypic-related information on a given gene.
Architecture of the application

The relevant corpus has been fully pre-processed by the AlvisNLP pipeline and indexed for query through the AlvisIR engine available from any web browser.
User manual, How-to

Launching the application

The URL to run the AlvisIR search engine on Wheat Phenotypes is: http://bibliome.jouy.inra.fr/demo/wheat/alvisir/webapi/search

Clicking on the “i” at the left of the query field displays the query language.

Query

The query is typed in the query field.
Results

The results of the query are displayed in four frames: *the query interpretation (blue frame)* the list of hits (pink frame) *the facets (yellow frame) *the page number (green frame).

Query interpretation

The query is interpreted according to the syntax of the query language. The terms of the query that designate phenotypes with respect to the Wheat Phenotype ontology are interpreted as concepts of this hierarchy.

The query interpretation frame displays the interpretation of query terms as concepts, the list of their synonyms and subconcepts (more specific). The number of occurrences for each term is also displayed. All documents that are indexed by a synonym or a more specific term of the query term are retrieved.

Hits

Each hit represents a document that verify the query constraints. It is composed of: *the title of the document, *the publication date, *the name of the journal, *the text of the abstract (whenever available), and *the list of the indexed entities (displayed by clicking on the >> symbol) The taxa, gene, marker, and phenotype names are highlighted. The colour for each entity type corresponds to the colour of the query interpretation frame.

Facets

There is one facet table per index type: Gene, Marker, Phenotype, Taxon, Journal and Year.
They give information about the set of retrieved hits. Each table gives the list of values, the total frequency and the number of documents. 10 (by default), 20 or all of them can be displayed. Clicking on any line refines the query: it adds to the query the term in the facet that has been selected.

**Page number**

The number of hits displayed per page can be 10, 20, 30 or 50.

**Phenotype selection in the ontology**

Clicking on the tree icon at the right of the Search button displays the Wheat Phenotype ontology that indexes the plant traits and a query composer.

The ontology window is composed of two parts: *ontology navigation (pink frame) * the query composer (yellow frame)

Clicking on the main window in the back closes the ontology window without query composition.

**Ontology navigation**

The levels of the ontology are displayed from the most general to the specific, from left to right.
The triangle at the right of a concept name represents the size of the subtree it is the root of. The surface of the triangle is proportional to the depth and the branching factor. Clicking on the triangle opens the subtree.

Zooming is done by using the sliding button on the top of the frame.

**Query composer**

Clicking on the name of the concept adds it to the query that is displayed as a term stack. The two operators Or or And are available.
Clicking on Refine button adds the terms to the current query. Clicking on the Search button replaces the current query by the new one.

Last updated 2017-03-13 16:13:34 EET
6. **AS-E: Extracting gene regulation networks involved in seed development (SeeDev)**
Introduction

At the first release of this application, we have developed a text-mining pipeline using the AlvisNLP/ML platform, as well as a web application (semantic search engine) which makes use of the text-mining results and allows the user to search information about seed development (available at http://bibliome.jouy.inra.fr/demo/seedeve/alvisir/webapi/search).
Data Processing Scenarios

In AS-E, we have outlined the following data processing scenarios:

- **Application design** (corpus building, resources gathering, etc.)
- **Production processing** (automatic annotation of a corpus by a text-mining pipeline and integration of the results into a parent application)
- **Resource update** (corpus increment, external resource update)

The web application of this first release is part of the "Production processing" scenario and is fully documented [here](#). As the results of the text-mining pipeline could not be integrated into the host application targeted by this use case at the time of the first release, we integrated them into the above-mentioned web application instead. This semantic search engine is close to what will be available in the final application since it provides an easier access to information about seed development from the scientific literature.

Last updated 2017-03-13 18:11:11 EET
Automatic annotation of a corpus

The AS-E text-mining pipeline automatically extracts the following entities:

- Genes
- Proteins
- RNA
- Gene families
- Protein families

All configuration files, AlvisNLP workflow plans, scripts, and resources used to create the AS-E text-mining pipeline are provided in the uc-tdm-AS-E repository in the openminted github (https://github.com/openminted/uc-tdm-AS-E). Documentation for running the pipeline to annotate and index a corpus is also included in the repository.
Integration into a parent application

As the results of the text-mining pipeline could not be integrated into the parent application targeted by this use case at the time of the first release, we integrated them into the semantic search engine instead. This semantic search engine is close to what will be available in the final application since it provides an easier access to information about seed development from the scientific literature. It is a web application, available at: http://bibliome.jouy.inra.fr/demo/seedev/alvisir/webapi/search.

