OSCAR4: a flexible architecture for chemical text-mining

Authors: David M. Jessop, Sam Adams, Egon L.Willighagen, Lezan Hawizy, Peter Murray-Rust*

Unilever Centre for Molecular Science Informatics, Department of Chemistry, Lensfield Road, Cambridge CB2 1EW (* Corresponding author: pm286@cam.ac.uk)

Abstract

The Open-Source Chemistry Analysis Routines (OSCAR) software, a toolkit for the recognition of named entities and data in chemistry publications, has been developed since 2002. Recent work has resulted in the separation of the core OSCAR functionality and its release as the OSCAR4 library. This library features a clean API that permits client programmers to easily incorporate it into external applications. OSCAR4 offers a foundation upon which chemistry specific text-mining tools can be built, and its development and usage are discussed.
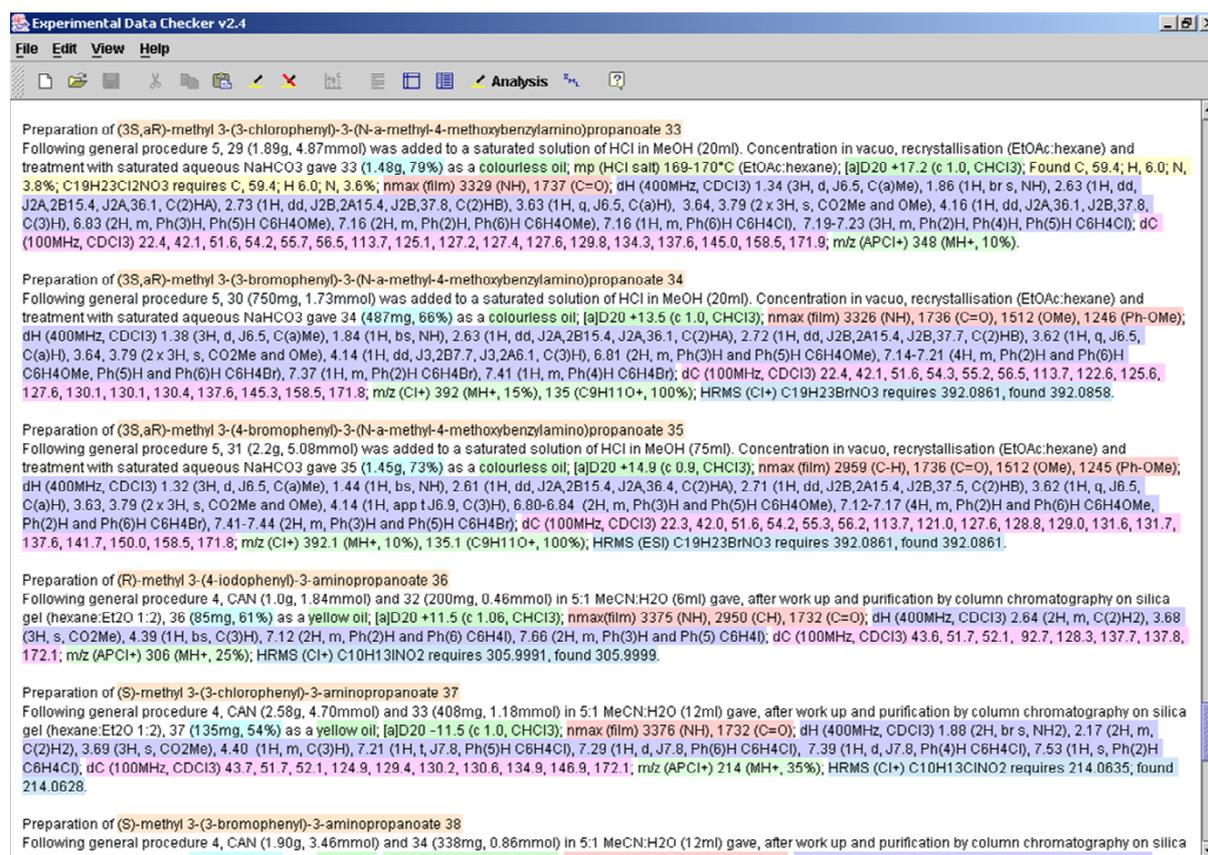
Introduction

A large amount of factual data in chemistry and neighbouring disciplines is published in the form of text and components within text rather than as structured semantic information. If we can discover and extract this information, the textual literature becomes an enormous additional chemical resource. As an example, we estimate that about 10 million chemical syntheses per year are published in the public literature (articles, patents, theses) and the conventional method is a natural language narrative (most commonly in English). It is extremely tedious and error-prone to extract information from this by hand, and for this reason many chemical abstracting services limit their scope and also frequently lag behind the current publication list.

The discipline of text-mining has now reached a state where much natural language in textual form can be analysed rapidly and with high precision and recall. Methodologies applied to the problem of chemical named entity recognition include dictionary- and rule-based methods, as well as machine learning and hybrid approaches[1][2][3][4][5][6][7][8][9][10][11]. We have been working in this area for approximately 10 years and the OSCAR4 software, together with OPSIN (the Open Parser for Systematic IUPAC Nomenclature)[12][13] and ChemicalTagger[14][15], represent the public state-of-the-art in chemical text analysis and extraction.

The OSCAR (Open-Source Chemistry Analysis Routines) software has been developed over a period of years and a number of projects. Between 2002 and 2004, sponsors including the Royal Society of Chemistry (RSC), Nature and the International Union of Crystallography (IUCr) supported a number of summer studentships. These projects were focused on the development of software with limited

capacity for the automated interpretation of chemical documents, and resulted in two main software components – the Experimental Data Checker[16][17] and OSCAR2.

The Experimental Data Checker was conceived as a tool to be used as part of the RSC's publication process. The tool is capable of recognising sections of reported experimental data within plain text input using regular expressions to match the highly-stylised and journal-mandated formats in which they are reported in the literature (as shown in Figure 1). Once this information has been identified and interpreted, the tool performs elementary checks on the characterisation data to ensure that it does not conflict with the reported structure.

Experimental Data Checker v2.4

File   Edit   View   Help

Analysis

Preparation of (3S,aR)-methyl 3-(3-chlorophenyl)-3-(N-a-methyl-4-methoxybenzylamino)propanoate 33
Following general procedure 5, 29 (1.89g, 4.87mmol) was added to a saturated solution of HCl in MeOH (20ml). Concentration in vacuo, recrystallisation (EtOAc:hexane) and treatment with saturated aqueous NaHCO3 gave 33 (1.48g, 79%) as a colourless oil; mp (HCl salt) 169-170°C (EtOAc:hexane); [a]D20 +17.2 (c 1.0, CHCl3); Found C, 59.4; H, 6.0; N, 3.8%; C19H23Cl2NO3 requires C, 59.4; H 6.0; N, 3.6%; nmax (film) 3329 (NH), 1737 (C=O); dH (400MHz, CDCl3) 1.34 (3H, d, J6.5, C(2)Me), 1.86 (1H, br s, NH), 2.63 (1H, dd, J2A,2B15.4, J2A,36.1, C(2)HA), 2.73 (1H, dd, J2B,2A15.4, J2B,37.8, C(2)HB), 3.63 (1H, q, J6.5, C(a)H), 3.64, 3.79 (2 x 3H, s, CO2Me and OMe), 4.16 (1H, dd, J2A,36.1, J2B,37.8, C(3)H), 6.83 (2H, m, Ph(3)H, Ph(5)H C6H4OMe), 7.16 (2H, m, Ph(2)H, Ph(6)H C6H4OMe), 7.16 (1H, m, Ph(6)H C6H4Cl), 7.19-7.23 (3H, m, Ph(2)H, Ph(4)H, Ph(5)H C6H4Cl); dC (100MHz, CDCl3) 22.4, 42.1, 51.6, 54.2, 55.7, 56.5, 113.7, 125.1, 127.2, 127.4, 127.6, 129.8, 134.3, 137.6, 145.0, 158.5, 171.9; m/z (APCI+) 348 (MH+, 10%).

Preparation of (3S,aR)-methyl 3-(3-bromophenyl)-3-(N-a-methyl-4-methoxybenzylamino)propanoate 34
Following general procedure 5, 30 (750mg, 1.73mmol) was added to a saturated solution of HCl in MeOH (20ml). Concentration in vacuo, recrystallisation (EtOAc:hexane) and treatment with saturated aqueous NaHCO3 gave 34 (487mg, 66%) as a colourless oil; [a]D20 +13.5 (c 1.0, CHCl3); nmax (film) 3326 (NH), 1736 (C=O), 1512 (OMe), 1246 (Ph-OMe); dH (400MHz, CDCl3) 1.38 (3H, d, J6.5, C(a)Me), 1.84 (1H, bs, NH), 2.63 (1H, dd, J2A,2B15.4, J2A,36.1, C(2)HA), 2.72 (1H, dd, J2B,2A15.4, J2B,37.7, C(2)HB), 3.62 (1H, q, J6.5, C(a)H), 3.64, 3.79 (2 x 3H, s, CO2Me and OMe), 4.14 (1H, dd, J3,2B7.7, J3,2A6.1, C(3)H), 6.81 (2H, m, Ph(3)H and Ph(5)H C6H4OMe), 7.14-7.21 (4H, m, Ph(2)H and Ph(6)H C6H4OMe, Ph(5)H and Ph(6)H C6H4Br), 7.37 (1H, m, Ph(2)H C6H4Br), 7.41 (1H, m, Ph(4)H C6H4Br); dC (100MHz, CDCl3) 22.4, 42.1, 51.6, 54.3, 55.2, 56.5, 113.7, 122.6, 125.6, 127.6, 130.1, 130.1, 130.4, 137.6, 145.3, 158.5, 171.8; m/z (CI+) 392 (MH+, 15%), 135 (C9H11O+, 100%); HRMS (CI+) C19H23BrNO3 requires 392.0861, found 392.0858.

