Mining biological parameters from literature: an application to environmental decision support systems

André Fernandes dos Santos
(**andrefs@cpan.org**)

Dissertation submitted in partial fulfillment of the requirements for the degree of Master in Bioinformatics at the University of Minho, under the supervision of
Anália Maria Garcia Lourenço
Regina Maria de Oliveira Barros Nogueira

Departamento de Engenharia Biológica
Escola de Engenharia
Universidade do Minho

Braga, October 2012

Acknowledgments

- First of all, many thanks to my supervisors, Anália Lourenço and Regina Nogueira, for the guidance, the encouragement and specially for putting up with my chaotic work methodologies.
- Thanks again to Anália for the proof-reading, and for applying the right amount of pressure whenever I needed the most.
- Thanks again to Regina for being able to perform an exhaustive annotation work while maintaining the enthusiasm for the text mining domain.
- Thanks to Nuno Carvalho, Alberto Simões and José João for always being available to discuss ideas and answering my doubts.
- Thanks to everyone at GroupBuddies for understanding my constant reduced availability during the last year.
- Thanks to Clinton Gormley for the advices and the quick bug fixes.
- Thanks to my parents and my brother for supporting me in this additional entreprise – no more master's, I promise!
- And, yet again, a very special thanks to Dominique, who somehow also thought this was a good idea :)

Preface

This document is a master thesis in Bioinformatics (area of Biomedical Text Mining) submitted to Universidade do Minho, Braga, Portugal.

Throughout the document the academic plural appears often in the text to describe the work developed. This form was intentionally used for two reasons: first, some of the work here presented was done in cooperation; and second, the plural can help the reader to feel more closely connected with the work done.

Document structure

Chapter 1: introduces the subject of this dissertation, defining the basic concepts and ideas developed throughout the document.

Chapter 2: presents some background on the concepts on which this dissertation is based, along with a brief overview of the state-of-the-art concerning these subjects.

Chapter 3: describes the problem of recognizing numerical parameters in scientific text, possible strategies and its main limitations.

Chapter 4: details the implementation of a text mining framework for performing extraction of numerical parameters scientific texts.

Chapter 5: relates the application of the framework to the wastewater domain, describing the creation of a domain-specific ontology and the results obtained.

Chapter 6: presents some conclusions and points out possible directions for future work.

Some complementary information is presented on the appendixes:

Appendix A: includes the documentation for some of the tools developed in the context of this thesis (also available as `man` pages).

Contents

List of Figures

List of Tables

List of Examples

Acronyms

API	Application Programming Interface
CSV	comma-separated values
EMBL-EBI	European Molecular Biology Laboratory-European Bioinformatics Institute
GUI	Graphical User Interface
GPL	General Purpose Language
HTML	HyperText Markup Language
IT	Information Technology
JSON	Javascript Object Notation
MAS	Microsoft Academic Search
MEDLINE	Medical Literature Analysis and Retrieval System Online
NLP	Natural Language Processing
NER	Named Entity Recognition
OA	Open Access
OASIS	Organization for the Advancement of Structured Information Standards
OO	Object-Oriented
OBO	Open Biomedical Ontologies
OCR	Optical Character Recognition
ODF	Open Document Format
OOXML	Office Open XML
PDF	Portable Document Format

POS	Part-of-Speech
PMC	PubMed Central
PMC-OAS	PubMed Central Open Access Subset
RE	Relationship Extraction
SV	Scalar Value
UCS	Universal Character Set
UTF-8	UCS Transformation Format – 8-bit
UKPMC	UK PubMed Central
WWTP	Waste Water Treatment Plant
XML	Extensible Markup Language
XSLT	Extensible Stylesheet Language Transformations
YAML	YAML Ain't Markup Language

Chapter 1

Introduction

Recognizing spelled-out numbers is a language-dependent task. Given the importance and ubiquity of the English language in scientific communication [Kaplan, 2001, Ammon, 2001], ,

1.1 Context

Waste water treatment is a traditional area of intervention of civil engineers. For decades, Waste Water Treatment Plants (WWTPs) were dimensioned and operated based on empiric criteria and experience gained by practice. Many of these plants work seamlessly, while others face severe problems of sludge sedimentation, due to excessive growth of filamentous microorganisms, and poor removal of nutrients, namely nitrogen and phosphor.

Until the eighties, methods used to study biocenosis in WWTPs were based on the culture of microorganisms in laboratories. As such, obtaining results was a slow process. These methods were not used to diagnose cases of sudden malfunction in WWTPs because by the time the results were available, the problem would already have been solved by itself or reached proportions beyond any hope of solution.

Additionally, these methods are not "unbiased", that is, they encourage the growth of the microorganisms which better adapt to the medium, and, as such, they introduce significant deviations in the results obtained. An example of this situation which is still described in several books is the oxidation of the nitrite ion to nitrate ion. This biological reaction was thought to be caused by bacteria of the genus Nitrobacter. In recent years, molecular biology methods used in the detection and identification of microorganisms in waste waters allowed to expand the knowledge about the diversity of microorganisms present in waste waters. The results obtained by such techniques have shown that in most WWTPs the oxidation of the nitrite ion is performed by bacteria of the Nitrospira genus.

Currently, thanks to the molecular biology methods used in the study of

biocenosis in WWTPs, we know much more about waste water microbiology; however, we still do not know how to put that knowledge to use in the optimization of the operation and performance of WWTPs, and to streamline the engineers decision processes. To help solving this issue, this project proposes the creation of a communication channel between waste water scientists and engineers. More specifically, this project aims to develop a computational approach to modeling possible relations between the microorganism populations and the operation/performance of waste water treatment processes. The methodology for this project comprises three main steps:

1. automatic retrieval and processing of the available literature, and the intersection of these references with the contents of public databases;

2. modeling and statistical analysis of the relations automatically extracted using supervised data mining techniques; and

3. validation of the statistically significant relations by performing new laboratorial experiments on the same biological systems and keeping the same operation and treatment.

