Chapter 13

ONTOLOGY DESIGN FOR BIOMEDICAL TEXT MINING

René Witte[1,2], Thomas Kappler[1], and Christopher J. O. Baker[2,3]
[1]*Universität Karlsruhe (TH), Germany;* [2]*Concordia University, Montréal (Québec), Canada;*
[3]*Institute for Infocomm Research, Singapore*

Abstract: Text Mining in biology and biomedicine requires a large amount of domain-specific knowledge. Publicly accessible resources hold much of the information needed, yet their practical integration into natural language processing (NLP) systems is fraught with manifold hurdles, especially the problem of semantic disconnectedness throughout the various resources and components. Ontologies can provide the necessary framework for a consistent semantic integration, while additionally delivering formal reasoning capabilities to NLP.

In this chapter, we address four important aspects relating to the integration of ontology and NLP: (i) An analysis of the different integration alternatives and their respective vantages; (ii) The design requirements for an ontology supporting NLP tasks; (iii) Creation and initialization of an ontology using publicly available tools and databases; and (iv) The connection of common NLP tasks with an ontology, including technical aspects of ontology deployment in a text mining framework. A concrete application example—text mining of enzyme mutations—is provided to motivate and illustrate these points.

Key words: Text Mining; NLP; Ontology Design; Ontology Population; Ontological NLP

1. INTRODUCTION

Text Mining is an emerging field that attempts to deal with the overwhelming amount of information available in non-structured, natural language form [1, 14, 40, 46]. Biomedical research and discovery is a particularly important application area as manual database curation—groups of experts reading publications and extracting salient facts in structured form for entry into biological databases—is very expensive and cannot keep up with the rapidly increasing amount of literature.

Developing suitable NLP applications requires a significant amount of domain knowledge, and there already exists a large body of resources for the biomedical

domain, including taxonomies, ontologies, thesauri, and databases [8]. Although most of these resources have not been developed for natural language analysis tasks but rather for biologist's needs, text mining systems typically make use of several such resources through a number of ad-hoc wrapping and integration strategies.

In contrast, in this chapter we show how to *design* an ontology specifically for NLP, so that it can be used as a single language resource throughout a biomedical text mining system. Hence, our focus is on analysing and explicitly stating the requirements for ontologies as NLP resources. In particular, we examine *formal* ontologies (in OWL-DL format) that, unlike the informal taxonomies typically used in NLP, also support automated reasoning and queries based on Description Logics (DL) [2] theorem provers.

After completing this chapter, the reader should be able to decide whether (and how) to employ ontology technology in a text mining application, based on the discussed integration alternatives and their respective properties. The application scenario, a text mining system analysing full-text research papers for enzyme mutations, provides the background for a detailed discussion of ontology design, initialization, and deployment for NLP, including technical challenges and their solutions.

Chapter outline. The next section analyses and motivates the connection between NLP and ontology in detail. A real-world scenario for biological text mining—enzyme mutations—is introduced in Section 3. We then provide a requirements analysis for ontology design in Section 4. How a concrete ontology fulfilling these requirements can be designed and initalized from existing resources is demonstrated in Section 5. And finally, we show in Section 6 how NLP tasks in a complex workflow can make use of the developed ontology, followed by a discussion and conclusions in Sections 7 and 8.

2. MOTIVATION FOR ONTOLOGY IN BIOMEDICAL TEXT MINING

Very little research has been done to show precisely what advantage ontologies provide vs. other representation formats when considering an NLP system by itself, i.e., not within a Semantic Web context. This discussion is split into two separate aspects: (1) Exporting NLP results by populating an ontology; and (2) Using an ontology as a language resource for processing documents.

2.1 Ontology as Result Format

Text mining results are typically exported in a (semi-)structured form using standard data formats like XML or stored in (relational) databases for further browsing or data mining.

Exporting text analysis results by instantiating a pre-modeled ontology, so-called *ontology population*, is one of the most common applications of ontology in NLP [29]. In [34] this is also referred to as "ontology-based processing," where the ontology is not necessarily used during the analysis process itself, but rather as a container to store and organize the results.

An obvious advantage of ontology population is that text analysis results are exported according to a standardised format (like OWL-DL), which can be stored, viewed, and edited with off-the-shelf tools. However, in cases where NLP results are fed directly into subsequent analysis algorithms for further processing, this advantage does not necessarily hold. Even so, there are further benefits that, in our view, outweigh the additional costs incurred by the comparatively complex ontology formats.

Result Integration. In complex application domains, like biomedical research and discovery, knowledge needs to be integrated from different resources (like texts, experimental results, and databases), different levels of scope (from single macromolecules to complete organisms), and across different relations (temporal, spatial, etc.). No single system is currently capable of covering a complete domain like biology by itself. This makes it necessary to develop focused applications that can deal with individual aspects in a reliable manner, while still being able to integrate their results into a common knowledge base. Formal ontologies offer this capability: a large body of work exists that deals with ontology alignment and the development of upper level ontologies [36], which can serve as a superstructure for the manifold sub-ontologies, while DL reasoners can check the internal consistency of a knowledge base, ensuring at least some level of semantic integrity.

Queries and Reasoning. By linking the structured information extracted from unstructured text to an ontology, semantic *queries* can be run on the extracted data. Moreover, using DL-based tools such as *Racer* [22] and its query languages, RQL and nRQL [52], *reasoning* by inference on T-Boxes (classes; concepts) and A-Boxes (individuals; instances) becomes possible. User-friendly interface tools like OntoIQ [5] allow even users without knowledge of DL to pose questions to an ontological knowledge base populated from natural language texts. Such functionality means that NLP-derived text segments used for automatically populating ontology concepts can subsequently be queried according to a user's familiarity with the domain content of the ontology.

Given that a multitude of specific text segments are generated when text mining a large body of scientific literature, querying the ontology is the equivalent of interrogating a summary of the whole domain of discourse, saving significant time in finding and reading relevant literature. This may in turn lead scientists to adopt a new approach to information retrieval, which is cross-platform and

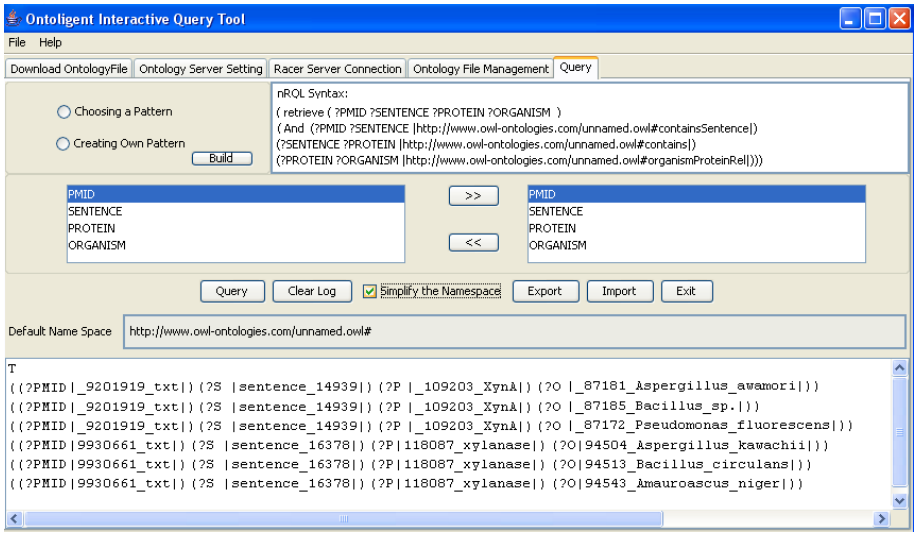

Figure 13-1. Querying an OWL-DL ontology populated by text mining full-text papers

content-specific rather than document-centric. Accessing the full text of a paper may become a secondary step occurring after the query of keyword-specific text segments or tiles from an NLP-instantiated ontology, invoked effortlessly from a user's desktop.

An example for this is depicted in Figure 13-1, which shows the query interface of OntoIQ [5]. The nRQL syntax of the query *"Find all references to organisms that are known to produce xylanases"* appears in the uppermost frame. The descriptors (Document-PMID, Sentence, Protein, and Organism) selected to appear in the query result are listed in the right hand frame below. The bottom frame shows the results returned through the interrogation of an NLP-populated ontology from the protein mutation domain that has been loaded into Racer. A user could now continue by examining the selected document sentences, connect with another ontology for further queries, or forward the selected instances to other (bioinformatics) tools for further automated processing.

2.2 Ontology as NLP Resource

Text mining systems require various language- and domain-specific resources, such as lexicons, gazetteer lists, or wordnets. These are typically accessed through ad-hoc data formats, such as flat files or databases. On a purely technical level, everything that can be expressed in an (OWL) ontology can be represented in another format, which in addition often can be simpler to develop and process. So what precisely is the motivation for using an ontology? Two important reasons are their representational capabilities and the improved semantic consistency they bring within a text mining system.

Semantically Richer Representation. An ontology allows for a more structured and semantically richer representation than many of the resources typically used in text mining systems, like simple gazetteer lists. This is particularly useful when the application domain of the texts is complex, as in biology; In such cases, the additional capabilities of ontologies, like relations, restrictions, and subsumption, allow for more efficient domain representations than simple templates. An example of this can be seen in [54], where an ontology guides information extraction from botanical texts.

Consistent Data Integration. Similar to the problem of result integration mentioned above, the various resources used throughout an NLP system need to be carefully managed to ensure semantic integrity. Currently, resources are typically not shared between analysis components (like a tokeniser, a noun phrase chunker, or a coreferencer), which can easily lead to inconsistencies. If an ontology can hold all the information necessary for the various analysis steps, only a *single* resource in *one* format needs to be developed and managed for the complete text mining system, thereby decreasing development effort while increasing overall semantic integrity.

3. CASE STUDY: TEXT MINING ENZYME MUTATIONS

In this section, we introduce a concrete application scenario for biological text mining, enzyme mutation mining. This example will be revisited several times in the following sections, e.g., in order to derive the requirements for an ontology supporting such an NLP system.

3.1 Biological Scenario

A large amount of biological knowledge today is only available from full-text research papers. Since neither manual database curators nor users can keep up with the rapidly expanding volume of scientific literature, natural language processing approaches are becoming increasingly important for bioinformatics projects.

