OntoChem GmbH, Blücherstr. 24, D-06120 Halle (Saale), Germany

White Paper Documentation

OC|processor Annotation Modules

For Named Entity Recognition

AUTHORS

Matthias Irmer, Claudia Bobach and Timo Böhme

Halle (Saale), September 28, 2019

version 1.3

last change 2019-09-28 02:06

ONTOCHEM GMBH
Blücherstraße 24
D-06120 Halle (Saale)
CEO:
Dr. Lutz Weber

Contact
T +49 345 478047-0
F +49 345 478047-1
info@ontochem.com
www.ontochem.com

Commercial Register
register court:
Amtsgericht Stendal
register no.: HRB 215461
USt/VAT: DE246232735

Bank Details
BIC/SWIFT:
SOLADEST600
IBAN:
DE41600501010004088854

ontochem

TABLE OF CONTENTS

CHANGELOG OF THIS DOCUMENT

- 2019-09-28 adaption to new CI design
- 2018-09-13 actualization of module descriptions (merged with frame extraction whitepaper); restructuring
- 2018-07-17 refinement of Section 4.6 Deconvolution of labels for chemical compounds
- 2017-06-08 initial version describing annotation modules of OC|miner

1 INTRODUCTION

This document contains a description of the annotation process performed by the OC|processor pipeline. Emphasis has been placed on all modules which influence the annotation process. Some modules can be used interchangeably, customized regarding their settings or are optional. Examples are given from different domains to illustrate functionality, excerpts from resource files given where valuable for understanding. The OC|processor pipeline is based on a UIMA framework being highly flexible. Settings should be customized according to the project needs.

Information regarding readers and input formats, knowledge extraction modules, consumers and output formats, as well the integration into an OC|miner Suite are available in additional OntoChem White Paper Documentations.

2 DOCUMENT PROCESSING WORKFLOW

The OC|processor pipeline is depicted in **Fig. 1**. The focus in the present document is on the process of annotating named entities.

Input documents in heterogeneous formats as available from publishers (PDF, XML, HTML,...) are converted into a uniform OC XHTML format by a range of different **reader modules**.

Pre-dictionary modules take care of the preparation of documents for annotation, e.g. removal of XML tags, language detection, text normalization and tokenization, abbreviation recognition, document structure partition.

Then, **annotation** of named entities (NE) takes place. The preferred annotation technique for highest throughput is based on **dictionaries**. For each domain, a separate dictionary is used. Alternatively, annotators based on rules or machine learning can be employed.

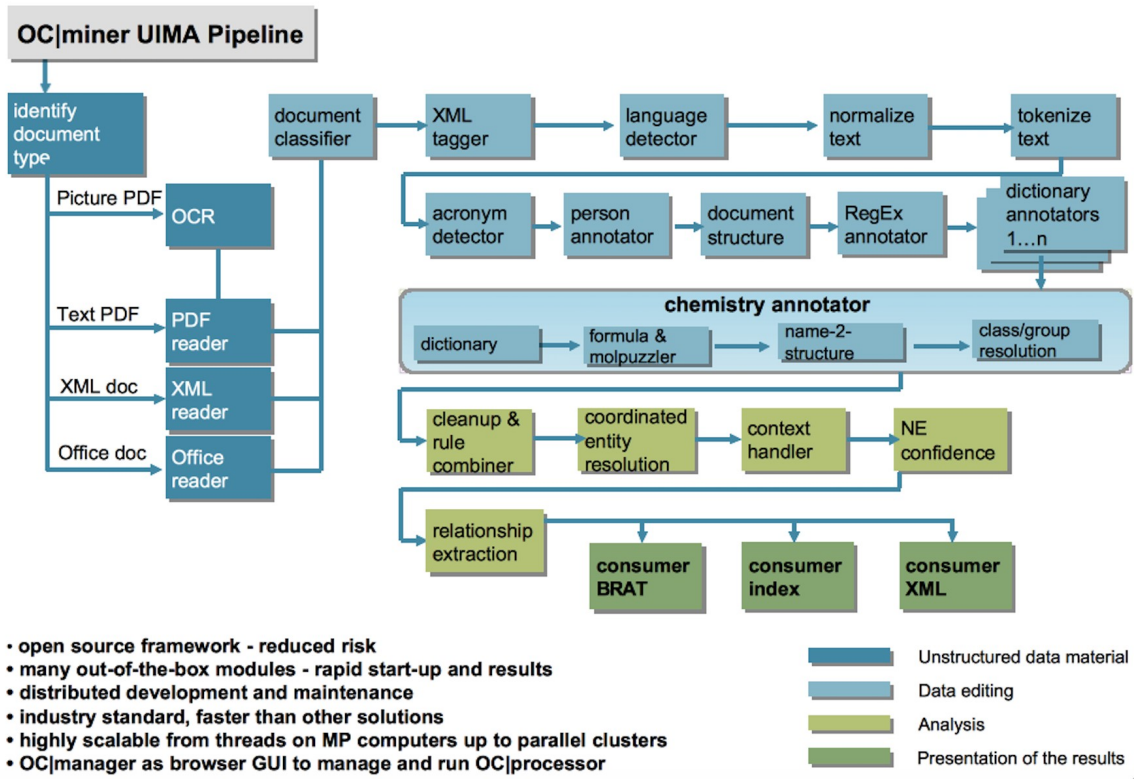
After that, **post-dictionary modules** take care of annotation clean-up, e.g. overlapping annotations, confidence adjustment and combination of simple annotations to complex entities.