Detailed documentation on how to use the search engine can be found in the AS-E web application documentation.

Last updated 2017-03-15 18:11:16 EET
Application goal

One of the main goals of the SeeDev TDM use case is aimed at researchers in plant breeding, especially those interested in plant reproduction and seed development. The main objective of research laboratories involved in plant development research is the understanding of biological mechanisms at different scales: molecular, biological, physiological and the whole plant in its environment (i.e. in field). They need to know about relations between different biological entities, such as gene, protein and the conditions in which the regulation occurs (e.g. tissue and so on). A better understanding of regulations allows improvement of seed development that is the basis for many industrial uses. This kind of information is described in the literature but not in structured databases that the user can query and cannot be found or directly extracted from experimental or bioinformatics data. Biologists regularly express a need for an improved access to documents related to their study subjects within a familiar environment.

The integration of a TDM module aims at improving the data discovery, and at assisting users in their bibliography search. The TDM module will provide normalization of unstructured entities and enhanced search capabilities. Here analysis means not only the extraction of the relevant spans of text, but also the normalization or categorization with reference resources (e.g. Gene and Protein list from TAIR).
Application description

The end-user can use a semantic search engine to search the extracted information. The corpus is composed of 45 full scientific texts (including whose from SeeDev BioNLP-ST 2016), and 2000 abstracts from WebOfKnowledge. This version of the application consists of the automatic recognition and normalization of Protein (17968), Gene (9490), theirs families (188), and RNA (55). The Gene and Protein entities are normalized according to TAIR list (release 10). The Gene and Protein families are normalized according to TAIR list and, if not available, by the gene/protein identifiant from TAIR plus the word "family". The RNA are normalized from the miRTarBase list. The TDM components and the workflow application are deployed on Alvis platform at INRA MaIAGE lab and available on-line on GitHub. Alvis Semantic Information Retrieval engine is web application, publicly available usable with any web browser.

Reference

SeeDev
User profile

The user of the SeeDev search engine is a plant biologist who wants to collect information about Gene, Protein, and their respective families, but also RNA. = Architecture of the application
The relevant corpus has been fully pre-processed by AlvisNLP pipeline and indexed for query through AlvisIR engine available from any web browser. = User manual, How-to ===
Launching the application The URL to run AlvisIR search engine on Microbial Biodiversity is:
http://bibliome.jouy.inra.fr/demo/seedev/alvisir/webapi/search

A click on the icon at the left of the query field displays the query language.

Query

The query is typed in the query field.
Results

The results of the query are displayed in four frames: the query interpretation (blue frame), the list of hits (pink frame) the facets (yellow frame) the page number (cyan frame).

Query interpretation

The query is interpreted according to the syntax of the query language. The terms of the query that designate gene and protein list with respect to the TAIR list are interpreted as concepts of these lexicons, (protein-family)* is interpreted as the matches with concept from the Protein family list from TAIR, but also all term identified as (protein + term which indicates the family apparenance. The query interpretation frame could displays the interpretation of query terms as concepts, the list of their synonyms and subconcepts (more specific). The number of occurrences for each term is also displayed. All documents that are indexed by a synonym or a more specific term of the query term are retrieved if they respect the relationship constraint if ever. The number of hits displayed per page can be 10, 20, 30 or 50.
Hits

Each hit represents a document that verify the query constraints. It is composed of the title of the document, (click on the title display the full reference on PubMed or DOI websites.) the text of the abstract and an extract of the full text if matches and if available the list of the indexes that is displayed by a click on the >> symbol. === Facets There is one facet table per index type: Gene, Protein, Protein Family Gene Family, RNA and Journals and Publication years. They give information about the set of retrieved hits. Each table gives the list of values, the total frequency and the number of documents. 10 (by default), 20 or all of them can be displayed. Click on any line refines the query: it adds to the query the term in the facet that has been selected. Results could be viewed by a click on the symbol.
Further information

This TDM application will be integrated into the FLAGdb (http://tools.ips2.u-psud.fr/projects/FLAGdb/HTML/index.shtml), a reference tool for biologists who work on the genetics of plants.

Last updated 2017-03-13 16:26:02 EET
7. **SS-A: Facilitation of complex information linking and retrieval from social sciences publications**
Introduction

This software package contains components for text mining in scientific publications. Current package contains the following components:

- PDF-to-XMI converter: for transforming PDF documents (publication full-texts) to XMI format.
- Named entity recognizer: for identifying named entities in XMI document (supports English and German)

To learn more about each component you can refer to its corresponding section in this manual, and to find example code you can refer to the corresponding test classes for each component.