Enzymes have widespread industrial applications and significant resources are devoted to the discovery of new enzymes and their development into commercial enzyme products with enhanced or new capabilities. Within the gene discovery process, there are numerous tests that newly discovered enzymes must pass before they can be considered for development into commercial products. Even enzymes with positive performance characteristics undergo mutational changes to improve their properties. The technologies used to design better enzymes involve either random or targeted mutagenesis, but in both cases scientists will

at some point review mutated residues in the 3D context of the protein structure. At this time the results of previous mutational analyses of the same or similar proteins are relevant and a review of the literature describing the mutations is necessary.

For protein engineers, understanding the impact of all mutations carried out on a protein family requires a complex mapping of sequence mutants to a common structure. Concurrent access to protein structure visualisations and annotations describing the impacts of mutations is possible using the *Protein Mutant Database* (PMD).[1] The content of this database is limited, however, by the speed at which newly published papers can be processed: In 1999, the PMD authors already reported a three-year backlog of unprocessed publications [27]. Thus, there exists a pronounced need to speed up the extraction of mutation-impact information from the scientific literature and make it more readily available to protein engineers. This has been our motivation for designing a text mining system capable of analysing enzyme mutation experiments described in full-text research papers: *Mutation Miner*.

3.2 Mutation Miner

The goal of this work is the annotation of 3D protein structures with segments of literature detailing the consequences of specific mutations. Mutation Miner [6, 53] is a sophisticated information system designed for this purpose that comprises an initial stage text mining subsystem linked to subsequent protein sequence retrieval and analysis subsystems. With Mutation Miner, a protein engineer can view structural representations of proteins (obtained from protein databases) combined with annotations describing mutations and their impacts (extracted through text mining from publications) within a unified visualisation using a tool like ProSAT [20] (Figure 13-2).

3.2.1 Implementation

The natural language analysis subsystem has been developed based on the GATE *(General Architecture for Text Engineering)* framework [16]. GATE is a component-based architecture, where documents are processed through *pipelines* of NLP components. This permits the dynamical assembly of a text mining application through adding, swapping, or re-ordering its components. Several standard components are supplied with the architecture, like a part-of-speech (POS) tagger, a gazetteer that assigns semantic labels to tokens (words) in a text, and the JAPE language [17] for expressing grammar rules, which are compiled into finite-state transducers. Results are exchanged between the

[1] Protein Mutant Database (PMD), `http://pmd.ddbj.nig.ac.jp/`

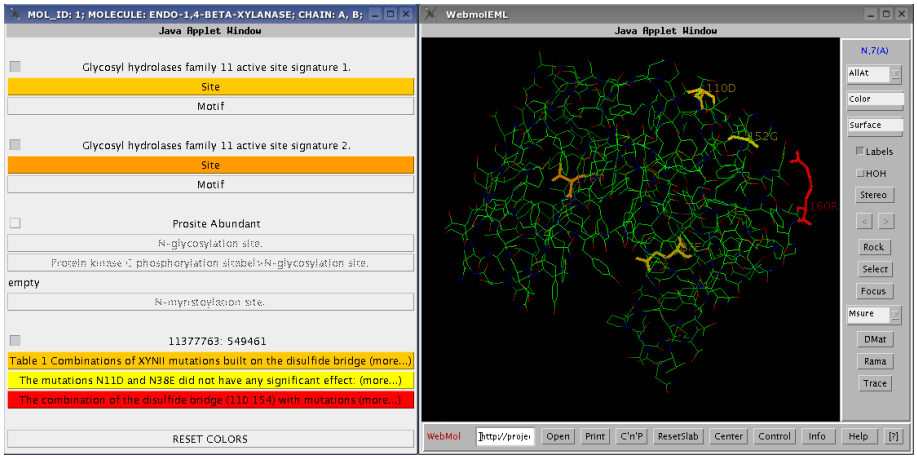

Figure 13-2. ProSAT showing a 3D (Webmol) visualisation of the endo-1,4-β-xylanase protein with mutations extracted through text mining, selected with the interface on the left. Sections of the extracted information are displayed on the buttons, the PMID for the original publication and the GI for the mutated protein are located above.

components through document *annotations* using a form of stand-off markup. For more details on GATE, we refer the reader to the online documentation.[2]

3.2.2 Ontology Extensions

Mutation Miner has originally been developed without innate support for ontologies: Resources were converted from external formats (like databases or taxonomies) into structures supported by GATE (like gazetteer lists). For the reasons stated above, we pursued the integration of the various disparate NLP resources into a single ontology shared by all NLP analysis components within the system.

At the same time, we also provide for result output in OWL-DL format (i.e., NLP-driven ontology population), which additionally enables semantic queries to instances of an ontological conceptualization, as shown in Figure 13-1. This becomes particularly interesting when the Mutation Miner ontology is integrated with other ontologies, as it allows cross-domain queries and reasoning. Instances generated by Mutation Miner alone provide information about impacts of mutational change on protein performance. These instances permit queries such as: *"Find the locations of amino acids in xylanase proteins, which when mutated have resulted in enhanced enzyme thermostability."* Integration of the Mutation Miner ontology with the instantiated FungalWeb ontology [44] that

[2]GATE documentation, `http://gate.ac.uk/documentation.html`

represents knowledge about the enzyme industry and fungal species additionally permits cross-disciplinary queries. For example, queries asking *"Identify the industrial benefits derived from commercial enzyme products based on mutated xylanases"* or *"What commercial enzyme products are not the result of mutational improvement"* become now possible. Depending on the user, access to this knowledge can assist in decision making for experimental design or product development. For further examples illustrating the use of formal ontology reasoning and querying in concrete application scenarios from fungal biotechnology, we refer the reader to [4, 7].

4. REQUIREMENTS ANALYSIS FOR ONTOLOGIES SUPPORTING NLP

In this section, we discuss how to design an ontology explicitly for supporting NLP-related tasks. We do this in two steps: Section 4.1 briefly discusses the typical tasks performed by a (biomedical) text mining system. This is followed by a requirements analysis in Section 4.2, where we state what information precisely needs to be in an ontology to support the various NLP tasks.

4.1 NLP Tasks

In order to motivate our requirements for designing ontologies as NLP resources, we briefly outline some of the major subtasks during the analysis of a biomedical document. These processing steps are shown in the left half of Figure 13-3.

4.1.1 Named Entity Recognition

Finding *Named Entities* (NEs) is one of the most basic tasks in text mining. In biological texts, typical examples for NEs are *Proteins*, *Organisms*, or *Chemicals*.

Named entity recognition, often also called *semantic tagging*, is a well-understood NLP task. Basic approaches to finding named entities include rule-based techniques using finite-state transducers [17, 42] and statistical taggers, e.g., using Support Vector Machines (SVMs) [32] or Hidden Markov Models (HMMs) [33].

Scientific publications and other knowledge resources containing natural language text in the biomedical domain show certain characteristics that make term recognition unusually difficult [37]. There is a high degree of term variation, partly caused by the lack of a common naming scheme for the above mentioned entities, like proteins or organisms. Often, identical names are used for a gene and the protein encoded by it, further complicating the automatic identification of genes and proteins. Moreover, there is an abundant use of abbreviations in the field, where their expansion into the non-abbreviated form is easy for expert human readers, but difficult for text mining systems.

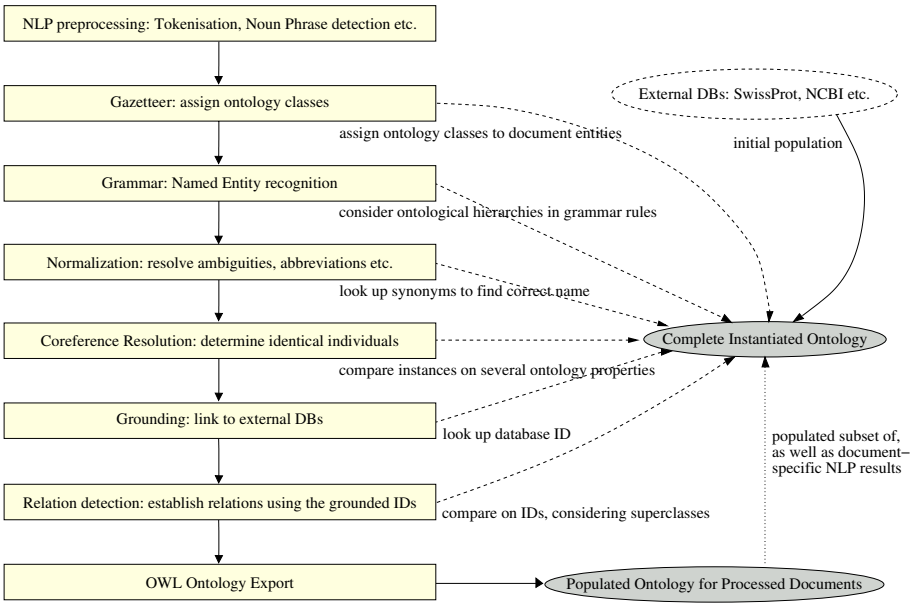

Figure 13-3. Workflow of the Mutation Miner NLP subsystem

While NE recognition is a well analysed task for the domain of newspaper and newswire articles, biomedical text mining requires further processing of detected entities, especially *normalization* and *grounding*.

4.1.2 Entity Normalization

Entities in natural language texts that occur in multiple places are often written differently: Person names, for example, might omit (or abbreviate) the first name, and include or omit titles and middle initials. Similarly, in biological documents, entities are often abbreviated in subsequent descriptions, e.g., the same organism can be referred to by both of the different textual descriptors, *Trichoderma reesei* and *T. reesei.* Likewise, the same protein mutation can be encoded using single-letter or three-letter amino acid references. It is important for downstream processing components that these entities are *normalized* to a single descriptor, e.g., the non-abbreviated form. For a thorough discussion on abbreviations in the biomedical domain, we refer the reader to [13].

4.1.3 Coreference Resolution

A task related to normalization is *coreference resolution*. In addition to abbreviations, other variations in names often exist. Within a biological text for example, the same protein might be referred to as *Xylanase II* and *endo-1,4-β-*

Xylanase II. In addition, *pronominal* references like *it* or *this* can also refer to a particular entity [12]. Consider the following sentence:[3]

> Interestingly, the Brønsted constants for the hydrolysis of aryl β-glucosides by Abg, a β-glucosidase from Agrobacterium faecalis, and its catalytic nucleophile mutant, E358D, [...] are also identical, as also are β_{1g} values for wild-type and E78D Bacillus subtilis xylanase (Lawson et al., 1996).