Dedicated **knowledge extraction modules** can build on top of this annotation base and detect

relationships between annotated concepts or extract complex knowledge configurations in the form of semantic frames.

Finally, **consumer modules** deliver annotated documents and/or extracted data in various formats (XML/XHTML, CSV, Brat).

Fig. 1: OC|miner processing pipeline



3 ANNOTATION OF NAMED ENTITIES

3.1 Dictionary-based annotation of domain-specific concepts

Starting from domain-specific ontologies in form of OBO taxonomies, dictionaries are built for annotation with OntoChem's proprietary dictionary annotation system. Dictionaries are generated out of the terms within the respective domain ontologies and additionally expanded by generating desired term variants (e.g. automatic plural forms, case sensitivity) and desired term normalization procedures, e.g. AE/BE English, apostroph "s", hyphen/space conversion etc.

3.1.1 Black, white and grey lists (BWG lists)

The dictionary annotation is guided by a number of black and white lists:

- **<DOMAIN>_bl_pre.txt**: blacklisted terms (terms from ontology plus term variants are blacklisted, i.e. CAR is a protein, but too ambiguous and therefore if CAR is blacklisted here automatically all term variants such as CARS are also blacklisted). Thus, terms in this list will not enter the annotation dictionary.

- **<DOMAIN>_bl_idx.txt**: blacklisted terms (only specific terms are blacklisted, e.g. if CAR is blacklisted here, CARS, being a term variant of CAR, will be annotated). Thus, terms in this list will enter the annotation dictionary but shall not be annotated.

- **<DOMAIN>_cond.txt**: white or black listing of specific terms depending on certain environmental conditions. This list allows very specific black/white listing for fine tuning the annotation. Specific terms can be modified by specific contextual expressions, e.g. "gold" is blacklisted if followed by "standard", whereas "standard" is a valid contextual surrounding for other chemicals such as "trimethylsilane". Example file from **chemistry_cond.txt** below:

```
# Conditional blacklist / whitelist
# - one trigger (pattern) per line
# LINE ::= PATTERN TAB ('bl'|'wl') TAB CONDITION ( ';' CONDITION )*
# CONDITION ::= ( 'pre' || 'suf' ) ':' TEXT
# TEXT      ::= characters except newline or ';'
# lines starting with '#' or empty lines are ignored
gold  bl suf:standard;suf:set
```

- **<DOMAIN>_bl_cntxt_cond.txt**: context-sensitive blacklisting of all terms from a domain.

- **<DOMAIN>_wl_cntxt_cond.txt**: context-sensitive whitelisting of all terms from a domain.

- **<DOMAIN>_wl_pre.txt**: terms which would not be annotated using default annotation configuration parameters, but should be annotated. For example, when units shall be annotated it is necessary to explicitly whitelist single character units such as "%" or "g".

- **<DOMAIN>_gray.txt**: terms listed here get a starting basic confidence score of 0. If other terms from the same knowledge domain appear within a certain textual distance the confidence score of these annotations is increased. Finally, this leads to a valid annotation with a confidence score of >0, otherwise the respective "gray" terms will be discarded and not annotated.

- **<DOMAIN>_bl_regexp.txt**: blacklist of terms matching regular expressions, e.g. [0-9][A-Za-z]{3} to disallow PDB entries as chemical entity synonyms

3.1.2 Case sensitivity: smart case treatment

This module automatically recognizes terms which should be treated in a case sensitive annotation

mode, such as for example abbreviations or acronyms, resulting in case sensitive term variants, e.g. plural forms if desired such as "ELISAs" for "ELISA".

In addition to this automated smart case treatment, synonym terms can be marked within the ontology explicitly as case sensitive or stored in a **<DOMAIN>_case.txt** list to only allow exact case sensitive matches for annotations.

Further annotation settings which can be **customized**:

- The minimal length of an annotation can be determined. This allows for example to prevent annotations of single characters.
- Handling of overlapping annotations: longest named entity wins (default).

3.1.3 Hidden concepts

Concepts may carry the attribute "hidden". This attribute has the effect that all synonyms of a hidden concept will be annotated but not shown nor extracted unless they are part of an extracted relation/frame.