Preparation of (3S,aR)-methyl 3-(4-bromophenyl)-3-(N-a-methyl-4-methoxybenzylamino)propanoate 35
Following general procedure 5, 31 (2.2g, 5.08mmol) was added to a saturated solution of HCl in MeOH (75ml). Concentration in vacuo, recrystallisation (EtOAc:hexane) and treatment with saturated aqueous NaHCO3 gave 35 (1.45g, 73%) as a colourless oil; [a]D20 +14.9 (c 0.9, CHCl3); nmax (film) 2959 (C-H), 1736 (C=O), 1512 (OMe), 1245 (Ph-OMe); dH (400MHz, CDCl3) 1.32 (3H, d, J6.5, C(a)Me), 1.44 (1H, bs, NH), 2.61 (1H, dd, J2A,2B15.4, J2A,36.4, C(2)HA), 2.71 (1H, dd, J2B,2A15.4, J2B,37.5, C(2)HB), 3.62 (1H, q, J6.5, C(a)H), 3.63, 3.79 (2 x 3H, s, CO2Me and OMe), 4.14 (1H, app t J6.9, C(3)H), 6.80-6.84 (2H, m, Ph(3)H and Ph(5)H C6H4OMe), Ph(2)H and Ph(6)H C6H4Br), 7.41-7.44 (2H, m, Ph(3)H and Ph(5)H C6H4Br); dC (100MHz, CDCl3) 22.3, 42.0, 51.6, 54.2, 55.3, 56.2, 113.7, 121.0, 127.6, 128.8, 129.0, 131.6, 131.7, 137.6, 141.7, 150.0, 158.5, 171.8; m/z (CI+) 392.1 (MH+, 10%), 135.1 (C9H11O+, 100%); HRMS (ESI) C19H23BrNO3 requires 392.0861, found 392.0861.

Preparation of (R)-methyl 3-(4-iodophenyl)-3-aminopropanoate 36
Following general procedure 4, CAN (1.0g, 1.84mmol) and 32 (200mg, 0.46mmol) in 5:1 MeCN:H2O (6ml) gave, after work up and purification by column chromatography on silica gel (hexane:Et2O 1:2), 36 (85mg, 61%) as a yellow oil; [a]D20 +11.5 (c 1.06, CHCl3); nmax(film) 2950 (CH), 1732 (C=O); dH (400MHz, CDCl3) 2.64 (2H, m, C(2)H2), 3.68 (3H, s, CO2Me), 4.39 (1H, bs, C(3)H), 7.12 (2H, m, Ph(2)H and Ph(6 C6H4I), 7.66 (2H, m, Ph(3)H and Ph(5) C6H4I); dC (100MHz, CDCl3) 43.6, 51.7, 52.1, 92.7, 128.3, 137.7, 137.8, 172.1; m/z (APCI+) 306 (MH+, 25%); HRMS (CI+) C10H13INO2 requires 305.9991, found 305.9999.

Preparation of (S)-methyl 3-(3-chlorophenyl)-3-aminopropanoate 37
Following general procedure 4, CAN (2.58g, 4.70mmol) and 33 (408mg, 1.18mmol) in 5:1 MeCN:H2O (12ml) gave, after work up and purification by column chromatography on silica gel (hexane:Et2O 1:2), 37 (135mg, 54%) as a yellow oil; [a]D20 -11.5 (c 1.0, CHCl3); nmax (film) 3376 (NH), 1732 (C=O); dH (400MHz, CDCl3) 1.88 (2H, br s, NH2), 2.17 (2H, m, C(2)H2), 3.69 (3H, s, CO2Me), 4.40 (1H, m, C(3)H), 7.21 (1H, t, J7.8, Ph(5)H C6H4Cl), 7.29 (1H, d, J7.8, Ph(6)H C6H4Cl), 7.39 (1H, d, J7.8, Ph(4)H C6H4Cl), 7.53 (1H, s, Ph(2)H C6H4Cl), dC (100MHz, CDCl3) 43.7, 51.7, 52.1, 124.9, 129.4, 130.2, 130.6, 134.9, 146.9, 172.1; m/z (APCI+) 214 (MH+, 35%); HRMS (CI+) C10H13ClNO2 requires 214.0635, found 214.0628.

Preparation of (S)-methyl 3-(3-bromophenyl)-3-aminopropanoate 38
Following general procedure 4, CAN (1.90g, 3.46mmol) and 34 (338mg, 0.86mmol) in 5:1 MeCN:H2O (12ml) gave, after work up and purification by column chromatography on silica

FIGURE 1: A screenshot of the Experimental Data Checker (OSCAR-Data) showing identification and markup of plain text experimental data. The initial application of OSCAR was to parse the highly stylised data used to report spectra and other analytical proofs of synthesis. This functionality is very widely-used and has been re-integrated into OSCAR4 rather than being a separate application.

The Experimental Data Checker application relied upon a core library of analysis routines, and it was this library that was the first to bear the name OSCAR. Further development of this library in the summer of 2004 resulted in OSCAR2, which used XML formatting to represent the document undergoing processing, and applied XML annotations to the document to indicate recognised sections of text. OSCAR2 implemented a naïve Bayesian system based on *n*-grams and a simple grammar in order to identify chemical names within a text. These improvements were later extended as part of the OSCAR3 project.

In 2005, the EPSRC awarded a grant ("Sciborg") to develop natural language processing (NLP) tools for chemistry and science. The chemistry component of this project focused on the development of

2

the OSCAR2 methodology and resulted in the creation of OSCAR3[18]. OSCAR3 focuses on the recognition of and, where appropriate, the resolution of connection tables for chemical named entities. OSCAR3 employs a naïve Bayesian model to identify "chemical" tokens in text and offers a choice of two methods for the identification of multi-token named entities. The first of these, the `PatternRecogniser`, uses predetermined regular-expression style heuristics while the second, the `MEMMRecogniser`[19], employs machine learning in the form of a Maximum Entropy Markov Model (MEMM). OSCAR3 uses these methods to identify four classes of named entity (Chemical, Reaction, Chemical Adjective and Enzyme) as well as dictionary lookup to identify a pre-determined set of ontology terms and a discrete finite automaton based method to identify chemical prefixes.

Preparation of N-(2,5-dichlorobenzyl)-8-hydroxy-1,6-naphthyridine-7-carboxamide

Triphosgene (0.556g, 1.87 mmol) was added over 20 mins to a solution of

8-hydroxy-1,6-naphthyridine-7-carboxylic acid (0.89g, 4.68 mmol) and diisopropylethylamine (3.26 ml, 18.7 mmol) in DMF (22 ml) at 0°C.

2,5-dichlorobenzylamine (0.142 ml, 1.05 mmol) was treated with a portion of the above solution (0.58ml, 0.07...) ...
for 16 hrs. 1H NMR
d, J=4.0 Hz), 8

id = o11; surface = 2,5-dichlorobenzylamine; type = CM; confidence = 0.9293289445146479; SMILES = [H]C1=C([H])C(=C(C([H])=C1Cl)C([H])([H])N([H])[H])Cl; InChI = InChI=1/C7H7Cl2N/c8-6-1-2-7(9)5(3-6)4-10/h1-3H,4,10H2; cmlRef = cml6;

4.0 Hz), 7.54 (1H, d, J=8.0Hz), 7.50-7.30 (2H, m), 4.64 (2H, d, J=5.0 Hz) ppm.

FAB MS calcd for C16H11N3O2Cl2 348 (MH+), found 348.
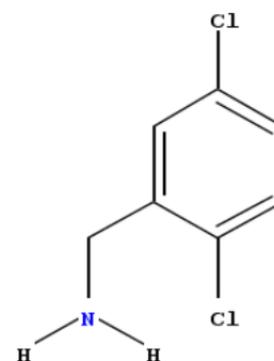
- Experimental data
- Ontology term
- Chemical (etc.) with structure
- Chemical (etc.), without structure
- Reaction
- Chemical adjective
- enzyme -ase word
- Chemical prefix

FIGURE 2: OSCAR3 markup displaying recognised chemical entities (CM). A mouse-over action on an annotated term displays the associated metadata, in this case for 2,5-dichlorobenzylamine, and displays an image representing the structure generated by the Chemistry Development Kit (CDK)[20][21][22] (right). OSCAR3 concentrated on the identification and interpretation of chemical entities in text (named entity recognition, NER). The primary purpose was to identify and extract the following types of object: chemicals (CM), ontology terms (ONT; looked-up from ChEBI[23][24][25], FIX[26] and REX[27] *etc.*), reactions (RN; as identified by linguistic constructs, *e.g.* "methylated"), chemical adjectives (CJ) mainly formed from chemical nouns), enzymes (ASE) and chemical prefixes (CPR), highlighted in different colours. These concepts are maintained in OSCAR4.

In order to convert chemical names to connection tables (Figure 2), OSCAR3 uses dictionary-based methods and, where this is not successful, OPSIN. Early versions of OSCAR directly included the OPSIN code, but this was later re-factored into a separate library.

By 2008, OSCAR was in common use in many laboratories for the identification and extraction of chemical terms (chemical named entities) in a variety of texts. Our original metrics[18] showed that the precision and recall were domain-dependent and varied considerably with the purpose and style of chemical texts. Feedback from users was informal but it was clear that they were modifying OSCAR for their particular purposes both in vocabulary and recognition methods. As a result we embarked on a major re-factoring program in order to robustify the OSCAR software and simplify the API, and this paper describes the results.

It is very difficult to get funding for software engineering projects, especially when apparently little changes on the surface. We are grateful to the following bodies for their funding and interest:

1. OMII-UK. This organisation existed to support and robustify the products of the UK eScience program. Many of these were middleware products but OSCAR was seen by the UK eScience community as an example of a widely-deployable component that could be used in a modern manner in many branches of science. The OMII-UK project carried out an initial scoping and re-factoring of the OSCAR3 source.