In the part "hydrolysis of aryl β-glucosides by Abg, a β-glucosidase from Agrobacterium faecalis, and its catalytic nucleophile mutant, E358D," the pronoun *its* refers to the β-glucosidase protein *Abg*, however, this is not obvious for an NLP system.

Finding all the different descriptors referring to the same entity (both nominal and pronominal) is the task of coreference resolution. The resulting list of entities is collected in a *coreference chain*. Note that even after successful resolution, a normalized name still needs to be picked from the coreference chain.

4.1.4 Grounding

As a final step in NE detection, many entities need to be *grounded* with respect to an external resource, like a database. This is especially important for most biological entities, which have corresponding entries in various databases, e.g., *Swiss-Prot* for proteins. When further information is needed for downstream analysis tasks, like the automatic processing of amino acid sequences, grounding the textual entity to a unique database entry (e.g., assigning a Swiss-Prot ID to a protein entity) is a mandatory prerequisite. Thus, even if an entity is correctly detected from an NLP perspective, it might still be ambiguous with respect to such an external resource (or not exist at all), which makes it useless for further automated processing until the entity has been grounded.

4.1.5 Relation Detection

Finding entities alone is not sufficient for a text mining system: most of the important information is contained within the *relations* between entities. For example, the Mutation Miner system described above needs to determine which organism produces a particular protein (*protein*↔*organism* relation) and which protein is modified by a mutation (*mutation*↔*protein* relation).

Relation detection can be very complex. Typical approaches employ predefined patterns or templates, which can be expressed as grammar rules, or a deep syntactic analysis using a full or partial parser for the extraction of

[3] Example sentence from: A. M. MacLeod, D. Tull, K. Rupitz, R. A. J. Warren, and S. G. Withers: "Mechanistic Consequences of Mutation of Active Site Carboxylates in a Retaining beta-1,4-Glycanase from Cellulomonas fimi," *Biochemistry* 1996, 35(40), PMID 8855954.

predicate-argument structures [34]. The performance of a relation detection component can be improved given information about semantically possible relations, thereby restricting the space of possible combinations.

4.2 Detected Requirements

We can now state a number of requirements that an ontology needs to fulfill in order to support NLP analysis tasks. Note that, although we illustrate these requirements with the Mutation Miner scenario, they apply equally to a wide range of biomedical text mining systems.

Requirement #0: Domain Model. As a prerequisite, the ontology needs to be structured according to the domain of discourse. Entities that are to be detected in an NLP system need to be contained in the ontology in form of classes (T-Boxes).

Requirement #1: Text Model. Concepts that model a document's components are needed in the ontology in addition to the domain concepts, e.g., classes for *sentences, text positions,* or *document locations*. These are required for anchoring detected entities (populated instances) in their originating documents.

Location is important to differentiate entities discovered in e.g. the list of *references* from those in e.g. *abstract* or *introduction*. Note that detecting the location requires additional text tiling algorithms, which we do not discuss within this chapter.

Additional classes are needed for NLP-related concepts that are discovered during the analysis process, like the *noun phrases* (NPs) and *coreference chains* discussed above.

Requirement #2: Biological Entities. The ontology needs instances (in form of A-Boxes) reflecting biological entities in order to be able to connect textual instances with their real-world counterparts. That is, if a biological entity is known to exist (for example, *Laccase IV*), it must have a counterpart in the ontology (namely, an instance in the enzyme subclass *oxidoreductase*).

It might appear naïve to assume that entities under consideration for text analysis are already available in biological databases, yet this is often the case: Publication in this subject domain requires the deposition of the entities under analysis (e.g., proteins) in publicly accessible databases. The challenge for text mining is in fact to discover within texts larger semantic connections between targeted entities (e.g., protein-protein interactions), which are not necessarily available in databases since it is access to this implicit knowledge that provides a competitive advantage to scientists.

In addition to the main entities of the domain in question, the ontology might include supplementary classes and relations, like fundamental biological, medical, or chemical information, which facilitate entity detection and other text analysis tasks.

Requirement #3: Lexical Information. In order to enable the detection of named entities in texts, the ontology needs lexical information about the biological instances stipulated in requirement #2. Lexical information includes the full names of entities, as well as their synonyms, common variants and misspellings, which are frequently recorded in databases. If unknown or highly varying expressions need to be detected in texts, entity-specific pre- and postfixes (e.g., *endo-* or *-ene*) can also be recorded in the ontology.

In addition, specialized NLP analysis tasks usually need further information, like subcategorization frames. For example, in order to correctly determine predicate-argument structures for proteins, postnominal phrases need to be attached to the correct noun phrase [43]. Storing the frame structures required for this step together with the entities in the ontology helps to maintain the overall semantic integrity of a system.

Requirement #4: Database Links. As mentioned before, entities detected in documents need to be connected with their real-world counterparts in a so-called *grounding* step. In order to support this task, the ontology must contain information about database locations and IDs (unique keys) of the various entities.

Grounding is needed in order to allow downstream analysis tasks to actually process entities detected in documents. For example, once a protein has been linked to a database like Swiss-Prot, its particular amino acid sequence can be retrieved from the database and processed by bioinformatics algorithms (e.g., *BLAST*[4] for sequence alignment).

Requirement #5: Entity Relations. Where available, biologically relevant relations between entities have to be encoded semantically in the ontology as well. This information is important for many steps, not only relation detection, where it helps disambiguating possible PP-attachments, but also for coreference resolution, normalization, and grounding. For instance, the normalized name of a protein can reflect both the protein function and the originating organism, which is important semantic information for the protein↔organism relation detection task.

[4]Basic Local Alignment Search Tool (BLAST), `http://www.ncbi.nlm.nih.gov/BLAST/`

Table 13-1. Ontological concept definitions and instance examples for Mutation Miner

Concept	Definition	Example Instances
Cellular Component	Subcellular structures, locations, and macromolecular complexes	Ribosome, Golgi, Vesicle
Plasmid	Circular double-stranded DNA capable of autonomous replication found in bacteria	pPJ20
Protein	A complex natural substance that has a high molecular weight and a globular or fibrous structure composed of amino acids linked by peptide bonds	Protein, Immunoglobulin
Organism	A virus or a unicellular or multicellular prokaryote or eukaryote	S. lividans, Clostridium thermocellum
Enzyme	A protein that acts as a catalyst, speeding the rate at which a biochemical reaction proceeds but not altering the nature of the reaction	Xylanase A, endo-1,4-β-xylanase
Recombinant Enzyme	Enzymes produced from new combinations of DNA fragments using molecular biology techniques	Xylanase A+E210D
Mutant	Indicates that something is produced by or follows a mutation; also a mutant gene or protein	E210D, Phe37Ala, Arg115
Measurement	Units of measurement	half life (s), Kcat, hydrolysis efficiency, pH
Property	The description of a biological, chemical or physical property of a protein that can be quantified	denaturation, catalysis, stabilization, unfolding
Impact	An examination of two or more enzymes (wild type or mutant) to establish similarities and dissimilarities	shift, increase, more active, fold, destabilize

5. BUILDING ONTOLOGICAL RESOURCES FOR BIOMEDICAL TEXT MINING

This section shows in detail how to design and initialize an ontology that supports the stated requirements. Although we focus our discussion on information required for the mutation scenario, the principles apply to other biological text mining tasks as well.

5.1 The Mutation Miner Ontology

An ontology that can house instances from Mutation Miner requires concepts for the main units of discourse—proteins, mutations, organisms—as well as supplementary concepts that characterize changes in enzyme properties, the direction of the change, and the biological property of the enzyme that has been altered (Req. #0). Table 13-1 shows the main concepts together with a brief definition and Figure 13-4 shows a part of the ontology graphically.

The ontology is represented in OWL-DL [45] and was created using the Protégé-OWL extension of Protégé,[5] a free ontology editor. Here, we made

[5]Protégé ontology editor, `http://protege.stanford.edu/`

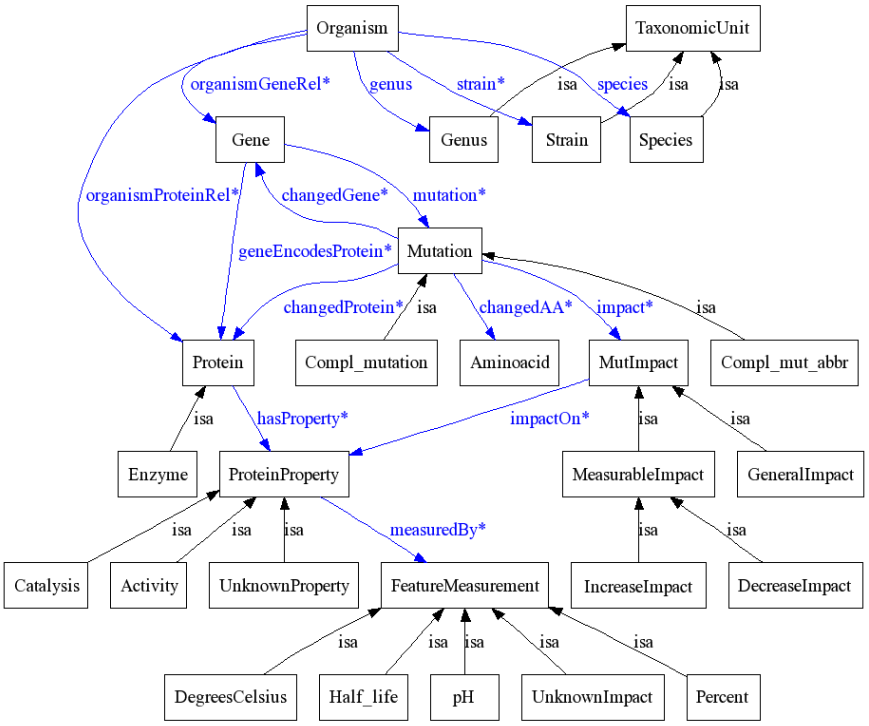

Figure 13-4. A part of the Mutation Miner ontology

use of two OWL language elements that model important information about the domain. Firstly, using *object properties*,[6] which specify relations between class instances, we register several relationships between instances of ontology classes. For example, the *Mutation* class has a *changedGene* object property, which is defined as having the domain "Mutation" and the range "Gene," linking a mutation instance to the instance of the gene it modifies. Secondly, cardinality restrictions are included to model the possible alternatives for denoting an organism. For example, the organism description in a text may consist of at most one genus, species, and strain, respectively, where strain is optional but only if both genus and species are given.