1.2 Motivation

The main motivation for this project is the creation of a communication channel suited to the transmission of knowledge between the distinct participants in the process of configuration and optimization of engineering systems. Particularly, we intend to develop a computational infrastructure capable of gathering and modeling relevant information from microbiological studies to configure and adapt WWTPs based on their environmental impact.

The novelty of this work lies in the scientific literature mining, and its integration in computational platforms for monitoring and control WWTPs systems. To date, there have been no efforts to compile, centralize and organize the enormous amount of information spread throughout the scientific literature, namely the results obtained from microbiological studies of different types of WWTPs. Literature mining can provide a valuable contribute, allowing to create a base of knowledge with immediate and obvious usefulness to biologists which address this domain. Furthermore, the possibility of indexing and examining automatically and under different perspectives microbiological parameters allows the application of supervised and unsupervised machine learning techniques to the inference of non-trivial knowledge about WWTP operability in regions with different microbiotical characteristics.

1.3 Objectives

This master dissertation is focused on scientific literature mining to automatically extract and process statistically significant microbiological relationships.

Specifically, the goals defined for this dissertation are:

1. gathering relevant scientific literature (i.e. scientific articles describing microbiological studies related to WWTPs);

2. semantic annotating the set of articles selected, identifying the information elements which are relevant to the characterization of this type of studies and, consequently, are to be extracted automatically;

3. building/adapting a controlled vocabulary on waste water treatment systems to ensure a semantic characterization which is comprehensive, scalable and virtually connectable to domain specific repositories;

4. designing and implementing an automatic system for extracting from scientific articles information comprising:

   (a) automatic recognition of relevant entities such as microorganisms, WWTPs typology and molecular methods supporting the studies;

   (b) automatic extraction/inference of biological parameters, particularly the name of the parameter and the related quantitative expression and unit of measure;

   (c) evaluation of content (relation) similarity between articles, and supervised learning of relevant relationships (or possibly, unsupervised, depending on the typology of the distinct data sets);

5. validating the system described above by using a specific type of WWTPs as case study and recurring to manual curation of experts, biologists and engineers which work directly with that type of WWTPs;

6. making freely available the base of knowledge built and validated, and making the project available through a web interface.

The next section presents the work developed in the context of this dissertation, stating some design goals which guided the project development and briefly presenting the tools created. Section 1.5 presents the structure of the document, including a summary of each chapter.

1.4 Project overview

1.4.1 Design Goals

A set of design goals was defined to guide implementation efforts. Some of these are practical requirements for the tools, while others can be viewed more as philosophical guidelines.

- Despite having the biomedical (and, more specifically, the ecotechnologies) as central concern, tools should be developed as generic as possible, because many of the problems they tackle are common to other fields.
- Develop tools which are independent from each other, but which can work together and with previously existing applications, and that can be assembled into a pipeline.
- Make the tools publicly available. This includes implementing them following the real-world standards and community practices, and making them installable and usable by third-party people and projects. This requires an extra effort given the difference between developing academic tools and real-world, usable applications.
- Avoid drowning users in data, presenting intelligent outputs instead which represent higher levels of analysis, preferably in intuitive and easy to understand formats.
- Output results in standard and widely used formats to facilitate their use in other pipelines.

1.4.2 Developed tools

This section presents and briefly describes the tools developed in the context of this master thesis, which include:

- A paragraph splitter
- A sentence splitter
- A text tokenizer
- A numerical expression extractor

The tools were developed according to what was most needed at the time. As such, in this document they will not be presented necessarily in the chronological order in which they were created, but instead in a way that helps getting a global view of how they can fit together.

1.4.2.1 Paragraph splitter

A Perl module for splitting text paragraphs. It performs the splitting based on the detection of several paragraph clues, and is highly configurable, allowing to define the weight of each clue and the confirmation threshold.

1.4.2.2 Sentence splitter

A rewrite of an existing Perl module for sentence splitting, adapting it to output the locations of the sentence boundaries instead of returning the original text split.

1.4.2.3 Text tokenizer

A rewrite of an existing Perl module for text tokenizing, adapting it to output the locations of the word boundaries instead of returning the original text tokenized.

1.4.2.4 Numerical expression extractor

A Perl module for extracting numerical expressions from plain text, capable of dealing with noisy text and conflicting or lack of a defined notation.

1.5 Document Summary

1.5.0.5 Extraction and Information Processing of Scientific Articles

The next chapter presents some background information and the state of the art in the areas of document manipulation and structure extraction, Named Entity Recognition (NER), Relationship Extraction (RE), ontologies, tool development and results visualization.

1.5.0.6 Recognition and processing of numerical parameters

This chapter introduces the problem of recognizing numerical parameters in scientific text, presenting limitations and proposing solutions for the different sub-tasks, and describes the implementation of a recognizer as a Perl module, `Text::Math::NumExp`.

1.5.0.7 A framework for the extraction of numerical parameters from scientific literature

This chapter presents the implementation of `T-Eng`, a modular text mining framework for performing the extraction of numerical parameters and other

relevant entities from scientific texts, comprising modules for building a customized workflow from the document importing to results visualization.

1.5.0.8 A case study: the ecotechnologies domain

This chapter introduces the necessity for automated methods for numerical parameter extraction in the wastewater domain, and relates the application of the framework in this domain, describing the development of a domain-specific ontology, presenting output samples and analyzing the results obtained.

1.5.0.9 Conclusions and future work

This chapter presents a review over the work performed, the problems tackled and the tools implemented. Several possible future work directions are described.

Chapter 2

Extraction and Information Processing of Scientific Articles

This chapter presents several distinct aspects related to the extraction of information from scientific literature. Some properties of the documents and techniques to handle them are described, followed by an introduction to NER and RE. The particular case of extraction numerical parameters from text is analyzed, and the usual methods for evaluating artificial intelligence tools are explained. The chapter ends with some remarks about forms of representing the information extracted from scientific articles.

2.1 Document manipulation

Several pre-processing tasks need to be performed in order to extract information from text documents. These tasks include finding the desired documents, retrieving them in a convenient format, and converting, adapting and cleaning them to make them more suitable to be automatically processed.