2. The OSCAR-ChEBI project. This was a competitive funding resource for eScience products and we worked with the European Bioinformatics Institute (EBI) to develop OSCAR as an appropriate tool for the extraction and verification of chemistry in the ChEBI ontology.

3. CheTA. This was a JISC-funded project led by our group in conjunction with the National Centre for Text Mining (NaCTeM) to evaluate the relative merits of human annotation and machine annotation of documents. Part of this project involved OSCAR running under the UIMA[28]/U-Compare[29][30] framework and required a re-factoring[31].

As a result of these projects, which probably amounted to two person-years of effort in the re-factoring, OSCAR4 has now been released in a usable form.

Limitation of OSCAR3 and design goals for OSCAR4

OSCAR3 is a powerful tool for chemical natural language processing, but early attempts to develop software using it as a library rather than as a standalone application – the ChemicalTagger[14] and PatentEye[32][33] projects – exposed weaknesses in the code in this regard. The architecture of the software was built around the principle that the software would be running as a server on the user's local machine. In order to function correctly, it required a properly configured workspace. Many key components were implemented as mutable singletons, compromising the thread-safety of the application and meaning that safe reconfiguration of a workflow required a complete shutdown and restart of the Java virtual machine (JVM). Furthermore, the implementations of the various OSCAR components required that a document be formatted in SciXML as it underwent processing. Consequently, the use of OSCAR3 by a client programmer to build secondary applications was unintuitive, and the distribution and successful use of such applications was found, as part of the Green Chain Reaction, to require an unacceptably high level of support.

Early attempts to resolve these problems[31] involved the extraction of the OSCAR3 tokeniser, `MEMMRecogniser` and `PatternRecogniser` components from the main OSCAR3 codebase and their conversion into modules suitable for use in the popular text-mining framework U-Compare. This work allowed the use of OSCAR as part of a drag-and-drop workflow, but not its direct integration into another application. Consequently, a comprehensive overhaul of the OSCAR3 code began in autumn 2010 with the aim of producing a well-engineered, simple, modularised version of OSCAR that retained the core OSCAR3 functionality and could be easily integrated into external applications. This most recent development has been designated OSCAR4 and is discussed in the remainder of this paper.

The development of OSCAR4 sought to address a number of specific issues. These are summarised below and subsequently discussed in greater detail.

1. To produce an OSCAR library with a simple API, suitable for use by client programmers who may not be familiar with the internal workings of OSCAR. Consequently, while it is desirable for users to be able to customise the behaviour of OSCAR in a number of ways, initialisation of OSCAR components must by default produce configurations that "just work" – the 'convention over configuration' paradigm.

2. In order to run, OSCAR3 required the existence of a properly configured workspace – a directory on the executing machine that contains the OSCAR chemical name dictionary, the InChI[34][35] binary file and a properties file along with subdirectories intended to contain further resource files. When OSCAR3 is first run this workspace is automatically created, and when OSCAR3 is used as a library the workspace is automatically created in the working directory. This behaviour was deemed undesirable, unnecessary and found to be a cause of difficulties in producing distributable OSCAR-dependent software. Consequently, the removal of the requirement for a workspace was considered a high priority of the OSCAR4 project.

3. Much of the OSCAR3 code required that a document undergoing processing is formatted in SciXML. Though converters are provided to transform HTML into plain text and plain text into SciXML, the requirement to perform this transformation is frustrating to the client programmer in that it prevents him from working directly with plain text or with a custom XML format which may very well be the native format of a document that he wishes to process. Consequently, the removal of this SciXML dependence was considered important.

4. In addition to its core functionality – the recognition and interpretation of chemical named entities – OSCAR3 included a wide range of secondary functions including the OSCAR3 server. This server runs on the local machine and provides an interactive demonstration of the capacity of OSCAR3 for text processing as well as a number of other utilities including the capacity to manually annotate a text from within a browser window, a servlet for the interconversion and depiction of chemical names and formats and an experimental Hearst pattern[36] based system for the extraction of chemical relations from text. The OSCAR3

codebase had the resemblance of a 'treasure trove' which made code maintenance a more complex task than necessary. The separation of a library containing the core OSCAR functionality from these secondary functions was therefore considered desirable.

5. Much of the architecture of OSCAR3 lacked clear definition. Excessive use is made of mutable singletons which, while aiding performance by eliminating the need for re-initialisation of components, allows for complex interactions in the code, making it difficult to understand, debug and re-factor. This problem was compounded by the manner in which program logic is partially controlled by a properties object backed by a serialised file. Some of the property values can be modified at runtime while others, once accessed by the objects that rely upon them, are duplicated in memory and cannot be further changed. Attempts to resolve these complex interactions can have unintended consequences since the unit test coverage in OSCAR3 is sparse. Consequently, the improvement of the architecture of the OSCAR software was considered a vital part of the OSCAR4 project.

6. It has been known for some time that the speed of OSCAR3 operation could be improved by introducing certain optimisations into the code. Using the YourKit Java profiler[37], a number of performance blackspots were identified and subsequently eliminated. This work was started after the final version of OSCAR3 (OSCAR3 alpha 5[38]) and continued as part of the OSCAR4 project.

Library as a design

OSCAR4 has been deliberately written as a Java library, rather than an application or service. Consequently, the decoupling of the core OSCAR functionality from applications that use this functionality has been achieved. The usage of the library has been simplified as much as possible with the introduction of the `Oscar` API object – a class intended to wrap the functionality of the wider library and provide default implementations of the various components. As a result, OSCAR4 can be called from external software, as shown in the examples in Figure 3.

```
a)   String text = "The quick brown ethyl acetate jumps over the lazy bromine";

     Oscar oscar = new Oscar();

     List <NamedEntity> neList = oscar.findNamedEntities(text);


b)   String text = "The quick brown ethyl acetate jumps over the lazy bromine".

     Oscar oscar = new Oscar();

     List <ResolvedNamedEntity> rneList = oscar.findAndResolveNamedEntities(text);
```

Figure 3: Java code using the OSCAR4 API to a) identify chemical named entities (CNEs) in a block of text and b) identify CNEs and resolve their connection tables where possible.

The quick brown ethyl acetate jumps over the lazy bromine

```
surface = ethyl acetate        surface = bromine
type = CM                      type = CM
startPos = 16                  startPos = 50
endPos = 29                    endPos = 57
confidence = 0.980             confidence = 0.990
```

```
type = SMILES
source = http://wwmm.ch.cam.ac.uk/dictionary/chebi/
value
  type = INCHI
  source = http://wwmm.ch.cam.ac.uk/dictionary/chebi/
  val
    type = SMILES
    source = http://wwmm.cam.ac.uk/oscar/dictionary/opsin/
    val
      type = CML
      source = http://wwmm.cam.ac.uk/oscar/dictionary/opsin/
      val
        type = INCHI
        source = http://wwmm.cam.ac.uk/oscar/dictionary/opsin/
        value = InChI=1/C4H8O2/c1-3-6-4(2)5/h3H2,1-2H3
```

```
type = CML
source = http://wwmm.cam.ac.uk/oscar/dictionary/opsin/
valu
  type = INCHI
  source = http://wwmm.cam.ac.uk/oscar/dictionary/opsin/
  valu
    type = SMILES
    source = http://wwmm.cam.ac.uk/oscar/dictionary/opsin/
    val
      type = INCHI
      source http://wwmm.ch.cam.ac.uk/dictionary/chebi/
      valu
        type = SMILES
        source http://wwmm.ch.cam.ac.uk/dictionary/chebi/
        value = [Br]
```

Figure 4: Graphic representing the structure of the OSCAR4 API output object. Named entities reference their position in the input text, the confidence in their identification and resolved structures in various formats (SMILES[39 40], InChI, CML[41] *etc.*)

In the first of the examples in Figure 3, OSCAR4 is used to detect named entities in an input string, returning a `List` of `NamedEntity` objects. In the second, it is used to both detect named entities and, where these named entities correspond to chemical names, to resolve these names to chemical structures – returning a List of `ResolvedNamedEntity` objects. The `ResolvedNamedEntity` class links a `NamedEntity` to a list of chemical structures in a number of formats – SMILES, InChI and CML – while the NamedEntity class stores such information as the surface (raw text) and type (*e.g.* compound or reaction) of the named entity and the indices that define its position within the source text. The outputs of these examples are illustrated in Figure 4.