Several other enhancements to the ontology's expressiveness are possible, like placing additional restrictions on relations. They are not necessary, however, for the ontology-enhanced NLP analysis, but could be added to improve reasoning over extracted entities, e.g., for advanced querying.

[6]OWL Web Ontology Language Guide, Object Properties, `http://www.w3.org/TR/2004/REC-owl-guide-20040210/\#SimpleProperties`

Before the ontology can be deployed in an NLP system, instances for the various classes like *protein* or *organism* need to be created. Since adding and maintaining these instances and their relations manually is not an option, we now show how ontology instances can be automatically created and updated with respect to external biological databases.

5.2 Initializing the Ontology for Organisms

The systematic classification of organisms is called *taxonomy*. The individual species are set in relation to each other according to the degree of their genetic relationship. The names of organisms consist of parts called *taxonomic units*, giving the position in the classification tree. Usually, the taxonomic units *genus* and *species* are used in biomedical texts, resulting in a name such as *Escherichia coli*. Sometimes a *strain* is also given, which designates a more precise identification.

5.2.1 The NCBI Taxonomy Database

We use the *Taxonomy database* [19] from NCBI[7] to initialize our ontology (Req. #2). The Taxonomy database is "a curated set of names and classifications for all of the organisms that are represented in GenBank" (see [19] for a detailed description). GenBank[8] is another NCBI database, containing "publicly available DNA sequences for more than 165,000 named organisms." As of 2006-06-05, the Taxonomy database contained 310,756 classified taxa, with 409,683 different names in total.

In NCBI's database, every species and taxonomic unit has exactly one entry with a name classified as *scientific name,* as well as other possible variants. The scientific name is the "correct" one, and the others can be synonyms, common misspellings, or past names if the organism has been reclassified. Table 13-2 shows an example entry, constricted to the most important columns, for the organism *Escherichia coli (E. coli)*. It can be seen that there are seven synonyms and two common misspellings recorded in addition to the scientific name.

5.2.2 Ontology Creation with Jena

To convert the taxonomy data, it is possible to download the whole database, which is available as structured plain text files from NCBI's FTP server. A Python program was developed for this purpose, which reads these files and inserts their contents into an SQL database, preserving the structure by directly mapping each file to a database table and its columns to SQL columns in that table.

[7] NCBI Taxonomy Homepage, `http://www.ncbi.nlm.nih.gov/Taxonomy/`
[8] GenBank sequence database, `http://www.ncbi.nih.gov/Genbank/index.html`

The Mutation Miner ontology can now be populated from the contents of this database with a custom Java program using the *Jena* library. Jena[9] is an open source "Semantic Web Framework for Java," providing an API for OWL generation. Figure 13-5 shows the function creating the *Organism* instances from the Taxonomy data.

```
1  public static OntModel populateOrganisms( OntModel m ) {
2    // Instantiate the necessary OWL properties. "mmNS" is the Mutation Miner namespace.
3    DatatypeProperty organismName = m.getDatatypeProperty( mmNS+"organismName" );
4    DatatypeProperty organismAllNames = m.getDatatypeProperty( mmNS+"organismAllNames" );
5    DatatypeProperty ncbiId = m.getDatatypeProperty( mmNS+"ncbiId" );
6
7    // Plain text lists with mappings written out from the SQL DB.
8    Map id2sciName = listToMap(id2sciNameFile);
9    Map id2nonsciName = listToMap(id2nonsciNameFile);
10
11   Set oids = id2sciName.keySet();
12   String curOid, orgName;
13   ArrayList otherNames;
14   Individual curOrg;
15   /* For each organism, get its scientific name and create the Individual, then
16    * get the other names and store them in the organismAllNames property. */
17   for( Iterator oidsIt = oids.iterator(); oidsIt.hasNext() ) {
18     curOid = (String) oidsIt.next();
19     orgName = (String)((ArrayList)id2sciName.get(curOid)).get(0);
20     curOrg = m.createIndividual( mmNS+createClassName(orgName, curOid), organismClass );
21     curOrg.addProperty( organismName, orgName );
22     curOrg.addProperty( ncbiId, curOid );
23     otherNames = (ArrayList)id2nonsciName.get( curOid );
24     if ( otherNames != null )
25       curOrg.addProperty( organismAllNames, otherNames.toString() );
26   }
27   return m;
28 }
```

Figure 13-5. Creating *Organism* instances in the Mutation Miner ontology using Jena

The resulting comprehensive set of instances can be queried by all language processing components through GATE's ontology layer (we explain the technical details for this in Section 6.1).

5.2.3 Adding Lexical Organism Information

In order to support named entity detection of organisms, the ontology must contain the taxonomical names so that they can be matched against words in a text using a gazetteer NLP component (Req. #3). This information can also be directly extracted from the NCBI database, including the names themselves and information like the hierarchical structure of taxa and organisms.

[9] Jena, `http://jena.sourceforge.net/`

Table 13-2. The NCBI Taxonomy entry for E. coli (`tax_id` 562, `rank="species"`)

name_txt	**name_class**
"Bacillus coli" Migula 1895	synonym
"Bacterium coli commune" Escherich 1885	synonym
"Bacterium coli" (Migula 1895) Lehmann and Neumann 1896	synonym
Bacillus coli	synonym
Bacterium coli	synonym
Bacterium coli commune	synonym
Escherchia coli	misspelling
Escherichia coli	scientific name
Escherichia coli (Migula 1895) Castellani and Chalmers 1919	synonym
Escherichia coli retron Ec107	includes
Escherichia coli retron Ec67	includes
Escherichia coli retron Ec79	includes
Escherichia coli retron Ec86	includes
Eschericia coli	misspelling

Together with the taxonomical information we store additional metadata, like the originating database and the "scientific name," for each instance. This becomes important when delivering provenance information to scientists working with the populated ontology. An additional advantage of replacing flat organism lists with an ontology is that the taxonomical hierarchy is directly represented and can be queried by e.g. grammar rules. An example for this is given in Section 6.2.

5.2.4 Entity Normalization and Grounding

The initialized ontology now also holds the information required for named entity normalization and grounding: Firstly, by encoding the taxonomic relations we can ensure that only valid organism names are extracted from texts. For example, we can reject a genus-species combination that might look like a valid name to a simple organism tagger, yet is not supported by the NCBI database and therefore cannot be grounded in the ontology. Secondly, by encoding the "scientific name" given by NCBI, we can assign each detected organism a normalized name, which is at the same time grounded in the taxonomic database. Here, we extract and encode the database IDs when creating the ontology, linking each instance to the external NCBI resource (Req. #4).

5.3 Ontology Initialization for Proteins

We now need ontology support for analysing protein information (Req. #2), just as for organisms.

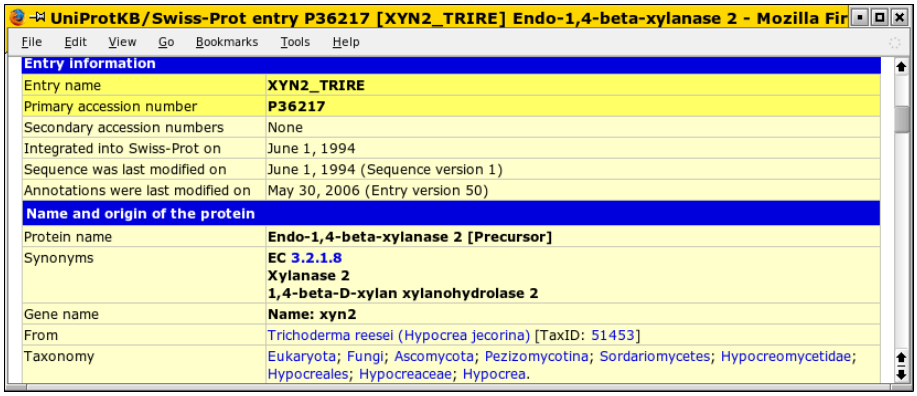

Figure 13-6. Swiss-Prot entry for *Xylanase II*

5.3.1 The Swiss-Prot Protein Database

The *UniProt Knowledge Base* [3] is a set of two protein databases, *Swiss-Prot*[10] and *TrEMBL*. Both hold entries about proteins appearing in published works, including information about protein functions, their domain structure, associated organisms, post-translational modifications, variants, among others. Swiss-Prot, which consisted of 228,670 entries as of 2006-07-02, contains "manually-annotated records with information extracted from literature and curator-evaluated computational analysis,"[11] while TrEMBL is populated by automatic analysis tools. In the Mutation Miner system, we use the manually curated Swiss-Prot database to gain reliable grounding (see Section 4.2) of proteins found in biological documents (Req. #4).

Figure 13-6 shows the Swiss-Prot entry for a variant of the *xylanase 2* protein. The entries most important for NLP analysis are the various "Synonyms," as they can all appear in a given biomedical document (Req. #3), the canonical name ("Protein name") that can depend on its host organism, and a unique ID ("Primary accession number") that allows unambiguous linking to the protein's entry.

A further essential feature of Swiss-Prot is that its entries are linked to other databases, notably to the NCBI Taxonomy database described in the previous section. This can be seen in the "From" line where the ID of the host organism ("TaxID") is recorded. Thus, proteins found in documents can easily be linked to their hosting organisms (Req. #5).

[10] Swiss-Prot protein database, `http://www.expasy.org/sprot/`
[11] Swiss-Prot manual, `http://www.expasy.org/sprot/userman.html`

The Swiss-Prot data can be downloaded from the Swiss-Prot website in XML, FASTA [38], and plain text format. We adapted our tool for writing NCBI data to an SQL database by exchanging its parser component in order to add the Swiss-Prot data to the database as well, thus enabling queries spanning the two datasets, using the NCBI ID recorded in both to join the results.