This section describes pre-processing tasks which are commonly performed in an information extraction pipeline. Several sources for scientific articles, formats in which they are available, and conversion tools are also presented.

Online repositories and download of electronic versions of scientific articles will be assumed as the main forms of obtaining documents, given the central role the Internet as assumed lately in scientific communication.

Different types of text documents (e.g. scientific articles, literary works, technical manuals) share some similarities in the methods used to process them in order to extract information. However, scientific articles present specific features, such as the average number of pages or the notion of document structure. Given that this work is focused mainly on scientific literature, the terms document, article and paper will be used interchangeably.

2.1.1 Document sources

The first task to be performed is the retrieval of documents of interest. Currently, the most used source of documents is, by far, the Internet. Several websites are specialized in making available scientific literature.

2.1.1.1 CiteSeer^X – `citeseer.ist.psu.edu`

CiteSeer was a search engine and a digital library for scientific articles, focused mainly on computer sciences and Information Technology (IT). Its successor, CiteSeer^X [Li et al., 2006b,a], incorporates more advanced features such as automatic citation indexing and reference linking, full-text indexing and complex search queries. It is being developed as an open source project[1], based on Apache Solr[Apache] and other open source tools.

2.1.1.2 GetCITED – `www.getcited.org`

GetCITED is a online database for scientific literature which includes articles and other types of publications (book chapters, conference papers, working papers, reports, etc), both peer-reviewed and not. Its contents are added and edited by members. It includes additional features such as a discussion forum, and it currently indexes over 3,400,000 publications [GetCITED, 2012].

2.1.1.3 Scirus – `www.scirus.com`

Scirus is a wide-spectrum scientific search-engine (not focused in one particular field of studies), whose results include non full text entries. Scirus is owned by Elsevier [Medeiros, 2002, McKiernan, 2005, Elsevier, 2004].

2.1.1.4 Microsoft Academic Search (MAS) – `academic.research.microsoft.com`

An academic search engine developed by Microsoft Research, MAS presents several interesting features, such as advanced visualization options and citation indexing.

2.1.1.5 UK PubMed Central (UKPMC) – `ukpmc.ac.uk`

UKPMC was initially created as a mirror of PubMed Central (PMC), but has since developed some different features. It is managed by the European Molecular Biology Laboratory-European Bioinformatics Institute (EMBL-EBI) and provides free access to more than 2 million full-text biomedical research articles[McEntyre et al., 2011].

[1] The source code for CiteSeer^X is available at `citeseerx.sourceforge.net`.

2.1.1.6 Google Scholar – `scholar.google.com`

Google Scholar is an online search engine for scientific literature [Jacsó, 2008, 2005, Cecchino, 2010]. It is capable of finding multiple entries for the same article (and even providing links validated to the users institution subscription), and allows to import the results in BibTEX format, among other features. However, Google Scholar does not provide an Application Programming Interface (API) for automated search and retrieval of articles or articles metadata, and apparently, there are no plans to do so in a near future [Code, 2012].

2.1.1.7 PubMed – `www.ncbi.nlm.nih.gov/pubmed`

It is a free database for literature on life sciences and biomedical topics which provides access mainly to the Medical Literature Analysis and Retrieval System Online (MEDLINE) [Greenhalgh, 1997] bibliographic database. PMC[2] is a subset of PubMed comprised only by full-text scientific literature in biomedical and life sciences. Articles from PMC are freely available.

PubMed is also the most frequently used source of articles for test and evaluation of biomedical text mining applications. It makes available tools for bulk retrieval and specialized search filters, among other features.

In addition to the article sources previously described, several other are available. Detailed reviews and comparisons of academic literature databases and repositories can be found in Giustini and Barsky [2005], Felter [2005], Anders and Evans [2010]

2.1.2 Document retrieval

Documents to be processed can be retrieved either one by one, possibly from distinct sources. Alternatively, some repositories allow to download entire collections at one time.

2.1.2.1 Single articles

Single articles can be downloaded from any of the previously mentioned sources and many others existing online. Despite the existence of initiatives such as Open Access [Suber, 2009], in which the costs of publishing the papers are supported by the authors and/or the publishers, it is common to find articles which are only available upon payment.

Downloading individual papers allows to carefully select which papers to include in the information extraction pipeline, but it also a time-consuming

[2] PubMed Central – `http://www.ncbi.nlm.nih.gov/pmc`

task, which typically involves searching and downloading from different sources and possibly getting documents in different formats.

2.1.2.2 Batch and bulk download

Some repositories allow to download entire sets of papers (for example, PubMed allows to download the subset of open access articles [NCBI, 2012a,b]). This allows to easily build a large-sized corpus which would be impossible when downloading single documents. The downside of this approach is the possibility of inclusion of articles which are less relevant for the task at hands.

Most pipelines for curation of articles (and specially those whose goal is to feed databases or collaborative annotation projects) tend to prefer the bulk download alternative, and using broad search filters. The filtering of the false positives is then performed only after the retrieval, reducing the number of calls to webservices (and having to face possible slow connections, time-outs, etc).

2.1.3 Document formats

Most text processing tools (such as stemmers, taggers, NER recognizers, etc) are only able to process plain text files, which means that documents in other formats need to be converted before being processed. Depending on the file format and the conversion tool used, additional metadata can also be extracted.

Tools which are capable of handling non-plain text documents usually come bundled with conversion libraries or third party applications, which are used to obtain the plain text documents.

2.1.3.1 Portable Document Format (PDF)

PDF is a file format developed by Adobe Systems. Initially developed as a proprietary format, it has evolved into an open standard [Adobe Systems Inc., 2008]. In 2010, Adobe claimed that more that 200 million PDF documents had been posted on the web [Adobe Systems Inc., 2010], which makes it one of the main formats for document exchange. The format itself has evolved and is now capable of supporting both vector and raster graphics, forms, attachments and other advanced features.

