Developer Documentation

(Applied Research Track)

Motivation

Scientists, use search engines or recommender systems to quickly find the most relevant scientific literature in their field. However, in the research field of chemistry, chemists are not only interested in textual relevance, but also want to make find and make sense of chemical formulas contained in the literature. For example, within a publication: $C_9H_8O_4$, C9H8O4, acetylsalicylic acid, and ASA all refer to the same compound, aspirin.

For chemists it is very challenging to know all chemical entities present in a paper, their abbreviations, their synonyms, and their properties. That problem is not addressed in any recommender system and there is no support for displaying chemical entities in a paper.

The overall goal of the project is to address that problem. Building a prototype that extracts chemical entities and their mentioned properties from text (academic paper) and visualize them in a meaningful way (graphical user interface).

The Prototype must be compatible with the RecVis literature recommendation system, which is an already existing literature recommendation system. This recommender system already supports a variety of different measures for the similarity of papers (Text, Citation, Image, Mathematical Formulas), which can be weighted different to get a new recommendation. This literature recommendation system uses the Hyplag Backend, and the developed prototype also must be compatible with it.

Approach

The project is divided into two sections, the backend, and the frontend.

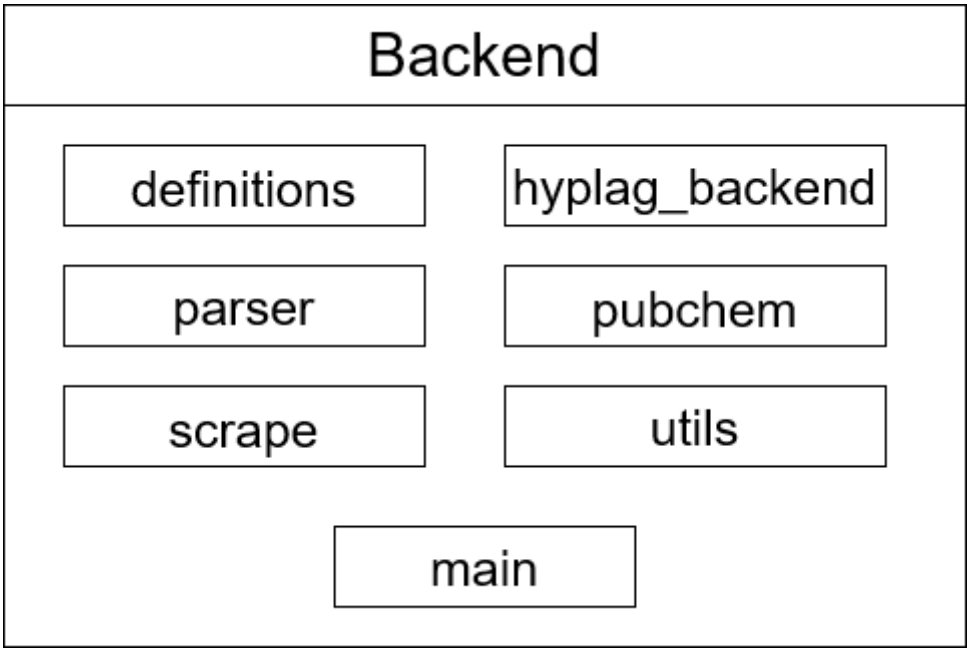


Figure 1. Backend overview

The backend is used to communicate with the Hyplag Backend and to parse and extract the chemical entities in each scientific paper. The frontend is embedded in RecVis and displays the extracted chemical entities.

Implementation

Backend

definitions.py

The Backend for the extraction of chemical formulas consists of multiple sections. The first one is the definitions.py. The file is used to make several definitions for the Hyplag backend and file paths. The definitions for the Backend are:

- Username and password for the Hyplag backend (need to be filled in)
- Hyplag URL
- URL for creating an authentication token (valid for 2 hours)
- Save path for the authentication token
- URL for posting documents to Hyplag
- ID for the document on Hyplag
- URL to get the parsed documents from Hyplag
- Save paths for pdf and xml files

hyplag_backend.py

The hyplag_backend.py file covers all communication with the Hyplag backend. The Hyplag backend itself offers a HTTP API, to which requests are sent. This API is based on Swagger.