The database entry corresponding to Figure 13-6 contains the fields `ID` for the unique identifier, `DE` for the possible names, `GN` for the corresponding gene's name, and `OX` for the identifier linking to the Taxonomy database:

```
ID   XYN2_TRIRE     STANDARD;      PRT;   222 AA.
DE   Endo-1,4-beta-xylanase 2 precursor (EC 3.2.1.8) (Xylanase 2) (1,4-
DE   beta-D-xylan xylanohydrolase 2).
GN   Name=xyn2;
OS   Trichoderma reesei (Hypocrea jecorina).
OX   NCBI_TaxID=51453;
RX   MEDLINE=93103679; PubMed=1369024;
[...]
```

The protein data is then encoded in the ontology, similar to the information concerning organisms. Thus, the ontology now has all the required information for detecting protein named entities, as well as assigning normalized names and grounding them to Swiss-Prot IDs (note that some additional processing is required for Protein analysis, including abbreviation detection [13], however, we cannot cover these steps within the scope of this chapter).

Of particular interest are the *relations* between proteins and organisms inferred from the NCBI `TaxID` value, which are also transferred into our ontology according to Req. #5 (note the `organismProteinRel` relation in Figure 13-4). We can now create relation instances, again using Jena (cf. Figure 13-5):

```
ObjectProperty organismProteinRel = m.getObjectProperty( mmNS+"organismProteinRel" );
for( Iterator protIt = proteinClass.listInstances(); protIt.hasNext() ) {
  [...]  // Find the ncbiId stored in the protein's record.
  // Query for the organism with this id
  org = (Object)rdfLiteralQuery( ox, ncbiId, organismClass, m );
  prot.addProperty( organismProteinRel, org );
}
```

How we exploit the relation information from the ontology for the NLP analysis of entity relations is covered in Section 6.5.

There is further potentially interesting information available in Swiss-Prot records that could also be transferred to the ontology, for instance the Medline and Pubmed IDs of the publications where primary information concerning the protein is found (shown in the `RX` line of the listing), as well as the protein *sequence* (see Figure 13-9) needed for further automatic processing of text mining results.

5.4 Ontology Initialization for Mutations

In protein engineering literature, mutations describe changes to amino acid or gene sequences. Mutations are somewhat different from the previously discussed

entities like proteins and organisms, in that they are not exhaustively listed in some database, which could be converted into an ontology. However, it is still necessary to model the different kinds of mutations to allow the population of the result ontology with the detected instances (Req. #0, see Figure 13-4).

Mutations are typically identified using NLP techniques, like transducers (see, e.g., [26, 41]) or HMMs. To facilitate their detection, the ontology needs lexical information concerning *amino acids*, with their various textual representations (for instance, *"Asn"= "N"="Asparagine"* all denote the same amino acid). This lexical information is then evaluated for the detection of Mutation entities (Reqs. #2 and #3).

6. NLP-DRIVEN ONTOLOGY POPULATION

This section discusses how to employ the modeled and initialized ontology for the various NLP analysis tasks stated in Section 4.2 (see Figure 13-3). For the sake of brevity, we omit several standard NLP analysis steps in this discussion, like part-of-speech (POS) tagging, noun phrase (NP) chunking, or stemming. Readers unfamiliar with these tasks should consult [23] and the GATE user's guide.[12]

6.1 Interfacing Ontology and NLP

Before we go into detail on individual NLP analysis steps, we discuss some technical issues concerning current implementations when interfacing ontologies with NLP systems. This is an essential part of an ontology-centered system as outlined in Section 2, as it allows replacement of the different data resources needed within the various NLP tasks with an ontology as a single source that can then be queried by each component in different ways.

Ontology Support in GATE. Starting with version 3.0, GATE has been featuring built-in ontology support in form of an abstraction layer between the components of an NLP system and the various ontology representations [9]. This layer is built on Jena as RDF-Store, enabling the use of OWL ontologies from within GATE. Also, an integrated *SPARQL*[13] query engine allows querying the ontology's RDF graph. With SPARQL it is possible to perform SQL-like queries, e.g., for selecting instances based on their ID.

For example, in order to construct a SPARQL query for the Mutation Miner ontology to retrieve the scientific name of the organism with NCBI ID 1423, one has to ask for a name (variable `?name`) that is the value of a `scientificName`

[12]GATE user's guide, `http://gate.ac.uk/sale/tao/index.html`
[13]SPARQL RDF query language, `http://www.w3.org/TR/rdf-sparql-query/`

property of an organism (variable `?organism`), which in turn also has an `ncbiTaxId` property with the value "1423":

```
SELECT ?name
WHERE { ?organism mm:scientificName ?name
        ?organism mm:ncbiTaxId 1423 }
```

However, SPARQL is not OWL-capable in the sense that semantically richer queries considering the ontology classes and the class hierarchy, e.g., formally restricting the queried subjects to instances of the Organism class, can not be expressed. If this functionality is required, interfacing with an ontology reasoner (like *Racer* [22]) and using one of its supported query languages (like nRQL [52]) becomes necessary.

Limitations of GATE's Ontology Support. While the GATE *architecture* supports OWL-DL, very few NLP *components* are ontology-aware. In particular, the gazetteer as well as the JAPE transducer component can evaluate information from an ontology. However, at present they only make use of *is-a* relations between classes. For the gazetteer, this is sufficient because its sole purpose is to map ontology classes to names. It should be noted, however, that it currently cannot access an existing ontology via Jena, instead it must be provided with plain text lists whose entries are then mapped to ontology classes. Nevertheless, this is an implementation detail with little impact on the general ontology design; these lists can easily be generated from an ontology filled with the NCBI and Swiss-Prot data as described in Section 5. For an alternative approach to ontological gazetteering, see the *Semantic Gazetteer* component [39] developed within the KIM platform [29], which is also based on GATE.

The JAPE transducer component also features only limited ontology support. It currently considers the feature `class` of an annotation to be special and takes the ontological hierarchy into account when equality tests are performed on its value in grammar rules. For example, if a grammar contains the pattern `Token.class == "TaxonomicUnit"`, the rule will also match if the value of `class` is "Species," as Species *is-a* TaxonomicUnit in the Mutation Miner ontology.

Consequences for Ontology Design. The discussed implementation restrictions also have an impact on ontology design, as illustrated in Figure 13-7. The left part shows the protein section, initialized with the *Xylanase 2* protein, modeled using the full capabilities of OWL-DL: All proteins are instances of a single class and have a *name* property that is further subclassed to distinguish the standard name from its variants. On the right side, a design alternative is shown, where each protein is represented by its own subclass.

The second design alternative allows direct leverage of the capabilities of GATE components to analyse texts with respect to an ontology despite their being

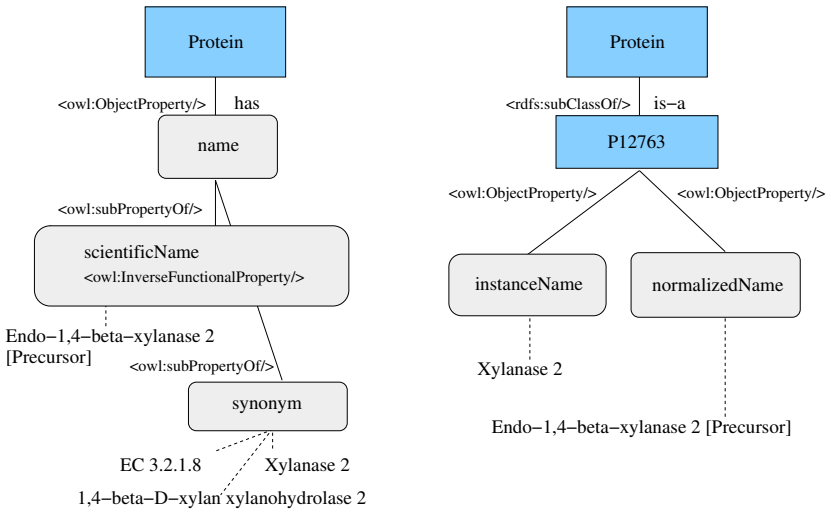

Figure 13-7. Ontology design alternatives for NLP analysis using GATE

limited to *is-a* class relationships. When the first, somewhat cleaner version is used, it becomes necessary to use a custom query interface for accessing the encoded information. These implementation issues will most likely change, however, in future versions of GATE.

6.2 Named Entity Detection

The basic process in GATE for recognizing entities of a particular domain starts with the gazetteer component. It matches given lists of terms against the tokens of an analysed text and, in case of a match, adds an annotation named `Lookup` whose features depend on the list where the match was found. Its ontology-aware counterpart is the *OntoGazetteer*, which incorporates mappings between its term lists and ontology classes and assigns the proper class in case of a term match. For example, using the instantiated Mutation Miner ontology, the gazetteer will annotate the text segment *Escherichia coli* with two `Lookup` annotations, having their `class` feature set to "Genus" for *Escherichia* and "Species" for *coli*.

In a second step, grammar rules written in the JAPE language are used to detect and annotate complex named entities. Those rules can refer to the `Lookup` annotation generated by the OntoGazetteer, and also evaluate the same ontology. For example, in a comparison like `class=="Species"`, the ontological hierarchy is taken into account so that also subspecies match, since a Subspecies *is-a* Species in the ontology. This can significantly reduce the overhead for grammar development and testing.

Hence, to detect *Organisms* in texts, an OntoGazetteer instance first annotates all tokens in a text that match instances in the ontology corresponding to Genus

or Species (additional grammar rules are employed to detect Strains). Specific grammar rules can then detect legal organism notations, for example, `[genus species strain?]`, which can be encoded in JAPE as:

```
Rule: OrganismRule1
Priority: 50
(
   ({Genus}   ):gen
   ({Species} ):spec
   (({Strain} ):str)?
):org1 --> (right hand side of the rule)
```

Similar processing takes place for detecting proteins, mutations, and other entities. The result of this stage is a set of named entities, which are, however, not yet normalized or grounded.

6.3 Normalization and Grounding

Normalization needs to decide on a canonical name for each entity, like a protein or an organism. Since the ontology encodes information about e.g. scientific names for organisms, a corresponding normalized entry can often be uniquely determined with a simple lookup. In case of abbreviations, however, finding the canonical name usually involves an additional disambiguation step.