2.1.3.2 Microsoft Word

It is a is a text processor developed by Microsoft. Despite being proprietary software, it is commonly used to produce text documents. Despite the most recent versions being able to export documents to other formats (for example, PDF), the default file format for documents created with Microsoft

Word is `docx`, which is part of the Office Open XML (OOXML), a Extensible Markup Language (XML) specification for office documents created by Microsoft [Paoli et al., 2006]. Older versions of Microsoft Word use `doc` files.

2.1.3.3 Open Document Format (ODF)

The ODF is a XML-based file format for documents such as word processing, spreadsheets, presentations, etc. It was developed by a technical committee in the Organization for the Advancement of Structured Information Standards (OASIS), and has been published as an ISO/IEC international standard [ISO, 2006].

2.1.3.4 Extensible Markup Language (XML)

XML is a markup language which defines rules for writing documents which are both human and machine readable [Bray et al., 2000]. XML schemas can be defined which impose additional constraints to the format, allowing the definition of XML-based languages. The previously mentioned OOXML and ODF formats are both XML-based.

Additionally, there are several XML schemas commonly used to represent scientific articles. PubMed Central offers documents in NXML format, defined by the National Library of Medicine Journal Archiving and Interchange DTD [Rosenblum, 2010].

2.1.4 Document conversion methods and tools

2.1.4.1 Optical Character Recognition (OCR)

When the desired documents cannot be found online, and only a printed version can be accessed, they need to be converted to digital text. This may be performed by manually typing the documents (which is hardly possible when the set of documents to be typed is large and will always be prone to errors), or by using an OCR tool.

OCR tools are available in several different prices and complexity levels. Examples of OCR tools are OCROpus [Breuel, 2008], Tesseract [Smith, 2007] and GOCR [Schulenburg, 2004]. Comparative analysis of OCR tools can be found in Lilleleht [2011], Gohr [2010], de Mello and Lins [1999]

2.1.4.2 PDF to text

There are several tools available which perform the conversion of PDF files to plain text. The main difficulty while converting PDF documents to plain text is that in PDF documents there is a poor notion of document structure – text elements are positioned in pages through graphical directives which

specify the horizontal and vertical offset, and the font to be used. As such, most conversion tools have a hard time determining the order of the textual elements, and are easily confused by multiple columns, headers and footers, etc.

Examples of PDF conversion tools are `pdftotext`, based on Xpdf [Noonburg, 2001], and Apache Tika [Mattmann and Zitting, 2011, Noonburg, 2001]. An in-depth review of several PDF to plain text converters can be found in [Robinson, 2001].

2.1.4.3 XML to text

The conversion of XML documents to plain text depends on the specific XML schema used. Nevertheless, after identifying the XML elements of interest, a generic XML parser can be more or less easily implemented to perform the conversion. Extensible Stylesheet Language Transformations (XSLT) is a declarative language used for transforming XML documents. Most General Purpose Languages (GPLs) have their own XML libraries – Perl, for example, has XML::DT [Simões, 2012b] and XML::Twig [Rodriguez, 2012], among many others.

2.1.4.4 Conversion of other formats

Despite the previously mentioned formats being the most common ones, as long as a conversion tool to plain text exists, almost any text file format can be processed. The main difference between file formats is the amount of metadata which can be extracted, and how rich is the notion of document structure (if any).

2.2 Document structure

The identification of the different sections in a scientific article allows to process those sections differently, in order to achieve different results. For example, information retrieval tends to rely on abstracts (i.e. the summary of the meaningful contents of the article as described by the authors), whereas information extraction often targets sections such as Materials and Methods and Results (i.e. sections that are considered to be focused on given topics of interest, such as the experimental methods used, the environmental conditions of the study, or the results obtained in the study) [Agarwal and Yu, 2009].

2.2.1 Common sections

The basic structure of a scientific paper is often summarized as Inroad: introduction, methods, results and discussion [Peh and Ng, 2008]. This structure was first used in the 1940s, adopted as a majority in the 1970s and is now

the standard for publication [Sollaci and Pereira, 2004]. Often, this structure transpires to the section names: papers with sections named Introduction, Methods, Results and Discussion are a common occurrence. A quick research has revealed that other common section names a include Background, Context, Motivation, Evaluation, Conclusion(s) and so forth.

These common section names can be used to identify and delimit the sections of an article. Then, the relevant sections can be processed, and, furthermore, specific processing algorithms can be applied to specific sections.

2.2.2 Recognition of other sections

Nevertheless, not all article sections belong to a well-defined set, and many sections or subsections have a name related to the subject being discussed. In this cases, other means have to be used to discover the structure of the article.

If the article was retrieved in a XML format, it may be possible to extract the table of contents, as XML schemas such as the previously mentioned NXML maintain the article structure. On the other hand, the Portable Document Format specification support a *Document outline*, which can be extracted with the help of tools such as PDFMiner [Shinyama, 2010].

2.3 Recognition of concepts of interest

The recognition of concepts of interest in a text is a central task in many information extraction applications. Also know as Named Entity Recognition (NER), its goal is to identify all the instances of a name for a given type of entity within a given set of texts, and it is a starting point for further extraction of relationships. Due to its potential utility and complexity, this field has been extensively studied.

A simple example of NER is the identification of enzyme references, taking free text as input, and outputting a score designating the confidence of a word or phrase belonging to a given type, or a set of tags assigning a predicted type to each word or phrase of interest (tagging can be used to mark grammatical function, as in part-of-speech, or may be used to identify biological membership, e.g, annotate a given entity as being a gene reference).

There are several aspects which increase the complexity of this task [Cohen and Hersh, 2005]:

- the non existence of complete dictionaries for most types of biological named entities;
- the interpretation of many words or phrases interpretation depends on the context in which their are found;

- many biological entities may be referred to by multiple names (synonymy), and the same name may apply to a number of entities in different contexts (homonymy).

2.3.1 NER approaches

Different types of approaches are usually considered, the most common being lexicon-based, rules-based and statistically based.