To communicate, post and get documents from the Hyplag backend, the creation of a JSON Web Token (JWT) is necessary. To generate that token, a username and password need to be provided in the definitions. To get the token a HTTP GET request with the username and password as payload is sent. If the username and password are correct the response content, is the JSON Web Token, which is valid for two hours. For all HTTP requests, the python requests library (https://pypi.org/project/requests/) is used.

The general usage after the generation of a JWT is:

- Upload PDF file of a scientific paper
- Hyplag backend processes the PDF and returns a document id
- Hyplag Backend parses the PDF file to an XML file
- Use document ID to get the XML file

PDF files should be placed in the folder specified in the definitions. The upload of one PDF file is done with a HTTP POST request, which uploads the file to the Hyplag backend. The Hyplag backend returns a document id, which is used to do a HTTP GET request for getting the xml file, which is then used to extract the different chemical entities of the scientific paper.

parser.py

For the extraction of the chemical entities from a scientific paper the ChemDataExtractorV2 (CDE) is used (http://www.chemdataextractor2.org/). The output xml file of the Hyplag backend is used as the input for ChemDataExtractorV2. To extract the chemical entities, present in a chemical paper the CDE is using a document class, which separates the scientific paper in multiple elements:

- Metadata
    - Authors
    - DOI
    - Etc.
- Title
- Heading
- Figure
- Table
- Reference
- Paragraph

The xml file returned from the Hyplag backend uses xml tags to mark the different elements of the document. The elements are specified with a specific xpath expression:

- Title: .//titleStmt/title
- Heading: .//div/head
- Figure: //div//*//figure[contains(@xml:id, 'fig')]
- Table: //div//*//figure[contains(@xml:id, 'tab')]

If any of the elements is not marked correctly, they are not extracted. It is easy to add new xpath expressions or to change them if the output layout of the xml files from Hyplag changes.

The xml tags are used to identify the different elements of the document. The parsing of the xml file is the same for all xml layouts, they only differ in the xpath expression. First, all the specified xpath expressions are used to identify the different elements of the document and create the corresponding classes of the CDE. After that, the whole xml file is parsed recursively through all its elements. If the tag of the xml file matches one of the created objects the corresponding parser for that type of element is used. The smallest Element is a paragraph. If an element consists of multiple elements, they are split until there are only paragraphs left.

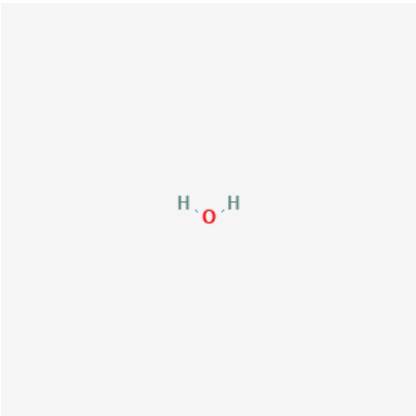
pubchem.py

PubChem is the largest open-source database for chemical information. After the extraction of chemical entities from a scientific paper, this database is used to collect a specified set of properties for those chemical entities. There are two search modes, which are used. The first one is using the name of the chemical and the second one is using the chemical formula. The following properties are extracted:

- IUPAC name of the chemical
- PubChem ID
- Molecular structure
- Molecular formula
- Molecular weight

Other properties can be added quickly, by adding the name of the property to the list of extracted properties.

The molecular structure is an PNG image, that is encoded as a Base64 string. This string can then be used in the frontend, to display the image of the molecular structure of the chemical entity.



iVBORw0KGgoAAAANSUhEUgAAASwAAAEsCAMAAAE5p
E7RAAAAn1BMVEX19fVmi4vv8PC9zMx+nZ2wwsL/AAD+Cw
v3tbX17e34oKD+AQH6bW313d3+AwP5jo717+/4lJT5kpL3sbH3
r6/9GRn+ExP3s7P+FRX7WFj6dXX7UVH8Ozv8RUX15eX8SUn
8Pz/21NT229v+CQn+Bwf8Njb6aGj6YmL8MjL4o6P9Kir5e3v4lp
b4mpr5fX39MDD5hYX5gYH22dn6bGz22NhrOeqlAAAACXBI
WXMAAA7EAAAOxAGVKw4bAAABWElEQVR4nO3d12rDM
BQGYDlNnJ107733fv9nawYUSkIWLnLi77vwMUIcfoTxhTByC
LAykplnPv1jCgDmMfu7GwBYVcnITT56DZy+ZdQIAAAAyFxe
NxUy3qBIQzeTPn274TuzXgAAAAAAAABQGMnIVwCjIxHkN
Nav7ZN2GjvDGL1ML1+xQ4xxft2IHQEAAAAAAAAAAAAA
AAAAIC4SuVQWZsyEoFY8xBrAUkIzYvb91ydmzFQOxicVJE3
m61cxqpXQ+jEDjFGrXofOwIAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAroJSUe9dKMuk38bPMKQS
LNYdSMjR5sabPKQRP1hws1qKSYdk7rl7dPPwdY6z1nf16rzx2P
mMnWQIb6WG/PKet2EmWwFZ61C+vaTd2kmXQaJ5dtu/Sj9g5