The examples above show how OSCAR4 can be used without the need for any understanding of the underlying technology or implementations. An overview of the workflow managed by the `Oscar` API object is shown in Figure 5.
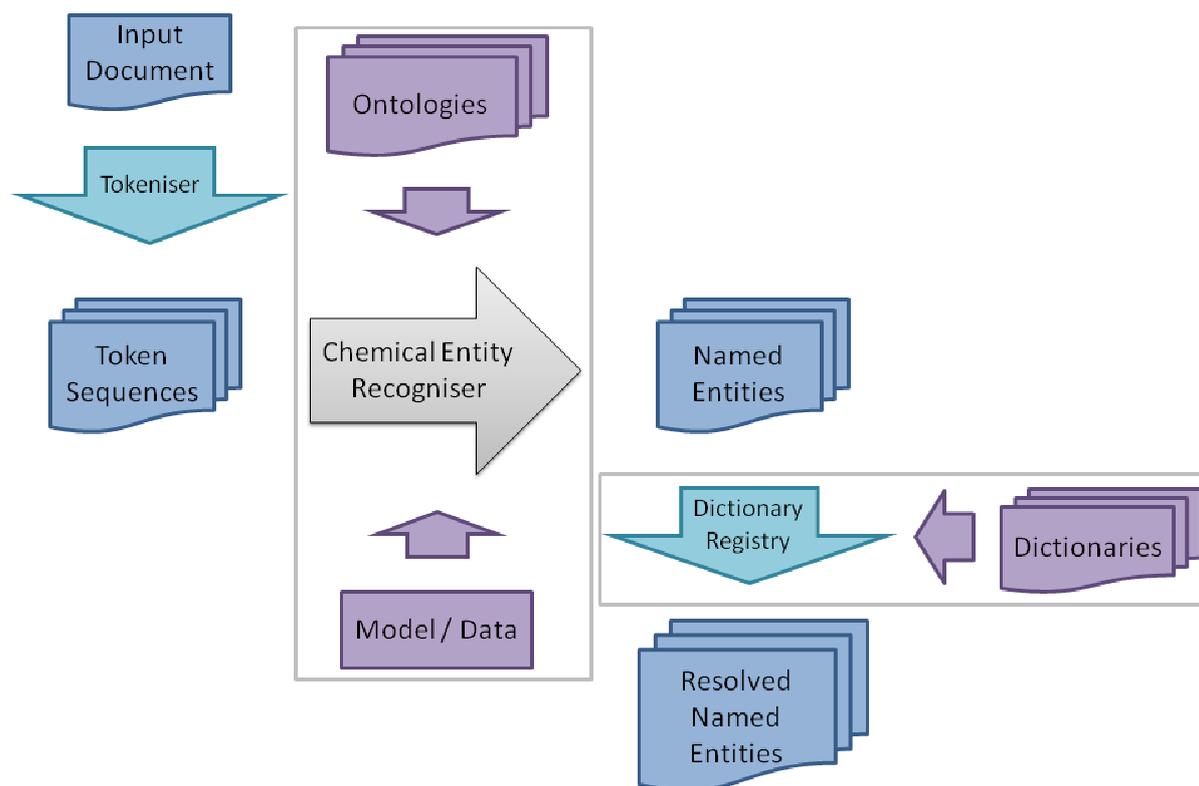
Figure 5: Workflow of the OSCAR4 API object

The input is first passed to the `Tokeniser` to produce a list of `TokenSequence` objects, each of which roughly corresponds to a paragraph of text and contains a list of `Token` objects. The `Token` represents a string of characters that mostly correspond to words but also to punctuation or other discrete units of text *e.g.* "C_2H_6O" or "42". In NLP tools, tokenisation commonly occurs at whitespace or punctuation boundaries, however due to the form of some of the domain-specific entities found in chemical texts such as "C-H" a custom `Tokeniser` is used. The `TokenSequences` are then passed to a `ChemicalEntityRecogniser` – an interface for a class capable of identifying a list of `NamedEntities`, which are subsequently passed to the `ChemNameDictRegistry` to create a list of `ResolvedNamedEntities` if required.

This workflow can be customised by the user, who can use the `set()` methods of the `Oscar` class to replace the components of the default configuration with suitable customised or custom-built alternatives. Specifically, the user can select which implementation of `ChemicalEntityRecogniser` to use or can specify which set of ontology terms are to be recognised and which model the default `ChemicalEntityRecogniser` should use, and which dictionary registry, *i.e.* set of chemical name dictionaries, to use for name to structure resolution. In addition to this, the public APIs of the individual components can be used to assume a greater degree of control over the execution of the workflow.

OSCAR4 provides three implementations of the `ChemicalEntityRecogniser`. The first, the `RegexRecogniser`, finds terms that match a given regular expression and is intended to find serial numbers corresponding to compounds *e.g.* "NSC-2648". The others, the `PatternRecogniser` and the `MEMMRecogniser`, use more complex strategies to identify chemical named entities and feature subcomponents that can be customised by the user to produce the desired behaviour.
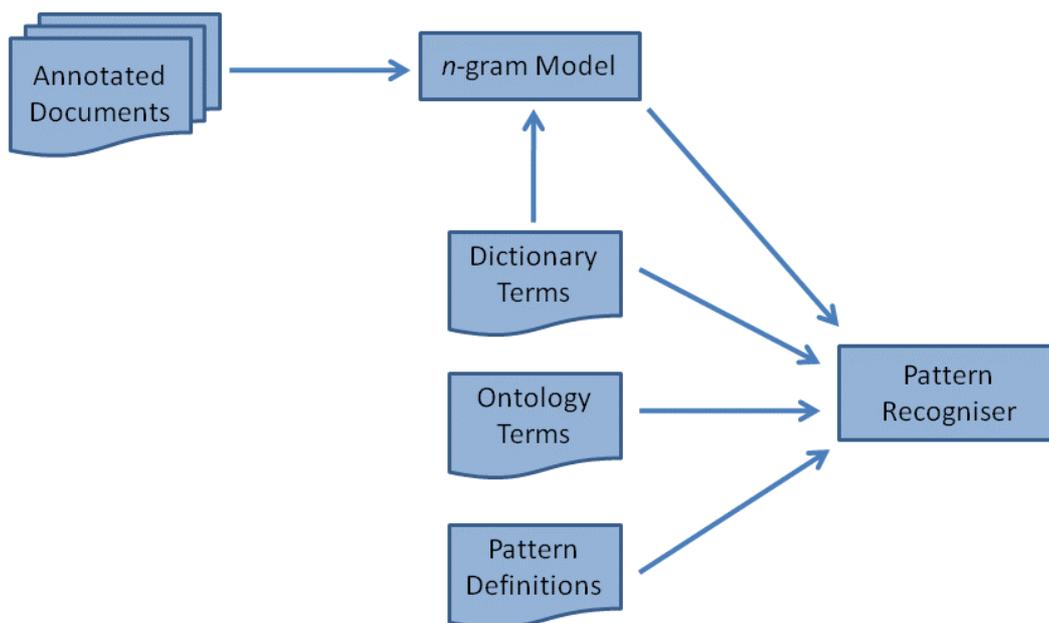
Figure 6: `PatternRecogniser` architecture

The architecture of the `PatternRecogniser` is shown in Figure 6. A list of "chemical" words is drawn from an internal dictionary composed mostly of words derived from the ChEBI database and from a corpus of manually-annotated documents, while a list of "non-chemical" words is determined by removing those words that occur in the chemical word list from a standard English dictionary. These lists are used to build an *n*-gram model which is used by a naïve Bayesian classifier to determine whether novel tokens are "chemical" or "non-chemical". Multi-token named entities, *e.g.* "ethyl acetate", that occur within the input text are then identified by regex-style matching of chemical tokens to a set of pre-specified pattern definitions such as "*yl *ate".
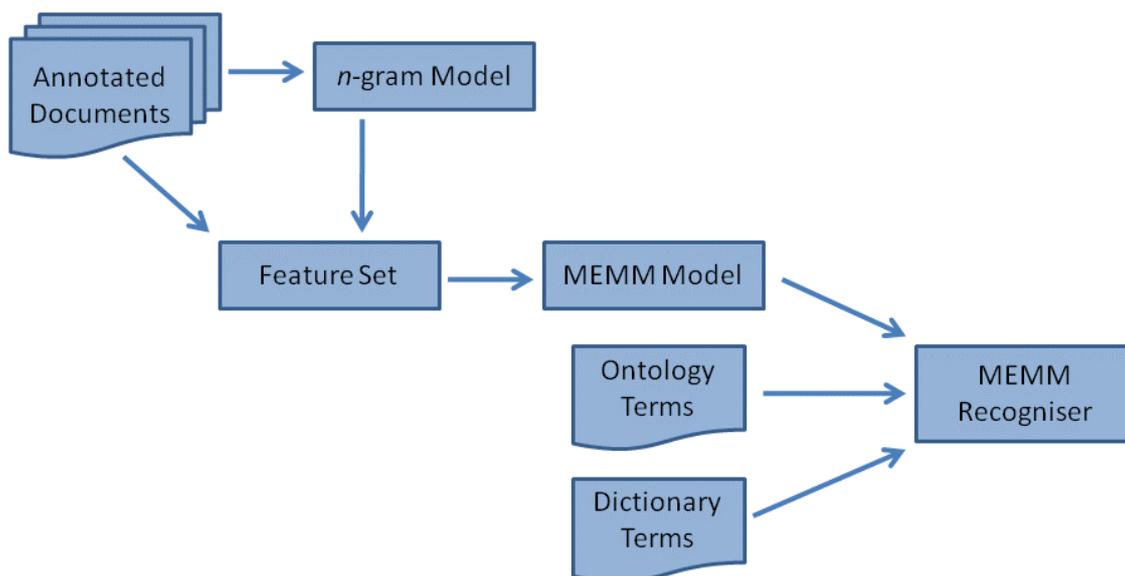
Figure 7: `MEMMRecogniser` architecture

The architecture of the `MEMMRecogniser` is shown in **Figure** 7, in which chemical named entities are identified using a Maximum Entropy Markov Model (MEMM). The feature set that is generated for each token includes features that describe the token in question, such as the *n*-grams that describe it and the probability that it is chemical as predicted by the *n*-gram model as previously, as well as contextual features that describe its neighbouring tokens. Using these features, the MEMM model assigns a chemical token as being either the first token in a named entity or a subsequent token in a named entity. Given these assignments, multi-token named entities can be constructed. Novel MEMM models can be built from a corpus of hand-annotated documents by the user, and OSCAR4 is supplied with two pre-generated models. One of these models was built from a set of papers from RSC journals[42], while the other was built from a set of abstracts retrieved from PubMed[43].

Architecture and tests

The OSCAR4 library has been separated into a number of modules with each performing a defined role in the operation of the OSCAR code, such as the tokenisation of text or the provision of chemical

name dictionaries. This allows client programmers to use as much or as little of OSCAR in their applications as required, without the need to unnecessarily pull in a large, comprehensive, single JAR. The process of creating the sub-projects had the additional advantage of highlighting the ways in which the separate components interact. During this process, the readability of the OSCAR code was improved by imposing a number of the idioms of 'clean code', and the reliability of the code was improved by the creation of appropriate unit and regression tests. At the time of writing, OSCAR4 has nearly 500 tests. As a result, the OSCAR4 code is far more robust than OSCAR3, so a developer can work both with and on the core OSCAR code with a far greater degree of confidence.