The lexicon-based (or dictionary) approach requires the candidate words to have a match in a previously built list. This approach may be improved by using techniques such as stemming (stripping off both inflectional and derivational affixes) or fuzzy matching (see Section 2.3.3.2) the candidates, or the list itself may be increased using machine learning algorithms [Nadeau and Sekine, 2007, Mansouri et al., 2008].

The rules-based approach defines patterns which candidate words must comply with to be considered entities, and additional rules may be used to separate different classes. These rules may be defined by experts or may be learned by the computer through the analysis of manually annotated corpora.

The statistical (or classification) approach works by transforming the NER problem into a classification one. The computer is provided with a set of manually annotated documents, which are analysed with machine learning techniques, creating a classifier based on what was learned.

Combined approaches have also been used [Hettne et al., 2010]. Further types of approaches are described in [Leser and Hakenberg, 2005].

Currently, NER tools often use several of this techniques, combining them in a more complex analysis which tries to take advantage of the best features of each approach. Typically pre-processing prepares texts, splitting sentences with Natural Language Processing (NLP) methods and Part-of-Speech (POS) tagging and using external tools to add annotations. Later, machine learning techniques are used to tag biological entities and rules are applied to disambiguation [Leser and Hakenberg, 2005].

2.3.2 Existing NER tools

Given the interest of the community on this field, currently there are a number of biomedical NER tools available.

2.3.2.1 ABNER

ABNER (A Biomedical Named Entity Recognizer) is an open source application for molecular biology text mining, based on a machine learning system using conditional random fields with a variety of orthographic and contextual features [Settles, 2005].

In addition to its Graphical User Interface (GUI), this tool also provides a Java API which allows its integration with third party tools. ABNER is capable of recognizing entities such as proteins, DNA, RNA, cell lines, cell types and genes, and it has achieved an overall performance of 0.72, a recall of 0.69 and an F-score of 0.70.

2.3.2.2 OSCAR4

OSCAR4 is a robust open source system for the automated annotation of chemistry-related terms in scientific literature. The results of processing a document is a XML annotated version of the document. This feature, in addition to the OSCAR's modular architecture, eases its integration with other systems.

OSCAR is able to annotate chemical names, including enzymes, chemical compounds, reaction terminology and generic chemical terms [Jessop et al., 2011, Corbett and Murray-Rust, 2006, Batchelor and Corbett, 2007], and has achieved performance, recall and F-score values of 0.70, 0.81 and 0.75.

2.3.2.3 Linnaeus

Linnaeus is an open-source software tool for recognizing and normalizing species names which uses a dictionary-based approach implemented as a finite- state automaton [Gerner et al., 2010].

Linnaeus uses the NCBI Taxonomy [Wheeler et al., 2007, Benson et al., 1997] for the normalization of species names. It comes bundled with a default dictionary file (which contains the 10,000 most frequently occurring species in MEDLINE), but it allows to use a custom file. Linnaeus has a performance value of 0.97 and a recall of 0.94.

2.3.2.4 OrganismTagger

OrganismTagger is a hybrid rule-based/machine learning system, based on the GATE language engineering framework [Cunningham et al., 2002, Bontcheva et al., 2004], capable of extracting organism mentions from literature, normalizing them, and linking them to the respective NCBI database ID.

OrganismTagger has been evaluated with the help of a manually annotated corpus, revealing a precision of 0.95 and a recall of 0.94. Slightly higher values have been achieved when evaluating with the same corpus as Linnaeus.

2.3.3 Common optimization techniques

2.3.3.1 Stemming

One of the main obstacles to entity recognition is word inflection: the modification of a term depending on the context in which it is being used (tense, number, gender, ...). One possible approach to mitigate this problem is to perform a technique called stemming, which consists of reducing each word in the corpus to its stem or root. This way, when looking for the named entities, the match is performed against the stemmed version of each term, reducing the entropy of the process. There are multiple methods and variants of stemming [Frakes, 1992], which have different levels of impact on the precision and recall values [Kraaij and Pohlmann, 1996, Paice, 1994].

2.3.3.2 Fuzzy matching

Fuzzy matching consists on the possibility of allowing non-perfect matches. In this specific context of matching text elements, this technique is also known as approximate string matching [Chaudhuri et al., 2003, Navarro, 2001]. Allowing approximate string matching means accepting terms found in the corpus which are, to a given degree, similar to the ones being looked for. This similarity can be measured in several different ways, but is frequently related to some kind of string edit distance, such as the Levenshtein distance [Levenshtein, 1966], the Hamming distance [Hamming, 1950] or the Jaro–Winkler distance.

2.3.4 Domain ontologies

2.3.4.1 What is an ontology?

In a broader sense, an ontology may be defined as a shared understanding of some domain of interest, to be used as a unifying framework to solve problems from that domain. In the context of information science, an ontology is perceived as:

Definition 1 *A representation of knowledge from a given domain as a set of concepts and the relationships between them.* ◇

When it is intended to be used by software tools, an ontology frequently takes the form of a file (or set of files) written in an ontology language – a formal language which facilitates the automatic processing of an ontology. Several ontology languages are available [Corcho and Gómez-Pérez, 2000].

Independently of the language used to describe them, ontologies often present structural similarities, namely on the components used to build them. In fact, most ontologies are formed by individuals, classes, attributes and relations.

Individuals: Objects or instances, the basic elements of the ontology.

Classes: Types or groups of objects, they allow to establish hierarchies.

Attributes: Characteristics, properties or features of the individuals or classes.

Relations: How the individuals or classes relate to each other.

In domains related to computer science, ontologies are usually described using formal languages called ontology languages [Corcho and Gómez-Pérez, 2000, Pulido et al., 2006]. Commonly used ontology languages are RDF [Klyne and Carroll, 2004] and OWL [Antoniou and Harmelen, 2009].

Open Biomedical Ontologies (OBO) is a project for the development of ontologies to be used in biological and medical contexts. This initiative is responsible for OBO Foundry[3], a website which, in addition to the list of the OBO ontologies, provides the community with several discussion forums, description of the OBO principles and other services. The OBO project has developed its own ontology language.