3.2 Rule-based named entity recognition

The dictionary-based annotation procedure can be complemented by a module called RegexpAnnotator identifying named entities on the basis of rules in form of regular expressions. While such an approach based on regular expressions is very flexible, there are disadvantages: it possibly over-generates named entities leading to false positives. Furthermore, mapping of synonyms to newly created concepts is theoretically possible, but would result in the same amount of work compared to an ontology update by inserting missing terms, which is not the purpose of this tool. The advantage is to recognize in a simple and efficient way so far unknown named entities following rules. This approach is very valuable, for example in the area of natural products, for recognizing new compounds isolated and published for the first time. Here, a dictionary-based approach would fail. The regular expression tool is working well when run on a set of domain relevant documents and combined with relation extraction. The contextual surroundings of the relation extraction procedure act as a confidence enhancer, so hardly any false named entities due to over-generation are extracted as participants of relations, finally resulting in improved recall and precision in relation extraction.

Example rule:

```
(?<![\p{L}\d])({{chemClass:150000001875}}(?i)(?:chemical\s)?compounds?)(?![\p{L}\d])
```

3.3 Machine-learning based named entity recognition

Named entity recognition based on machine learning may complement dictionary and rule based modules. Currently, we have a trained model for the recognition of chemical terms.

Note that machine-learning based NE recognition may do a good job in recognizing an expression in a

text as referring to a entity belonging to a given domain. However, such a technique is not capable of determining the exact concept (i.e. for a chemical entity its chemical structure or its ontological class) it refers to.

4 SPECIALIZED RECOGNITION OF CHEMICAL ENTITIES

OCMiner is especially suited for the recognition of entities in chemistry and related domains. Chemical entity recognition performed by OCMiner basically consists of dictionary-based annotation plus further chemistry-specific modules which refine the annotation.

- Dictionary annotation
- Name-to-structure conversion
- Chemical formula recognition
- Chemical image recognition (Optical structure recognition, OSR)
- Distinction of chemical compounds vs. chemical compund classes vs. chemical substituent group
- Deconvolution of labels for compounds and/or patent examples

Note that O.I.S. chemical entity annotation works in a semantic way: i.e. a textual expression annotated as a chemical entity is always mapped either to an ontological concept (e.g. chemical classes like "pyridine derivatives") or to a specific compound to which a chemical structure can be assigned. In other words, if an expression is annotated as a chemical entity, then we know not only that it is very likely to be a chemical name, but also which chemical concept is referred to by the expression.

4.1 Dictionary annotation

The chemistry dictionary is build on the basis of the chemistry ontology. Term variant settings are generated (AE/BE English, collapse spaces...). The dictionary annotation is guided by the following black and white lists (see also Section 3.1.1 above):

- **chemistry_bl_pre.txt**: blacklisted terms (dictionary: terms from ontology plus term variants are blacklisted, i.e. crack is a synonym for cocaine, but too ambiguous and therefore blacklisting of crack also discards all term variants)
- **chemistry_bl_idx.txt**: blacklisted terms (index: only specific terms are blacklisted, e.g. if leads are blacklisted here only this term variant of lead is blacklisted. Lead is still annotated in this case.)
- **chemistry_cond.txt**: specific terms white/black listing depending on certain environmental conditions. This list allows very specific black/whitelisting for fine tuning. Specific terms can

be modified by specific contextual expressions, e.g. “gold” is blacklisted if followed by “standard”, whereas “standard” is a valid contextual surrounding for other chemicals such as “trimethylsilane”

- **chemistry_wl_pre.txt**: terms which would not be annotated using default annotation configuration parameters, but should be, e.g. very short but common abbreviations

- **chemistry_bl_regexp.txt**: blacklist of terms matching regular expressions, e.g. [0-9][A-Za-z]{3} to disallow PDB entry IDs to be annotated.

- **chemistry_bl_cntxt_cond.txt**: context-sensitive blacklisting of all terms from chemistry domain

- **chemistry_gray.txt**: terms listed here get a basic confidence score of 0. If other terms from this domain appear within a certain distance the confidence score of these annotations is increased. This leads to a valid annotation, otherwise terms will be discarded.

Blacklisted terms are not annotated unless they figure in a white list. The minimal length of an OC chemistry annotation is 2 characters. Special preferences are needed when single characters such as "B" as the element symbol for boron should also be annotated.

4.2 Name-to-structure conversion

A specialized module (**ChemPotIupac**) identifies potential regular chemical names (IUPAC names) and submits identified character sequences from the text to name-to-structure conversion (N2S) engines. Currently used N2S engines include

- OPSIN (modified from the original open access form by OIS) and

- ChemAxon’s N2S,

and others can be applied as well if needed.

4.3 Chemical formula recognition

The FormulaDetector is working on the basis of regular expressions recognizing potential chemical formulas (chemPotFormula), such as semi-structural chemical formulas like "CH3CH2OH" or alloys like "FeAlSiTiRu" or "YBa2Cu3O7-δ". The recognized potential formulas are subsequently passed on to specialized chemoinformatics tools to generate the structure.

For using the FormulaDetector, a valid ChemAxon license is needed.