The mutable singletons that were commonplace in OSCAR3 have been largely removed. Instead, when setting up custom workflows, a user has the choice of either calling the `getDefaultInstance()` method or the default constructor as appropriate – each of which returns a preconfigured instance of the class – or using the custom constructor which uses dependency injection to supply the OSCAR components upon which the class depends. For example, the `OntologyTerms` class represents a set of ontology terms and their corresponding ontology IDs. The following two methods of obtaining an `OntologyTerms` object are available:

```
OntologyTerms.getDefaultInstance();

new OntologyTerms(ListMultimap <String, String> terms);
```

The first method returns the default OSCAR4 `OntologyTerms` object, which contains an amalgamation of the terms from the ChEBI, FIX and REX ontologies while the second supplies a multimap of ontology terms to IDs. The use of this design pattern throughout the codebase permits, but by no means requires, a user to assume a high degree of control over the functioning of OSCAR.

The use of the properties file and object to control elements of the program execution has been removed. Instead, the required information is either specified as part of a constructor's signature or using a `set()` method on the object in question. This improves the thread-safety of OSCAR, particularly in a multiuser environment, and contributes to its usability since a user can now trivially see what features may be customised from the outline of the class as opposed to needing to know which and how properties are used by which components.

Input and Output Formats

As previously discussed, OSCAR3 required that input documents be converted into SciXML before processing can occur, using the document formatting as a base against which annotations for identified named entities can be referenced – whether as inline or standoff annotations. OSCAR4 removes this requirement by operating on plain text and producing `NamedEntity` and `DataAnnotation` objects to represent recognised sections of text and does not currently produce serialised output, though some support for the serialisation of annotations into XML documents is

planned for future releases. It should be realised, however, that there is no single, fool-proof approach to this problem. Different XML schema may use different methods to indicate where in the document section breaks and even text content occur, while it cannot be guaranteed that well-formed inline annotations can be generated for a given input document. Client programmers are therefore recommended to consume `NamedEntity` objects directly rather than rely upon serialised output, though it is realised that users are likely to want to be able to create serialised, marked-up copies of their documents as well.

Non-core functionality

As much non-critical code as possible has been removed from the OSCAR4 codebase to reflect the philosophy that OSCAR4 should act as a library. While some minor supporting code remains, such as that required for generation of key resource files, the majority has been removed entirely as it is envisaged that much of the former functionality could be better implemented by developers with specific use cases.

A number of useful non-core functions are provided in dependent libraries developed at the Unilever Centre in Cambridge. Specifically, subsidiary modules exist to provide the capacity to run OSCAR4 from the command-line, as part of UIMA or Taverna[44] workflows and from the Bioclipse[45] scripting interface, as shown in Figure 8.

Figure 8: OSCAR4 run within Bioclipse's scripting interface (centre pane) to identify named entities in a block of text and save the connection tables to file (`extractedMols.sdf`) for viewing (right pane).

Performance

A number of modifications were introduced to the OSCAR code with the aim of reducing the time required to process documents. Performance hotspots were identified using the YourKit Java profiler and where possible eliminated. Some such improvements focused on the time taken to initialise the various OSCAR components, such as supplying a pre-calculated, serialised copy of the *n*-gram models used for named entity recognition rather than regenerating them each time OSCAR is loaded. Others improved the speed at which OSCAR can process a document by optimising extremely tight loops in the code, such as eliminating unnecessary string declaration while calculating *n*-gram features and avoiding recompilation of regular expressions. Further improvements were made *ad hoc*, as the OSCAR4 developers encountered obvious bottlenecks while working on the code.

In order to quantify the improvement in speed of operation, the time taken by both OSCAR4 version 4.0.1 and OSCAR3 alpha 5 to perform two tasks was measured. The first task measured the time taken to initialise the software to the point that it was ready to begin the task of finding named entities in text; the second task aimed to measure the speed at which the software could process bulk text and consisted of processing the full text of the 68 patents published by the European Patent Office in the week of 2009-05-06 – a total of 11468 paragraphs of text. All the tasks were run on a desktop computer equipped with an Intel Pentium 4 (3.00GHz) CPU and 1GB of RAM, purchased *c*. 2005, running openSUSE 11.1 and using the Java 1.6.0_22 32-bit virtual machine with a maximum heap size of 512MB. The results are summarised in **Table 1** and **Table 2**.

Software version	OSCAR4 4.0.1	OSCAR3 alpha 5	OSCAR4 4.0.1	OSCAR3 alpha 5
Recogniser	MEMMRecogniser	MEMMRecogniser	PatternRecogniser	PatternRecogniser
Mean time (s)	14.4	17.3	19.7	24.6
Standard deviation (ms)	40.8	40.0	72.6	88.6

Table 1: Results of the initialisation task

Software version	OSCAR4 4.0.1	OSCAR3 alpha 5	OSCAR4 4.0.1	OSCAR3 alpha 5
Recogniser	MEMMRecogniser	MEMMRecogniser	PatternRecogniser	PatternRecogniser
Mean time (s)	446	541	150	276

Standard deviation (s)	1.85	1.14	0.556	1.53

Table 2: Results of the bulk processing task

From these data, it can be seen that OSCAR4 performs significantly faster than OSCAR3. Initialisation times for the `MEMMRecogniser` and `PatternRecogniser` have been reduced by 17% and 20% respectively, while bulk processing times have been reduced by 18% and 46% respectively. The `OSCAR4 MEMMRecogniser` and `PatternRecogniser` processed approximately 26 and 76 paragraphs per second respectively, demonstrating that bulk processing of text is achievable on an acceptable timescale on desktop computers.

Deployment

OSCAR4 has generated significant interest in the community, and has been the subject of two meetings at the Unilever Centre for Molecular Science Informatics in Cambridge. The talks from the second of these are available to view online[46]. To our knowledge, the software has been adopted by the National Centre for Text Mining (NaCTeM), the European Bioinformatics Institute (EBI) and the European Patent Office (EPO) as well as various pharmaceutical companies.

We are aware of successful and straightforward integrations into the Bioclipse and Taverna frameworks, and believe that this is similarly straightforward for other Java environments. We were also pleased to see that at the recent MIOSS meeting at the EBI, OSCAR and OPSIN had been integrated into the .NET environment. For example, OPSIN was demonstrated as running within the JVM in Microsoft Excel, which is acceptable to commercial organisations as the JVM is of proven security.

Conclusions

This is a useful opportunity to reflect on the high cost of producing robust, re-usable software. OSCAR3, and OPSIN, were produced as a continuing activity by a mixture of summer students, PhDs and PDRAs and, until *ca.* 2009, evolved rather than having a top-down software design. When the project became valuable to the world, it was a clear indication that re-factoring was going to be essential, and it is important to realise the necessary but high cost of doing this. In times of lean funding, it will become increasingly difficult to obtain this type of support, and therefore it is always tempting to transfer academic code to commercial entities which can raise revenue.

The downside of this is that we know of very few commercial codes, and certainly none in chemical text analysis, that provide public metrics let alone expose the architecture on which the program is based. Text-mining as an academic subject requires metrics and increasingly requires Openness of

the components of the system, as we have done in OSCAR and OPSIN. We are investigating continuing business models where we can continue to re-factor and improve the product while not closing the code and therefore reducing scientific credibility and innovation.

Further Reading

The OSCAR4 libraries are available from https://maven.ch.cam.ac.uk

The OSCAR4 Javadoc is available at http://apidoc.ch.cam.ac.uk/oscar4-4.0.1

The source code, mailing list, tutorials and support are available at
https://bitbucket.org/wwmm/oscar4/wiki/Home

The source code used to measure OSCAR performance is available at
https://bitbucket.org/dmj30/oscar-performance

Acknowledgements

Authors' Contributions

DMJ wrote the manuscript and was lead developer in the OSCAR4 re-factoring.

SEA was the architect and project manager for the OSCAR4 re-factoring and was involved in the original Experimental Data Checker project.

ELW contributed to the OSCAR4 re-factoring and investigated its use in Bioclipse.

LH contributed to the OSCAR4 re-factoring and was involved in the CheTA project.

PMR had the overall vision for, and was involved in, all stages of the various OSCAR projects, and wrote the manuscript.