For example, if we encounter *E. coli* in a text, it is first recognised as an organism from the pattern "species preceded by abbreviation." The NLP component can now query the ontology for a genus instance with a name matching `E*` and a species named `coli`, and filter the results for valid genus-species combinations denoting an existing organism. Ideally, this would yield the single combination of genus *Escherichia* and species *coli,* forming the correct organism name. However, the above query returns in fact four entries. Two can be discarded because their names are classified by NCBI as misspellings of *Escherichia coli*, as shown by the identical `tax_id` (cf. Table 13-2). Yet the two remaining combinations, with the names *Escherichia coli* and *Entamoeba coli,* are both classified as "scientific name." A disambiguation step now has to determine which one is the correct normalized form for *E. coli:* This is the task of coreference resolution covered in Section 6.4 below.

Once the normalized name (and thus the represented ontology instance) has been determined, in the case of organisms and proteins the corresponding database ID can be trivially retrieved from the instance, where it was stored as an OWL datatype property as described in Section 5.1. Since the database record can now be unambiguously looked up, the entity is grounded with respect to an external source. For our examples, these IDs are `P36217` for the xylanase variant shown in Figure 13-6, and `562` for E. coli, whose database entries are shown in Figure 13-2.

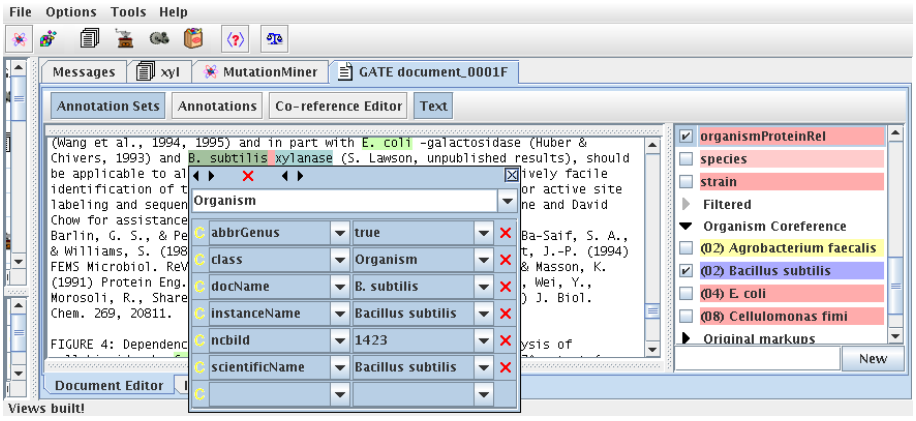

Figure 13-8. An organism annotation in GATE showing normalization and grounding of the textual entity *B. subtilis* to *Bacillus subtilis* with the NCBI database ID 1423

The end result of this step is a semantic annotation of the named entities as they appear in a text, which includes the detected information from normalization and grounding, as shown in Figure 13-8.

Mutation Normalization and Grounding. Mutation normalization and grounding exhibits some interesting additional properties. As mentioned in Section 5.4, protein mutations are first normalized to a single-letter format from their textual description, which can be easily achieved using the amino acid information stored in the ontology.

More involved is the grounding of a mutation with respect to its protein sequence. Using the already grounded protein information, an amino acid sequence is retrieved from *Entrez*[14] using eFetch[15] (see Figure 13-9). Mutated residues can then be located on the retrieved sequences and only those mutation/sequence combinations bearing the declared wild type residues at the specified coordinates with the correct offset between multiple mutations are eligible for subsequent processing. Single point mutations must match the amino acid at the designated coordinate exactly. Mutations detected in a text that cannot be grounded to its designated protein are discarded [53].

6.4 Coreference Resolution

Coreference resolution (see Section 4.1.3) is another important step in a text mining system, as its results, coreference chains, form the basis for many

[14]Entrez, `http://www.ncbi.nlm.nih.gov/gquery/gquery.fcgi`
[15]NCBI Entrez Programming Utilities (eUtils), `http://eutils.ncbi.nlm.nih.gov/entrez/query/static/eutils_help.html`

1: P36217. Reports Endo-1,4-beta-xyl...[gi:549461] BLink, Domains, Links

```
>gi|549461|sp|P36217|XYN2_TRIRE Endo-1,4-beta-xylanase 2 precursor
 (Xylanase 2) (1,4-beta-D-xylan xylanohydrolase 2)
MVSFTSLLAASPPSRASCRPAAEVESVAVEKRQTIQPGTGYNNGYFYSYWNDGHGGVTYTNGPGGQFSVN
WSNSGNFVGGKGWQPGTKNKVINFSGSYNPNGNSYLSVYGWSRNPLIEYYIVENFGTYNPSTGATKLGEV
TSDGSVYDIYRTQRVNQPSIIGTATFYQYWSVRRNHRSSGSVNTANHFNAWAQQGLTLGTMDYQIVAVEG
YFSSGSASITVS
```

Figure 13-9. Protein sequence data in FASTA format for *xylanase 2* retrieved from *Entrez* using the grounded protein entity P36217 obtained by NLP analysis

downstream analysis tasks. Mutation Miner, for example, needs to identify the *impact* of a certain enzyme mutation. This requires the identification of *all* mentions of a mutation throughout the text, in order to examine their context, thereby extracting and summarizing the impact descriptions.

While coreference resolution has been studied extensively in the general newspaper/newswire domain, the resolution of biological entities (nominal and pronominal) is a rather new area of research. Here, we only focus on the ontological extensions of coreference resolution, not the basic approaches covered in the literature [12, 21, 28, 50]. In our system, we employ a fuzzy-based coreference resolution strategy using a number of heuristics that can use the instantiated ontology as a knowledge source. For example, coreference between an organism entity in abbreviated and several candidates in non-abbreviated form (cf. the last section) can be resolved by examining their context and picking the closest one of the candidates that was previously mentioned in non-abbreviated form. Entities that have been successfully grounded can be unambiguously identified as being equal by comparing their unique database IDs recorded in the ontology and thusly grouped in a coreference chain.

A common problem during coreference analysis are ambiguities occurring at the linguistic level. Here, the ontology can facilitate disambiguation by allowing comparisons considering different hierarchy levels in the ontology. For example, the NCBI Taxonomy database records the "parent" for each species. Thus, when testing for coreferring entities of an organism classified as "species" in the taxonomic tree, not only other species but also all subspecies can be taken into account by retrieving their parent IDs and using them in the comparison. For the subspecies *Batis mixta mixta*, for instance, the hierarchical relationship to its parent species *Batis mixta* can be established without resorting to substring tests by comparing the parent ID of the subspecies with the species' ID.

An example for successful coreference resolution on organisms can be seen in Figure 13-10, which shows GrOWL[16] visualising a segment of the result ontology for a document, with ontology classes depicted by filled boxes and class instances

[16]GrOWL ontology visualiser, `http://ecoinformatics.uvm.edu/dmaps/growl`

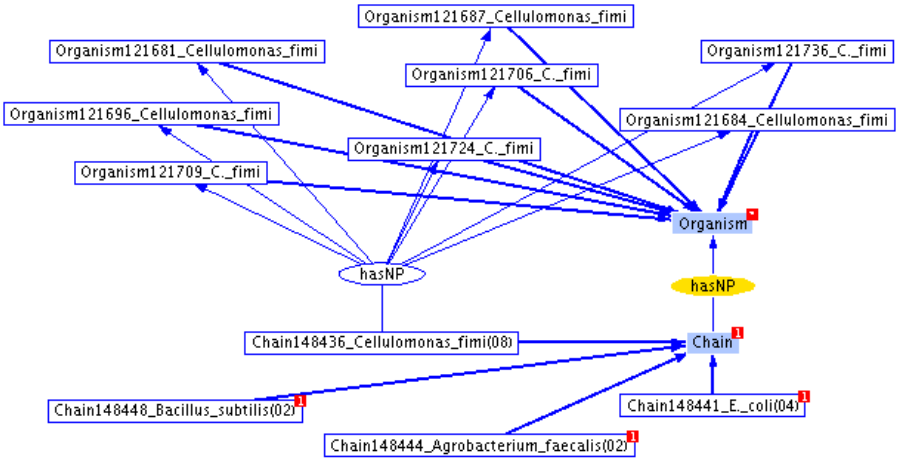

Figure 13-10. Organism coreference chains from the NLP-populated result ontology

by boxes with empty background. The *Chain* class representing coreference chains, here confined to Organism chains, is connected to its members by the object property `hasNP`. On the instance level, we see four chains, one for each organism found in the document. The chain for *Cellulomonas fimi* is expanded in the figure to show its eight members, which are instances of the Organism class.

6.5 Relation Detection

Relation detection, for example between organisms and proteins, requires more involved NLP analysis, like full or partial parsing for predicate-argument extraction [23, 31, 51].

A common problem in relation extraction is the high amount of ambiguity, especially when using full parsers [55]. Employing an ontology encoding semantically valid relations (Req. #5) allows to constrain the number of detected relation candidates to the semantically valid ones, which ideally results in a unique relation and otherwise boosts precision [30].

We give an example for detecting and disambiguating protein-organism relations, which is illustrated in Figure 13-11. Information from Swiss-Prot, including protein synonyms and taxonomic origin, is encoded in our ontology as detailed in Section 5.3. We can use this information to resolve ambiguous entities in a relation by discarding possible combinations that are not supported by the ontology, as each protein in Swiss-Prot is linked to its hosting organism via the latter's NCBI Taxonomy ID.

In the given example sentence, the phrase "*Bacillus subtilis* xylanase" refers to a protein of the Xylanase family. This can be automatically determined by the

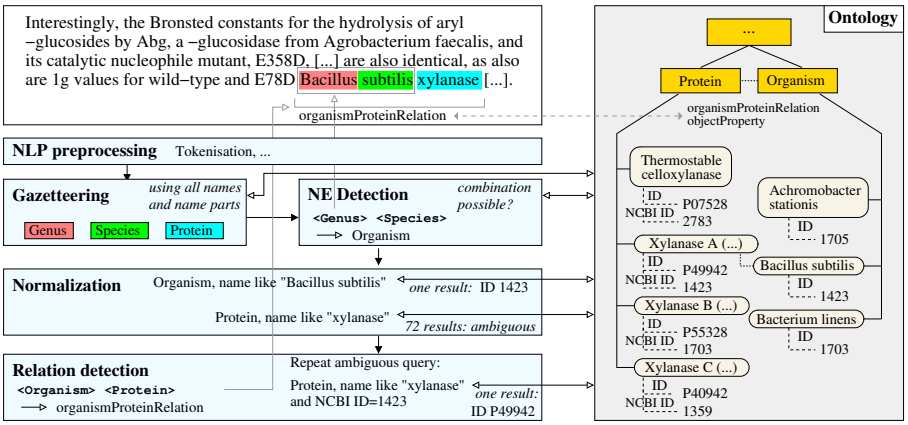

Figure 13-11. Protein disambiguation exploiting a detected relation

named entity detection (see Section 6.2), semantically annotating "xylanase" as `Protein` and "*Bacillus subtilis*" as `Organism`. But it is not yet clear which protein is meant precisely. As can be seen in Figure 13-6, canonical protein names can change according to the organism they have been generated from: *Xylanase 2* from *Trichoderma reesei* has the normalized name *Endo-1,4-beta-xylanase 2 [Precursor]* and a grounded ID in Swiss-Prot of P36217. Querying the ontology for proteins with "xylanase" in their name yields no less than 72 different proteins. However, in this example, *Bacillus subtilis,* which was tagged as organism by the NE component, can be unambiguously grounded, because it is a name occurring in the NCBI Taxonomy database, with the ID 1423 (see Figure 13-8).