4.4 Optical structure recognition

Documents with images containing drawings of chemical structures can be processed by an Optical Structure Recognition (OSR) module. Images are sent to OSR programs which interpret them and convert them into a suitable formal representation (e.g. SDF or smiles).

In principle, any OSR engine can be applied for processing. Currently, the OSR program OSRA (https://sourceforge.net/projects/osra) is used.

4.5 Class and group recognition

This module uses custom derived chemistry rules for distinguishing chemical classes and substituent groups from individual chemical compounds in a context-dependent manner. The annotated text, information about the chemical concept it refers to, and the surrounding context is taken into account. For example, "imidazole derivatives", "derivatives of imidazole", "substituted imidazoles" or "imidazolyl" are not mentions of the compound imidazole but of compounds containing an imidazole structural moiety.

- Entities recognized as chemical compounds are annotated as NE of domain "chemCompound"
- Entities recognized as chemical compound classes are annotated as NE of domain "chemClass"
- Entities recognized as chemical substituent groups are annotated as NEs of domain "chemGroup"

The knowledge of the type of chemical terms is used for correcting annotation errors in a post-processing step. For example, accumulations of various consecutive annotations can be combined or neglected, depending on the involved chemical term types.

The chemical type is first determined by the **CGC detector** (Compound-Group-Class detector) considering the surrounding text for each chemical entity (CE). The chemical type detection has been further developed by using the recognition of enumerations (combined NE annotator2 (enumerations)). Enumerations are very common, especially in patents, and are combined to one CE as an intermediate step (**neEnumerationRules.txt**, see 4.3.1). According to the surrounding text of this combined CE the chemical type is changed (**context_handling_CGC.csv**).

Excerpt from example file context_handling_CGC.csv:

```
bond   suf    changeNeType:chemGroup   chemCompound,chemClass,chemFormula
```

This rule will result in a change of all entities from the domain "chemCompound", "chemClass" and "chemFormula" to the chemType "chemGroup" when followed by the suffix term "bond". Finally, the combined CEs are decomposed again (combined NE filter) considering the chemical type changes.

4.6 Deconvolution of labels for chemical compounds

Scientific publications and patents in the chemistry domain make extensive use of document-specific identifiers (aka labels) standing for chemical compounds. In journal articles, these identifiers are usually typeset as bold face numbers, e.g. "compound **1**" or "**4c**". In patent texts, numbering of compounds can have much more diverse shapes. For composition-of-matter patents, relevant compounds are given as examples and are referred to by expressions like "EXAMPLE I-6".

A dedicated module (**DeconvolutionAnnotator**) resolves these references within a document's text and annotates occurrences of resolved identifiers as their corresponding chemical named entity. Two strategies are applied:

- Resolution of explicit bold face numbers (fast and precise) or example patterns,
- Resolution of numbers behind chemical NEs, e.g. in parentheses (slow and not always precise).

Occurrences of identifier candidates are validated against a series of constraints, some of which can be configured by specific black or white lists:

- blacklist of identifiers: This (case-sensitive) blacklist consists of chemical elements written as formula (e.g. "H" or "He"), which are not allowed as identifers for chemical compounds.
- deconvolution_blacklistRegEx.txt: This blacklist consists of regular expressions which are not allowed to match the text of potential identifiers.
- deconvolution_whitelistBefore.txt: This (case-insensitive) whitelist contains expressions that may precede a potential identifier occurrence (e.g. 'reactant' or 'substance'). In addition, a chemical entity (only compounds, not groups or solvents) before a potential identifier counts as a whitelist item.
- deconvolution_blacklistBefore.txt: This (case-sensitive) blacklist contains expressions that are not allowed to precede a potential identifier occurrence (e.g. 'Section').
- deconvolution_blacklistAfter.txt: This (case-sensitive) blacklist contains expressions that are not allowed to follow a potential identifier occurrence (e.g. 'min','+').
- Deconvolution to chemical groups (e.g. 'methyl') or to solvents (e.g. 'DMSO') is not allowed.

In addition, there is a module (**TableDeconvolution**) which resolves identifiers/labels for chemical compounds defined in tables which either by reference to a chemical name or a chemical structure image. This module is a specialized form of TableInterpreter and can be configured via a properties file (tableDeconvolution.properties).

Furthermore, there is a module (**TableMarkushExtractor**) which composes simple Markush structures in tables which contain a scaffold as image and chemical substitution groups to be attached at defined substitution points (e.g. 'R' or 'Ar' or 'X') in that scaffold. If the table also contains labels/identifiers for the structures to be composed, then all valid occurrences of these labels within the document will be resolved to their corresponding chemical compounds.

5 SCORING OF NAMED ENTITIES

5.1 Confidence scoring of named entities

To each annotated mention of a named entity a **confidence value** is assigned. This value is intended to reflect how certain the extraction machine is that the annotation is correct. Confidence values range from 0 (lowest confidence) up to 1 (highest confidence). The basic confidence scoring (set by dictionary annotators) basically depends on the length of an annotation – a longer recognized term will get a higher confidence. Shorter expressions are more likely to be ambiguous and it is more probable that a wrong concept is annotated.