2.4 Extraction of meaningful relationships

Relationship extraction (RE) attempts to detect occurrences of a prespecified pattern on the relationship between a pair of entities of given types. The entities must be clearly identified or the relation extraction performance will decrease. Usually, the type of entities is very specific, while the type of relationship may be either very general or very specific.

Bach and Badaskar [2007] provide the following definition of relation:

Definition 2 *A relation is defined in the form of a tuple* $t = (e_1, e_2, \ldots, e_n)$ *where the* e_i *are entities in a pre-defined relation* r *within document* D. $\diamond$

The characterization of a relation may depend on several distinct properties:

Cardinality: The number of participants in the relation.

Transitivity: A binary relation r can be defined as transitive if whenever an element a is related to an element b, and b is in turn related to an element c, then a is also related to c. More formally put:

$$\forall_{a,b,c}\, r(a,b) \wedge r(b,c) \Rightarrow r(a,c)$$

Direction: A relation can be classified as directed when it is not reflexive – the fact that a relation r exists between an element a and an element b

[3]OBO Foundry – `http://obofoundry.org/`

does not necessarily imply the existence of a relation between b and a. In a formal definition:

$$\forall_{a,b}\, r(a,b) \not\Rightarrow r(b,a)$$

Signal: When a relation is extracted from a negated affirmation, the result may be a negative relation.

2.4.1 RE approaches

There are several approaches to RE documented in literature [Cohen and Hersh, 2005]:

- Manually generated template-based methods use patterns created by domain experts to extract entities connected by a specific relation;
- Automatic template methods create similar templates by analyzing patterns in text in the surroundings of pairs of entities known to have the desired relationship;
- Statistical methods identify relationships by observing statistically significant co-occurrence of entities, i.e, entities which are found together more than they would by chance;
- NLP-based methods parse text into a structure from which relations can be extracted [Fundel et al., 2007].

A comprehensive review on the state of the art methods and tools for relationship extraction can be found in Bach and Badaskar [2007].

2.4.2 Existing RE tools

Usually, template-based and NLP-based methods are developed for a particular domain application. Notwithstanding, text miners can still take advantage of previously existing modules, libraries and frameworks, like the Lingua modules in Perl or tools like GATE [Cunningham et al., 2002, Bontcheva et al., 2004] and LingPipe [Baldwin and Carpenter].

Likewise, NetCutter is a tool for co-occurrence analysis based on a bipartite graph representation of data [Müller and Mancuso, 2008], and may be used to identify possible relations between named entities.

And, LAITOR is a text mining system that analyses co-occurrences of bioentities, biointeractions and other biological terms in MEDLINE abstracts [Barbosa-Silva et al., 2010]. A summary of the reference performance values for LAITOR may be found on Table 2.1.

Table 2.1: Precision, recall and F-score values for LAITOR (against BioCreative II IAS subtask) [Corbett and Murray-Rust, 2006].

LAITOR		
Precision	**Recall**	**F-score**
0.85	0.58	0.67

2.5 Ontologies

2.5.1 The Open Biomedical Ontologies (OBO) format

Ontologies have been widely used in the biomedical sciences [Schulz et al., 2009] as a shared understanding of some domain of interest, to be used as a unifying framework to solve problems from that domain.

OBO is a project for the development of ontologies to be used in biological and medical contexts. This initiative is responsible for OBO Foundry[4], a website which, in addition to the list of the OBO ontologies, provides the community with several discussion forums, description of the OBO principles and other services. The OBO project has developed its own ontology language, which is widely used within the biomedical sciences. Roundtrip transformation tools are available between OBO and OWL (another common ontology format).

2.5.2 Ontologies for units of measure

Several ontologies have been published covering the domain of units of measurement. Follows a list with the most relevant ones and their description.

The Ontology of Units of Measurement: Part of the OBO Foundry Initiative [Smith et al., 2007], it was developed to be used in conjunction with OBO Phenotype Group [2006].

Ontology of units of Measure and related concepts (OM): This ontology models concepts and relations important to scientific research, having a strong focus on units and quantities, measurement, and dimensions [Rijgersberg et al., 2012, 2011].

Measurement Units Ontology: MyMobileWeb is an open source, standards-based software framework that simplifies the rapid development of mobile web applications and portals. The similar need for a unifying definition of units of measure has resulted in the development of an ontology [Berrueta and Polo, 2012].

W3 Units Ontology: An ontology for units of measure created by the World Wide Web Consortium. This ontology defines a set of properties which

[4]OBO Foundry – `http://obofoundry.org/`

> may be used for representing physical quantities. Each property relates a quantity to a number which expresses the quantity in the given unit [World Wide Web Consortium, 2007].

The usage of ontologies for representing units of measure and similar concepts has been discussed in several articles, namely in Allen et al. [2004], Gruber and Olsen [1994] and Kuhn [2009].

2.6 Tool train, test and evaluation

Systems like the ones described in the previous sections are often improved by training, a process in which a given system is perfected by analyzing real data which was previously manually classified or annotated. The train process requires three sets of data: a training set, a test set and a verification set [Dayhoff and DeLeo, 2001]. The training set is used to adjust several parameters during the training in order to achieve better results. The test set is used to decide when to stop training the system. The verification set is used only after the training is concluded. This way it keeps its independence from the training performed, and may be used as as performance meter for the system.

The evaluation of the performance is usually measured in terms of *precision*, *recall* [Olson and Delen, 2008] and *F-score* [Rijsbergen, 1979].

Taking as example the task of Named Entity Recognition, these concepts can be simply explained as follows: given a set of documents D, there is a number E_{total} of named entities mentioned in them. After processing the texts with a NER tool N, $T_{N(D)}$ represents the total number of entities found by N in D, and $C_{N(D)}$ represents the number of correct entities found by N in D. The precision $P_{N(D)}$ and recall $R_{N(D)}$ are then given by:

$$P_{N(D)} = \frac{C_{N(D)}}{T_{N(D)}} \tag{2.1}$$

$$R_{N(D)} = \frac{C_{N(D)}}{E_{total}} \tag{2.2}$$

The *F-score* is the harmonic mean between precision and recall, obtained with the following equation:

$$\textit{F-score}_{\mathrm{N(D)}} = \frac{2 * P_{N(D)} * R_{N(D)}}{P_{N(D)} + R_{N(D)}} \tag{2.3}$$

Precision, recall and F-score are also easily defined in the context of other tasks, such as relationship extraction (metrics based on the number of existing and detected relationships).