utils.py

The utils.py file is used to create a session for the HTTP requests. Those requests need a user agent string to be recognized as valid HTTP requests. The user agent strings are generated random for each session created.

- Opera/9.80 (X11; Linux x86_64; U; fr) Presto/2.9.168 Version/11.50

main.py

The main.py file combines the several other files into one function, which processes scientific papers in pdf format and outputs a list of chemical entities in those papers.

The first step is to post the pdf files to the Hyplag backend. The Hyplag backend processes the files and returns a document id. This id is used to request the xml file of the processed pdf. The xml file is then read with the reader from the parser.py. All document objects from the

ChemDataExtractorV2 have a serialize function, which is used to create a dictionary of the chemical entities of the document.

All chemical models which were used to construct the document are extracted. The model for chemical entities is the compound model. Other models for specific chemical properties of those entities can be added.

After extracting the chemical entities, additional properties are extracted from pubchem. If there is no record on pubchem for the extracted chemical entities, some part of the chemical names are replaced and the pubchem search is done again. If there is no record on pubchem after that, the chemical entity is removed. This is done for all chemical entities in the document records. After that the list of chemical entities is send to the frontend.

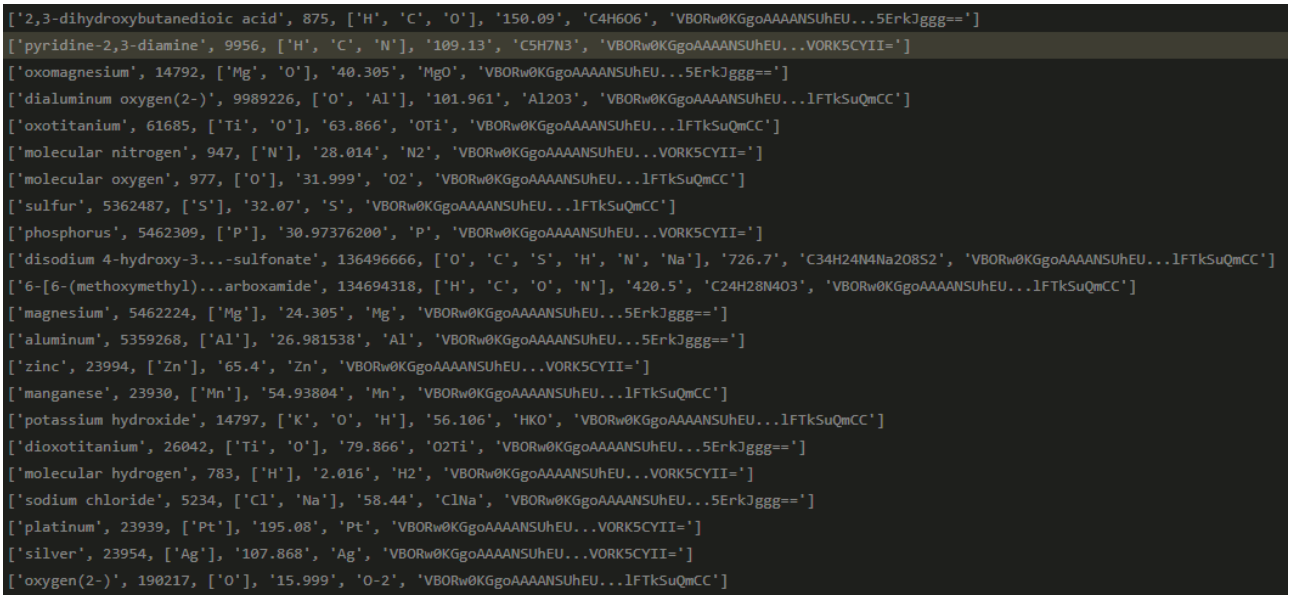


Figure 2: List of extracted chemical entities

Frontend

The frontend extends the current detailed comparison views. This includes

- changes in detailed.html and detailed.js
- creation of detailed-chemical-style.css and detailed-chemical.js

detailed.js

The detailed.js file requests all the necessary data for the frontend, after the DOM content is loaded. It enables the navigation and loads the different visulization approaches dynamically, after the requests are answered.