There is a dedicated OC|miner module (**ConfidenceAdjustment**) that makes specific adjustments to individual confidence values. The scores and factors can be **customized**.

- "list push" - if the term is found in a list with minimally 2 terms its confidence will be increased by 0.10.
- "synonym push" - if the term is found multiple times in the document – even using different synonyms of the same concept – all terms will get the confidence of the term with the maximum confidence.
- "taxonomy push" - if parent / grandparent or descendant terms are found for a given term in the document, the confidence of the term will be increased by 0.10. A parent or descendant term will have the same influence; grandparents have half of the parent influence. The number of those "influencers" is considered using a logarithmic function. The maximal confidence threshold is 0.90.

5.2 Relevance scoring of named entities

The relevance score of named entities reflects the relevance of the annotated concept, that means how relevant this named entity is regarding the focus of the article/document. The relevance score is an integer greater than 0 and increasing with increased relevance, e.g. up to 280. The process of relevance score calculation depends on the domain the named entity belongs to as well as on project-specific settings.

A concept-based relevance score for named entities has been developed and implemented. For each concept mentioned in a document, a relevance score within the document is calculated. All OC entities, including the chemistry domain, are mapped to some concept (i.e. an OCID). For entities without concept OCID, we can only map the same terms/expressions (i.e. the same text string and annotation type) onto equal concepts (for which a temporary OCID is generated within a document).

5.2.1 Basic relevance score calculation

The basic relevance score calculation of chemical concepts depends on:

- absolute number of occurrence within a document #DIRECT_OCCURRENCE(OCID)
- occurrence within boost areas (title/abstract/heading/claim section):
  - WITHIN_TITLE_PUSH + 75
  - WITHIN_ABSTRACT_PUSH + 37.5
  - SECTIONS_PUSH Claims + 20
- the relevance score is **pushed** according to the term length:
  - SCORE_PUSH_BY_TERM_LENGTH 20 characters + 20
  - SCORE_PUSH_BY_TERM_LENGTH 30 characters + 30
- the relevance score of a chemical concept is scored down by multiplying with a factor if it is only occurring once but not in title and abstract (#SINGLE_OCCURRENCE_FACTOR_VALUE = 0.25)
- if a named entity occurs within brackets it is scored down by multiplying with a factor (WITHIN_BRACKETS_FACTOR = 0.75)
- scores of children: child scores are summed up and multiplied by a factor (CHILD_SUM_FACTOR = 0.50), and subsequently added to the relevance score

5.2.2 Parent / grandparent push

The basic relevance score is subsequently modified by the scores of parents / grandparents. The scores of parents / grandparents are summed up and multiplied by

PUSH_BY_PARENT_FACTOR = 0.75

or

PUSH_BY_GRANDPARENT_FACTOR = 0.375

respectively, and added subsequently to the relevance score.

5.2.3 Occurrence class

The basic relevance score plus parent/grand-parent push is subsequently influenced by the occurrence class of domain concepts or terms. This is a factor resulting from the frequency of a

concept (or a term if the concept to which it is mapped is unknown) within a large document collection. It is obtained as follows: The number of mentions (i.e. occurrences) of all concepts in a large document collection are counted, and then clustered into "occurrence classes", a numerical value between 0 and 15. Very frequent named entity terms such as "ethanol" have an occurrence class of 0, while rarely mentioned entities have an occurrence class of 15. The influence of the occurrence class can be switched on or off.

The relevance score of a named entity is pushed if it is rarely occurring. For this purpose, the occurrence classes 0-15 are used, while 0 contains the most frequently and 15 (=minOccClass) the least frequently mentioned concepts. The relevance score is adjusted according to the following formula:

factor = (occClass + 1) / (minOccClass + 2)

5.2.4 List-based relevance score settings

- e.g. for **chemistry**: solvents and standards have a fixed relevance score of 7
- upscore list: the relevance is slightly pushed by +10.
- downscore list: fixed relevance score 5

6 CONTEXT-SENSITIVE REFINEMENT OF ANNOTATIONS

A number of post-dictionary modules are dedicated to the refinement and clean-up of annotations. As a common feature, they take the context of annotations into account. The distinguishing feature to the domain-specific dictionary annotator (see above), which to some extent also may take the context into account, is that at this post-dictionary step we may also take annotations of other domains or made by other, possibly external, sources into account.

These refinements also serve as preparatory steps to relation and frame extraction because some modules create complex meaning from specific annotation sequences.