Useful links

Get sources from GitHub:

https://github.com/openminted/uc-tdm-socialsciences

To use the latest release, please add the following piece to your pom file:

```xml
<dependency>
    <groupId>eu.openminted.uc-tdm-socialsciences</groupId>
    <artifactId>openminted.uc-tdm-socialsciences</artifactId>
    <version>1.0.0</version>
</dependency>
```

To use the latest snapshot, please add the following piece to your pom file:
Developers mailing list (request to join):

https://groups.google.com/forum/#!forum/uc-tdm-socialsciences-developers
Components

PDF processing

PDF to XMI conversion

The PDF-to-XMI conversion pipeline can be used to convert a collection of PDF files to XMI format.

Usage: command line

This pipeline can be used via calling the main method, like the following:

```bash
$ java -jar /PATH/TO/ss-io-pdf-x-xml-0.0.1-SNAPSHOT-pdf-xmi-pipeline-standalone.jar [args...]
```

The following arguments have to be provided for the program:

- **-i <path>** path to input PDF documents. This path can either point to a single file or a directory.
- **-o <path>** path to output directory.
- **-lang <value>** Language of input documents. Possible values: en or de
- **-overwrite** (Optional) Overwrite output flag. If this option is set program will overwrite any existing files in output directory.
- **-home** Path to application home where required files (e.g. dictionary files) are located. You can download the dictionary files from [here](https://github.com/openminted/uc-tdm-socialsciences/releases) (ss-io-dictionaries.zip).

Usage: java

Add the following dependency to your pom file:

```xml
<dependency>
    <groupId>eu.openminted.uc-tdm-socialsciences</groupId>
    <artifactId>ss-io-pdf-x-xml</artifactId>
</dependency>
```

Create an example method like the following:
import eu.openminted.uc.sciences.io.pdfx.Pipeline;

public class PipelineTest
{
   public void runPipelineExample()
   {
      Pipeline pipeline = new Pipeline();
      pipeline.setInput("my/input/directory");
      pipeline.setOutput("my/output/directory");
      pipeline.setLanguage(Pdf2XmConverter.LANGUAGE_CODE_EN);
      pipeline.setHomePath("my/dictionary/files/directory");
      pipeline.run();
   }
}

The language parameter is required to correctly post-process the converted text files (e.g. remove hyphenations inside words that are split into two lines). This attribute will also be stored in the output XMI files so it can be used in subsequent text analysis components (e.g. named entity recognition).

Currently PDF-to-XML converter can only convert files smaller than 5 MB and containing less than 100 pages. This limit is enforced by the PDFX service which is used internally to convert PDF documents to XML format.

PDF to PDFX-XML conversion

The PdfxXmlCreate class can be used to convert a collection of PDF files to PDFX XML format (schema (http://pdfx.cs.man.ac.uk/static/article-schema.xsd)). This class is used internally by PDF-to-XMI converter component. This component can be used via calling the main method, like the following:

```
$ java -cp PATH TO ss-i o-pdfx-xml-0.0.1-SNAPSHOT-pdf-xmli ne-st andal one.jar eu.openmint ed.uc.soci alsciences.io.pdf x.PdfxXmlCreate [args...]
```

The following arguments have to be provided for the program:

- `-i <path>` path to input PDF documents. This path can either point to a single file or a directory.
- `-o <path>` path to output directory.
- `-overwrite` (Optional) Overwrite output flag. If this option is set program will overwrite any existing files in output directory.
PDFX-XML to XMI conversion

The `Pdf xMl ToXmI Converte` class can be used to convert a collection of PDFX XML documents to XMI format. This class is used internally by PDF-to-XMI converter component. This component can be used via calling the main method, like the following:

```
$ java -cp /PATH TO ss-i o-pdf x- xml -0. 0. 1- SNAPSHOST- pdf - xml - pi pel i ne- st andal one. jar
eu. openmi nt ed. uc. soci al sciences. i o. pdf x. Pdf xMl ToXmI Converte [ a rg s . . ]
```

The following arguments have to be provided for the program:

- `-i <path>` path to input directory containing pdfx XML files
- `-o <path>` path to output directory to save converted XMI files
- `-overwrite` (Optional) Overwrite output flag. If this option is set program will overwrite any existing files in output directory.
- `-lang` language of input documents.
- `-home` Path to application home where required files (e.g. dictionary files) are located

Named Entity Recognition

With this module, you can perform Named Entity Recognition (NER) on your data.