A new container for the chemical detailed view is added, which is called the Chemical-Container. The EventListeners are added the same as in the original. If all callbacks are resolved, the corresponding function is called.

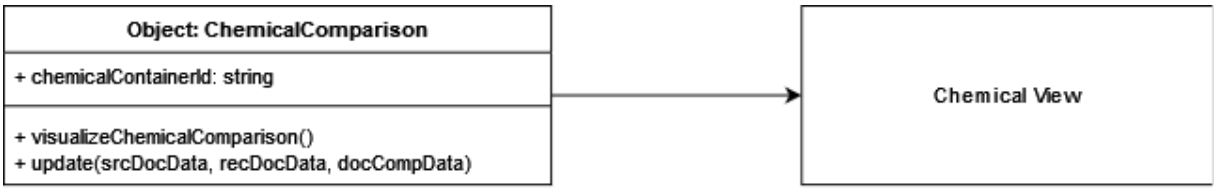


Figure 3. ChemicalComparison object creates the chemical view for the

detailed.html

The navigation is changed and a option for selecting the chemical detailed view is added.

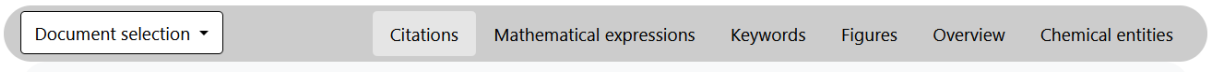


Figure 4. Selection of the different views

The different options for the chemical views, are displayed in the left flex column and can be selected by clicking them.