REFERENCES

[1] Kemp N, Lynch M: **Extraction of Information from the Text of Chemical Patents. 1. Identification of Specific Chemical Names.** *J. Chem. Inf. Comput. Sci.* 1998, **38**, 544-551

[2] Wilbur WJ, Hazard JF, Divita G, Mork JG, Aronson AR, Browne AC: **Analysis of Biomedical Text for Chemical Names: A Comparison of Three Methods.** *Proc. AMIA Symp.* 1999 176-180

[3] Wren JF: **A scalable machine-learning approach to recognize chemical names within large text databases.** *BMC Bioinformatics.* 2006 **7** (Suppl. 2) S3

[4] Klinger R, Kolářik C, Fluck J, Hofmann-Apitius M, Friedrich CM: **Detection of IUPAC and IUPAC-like chemical names.** *Bioinformatics* 2008 **24 i268-276**

[5] Hettne KM, Stierum RH, Schuemie MJ, Hendriksen PJM, Schijvenaars BJA, van Mulligen EM, Kleinjans J, Kors JA: **A dictionary to identify small molecules and drugs in free text.** *Bioinformatics* 2009 **25** 2983-2991

[6] Jiao D, Wild DJ: **Extraction of CYP Chemical Interactions from Biomedical Literature Using Natural Language Processing Methods.** *J. Chem. Inf. Comput. Sci.* 2009 **49** 263-269

[7] Kolářik C, Klinger R, Friedrich CM, Hofmann-Apitius M, Fluck J: **Chemical Names: Terminological Resources and Corpora Annotation.** In *Proceedings of the Workshop on building and evaluating resources for biomedical text mining (6*th* edition of the Language Resources and Evaluation Conference)* 2008 51-58*.* Marrakech

[8] Grego T, Pezik P, Couto FM, Rebholz-Chuhmann D: **Identification of Chemical Entities in Patent Documents.** In *Distributed Computing, Artificial Intelligence, Bioinformatics, Soft Computing, and Ambient Assisted Living, 2009 Berlin, Germany.* Springer-Verlag 942–949

[9] Hettne KM, Williams AJ, van Mulligen EM, Kleinjans J, Tkachenko V, Kors JA: **Automatic vs manual curation of a multi-source chemical dictionary: the impact on text mining.** *Journal of Cheminformatics.* 2010 **2**:3

[10] Müller B, Klinger R, Gurulingappa H, Mevissen H, Hofmann-Apitius M, Fluck J, Friedrich CM: **Abstracts versus Full Texts and Patents: A Quantitative Analysis of Biomedical Entities.** In *Advances in Multidisciplinary Retrieval.* Springer-Verlag 2006: 152-165

[11] Sun B, Mitra P, Giles CL, Mueller KT: **Identifying, Indexing and Ranking Chemical Formulae and Chemical Names in Digital Documents.** *ACM Trans. Inf. Syst.* **29** Article 12

[12] Lowe DM, Corbett PT, Murray-Rust P, Glen RC*:* **Chemical name to structure: OPSIN, an open source solution.** *J. Chem. Inf. Model.* 2011, **51**, 739-753. DOI: 10.1021/ci100384d

[13] OPSIN [http://opsin.ch.cam.ac.uk/] Accessed 2011-06-15

[14] Hawizy L, Jessop DM, Adams N, Murray-Rust P: **ChemicalTagger: A tool for semantic text-mining in chemistry.** *J.Cheminf.* 2011, **3**, 17. DOI: 10.1186/1758-2946-3-17

[15] ChemicalTagger [http://chemicaltagger.ch.cam.ac.uk/] Accessed 2011-06-15

[16] Adams SE, Goodman JM, Kidd RJ, McNaught AD, Murray-Rust P, Norton FR, Townsend JA, Waudby CA: **Experimental data checker: better information for organic chemists.** *Org. Biomol. Chem.* 2004, **2**, 3067-3070. DOI: 10.1039/b411699m

[17] RSC Experimental Data Checker [http://www.rsc.org/Publishing/Journals/guidelines/AuthorGuidelines/AuthoringTools/ExperimentalDataChecker/index.asp] Accessed 2011-06-15

[18] Corbett P. Murray-Rust P: **High-Throughput Identification of Chemistry in Life Science Texts.** In *Computational Life Sciences II, Second International Symposium, CompLife 2006, Cambridge, UK, September 27-29, 2006, Proceedings. Lecture Notes in Computer Science.* Edited by Berthold MR, Glen RC, Fischer I*.* Heidelberg: Springer; 2006: 107-118.

[19] Corbett P, Copestake A: **Cascaded classifiers for confidence-based chemical named entity recognition.** *BMC Bioinformatics*, 2008, **9**(Suppl 11)**:**S4. DOI:10.1186/1471-2105-9-S11-S4

[20] Steinbeck C, Hoppe C, Kuhn S, Floris M, Guha R, Willighagen EL: **Recent Developments of the Chemistry Development Kit (CDK) - An Open-Source Java Library for Chemo- and Bioinformatics.** *Curr. Pharm. Des.* 2006, **12**, 2111-2120.

[21] Steinbeck C, Han Y, Kuhn S, Horlacher O, Luttman E, Willighagen E: **The Chemistry Development Kit (CDK): An Open-Source Java Library for Chemo- and Bioinformatics.** *J. Chem. Inf. Comput. Sci.* 2003, **43**, 493-500. DOI: 10.1021/ci025584y

[22] Chemical Development Kit (CDK) [http://sourceforge.net/projects/cdk/] Accessed 2011-06-15

[23] Degtyarenko K, de Matos P, Ennis M, Hastings J, Zbinden M, McNaught A, Alcantara R, Darsow M, Guedj M, Ashburner M: **ChEBI: a database and ontology for chemical entities of biological interest.** *Nucl. Acids Res.* 2008, **36**(Suppl. 1), D344-D350. DOI: 10.1093/nar/gkm791

[24] de Matos P, Alcántara R, Dekker A, Ennis M, Hastings J, Haug K, Spiteri I, Turner S, Steinbeck C: **Chemical Entities of Biological Interest: an update.** *Nucl. Acids Res.* 2009 **38** (Suppl. 1) D249-D254, DOI: 10.1093/nar/gkp886

[25] ChEBI [http://www.ebi.ac.uk/chebi/] Accessed 2011-06-15

[26] FIX ontology [http://www.berkeleybop.org/ontologies/owl/FIX] Accessed 2011-06-15

[27] REX ontology [http://www.berkeleybop.org/ontologies/owl/REX] Accessed 2011-06-15

[28] UIMA [http://uima.apache.org/] Accessed 2011-06-15

[29] U-Compare [http://u-compare.org/] Accessed 2011-06-15

[30] Kano Y, Baumgartner W, McCrohon L, Ananiadou S, Cohen KB, Hunter L, Tsujii J: **U-compare: Share and compare text mining tools with UIMA**. *Bioinformatics,* 2009, **25**, 1997-1998. DOI: 10.1093/bioinformatics/btp289

[31] Kolluru B, Hawizy L, Murray-Rust P, Tsuji J, Ananiadou S: **Using workflows to explore and optimise named entity recognition for chemistry.** PLoS ONE, 2011, 6, e20181. DOI: 10.1371/journal.pone.0020181

[32] Jessop DM. *PhD thesis*. University of Cambridge 2011.

[33] Jessop DM, Adams S, Murray-Rust P: **Mining chemical information from Open patents.** *J. Cheminf.* 2011, this issue.

[34] McNaught, A: **The IUPAC International Chemical Identifier.** *Chemistry International* 2006, **November-December**, 12-14.

[35] IUPAC International Chemical Indentifier [http://www.iupac.org/inchi/] Accessed 2011-06-15

[36] Hearst MA: **Automatic acquisition of hyponyms from large text corpora.** In *COLING '92 Proceedings of the 14th conference on Computational linguistics – Volume 2* 539-545

[37] YourKit java profiler [http://www.yourkit.com/] Accessed 2011-06-15

[38] OSCAR3 alpha 5 [http://sourceforge.net/projects/oscar3-chem/] Accessed 2011-06-15

[39] Weininger, D: **SMILES, a chemical language and information system. 1. Introduction to methodology and encoding rules.** *J. Chem. Inf. Comput. Sci.* 1988, **28**, 31-36.

[40] Weininger D, Weininger A, Weininger JL: **SMILES. 2. Algorithm for generation of unique SMILES notation.** *J. Chem. Inf. Comput. Sci.* 1989, **29**, 97-101.

[41] Murray-Rust P, Rzepa HS: **Chemical Markup, XML and the Worldwide Web. 1. Basic Principles.** *J. Chem. Inf. Comput. Sci.* 1999, **39** 928-942.

[42] "Annotation of Chemical Named Entities", BioNLP 2007

[43] "Cascaded Classifiers for Confidence-Based Chemical Named Entity Recognition", BioNLP 2008 (This one is already cited above)

[44] Taverna [http://www.taverna.org.uk/] Accessed 2011-06-15

[45] Bioclipse [http://www.bioclipse.net/] Accessed 2011-06-15

[46] OSCAR4 Launch. [http://sms.cam.ac.uk/collection/1130934] Accessed 2011-06-15

Preparation of (3S,aR)-methyl 3-(3-chlorophenyl)-3-(N-a-methyl-4-methoxybenzylamino)propanoate 33
Following general procedure 5, 29 (1.89g, 4.87mmol) was added to a saturated solution of HCl in MeOH (20ml). Concentration in vacuo, recrystallisation (EtOAc:hexane) and treatment with saturated aqueous NaHCO3 gave 33 (1.48g, 79%) as a colourless oil; mp (HCl salt) 169-170°C (EtOAc:hexane); [a]D20 +17.2 (c 1.0, CHCl3); Found C, 59.4; H, 6.0; N, 3.8%; C19H23Cl2NO3 requires C, 59.4; H 6.0; N, 3.6%; nmax (film) 3329 (NH), 1737 (C=O); dH (400MHz, CDCl3) 1.34 (3H, d, J6.5, C(a)Me), 1.86 (1H, br s, NH), 2.63 (1H, dd, J2A,2B15.4, J2A,36.1, C(2)HA), 2.73 (1H, dd, J2B,2A15.4, J2B,37.8, C(2)HB), 3.63 (1H, q, J6.5, C(a)H), 3.64, 3.79 (2 x 3H, s, CO2Me and OMe), 4.16 (1H, dd, J2A,36.1, J2B,37.8, C(3)H), 6.83 (2H, m, Ph(3)H, Ph(5)H C6H4OMe), 7.16 (2H, m, Ph(2)H, Ph(6)H C6H4OMe), 7.16 (1H, m, Ph(6)H C6H4Cl), 7.19-7.23 (3H, m, Ph(2)H, Ph(4)H, Ph(5)H C6H4Cl); dC (100MHz, CDCl3) 22.4, 42.1, 51.6, 54.2, 55.7, 56.5, 113.7, 125.1, 127.2, 127.4, 127.6, 129.8, 134.3, 137.6, 145.0, 158.5, 171.9; m/z (APCI+) 348 (MH+, 10%).