2.7 Visualization and interpretability

Text mining methods such as Named Entity Recognition (NER) and Relationship Extraction (RE) allow to extract information from text documents. However, the universe of scientific articles (even when restricted to those belonging to a specific field of studies) is often too large, resulting in said methods outputting an amount of information much greater than users are capable of dealing with efficiently. In fact, simply extracting and presenting textual evidences found in the articles is of little help when the size of the corpus exceeds a few dozens of papers.

As such, information extraction techniques should be complemented with data mining methods, allowing to prevent users of becoming overloaded with information.

Additionally, traditional visualization forms are often too low-level, requiring from the user a deep knowledge of the field. For example:

- the fact that a given entity is mentioned in a document text is not very meaningful unless the user is able to evaluate the relevance of that entity;
- if the named entity is a property (either numeric or other type), it is important to know its value, and which values (or ranges of values) are more frequently mentioned in the corpus.

Based on the textual evidences extracted by the previously mentioned methods, it is possible to present the data in more sophisticated forms, providing the user with more intuitive and immediate ways of inspecting the corpus.

One possible form of representation is by using graphics. In fact, graphic visualization allows the user to quickly have a global understanding of properties of the documents included in the corpus.

EnvMine is a set of text-mining tools for extraction of contextual information (physicochemical variables and geographical locations) from textual sources. Figures 2.1 and 2.2, extracted from Tamames and De Lorenzo [2010], provide an example of the expressive power of graphics.

Being able to provide higher level representations of the corpus, however, requires from software a deeper understanding of the entities and relationships identified in the documents. The context in which entities are mentioned has to be inspected to guarantee that they are comparable across the several papers in which they appear.

KID is an algorithm developed for generating a database with kinetic information of enzymes by using text-mining methods [Heinen et al., 2010]. Figure 2.3 presents an example of a graphical representation of the values mentioned for the parameters K_M, K_i and K_d.

Figure 2.1: Locations of 1646 samples stored in EnvBD database [Tamames and De Lorenzo, 2010].

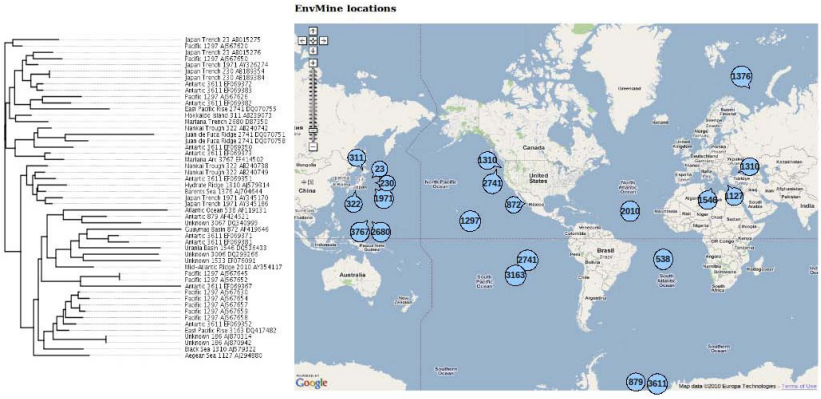


Figure 2.2: Sample clustering of EnvDB samples by geographical location [Tamames and De Lorenzo, 2010].

The ability to recognize numeric parameters in text documents is discussed in Chapter 3. However, in order to produce the kind of graphics presented in Figure 2.3, the software must understand the meaning of the parameters recognized. This means that the textual representations of numeric values and units of measure have to be parsed (for further details please refer to Sections 3.3.4 and **??**).

This additional step poses several additional challenges: if the lack of a standard way of representing numeric expressions in text makes difficult their recognition, parsing them to discover their true value is even more complex. A possible approach to this problem would be to perform a normalization of the numeric expressions (possibly by using a rewrite system such as Text::RewriteRules [Simões, 2012a]). The normalized numeric expressions could then resolved by an external program like Maxima [Rand, 2005]. Algo-

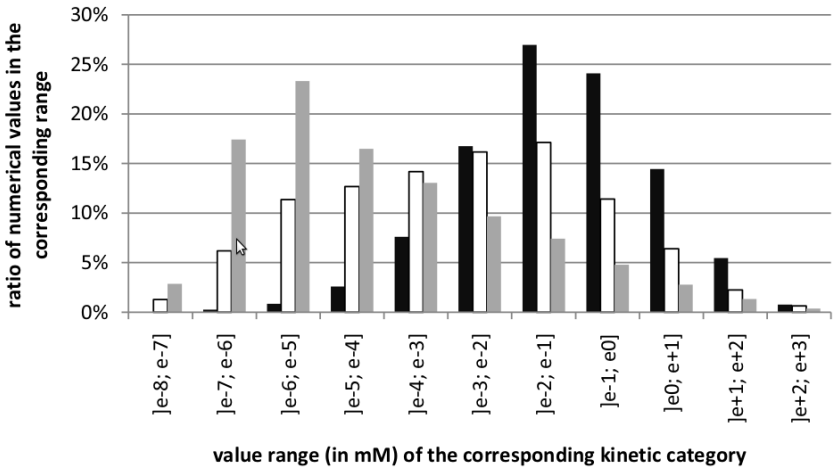


Figure 2.3: Range of extracted values for K_M, K_i and K_d [Heinen et al., 2010].

rithms for converting units of measure would also have to be implemented [Novak Jr, 1995], as different articles often use different units or subunits for the same parameters.