6.1 Coordinated entity annotator: non-continuous entities

This module recognizes enumerations of expressions like "vitamin A, B and C" as a coordinated entity and annotates "vitamin A" as such and "B" as "vitamin B" and "C" as "vitamin C". The reverse coordinated list is also recognized, e.g. “A, B and C vitamin” will be annotated as three named entities: "A" pointing to "vitamin A", "B" to "vitamin B" and "vitamin C" as such. Note that this module only works with dictionary-lookup and, as a consequence, does not work with contextually enhanced named entities such as "ethyl and methyl groups" or named entities resulting from the Rule-based annotation approach (see Section 3.1.2 above).

6.2 Context handler: adjustment of annotations

Domain-specific adjustments can be made for mentions of named entities that are either preceded ("prefixes") or followed ("suffixes") by specific expressions or other named entities in the textual context. The module works on the basis of existing annotations and their contextual surroundings and performs specific actions for each defined case. For example, the confidence value of a NE is scored down when found in the context of a term which strongly modifies the meaning of the named entity. For example, if a protein term is followed by "antibody", it shall not be annotated with the concept of the respective protein. Note that the context handler is able to take annotated named entities from all domains into account, unlike the conditional black and white list which are part of domain-specific dictionary annotators (see above).

Resource: ../data/resources/annotation/context_handling.txt

The resource file has the following format:

```
LINE     ::= CONTEXT '\t' POSITION '\t' ACTION '\t' DOMAINS
POSITION ::= ( 'pre' | 'suf' )
ACTION   ::= ( 'del' | 'wl' | 'conf' | 'changeNeType:'[A-Za-z_-]++ )
      'del' - delete NE;
      'wl' - set confidence to minimum threshold;
      'conf' -  set confidence to fixed value;
      'changeNeType:NE_TYPE - set NeType to given value
DOMAINS  ::= DOMAIN ( ',' DOMAIN )*
CONTEXT  ::= TEXT | PATTERN_TOKEN
TEXT     ::= [^[] ( CHAR )*
PATTERN_TOKEN ::= a PhraseToken definition which is parsed by
PhraseTokenParser, e.g. '[N/chem]'
```

Examples:

```
antibody          suf   del   proteins
Institute of      pre   del   proteins,diseases,substances,physiology
```

6.3 NERuleCombiner: Combination of named entities to complex semantic units

This mechanism combines named entities of certain (sub-)domains to complex annotation units. Several options can be executed such as

1. Combination to complex annotation units
2. Removal of annotations (e.g. removal of compound mixtures, removal of compounds being mixture parts)
3. change of properties of annotations (e.g. change of OCID, change of chemType)
4. extraction as relations in combination with the Relation matcher and relation ontology (e.g. isInstanceOf relations (see 4.3.2), properties such as „green compounds")

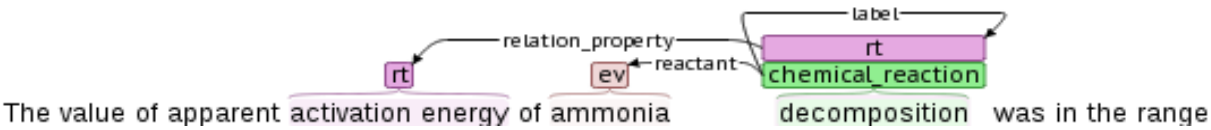
Named entities can be combined with other named entities, subtrees or text using regular expressions to combined named entities. This approach leads to

- simplification of linguistic syntax structures, e.g. enumeration RuleCombiner (see 4.3.1)
- enhancing named entities, e.g. [volatile]$_{text}$ + [compounds]$_{chem}$ to the combined NE [volatile compound]$_{chem}$
- change meaning by generating a combined NE, e.g. "ammonia decomposition", as described below.

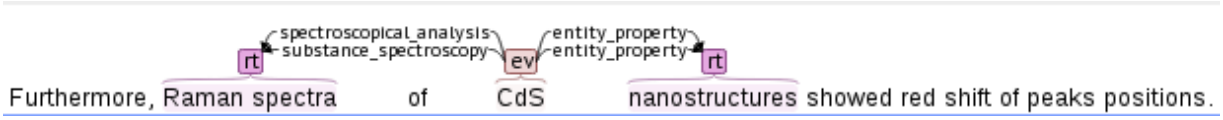
Example: In the sentence fragment "...apparent activation energy of ammonia decomposition ...", the pattern [$_{chem}$ ammonia] + [$_{reactions}$ decomposition] is combined to the complex unit [$_{reactions}$ ammonia decomposition].

This mechanism hides terms like "ammonia" in the above example for further relation extraction and prevents the erroneous recognition of a relation between "activation energy" and "ammonia" within the context of the example.