So, the ontology query can be refined by including the organism's NCBI ID, which is used in Swiss-Prot to record the organism producing a protein. The resulting query for a protein named "∗xylanase∗" that is linked to the NCBI entry 1423 yields exactly one result, the correct protein *"Endo-1,4-beta-xylanase A precursor (EC 3.2.1.8) (Xylanase A) (1,4-beta-D-xylan xylanohydrolase A)."*

6.6 Exporting the Populated Ontology

Finally, the instances found in the document and the relations between them are exported to an OWL-DL ontology. Note that for the instances and relations available in the external databases, the result ontology is a subset of the one populated initially (cf. Figure 13-3).

In our implementation, ontology population is done by a custom GATE component, the *OwlExporter*, which is application domain-independent. It collects two special annotations, `OwlExportClass` and `OwlExportRelation`, which specify instances of classes and relations (i.e., object properties), respec-

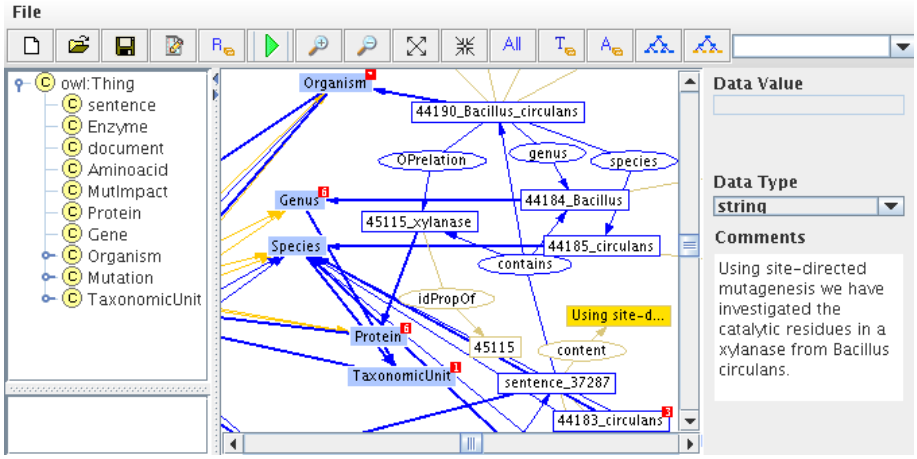

Figure 13-12. Mutation Miner ontology populated by NLP visualised in GrOWL

tively. These must in turn be created by application-specific components, since the decisions as to which annotations have to be exported, and what their OWL property values are, depend on the domain.

The class annotation carries the name of the class, a name for the instance like the Swiss-Prot official name for a protein, and the GATE internal ID of an annotation representing the instance in the document. If there are several occurrences of the same entity in the document, the final representation annotation is chosen from the ones in the coreference chain by the component creating the `OwlExportClass` annotation.

From the representative annotation, all further information is gathered. When it has read the class name, OwlExporter queries the ontology via Jena for the properties of the class and then looks for equally named features in the representation annotation, using their values to set the OWL properties.

The exported, populated ontology also contains document specific information; for example, for each class instance the sentence it was found in is recorded. Additional entity-specific information, like an automatically created summary for a mutation's impact, can also be exported.

Figure 13-12 shows an excerpt of such an ontology populated by Mutation Miner, visualised using GrOWL.

7. DISCUSSION

In this chapter, we motivated and illustrated the use of ontology within a text mining system from an NLP perspective. When deciding on whether to employ ontology technology in a (biological) text mining application, one needs to be clear about the motivation in order to properly assess its cost/benefit

ratio. Partly due to its novelty and complexity, semantic web technology still requires significant upfront investments before one can reap the benefits of their integration.

So what precisely are the benefits again? In Section 2 we discussed the various reasons for ontology integration. In short, exporting NLP into an OWL-DL ontology (ontology population) allows for standardised data exchange, which in particular includes reasoning tools that can be used to query the ontology, as shown in Figure 13-1. Using an ontology during NLP analysis allows one to consolidate the various resources, stored in different representational formats, into a single datastructure, thereby ensuring semantic integrity between the various analysis steps. In this case, however, ontology design needs to take the actual NLP analysis tasks into account, like named entity detection, normalization, entity grounding, coreference resolution, relation detection, and others. An ontology might be well-defined and instantiated, but lacking necessary relations, attributes, or other information to support those tasks, it will require expensive transformations or even a re-design before it can be used in a text mining system.

But we believe that the most interesting benefits will emerge when both approaches are combined in a unified, ontological NLP system. Tasks like normalization, relation detection, and coreference resolution can be seen as different facets of the same problem, namely, the construction of ontology concepts, instances, and relations. For example, every member of a coreference chain must be normalized and grounded to the same external protein instance, which in turn requires consistent relations between the chain members and other entities in a text. Inconsistencies, caused by e.g. a pronoun with an incompatible relation to another textual entity, would be immediately flagged by an automated reasoner. Thus, current algorithms for these tasks could be enhanced or replaced by new ones employing formal reasoning over the ontology. This is, however, an ongoing research target (with still diverging views [49]), requiring extensive re-design of existing NLP tools and algorithms, which is why we presented a more gentle, canonical extension of existing, standard NLP tasks in this chapter.

It is important to note that we covered only a single, very specific connection of ontology with biological text mining in this chapter. Other related work includes: Firstly, ontology learning, where NLP is used to determine potential classes and their relations from texts [10, 47]. However, at present these technologies are not capable of generating an ontology that would fulfill all the requirements we outlined in Section 4. Secondly, using text mining with existing ontologies, like the *Gene Ontology* (GO),[17] to annotate database entries with segments from the literature [11, 15, 48]. Recent work in this area has also been carried out within the *Critical Assessment for Information Extraction*

[17]The Gene Ontology, `http://www.geneontology.org/`

systems in Biology (BioCreAtIvE)[18] competition. Thirdly, information retrieval using ontologies that have been automatically linked to documents using NLP techniques. Examples for this category are systems like Textpresso [35] and GoPubMed [18]. And lastly, work concerning ontology proper, like ontology linking, merging, alignment, and ontology evaluation [47].

Note that we also did not discuss the *evaluation* of a text mining system [24, 25]. This is an issue largely orthogonal to ontology integration, since virtually all existing resources can, in a first step, be transformed from their ad-hoc representations into an ontology without impacting a system's performance. Ontological NLP, in this respect, addresses software engineering concerns of text mining systems—an issue for which computational linguists often seem to have little love left.

8. CONCLUSIONS

This chapter describes the combination of two still emerging technologies—Semantic Web Ontologies and Text Mining—for the biomedical domain. The integration can take several forms: Ontology-based NLP simply exports results by populating an ontology, using other resources for the actual processing. Ontology-driven NLP actively uses ontological resources for NLP tasks, which requires ontologies that hold all the information needed for the various language analysis algorithms. A combined approach—Ontological NLP—offers the most benefits, including semantic consistency within a text mining system and formal reasoning capabilities for querying NLP-populated ontologies.

We believe these advantages over ad-hoc NLP resource formats will lead to a rapid increase of ontology-enabled language tools, as well as ontologies encoding the necessary domain- and language-specific information. Frameworks like GATE already have basic ontology support; however, it will take much longer for individual NLP tools (like full or partial parsers, coreference resolution engines, word sense disambiguators) to adapt and make use of ontologies. This, in turn, requires more attention from the ontology community to recognize and deliver support for language analysis tasks.

The emergence of ontological NLP is also likely to give rise to an increase in the abundance of instantiated ontologies serving as knowledge bases. Having domain-specific text segments from the scientific literature available in a formal and interoperable format is consistent with the vision of the Semantic Web. Given that the scientific community can see beyond the challenges of new query tools and workflows for information retrieval, it is reasonable to expect that NLP techniques connected with ontologies will contribute significantly to the discovery processes in the life sciences.

[18] BioCreAtIvE, `http://biocreative.sourceforge.net/`

ACKNOWLEDGMENTS

The authors would like to thank Qiangqiang Li for implementing OWL-DL generation from GATE annotations and Vladislav Ryzhikov for his contributions to the Mutation Miner NLP subsystem.

REFERENCES

[1] Ananiadou S. and McNaught J., editors. *Text Mining for Biology and Biomedicine*. Artech House, 2006.

[2] Baader F., Calvanese D., McGuinness D.L., Nardi D., and Patel-Schneider P.F., editors. *The Description Logic Handbook: Theory, Implementation and Application*. Cambridge University Press, 2002.

[3] Bairoch A., Apweiler R., Wu C.H., Barker W.C., Boeckmann B., Ferro S., Gasteiger E., Huang H., Lopez R., Magrane M., Martin M.J., Natale D.A., O'Donovan C., Redaschi N., and Yeh L.S.L. The Universal Protein Resource (UniProt). *Nucleic Acids Research*, 2005.

[4] Baker C.J.O., Shaban-Nejad A., Su X., Haarslev V., and Butler G. Semantic Web Infrastructure for Fungal Enzyme Biotechnologists. *Journal of Web Semantics*, vol. 4(3), 2006. Special issue on Semantic Web for the Life Sciences.

[5] Baker C.J.O., Su X., Butler G., and Haarslev V. Ontoligent Interactive Query Tool. In M.T. Koné and D. Lemire, editors, *Canadian Semantic Web Series*, vol. 2 of *Semantic Web and Beyond*. Springer, 2006.