Preparation of (3S,aR)-methyl 3-(3-bromophenyl)-3-(N-a-methyl-4-methoxybenzylamino)propanoate 34
Following general procedure 5, 30 (750mg, 1.73mmol) was added to a saturated solution of HCl in MeOH (20ml). Concentration in vacuo, recrystallisation (EtOAc:hexane) and treatment with saturated aqueous NaHCO3 gave 34 (487mg, 66%) as a colourless oil; [a]D20 +13.5 (c 1.0, CHCl3); nmax (film) 3326 (NH), 1736 (C=O), 1512 (OMe), 1246 (Ph-OMe); dH (400MHz, CDCl3) 1.38 (3H, d, J6.5, C(a)Me), 1.84 (1H, bs, NH), 2.63 (1H, dd, J2A,2B15.4, J2A,36.1, C(2)HA), 2.72 (1H, dd, J2B,2A15.4, J2B,37.7, C(2)HB), 3.62 (1H, q, J6.5, C(a)H), 3.64, 3.79 (2 x 3H, s, CO2Me and OMe), 4.14 (1H, dd, J3,2B7.7, J3,2A6.1, C(3)H), 6.81 (2H, m, Ph(3)H and Ph(5)H C6H4OMe), 7.14-7.21 (4H, m, Ph(2)H and Ph(6)H C6H4OMe, Ph(5)H and Ph(6)H C6H4Br), 7.37 (1H, m, Ph(2)H C6H4Br), 7.41 (1H, m, Ph(4)H C6H4Br); dC (100MHz, CDCl3) 22.4, 42.1, 51.6, 54.3, 55.2, 56.5, 113.7, 122.6, 125.6, 127.6, 130.1, 130.1, 130.4, 137.6, 145.3, 158.5, 171.8; m/z (CI+) 392 (MH+, 15%), 135 (C9H11O+, 100%); HRMS (CI+) C19H23BrNO3 requires 392.0861, found 392.0858.

Preparation of (3S,aR)-methyl 3-(4-bromophenyl)-3-(N-a-methyl-4-methoxybenzylamino)propanoate 35
Following general procedure 5, 31 (2.2g, 5.08mmol) was added to a saturated solution of HCl in MeOH (75ml). Concentration in vacuo, recrystallisation (EtOAc:hexane) and treatment with saturated aqueous NaHCO3 gave 35 (1.45g, 73%) as a colourless oil; [a]D20 +14.9 (c 0.9, CHCl3); nmax (film) 2959 (C-H), 1736 (C=O), 1512 (OMe), 1245 (Ph-OMe); dH (400MHz, CDCl3) 1.32 (3H, d, J6.5, C(a)Me), 1.44 (1H, bs, NH), 2.61 (1H, dd, J2A,2B15.4, J2A,36.4, C(2)HA), 2.71 (1H, dd, J2B,2A15.4, J2B,37.5, C(2)HB), 3.62 (1H, q, J6.5, C(a)H), 3.63, 3.79 (2 x 3H, s, CO2Me and OMe), 4.14 (1H, app t J6.9, C(3)H), 6.80-6.84 (2H, m, Ph(3)H and Ph(5)H C6H4OMe), 7.12-7.17 (4H, m, Ph(2)H and Ph(6)H C6H4OMe, Ph(2)H and Ph(6)H C6H4Br), 7.41-7.44 (2H, m, Ph(3)H and Ph(5)H C6H4Br); dC (100MHz, CDCl3) 22.3, 42.0, 51.6, 54.2, 55.3, 56.2, 113.7, 121.0, 127.6, 128.8, 129.0, 131.6, 131.7, 137.6, 141.7, 150.0, 158.5, 171.8; m/z (CI+) 392.1 (MH+, 10%), 135.1 (C9H11O+, 100%); HRMS (ESI) C19H23BrNO3 requires 392.0861, found 392.0861.

Preparation of (R)-methyl 3-(4-iodophenyl)-3-aminopropanoate 36
Following general procedure 4, CAN (1.0g, 1.84mmol) and 32 (200mg, 0.46mmol) in 5:1 MeCN:H2O (6ml) gave, after work up and purification by column chromatography on silica gel (hexane:Et2O 1:2), 36 (85mg, 61%) as a yellow oil; [a]D20 +11.5 (c 1.06, CHCl3); nmax(film) 3375 (NH), 2950 (CH), 1732 (C=O); dH (400MHz, CDCl3) 2.64 (2H, m, C(2)H2), 3.68 (3H, s, CO2Me), 4.39 (1H, bs, C(3)H), 7.12 (2H, m, Ph(2)H and Ph(6) C6H4I), 7.66 (2H, m, Ph(3)H and Ph(5) C6H4I); dC (100MHz, CDCl3) 43.6, 51.7, 52.1, 92.7, 128.3, 137.7, 137.8, 172.1; m/z (APCI+) 306 (MH+, 25%); HRMS (CI+) C10H13INO2 requires 305.9991, found 305.9999.

Preparation of (S)-methyl 3-(3-chlorophenyl)-3-aminopropanoate 37
Following general procedure 4, CAN (2.58g, 4.70mmol) and 33 (408mg, 1.18mmol) in 5:1 MeCN:H2O (12ml) gave, after work up and purification by column chromatography on silica gel (hexane:Et2O 1:2), 37 (135mg, 54%) as a yellow oil; [a]D20 -11.5 (c 1.0, CHCl3); nmax (film) 3376 (NH), 1732 (C=O); dH (400MHz, CDCl3) 1.88 (2H, br s, NH2), 2.17 (2H, m, C(2)H2), 3.69 (3H, s, CO2Me), 4.40 (1H, m, C(3)H), 7.21 (1H, t, J7.8, Ph(5)H C6H4Cl), 7.29 (1H, d, J7.8, Ph(6)H C6H4Cl), 7.39 (1H, d, J7.8, Ph(4)H C6H4Cl), 7.53 (1H, s, Ph(2)H C6H4Cl); dC (100MHz, CDCl3) 43.7, 51.7, 52.1, 124.9, 129.4, 130.2, 130.6, 134.9, 146.9, 172.1; m/z (APCI+) 214 (MH+, 35%); HRMS (CI+) C10H13ClNO2 requires 214.0635; found 214.0628.

Preparation of (S)-methyl 3-(3-bromophenyl)-3-aminopropanoate 38
Following general procedure 4, CAN (1.90g, 3.46mmol) and 34 (338mg, 0.86mmol) in 5:1 MeCN:H2O (12ml) gave, after work up and purification by column chromatography on silica

Figure 1

Preparation of N-(2,5-dichlorobenzyl)-8-hydroxy-1,6-naphthyridine-7-carboxamide

Triphosgene (0.556g, 1.87 mmol) was added over 20 mins to a solution of
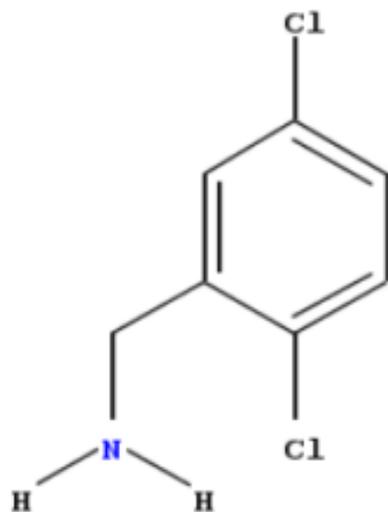
8-hydroxy-1,6-naphthyridine-7-carboxylic acid (0.89g, 4.68 mmol) and
diisopropylethylamine (3.26 ml, 18.7 mmol) in DMF (22 ml) at 0°C.

2,5-dichlorobenzylamine (0.142 ml, 1.05 mmol) was treated with a portion of the above
solution (0.58ml, 0.

for 16 hrs. 1H NMR   id = o11; surface = 2,5-dichlorobenzylamine; type = CM; confidence =
0.9293289445146479; SMILES = [H]C1=C([H])C(=C(C([H])=C1Cl)C([H])([H])N([H])[H])Cl;
d, J=4.0 Hz), 8   InChI = InChI=1/C7H7Cl2N/c8-6-1-2-7(9)5(3-6)4-10/h1-3H,4,10H2; cmlRef = cml6;   nd
4.0 Hz), 7.54 (1H, d, J=8.0Hz), 7.50–7.30 (2H, m), 4.64 (2H, d, J=5.0 Hz)
ppm.

FAB MS calcd for C16H11N3O2Cl2 348 (MH+), found 348.

- • Experimental data
  - • Ontology term
- • Chemical (etc.) with structure
  - • Chemical (etc.), without
    structure
    - • Reaction
- • Chemical adjective
- • enzyme -ase word
- • Chemical prefix

Figure 2

```
a)   String text = "The quick brown ethyl acetate jumps over the lazy bromine";
     Oscar oscar = new Oscar();
     List <NamedEntity> neList = oscar.findNamedEntities(text);


b)   String text = "The quick brown ethyl acetate jumps over the lazy bromine".
     Oscar oscar = new Oscar();
     List <ResolvedNamedEntity> rneList = oscar.findAndResolveNamedEntities(text);
```
Figure 3