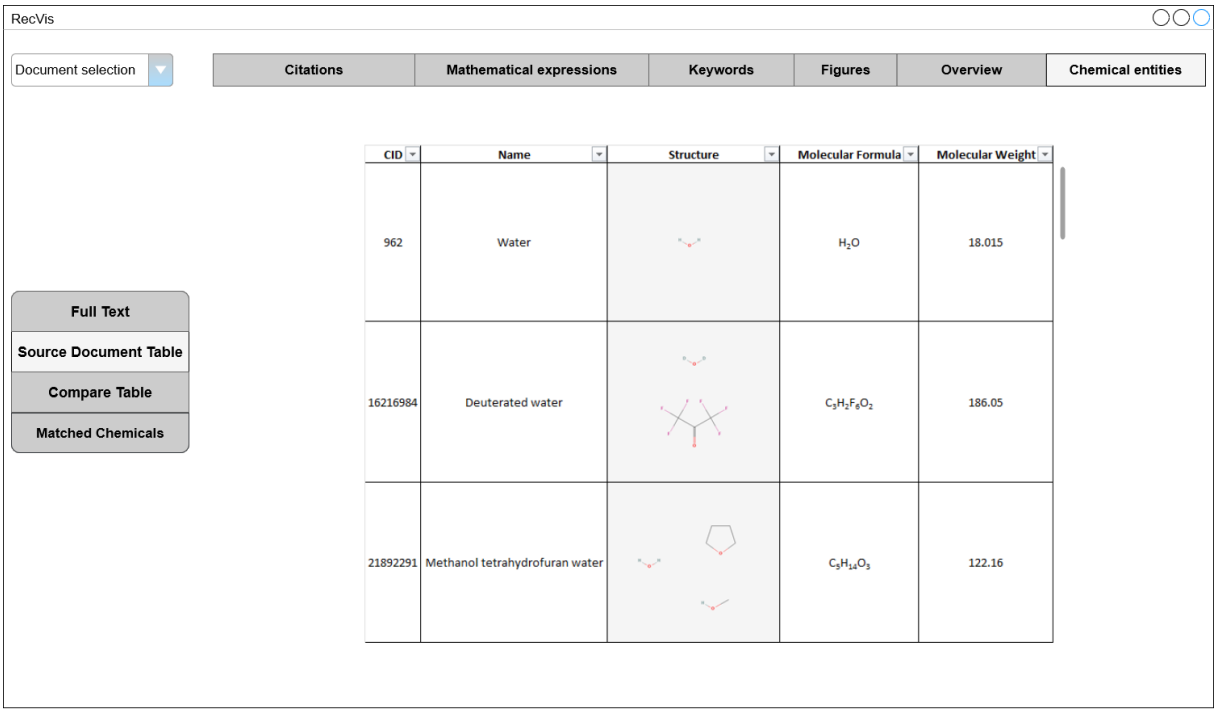


Figure 5. Source document table of chemical entities

Each option has a own flex container, which behaves similarily to the general view containers. If the option is not shown the display style of this flex container is set to none. If it is selected the display style is set to flex.

detailed-chemical.js

For the comparison of the chemical entities the tables are created dynamically from the comparison data, which is returned by the backend. After the creation of the tables, they are appended to their corresponding flexboxes.

For the comparison tables the flexbox is split in half. The source table is displayed on the left side and the recommendation table is displayed on the right side. Each chemical entity in the source, which is also present in the recommendation table is marked with the same color in both tables

Figure 6. Comparison of chemical entities in the source and recommended document

A barplot is used to display the percentage of matched chemical entities in the source and the recommended document. The barplot is created with d3.js and displayed as a svg (Scalable Vector Graphic).

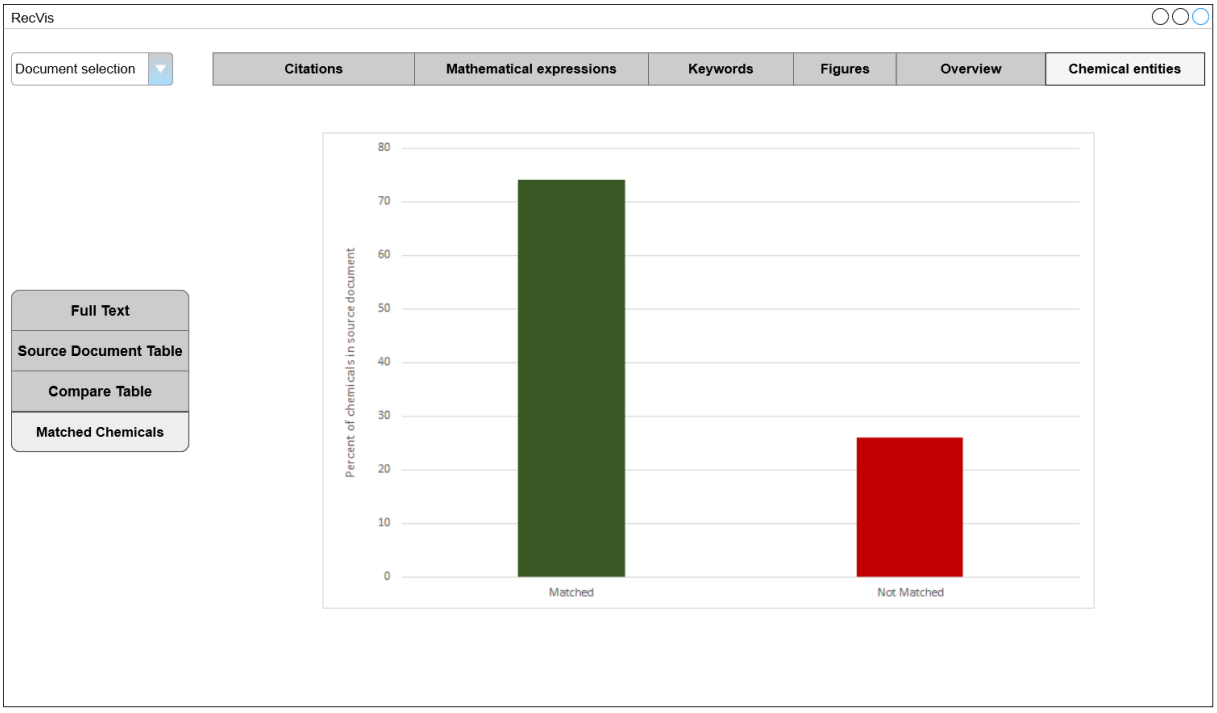


Figure 7. Barplot of matched chemical entities in the source and the recommended document