[6] Baker C.J.O. and Witte R. Mutation Mining—A Prospector's Tale. *Information Systems Frontiers (ISF)*, vol. 8(1):47–57, February 2006.

[7] Baker C.J.O., Witte R., Shaban-Nejad A., Butler G., and Haarslev V. The FungalWeb Ontology: Application Scenarios. In *Eighth Annual Bio-Ontologies Meeting*, pages 1–2. Detroit, Michigan, USA, June 24 2005.

[8] Bodenreider O. Lexical, Terminological, and Ontological Resources for Biological Text Mining. In Ananiadou and McNaught [1], chapter 3.

[9] Bontcheva K., Tablan V., Maynard D., and Cunningham H. Evolving GATE to Meet New Challenges in Language Engineering. *Natural Language Engineering*, 2004.

[10] Buitelaar P., Cimiano P., and Magnini B., editors. *Ontology Learning from Text: Methods, Evaluation and Applications*, vol. 123 of *Frontiers in Artificial Intelligence and Applications*. IOS Press, 2005.

[11] Camon E.B., Barrell D.G., Dimmer E.C., Lee V., Magrane M., Maslen J., Binns D., and Apweiler R. An evaluation of GO annotation retrieval for BioCreAtIvE and GOA. *BMC Bioinformatics*, vol. 6(Suppl 1), 2005.

[12] Castaño J., Zhang J., and Pustejovsky J. Anaphora Resolution in Biomedical Literature. In *International Symposium on Reference Resolution*. 2002.

[13] Chang J. and Schütze H. Abbreviations in Biomedical Text. In Ananiadou and McNaught [1], chapter 5.

[14] Cohen A.M. and Hersh W.R. A survey of current work in biomedical text mining. *Briefings in Bioinformatics*, vol. 6:57–71, 2005.

[15] Couto F.M., Silva M.J., and Coutinho P. ProFAL: PROtein Functional Annotation through Literature. In *VII Conference on Software Engineering and Databases (JISBD)*, pages 747–756. 2003.

[16] Cunningham H., Maynard D., Bontcheva K., and Tablan V. GATE: A framework and graphical development environment for robust NLP tools and applications. In *Proceedings of the 40th Anniversary Meeting of the ACL*. 2002. http://gate.ac.uk.

[17] Cunningham H., Maynard D., and Tablan V. JAPE: a Java Annotation Patterns Engine (Second Edition). Technical report, University of Sheffield, Department of Computer Science, 2000.

[18] Doms A. and Schroeder M. GoPubMed: Exploring PubMed with the GeneOntology. *Nucleic Acids Research*, vol. 33:W783–W786, 2005.

[19] Federhen S. The Taxonomy Project. In J. McEntyre and J. Ostell, editors, *The NCBI Handbook*, chapter 4. National Library of Medicine (US), National Center for Biotechnology Information, 2003.

[20] Gabdoulline R.R., Hoffmann R., Leitner F., and Wade R.C. ProSAT: functional annotation of protein 3D structures. *Bioinformatics*, vol. 19(13):1723–1725, 2003.

[21] Gasperin C. Semi-supervised anaphora resolution in biomedical texts. In *Proceedings of the HLT-NAACL Workshop on Linking Natural Language Processing and Biology (BioNLP)*. New York City, NY, USA, 2006.

[22] Haarslev V. and Möller R. RACER System Description. In *Proceedings of International Joint Conference on Automated Reasoning (IJCAR)*, pages 701–705. Springer-Verlag Berlin, Siena, Italy, June 18–23 2001.

[23] Hahn U. and Wermter J. Levels of Natural Language Processing for Text Mining. In Ananiadou and McNaught [1], chapter 2.

[24] Hirschman L. and Blaschke C. Evaluation of Text Mining in Biology. In Ananiadou and McNaught [1], chapter 9.

[25] Hirschman L., Yeh A., Blaschke C., and Valencia A. Overview of BioCreAtIvE: critical assessment of information extraction for biology. *BMC Bioinformatics*, vol. 6(Suppl 1), 2005.

[26] Horn F., Lau A.L., and Cohen F.E. Automated extraction of mutation data from the literature: application of MuteXt to G protein-coupled receptors and nuclear hormone receptors. *Bioinformatics*, vol. 20(4):557–568, 2004.

[27] Kawabata T., Ota M., and Nishikawa K. The protein mutant database. *Nucleic Acids Research*, vol. 27(1), 1999.

[28] Kim J.J. and Park J.C. BioAR: Anaphora Resolution for Relating Protein Names to Proteome Database Entries. In S. Harabagiu and D. Farwell, editors, *ACL 2004: Workshop on Reference Resolution and its Applications*, pages 79–86. Association for Computational Linguistics, Barcelona, Spain, 2004.

[29] Kiryakov A., Popov B., Terziev I., Manov D., and Ognyanoffe D. Semantic Annotation, Indexing, and Retrieval. *Journal of Web Semantics*, vol. 2(1), 2005.

[30] Leroy G. and Chen H. Genescene: An Ontology-enhanced Integration of Linguistic and Co-occurrence based Relations in Biomedical Texts. *Journal of the American Society for Information Systems and Technology (JASIST)*, vol. 56(5):457–468, March 2005.

[31] Leroy G., Chen H., and Martinez J.D. A shallow parser based on closed-class words to capture relations in biomedical text. *J. of Biomedical Informatics*, vol. 36:145–158, 2003.

[32] Li Y., Bontcheva K., and Cunningham H. Using Uneven Margins SVM and Perceptron for Information Extraction. In *Proceedings of Ninth Conference on Computational Natural Language Learning (CoNLL)*. 2005.

[33] Manning C.D. and Schütze H. *Foundations of Statistical Natural Language Processing*. The MIT Press, 1999.

[34] McNaught J. and Black W.J. Information Extraction. In Ananiadou and McNaught [1], chapter 7.

[35] Müller H.M., Kenny E.E., and Sternberg P.W. Textpresso: An Ontology-Based Information Retrieval and Extraction System for Biological Literature. *PLoS Biology*, vol. 2(11):1984–1998, November 2004.

[36] Niles I. and Pease A. Towards a Standard Upper Ontology. In C. Welty and B. Smith, editors, *Proceedings of the 2nd International Conference on Formal Ontology in Information Systems (FOIS)*. Ogunquit, Maine, 2001.

[37] Park J.C. and Kim J.J. Named Entity Recognition. In Ananiadou and McNaught [1], chapter 6.

[38] Pearson W.R. and Lipman D.J. Improved tools for biological sequence comparison. *Proceedings of the National Academy of Sciences of the USA*, vol. 85(8):2444–2448, April 1988.

[39] Popov B., Kiryakov A., Ognyanoff D., Manov D., Kirilov A., and Goranov M. Towards Semantic Web Information Extraction. In *Human Language Technologies Workshop at the 2nd International Semantic Web Conference (ISWC)*. Sanibel Island, Florida, USA, October 20 2003.

[40] Rebholz-Schuhmann D., Kirsch H., and Couto F. Facts from Text—Is Text Mining Ready to Deliver? *PLoS Biology*, vol. 3:188–191, 2005.

[41] Rebholz-Schuhmann D., Marcel S., Albert S., Tolle R., Casari G., and Kirsch H. Automatic extraction of mutations from Medline and cross-validation with OMIM. *Nucleic Acids Research*, vol. 32(1):135–142, 2004.

[42] Roche E. and Schabes Y., editors. *Finite-State Language Processing*. MIT Press, 1997.

[43] Schuman J. and Bergler S. Postnominal prepositional attachment in proteomics. In *Proceedings of the HLT-NAACL Workshop on Linking Natural Language Processing and Biology (BioNLP)*. New York City, NY, USA, 2006.

[44] Shaban-Nejad A., Baker C.J.O., Haarslev V., and Butler G. The FungalWeb Ontology: Semantic Web Challenges in Bioinformatics and Genomics. In *Springer LNCS 3729*, pages 1063–1066. 2005.

[45] Smith M.K., Welty C., and McGuinness D.L., editors. *OWL Web Ontology Language Guide*. World Wide Web Consortium, 2004. http://www.w3.org/TR/owl-guide/.

[46] Spasic I., Ananiadou S., McNaught J., and Kumar A. Text mining and ontologies in biomedicine: making sense of raw text. *Briefings in Bioinformatics*, vol. 6, 2005.

[47] Staab S. and Studer R., editors. *Handbook on Ontologies*. Springer, 2004.

[48] Stoica E. and Hearst M. Predicting Gene Functions from Text Using a Cross-Species Approach. In *Pacific Symposium on Biocomputing (PSB)*, pages 88–99. 2006.

[49] Tsujii J. and Ananiadou S. Thesaurus or logical ontology, which one do we need for text mining? *Language Resources and Evaluation*, vol. 39(1):77–90, 2005.

[50] Vlachos A., Gasperin C., Lewin I., and Briscoe T. Bootstrapping the Recognition and Anaphoric Linking of Named Entities in Drosophila Articles. In *Pacific Symposium on Biocomputing*, pages 100–111. 2006.

[51] Wattarujeekrit T., Shah P.K., and Collier N. PASBio: predicate-argument structures for event extraction in molecular biology. *BioMed Central Bioinformatics*, vol. 5(155), 2004.

[52] Wessel M. and Möller R. High Performance Semantic Web Query Answering Engine. In *International Workshop on Description Logics (DL)*. Edinburgh, Scotland, UK, 2005.

[53] Witte R. and Baker C.J.O. Combining Biological Databases and Text Mining to support New Bioinformatics Applications. In *10th International Conference on Applications of Natural Language to Information Systems (NLDB)*, vol. 3513 of *LNCS*, pages 310–321. Springer, Alicante, Spain, June 15–17 2005.

[54] Wood M.M., Lydon S.J., Tablan V., Maynard D., and Cunningham H. Populating a Database from Parallel Texts Using Ontology-Based Information Extraction. In *9th International Conference on Applications of Natural Language to Information Systems (NLDB)*, vol. 3136 of *LNCS*. Springer, 2004.

[55] Yakushiji A., Tateisi Y., Miyao Y., and Tsujii J. Event extraction from biomedical papers using a full parser. In *Proceedings of the 6th Pacific Symposium on BioComputing (PSB)*, pages 408–419. Hawaii, USA, January 2001.