You can:

- input some annotated data to train your own NER model
- apply a trained NER model to new, un-annotated data
- evaluate the performance of any NER model

Training your own model is optional, you might also already have a pre-trained model and use that, or use a model provided by third parties (e.g. Stanford website).

Training your own NER model

To train your own NER model, you will need annotated training data. The training data has to be in TSV format with one token per line, each sentence separated by a blank line, with the tokens in the first column and annotations in the second column.

Preparing training data

If your training data is in binary CAS format (e.g. exported from WebAnno), you can use the `Binar yCasToSt anf or dTsvConverte` to perform the conversion.
Usage:

$ java -cp /PATH/TO/ss-module-ner-0.0.1-SNAPSHOT:ss-ner-standalone.jar
eu.opennmt.ted.uc.soci.alsciences.nerelper.BinaryCasToTsvConverter [args...]

The following arguments have to be provided for the program:

- **-i <path>** path to input documents containing annotations in binary CAS format. This path can either point to a single file or a directory.
- **-o <path>** (optional) path of output file. Default: /stanfordTrain.tsv
- **-subtypes <value>** [optional] useSubTypes flag. If set, value and modifier of an annotation will be merged to create more fine-grained classes.

To see the difference the setting of the -subtypes flag makes, consider the following excerpt from a training data file. In the first case, the flag is set:

```
by     O
the    O
Community B-ORGpar
Party   I-ORGpar
.

For     O
instance O
.
researchers O
at       O
the      O
Institute B-ORGsci
of       I-ORGsci
Economics I-ORGsci
```

This results in different labels for the two entities that are different kinds of organizations (ORG). Whereas in the second case, the flag has not been set:
Here, both are labeled with the same coarse class 'organization' (ORG). Thus, setting the -subtypes flag allows to differentiate sub-types of annotations, but mind that this means an increase in the number of classes for training.

Model training

You can use the StanfordNer class to train a new NER model with training data. You will have to provide a file containing the training properties:

```
$ java -cp /PATH/TO/ss-modul e-ner-0.0.1-SNAPSHOT-ss-ner-standal one.jar eu.openmi nt.ed.uc.soci al.sci en ces.ner .tra in. StanfordNer [args...]
```

The following arguments have to be provided for the program:

- **-i <path>** path to file with training data in .tsv format
- **-o <path>** (optional) path of output file for the serialized model. Default: /omtd-ner-model.ser.gz
- **-t <value>** path to the training properties file

The file containing the parameters for training has to be in properties format, i.e. one parameter per line in key-value-pairs like this:
parameter  =  value

You can find detailed descriptions of available training parameters in the FAQ of the Stanford CoreNLP NER: http://nlp.stanford.edu/software/crf-faq.html

Mind that it is in general possible to set paths to training file(s) and model output file also in the training properties file, but these values will be overridden.

**Apply a NER model to un-annotated data**

With the Peline class, you can input un-labeled data and apply a NER model to it, such that the output will contain labels for all recognized Named Entities.

Input data has to be in XMI (UIMA) format, so if you want to label text from PDF, convert them first (see PDF to XMI conversion). You can provide the path to a model in case you pre-trained a model on your own data yourself. You can also specify to use one of the pre-trained models that are available (but mind that those models are mostly trained on newswire text, so if you apply those models to a different domain, the results may have not the quality you expect).

**Usage:**

```bash
$ java -cp /PATH/TO/ss-modul-e-ner-0.0.4-SNAPSHOT-ss-ner-standalone.jar eu.openen.it.eds.ueit.projecttms.nlp.Peline [args...]
```

The following arguments have to be provided for the program:

- **-i <path>** path to input data to be labeled. Can also be a pattern for matching files in a directory, e.g. /*/*.xmi
- **-o <path>** path to output directory.
- **-standardModel** (optional) Use standard stanford model flag. If this flag is set, standard Stanford models will be used instead of a custom model.

The results will be written again to XMI files, containing the annotations produced by the Named Entity Recognizer. Example:

```xml
<NamedEntity:LOC xml:id="46865" sof="46711" begin="7014" end="7027" value="LOC" />
```

**Evaluate the performance of NER model**

We also provide a means to evaluate the results of NER. Use `PerformanceMeasure` for evaluation. You will need gold data, i.e. manually annotated data with the correct NE labels.
And of course you will need the prediction data, i.e. documents annotated with the NER. Both have to be in XMI format again.