In a post-processing step, the complex unit [$_{reactions}$ ammonia decomposition] may be decomposed again into its constituents. Thus, we are able to correctly extract "activation energy" as a property of the reaction ("decomposition"):



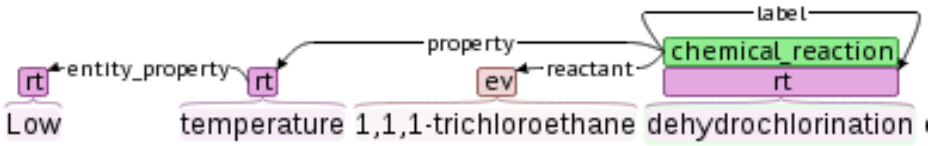
In a chemistry-centered point of view, the direction of combined named entities can be inverted in certain configurations. E.g. although the expression "CdS nanostructures" semantically denotes a specific kind of "nanostructures", the combined named entity can be inverted while decomposing. As a result, "CdS" is made available for further relation extractions. Hence, relations would be extracted in the following form:



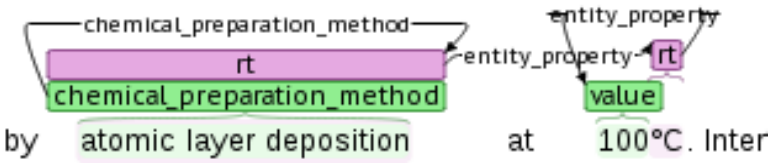
Combined named entities can also be extracted as relations between their constituents, or they may become part of a more complex relation or frame.

The mechanism permits to extract nested properties (a) and also specific values of properties (b), like in the following examples:

(a)



(b)



6.3.1 Enumerations of named entities

The RuleCombiner module is also used to automatically recognize enumerations of named entities, which is a quite common language pattern used in scientific publications. For the recognition of enumerations, a number of consecutive mentions of named entities, e.g., chemical entities, as well as additional regular expressions indicating trigger phrases such as "and", "other" or "and also other" must be present.

Example 1: "*...compounds, including alkaloids (echitamine, echitamidine, voacangine, akuammidine, N-formylechitamidine, Nα-formyl-12-methoxyechitamidine)...*"

This enumeration, highlighted in grey, will be captured by the following rule:

```
[REL|a=flatten] [N/chem|b|cl=enum] [T/chemCompound|i|cl=enum:r:[,;\]\\
s([^,;\]+)[,;\]\\s*+] [N/chem|i|cl=enum]
```

To a certain extent, the mechanism allows also to recognize unknown entities within an enumeration of already annotated entities of a homogenous domain:

Example 2: "*...[compounds]*$_{chem}$*, including [alkaloids]*$_{chem}$ *([echitamine]*$_{chem}$*, [echitamidine]*$_{chem}$*, [voacangine]*$_{chem}$ *, akuammidine, [N-formylechitamidine]*$_{chem}$*, [Nα-formyl-12-methoxyechitamidine]*$_{chem}$*)...*"

In example 2 "akuammidine" is not recognized as chemical entity by the annotation procedure. Employing the Enumeration RuleCombiner with the above shown rule "akuammidine" will be included in the combined NE and afterwards the property (chemType) will be changed to "chem".

6.3.2 Named entities referring to classes and instances

Furthermore, authors of scientific publications often create classes of concepts which are defined by larger enumerations of named entities. For instance, the linguistic expression given in the example above is defining and referring to a class of compounds (alkaloids) where the specific compounds mentioned in the enumeration are instances of (echitamine, echitamidine, voacangine, ...). We have thus a relation IS_INSTANCE_OF between the specific instances, e.g., echitamine, and the defined class, e.g. alkaloids. Further trigger phrases used by the RuleCombiner for recognizing such relations besides "including" are "such as" and "namely", as well as enumerations of named entities within parentheses after a class term.

The recognition of enumerations together with IS_INSTANCE_OF relations enables the system to syntactically simplify the text which is of great value for subsequent extraction steps which may consider these complex entity configurations as atomic units referring to a single (though in this case complex) named entity.

Thus, combined named entities forming a complex annotation unit are passed on as a single unit to relation extraction and are subsequently disentangled resulting in numerous relations.

For illustration, the figure below shows a typical sentence exhibiting these features, with annotations of named entities and relations between them.