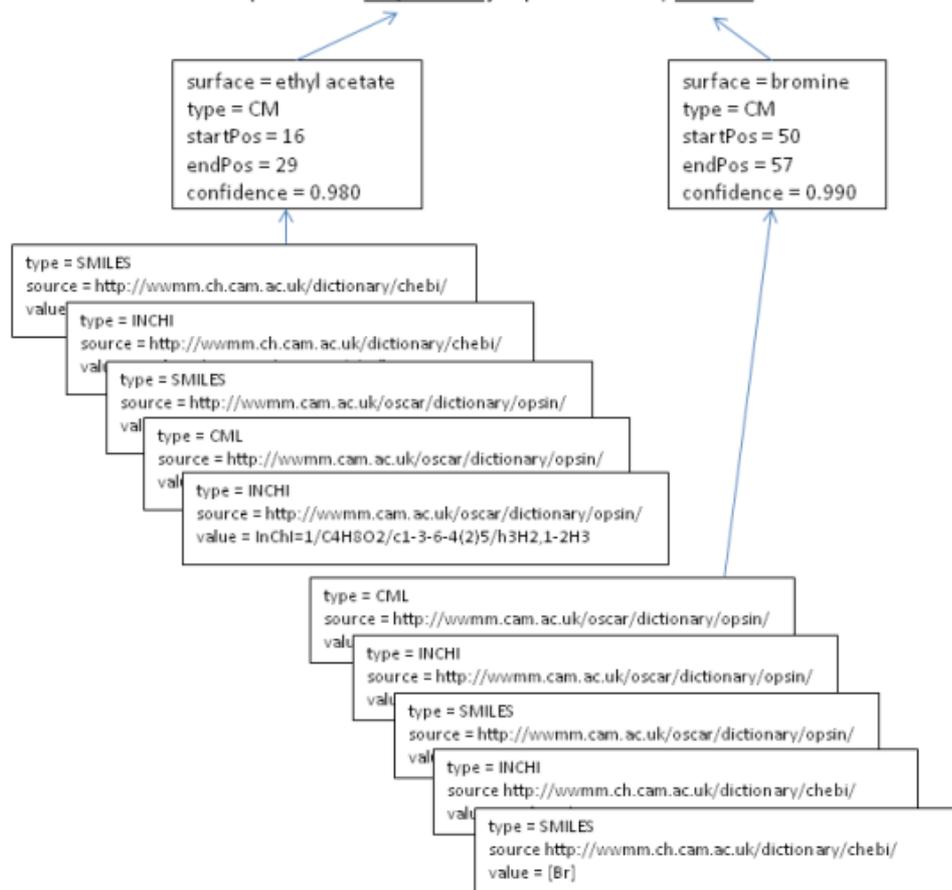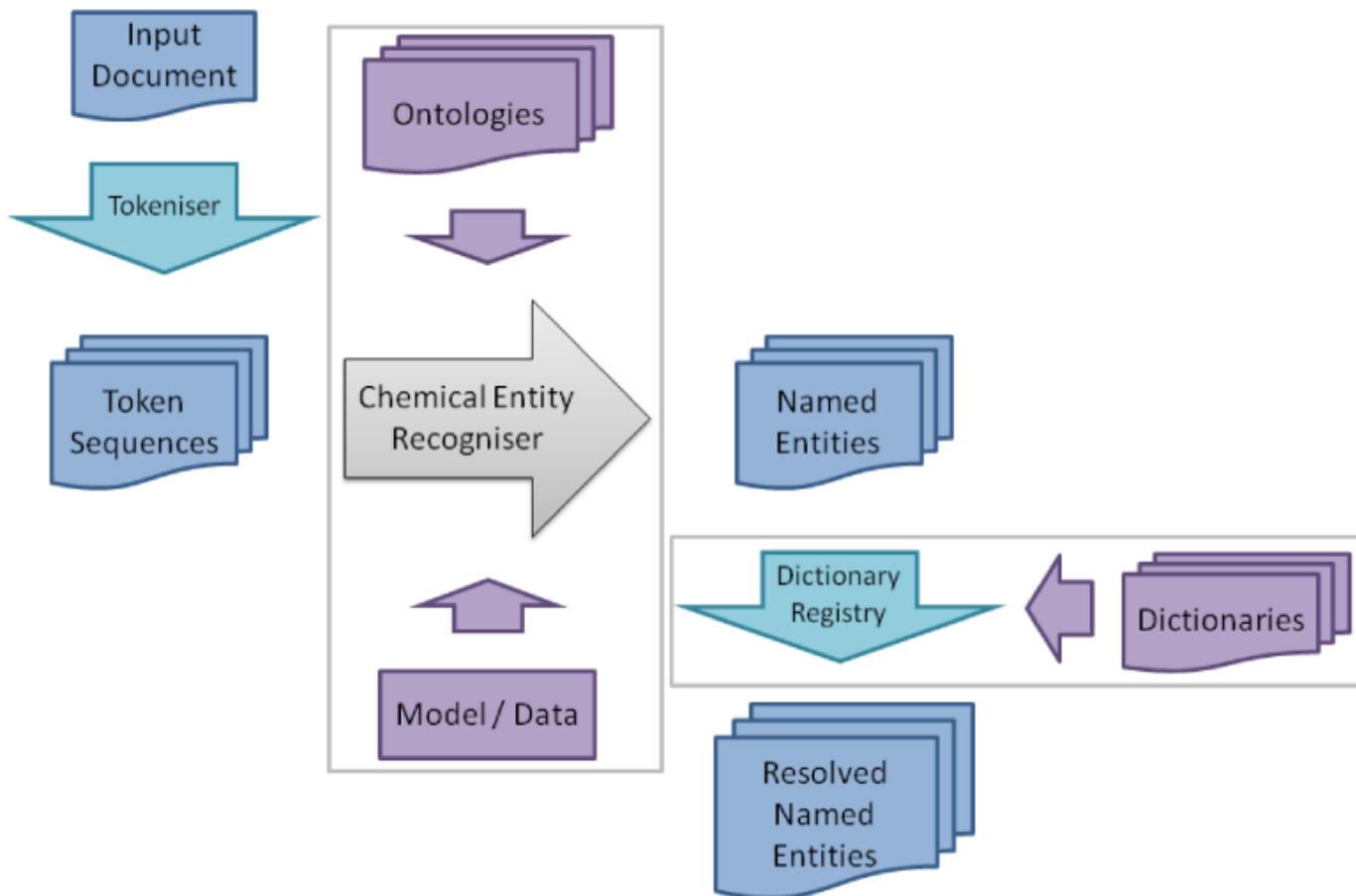The quick brown <u>ethyl acetate</u> jumps over the lazy <u>bromine</u>
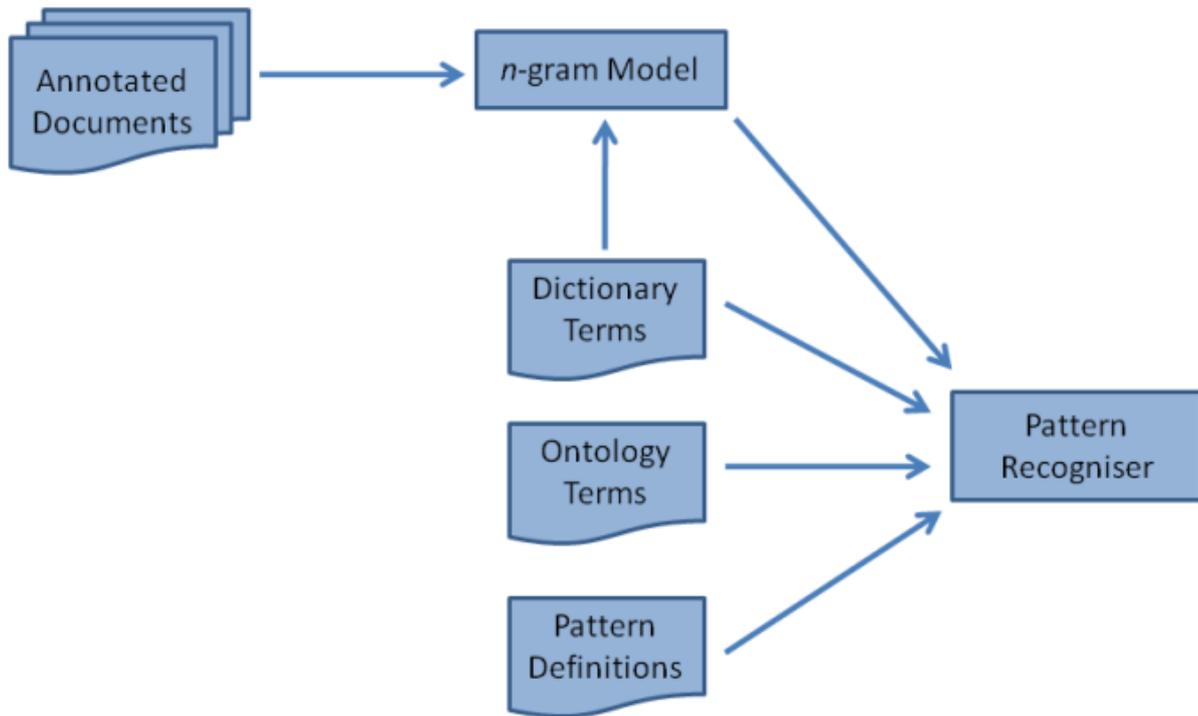
surface = ethyl acetate
type = CM
startPos = 16
endPos = 29
confidence = 0.980

surface = bromine
type = CM
startPos = 50
endPos = 57
confidence = 0.990

type = SMILES
source = http://wwwmm.ch.cam.ac.uk/dictionary/chebi/
value

type = INCHI
source = http://wwwmm.ch.cam.ac.uk/dictionary/chebi/
val

type = SMILES
source = http://wwwmm.cam.ac.uk/oscar/dictionary/opsin/
val

type = CML
source = http://wwwmm.cam.ac.uk/oscar/dictionary/opsin/
val

type = INCHI
source = http://wwwmm.cam.ac.uk/oscar/dictionary/opsin/
value = InChI=1/C4H8O2/c1-3-6-4(2)5/h3H2,1-2H3

type = CML
source = http://wwwmm.cam.ac.uk/oscar/dictionary/opsin/
val

type = INCHI
source = http://wwwmm.cam.ac.uk/oscar/dictionary/opsin/
val

type = SMILES
source = http://wwwmm.cam.ac.uk/oscar/dictionary/opsin/
val

type = INCHI
source http://wwwmm.ch.cam.ac.uk/dictionary/chebi/
val

type = SMILES
source http://wwwmm.ch.cam.ac.uk/dictionary/chebi/
value = [Br]

Figure 4

Figure 5

Figure 6

Figure 7

Figure 8