Usage:

```
$ java -cp /PATH/TO/ss-module-ner-0.0.1-SNAPSHOT-ss-ner-standalone.jar
eu.openmi.nt.ed.uc.soci.al sciences.ner.eval.PerformanceMeasure [args...]
```

The following arguments have to be provided for the program:

- **-iGold** `<path>` path to gold data with correct labels. Can also be a pattern for matching files in a directory, e.g. `/*/*.xmi`
- **-iPred** `<path>` path to prediction data, i.e. labeled by an algorithm. Can also be a pattern for matching files in a directory, e.g. `/*/*.xmi`
- **-strictId** (optional) If set, for each Gold-document there should be a Prediction-document in the prediction set with identical documentId (cf. documentId attribute in xmi file). If this requirement is not satisfied, program will not work properly.
- **-v** (optional) verbose output flag. If this flag is set, output will contain comprehensive information about tags found in gold and prediction sets.

The program will output agreement scores as well as precision and recall. The output will look similar to this:

```
Calculating agreement scores for doc [ 5]
Agreement scores on file [ 5]
- Alpha for category OTHevt: -0.000365
- Alpha for category LOC: 0.539081
- Alpha for category ORGsci: 0.555925
- Alpha for category PERnd: 0.641172
- Alpha for category OTHmed: 1.000000
- Alpha for category ORGoth: 0.817166
- Alpha for category OTHoff: -0.000134
- Alpha for category SUBhe: -0.000594
- Alpha for category ORGgov: 1.000000
- Alpha for category SUBres: 1.000000
Overall Alpha: 0.691863

Calculating precision/recall scores for doc [ 5]
FMeasure scores
Overall precision: 0.601852
Overall recall: 0.970149
Overall F-Measure: 0.742857
```

Last updated 2017-01-31 16:36:31 EET
8. *SC-A: Research Analytics – Funding Mining Services*
Introduction

The Funding Mining application mines the fulltext of publications and extracts links to projects. Currently, projects from EC (FP7/H2020), NSF (National Science Foundation, USA), NIH (National Institute of Health, USA), Wellcome Trust, FCT, ARC (Australian Research Council), NHMRC (National Health and Medical Research Council, Australia), CSF/HRZZ (Hrvatska Zaklada Za Znanost, Croatia), MSES-MZOS, SFI (Science foundation Ireland), NWO (Nederlandse Organisatie voor Wetenschappelijk Onderzoek, Netherlands) are supported, but new funders are added regularly. Initially developed as part of OpenAIRE's Inference (by mining) workflow to enrich the OpenAIRE information space, it is actively used as a standalone workflow in several E.U. funding & research evaluation tenders providing useful insight and timely intelligence of systematic research in E.U.
Data Processing Scenarios

In SC-A, we have outlined the following data processing scenarios:

- Adding documents to the database
- Extracting project links
- Storing the results as JSON

Some technical information for the application can be found at: Technical Information

Last updated 2017-03-30 12:51:33 EEST
Adding documents to the database

This data processing scenario is yet to be addressed in the current SC-A application.

Last updated 2017-03-30 13:42:44 EEST
Extracting project links

The application is currently focussed around this data processing scenario.

The application is implemented as a declarative query on top of madIS (https://github.com/madgik/madis). The query gets fulltext (in XML format) in its input and produces json in its output.

The following command is running the application:

```
cat input.xml | python madis/src/mexec.py -f projects.sql -d projects.db > output.json
```

projects.sql : The declarative query that parses the xmls and extracts the links to projects.

projects.db : An SQLite database that stores information about projects that are necessary for the extraction.

There is also a web interface for experimenting at http://mining.openaire.eu/openaireplus/analyze

The user uploads his text and gets the extracted links in JSON format.

Last updated 2017-03-30 13:39:34 EEST
Storing the results as JSON

This data processing scenario is yet to be addressed in the current SC-A application.

Last updated 2017-03-30 13:39:34 EEST
Technical Information for the madIS Funding Mining Application

FundingMining_OMTD

Run standalone

cat input.xml | python madis/src/mexec.py -f PATH/projects.sql -d PATH/projects.db > output.json

Docker

Build

docker build -t <name>

Run

docker run -it -v <host-dir>:/usr/app/src/data --rm --name <name> <name>

Remember to declare with an absolute path, for example ${PWD}/ or /home/user/data.

For example:

docker run -it -v ${PWD}/xml_files:/usr/app/src/data --rm --name <name> <name>

Results

The docker images expects an xml file inside the folder and writes the result in a results.json file that is saved in the host's filesystem.

Last updated 2017-03-28 16:05:31 EEST