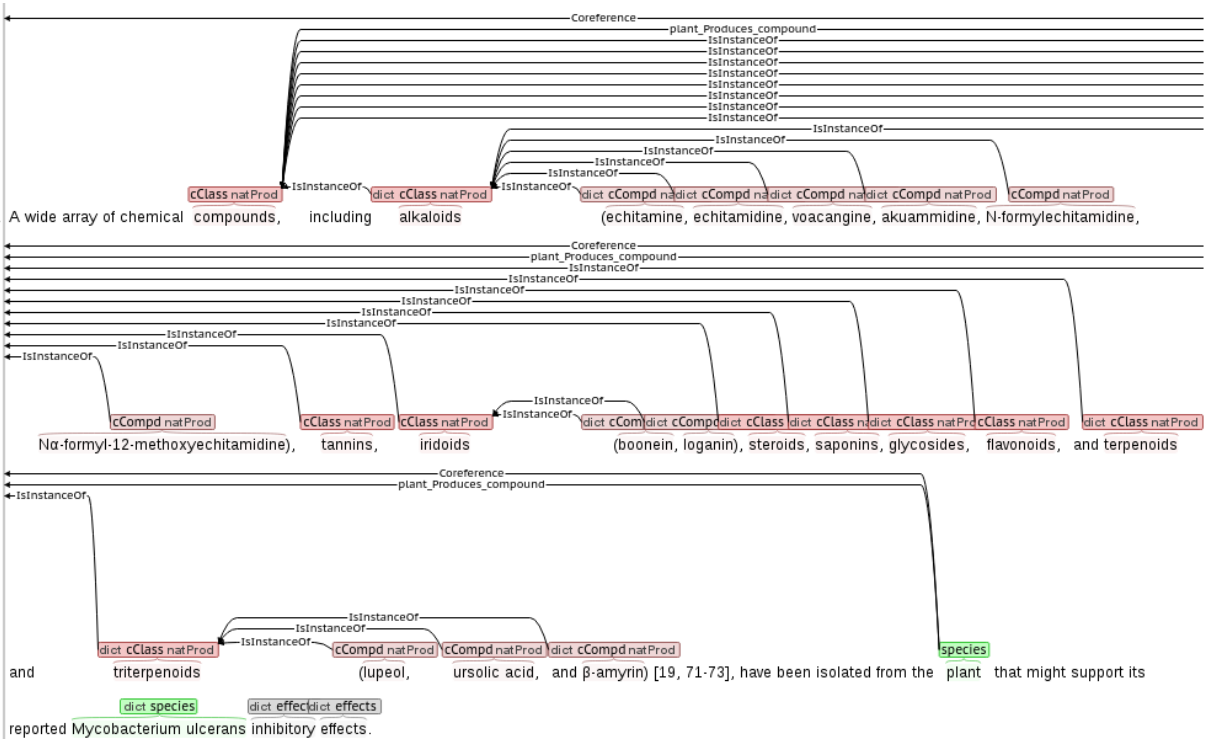


Fig. 2: A typical sentence expressing PLANT_PRODUCES_COMPOUND relations containing enumerations and IS_INSTANCE_OF relations.

For extracting the PLANT_PRODUCES_COMPOUND relations expressed in this sentence, it would be sufficient to recognize the relation between "compounds" and "the plant", given a correct resolution of what is meant by "compounds" and "the plant", respectively.

Resource: ../data/resources/annotation/neCombineRules.txt

The resource file consists of rule definitions in the following format:

```
RULE        ::= ( REL_TOKEN )? TOKEN TOKEN++
REL_TOKEN   ::= '[REL' ( '|O='RELATION_OCID )?
                       ( '|a='ACTION )?
                       ( '|s='SETTING )?
                '|]'
```

- 'O': if set it defines which relation is created by combining token
- 'a': action to be performed (default action is 'create')
- 's': rule-specific setting

```
TOKEN       ::= NE_TOKEN | TEXT_TOKEN
```

```
NE_TOKEN    ::= '[N' ( '/' DOMAIN ( '/' SUBDOMAIN )? )? ( '|O='ROOT_OCID )?
                    ( '|'COMB_TYPE )? ROLE? ( ':' TEXT )? ']'
TEXT_TOKEN ::= ONLY_TEXT_TOKEN | INCLUDE_TEXT_TOKEN
ONLY_TEXT_TOKEN ::= '[T:' TEXT ']'
      - if TEXT starts with 'r:' it is read as a regular expression;
      - phrases (multiple space separated words with no 'r:') will be
transformed to regular expressions with \\s++ as space
INCLUDE_TEXT_TOKEN::= '[T:/' DOMAIN ( '/' SUBDOMAIN )? '|' INCLUDE ( '|O='
                         OCID )?     ROLE? ':' TEXT ']'
   • text as in ONLY_TEXT_TOKEN
   • regular expression may contain a group which is used as match
     boundaries for creating include NE or defining enlarged span of
     'base' NE
COMB_TYPE   ::= BASE | INCLUDE
BASE        ::= 'b'
   • NE is base NE
INCLUDE     ::= 'i'
   • NE is include NE
ROLE        ::= '|cl=' ROLENAME
   • - semantic role (cl = class); e.g. gender, target, ...
TEXT        ::= ( [^|]] | \. )+
   • '\' as escape character
ACTION      ::= 'del' | 'atomize' | 'flatten' | 'changeProp' | 'custom'
   • action type 'delete': combined NE is not added to index but deleted
     (together with property NEs)
   • action type 'atomize': combined NE is atomized (property NEs are
     deleted from index and stripped off from combinedNE)
   • action type 'flatten': combined NE is flattened
   • action type 'changeProp': properties of combined NE will be changed
     according to rule-specific settings
   • action type 'custom': perform custom action on combined NE; specified
     via class implementing CombinedNEHandler
SETTING     ::= PROP(=VALUE)?(,PROP(=VALUE)?)*
   • rule-specific settings
   • property  'matchNeTxt': NE text must match TEXT given in rule
   • property-value pair: property of combined NE will be set to value if
     action type is 'changeProp'
```