Chapter 3

Recognition and processing of numerical parameters

This chapter introduces the problem of recognizing numerical parameters in scientific text, sub-divided in the problems of recognizing a numerical parameter three components: parameter name, numerical expression and unit of measurement. The main limitations for and possible solutions for this task are presented, and the implementation of a Perl module, `Text::Math::NumExp`, for recognizing numerical expressions is presented.

Text examples provided throughout this chapter have been extracted from scientific articles belonging to distinct areas such as astrophysics, biomedical engineering, accelerator physics and statistics, to emphasize the broad nature of the challenges here described.

3.1 Anatomy of a numerical parameter

A numerical parameter can be viewed as a tuple comprised of three elements:

Parameter name: a term, composed by one or more words, which represents the magnitude or a physical property being quantified (e.g. *height*).

Numerical value: a number or numerical expression stating the value of the parameter (e.g. *5* or *3.4*E*-03*).

Unit of measurement: a term, possibly abbreviated, which represents a standard value for the physical property, and provides dimension to the numerical value (e.g. *kilometer* or $m \cdot s^{-2}$).

The tasks of finding numerical parameters in text requires the detection of the parameter names, numerical values and units of measurement, and inferring relationships between them. More formally put, a scientific text

$\mathcal{T}$ can be viewed as a set containing a set of parameter names $\mathcal{N}$, a set of numerical values $\mathcal{V}$, a set of units of measurement $\mathcal{U}$ and a set of numerical parameters $\mathcal{P}$ (Equation 3.1), where $\mathcal{P}$ if formed by all the relationships r involving an element from $\mathcal{P}$, one from $\mathcal{N}$ and other from $\mathcal{U}$ (Equation 3.2).

$$\mathcal{T} = \{\mathcal{N}, \mathcal{V}, \mathcal{U}, \mathcal{P}\} \tag{3.1}$$

$$\forall r \in \mathcal{P} : r(n, v, u) \Rightarrow n \in \mathcal{N} \wedge v \in \mathcal{V} \wedge u \in \mathcal{U} \tag{3.2}$$

The process of finding numerical parameters in text comprises two main steps:

1. the parameter names, numerical values and units of measure must be found, using distinct methods for each one.

2. relationships between the elements found must be inferred, aided by additional information provided by other tools (sentence splitters, tokenizers, etc).

3.2 Parameter names

3.2.1 What is a parameter name?

Parameter names are the identifiers which allow to understand which concept is being characterized or quantified. Parameter names present the following set of features:

Domain-specific: Each domain of expertise has its own set of commonly used parameters (which may however be common to other domains) and, consequently, its own set of parameter names.

Language-dependent: Being natural-language expressions, parameter names are usually language-dependent. However, some terms are often used untranslated.

Single/multi-word: Parameter names can be either mono-word terms or multi-word terms.

3.2.2 Recognition

The problem of recognizing parameter names is actually a Named Entity Recognition (NER) problem, which means that many of the traditional techniques can still be applied. On the other hand, scientific domains often present specific terms which traditional tools are not prepared to handle. The recognition of parameter names presents the following challenges:

Synonyms: Often there are synonyms for parameter names (sometimes more than one synonym is used even within the same article).

Inflection and derived words: Parameter names are natural language expressions in the sense that they are formed by nouns, verbs, adjectives, adverbs, etc. As such, words in parameter names can be inflected or derived.

Abbreviations and acronyms: Often, the same parameter names are recurrently used in an article. For this reason, they are sometimes abbreviated or, when composed by several words, used as acronyms.

Partial names: Parameter names can sometimes be used with additional or missing words, such as adjectives.

Convolution: Multiword expressions may suffer from convolutions, that is, a change in the order of the words.

Due to their language and domain-dependent nature, it is difficult to implement generic methods for the recognition of parameter names. On the other hand, within a given domain (and a given language), the number of commonly used parameter names tends not to exceed a few dozens. As such, domain-specific dictionary-based approaches are a viable option from the several methodologies traditionally used in NER (see Section 2.3.1).

The strategy used in this project was to allow the inclusion of a domain-specific exhaustive ontology of the most commonly used parameter names, and use it to implement a dictionary-based approach. The ontology contains elements of the class `parameter_names`.[1] Each element includes the following fields:

Name: the name of the parameter. Usually it is the most common or less ambiguous term for that parameter.

Synonyms: a list of other terms used to refer to the same parameter. Each synonym can be an *exact*, *broad*, *narrower* or *related* synonym, depending if it represents exactly the same parameter, a broader definition, a narrower definition or is just similar, respectively.

Each element from the ontology can be loaded, and its `name` and `synonyms` used to build search patterns which allow to identify parameter names in the text of a scientific article. Additional methods such as stemming or approximated matching can be used to increase the system's performance. Example 3.1 presents an excerpt of an ontology specific for the waste water treatment domain [Santos et al., 2012].

[1] Additionally, the ontology also includes the `biological_entities` and `units_of_measurement` classes, whose application is described in Section 3.4).

```
[Term]
id:       ID:0000001
name:     variable_names
def:      "Names of variables commonly found on waste water
          treatment scientific texts."
[Term]
id:       ID:0000011
name:     bacterial ratio
synonym:  "autothrophic bacterial ratio" NARROW []
synonym:  "heterothrophic bacterial ratio" NARROW []
is_a:     ID:0000001 ! variable_names
```

Example 3.1: Excerpt from a domain-specific ontology.

3.3 Numerical expressions

3.3.1 What is a numerical expression?

More than the epistemological and ontological questions of how human beings perceive numerical values, or the technological ones of how computers store them internally, the main concern in this project is how such values are represented in (both electronic and physically printed) text, and therefore, which strategies can be used to recognize them automatically.

The most basic instance of a numeric expression is one that is composed by a single numerical value. Such a simple number recognizer can be easily implemented (using, for example, regular expressions). The recognizer can then be extended to allow positive and negative numbers, numbers with decimal mark and thousands separator. It gets complicated when it comes to scientific or engineering notation, and eventually it becomes unmanageable.

Additionally, with the ultimate goal of extracting numerical parameters, the term *numerical expression* is here used with a very comprehensive meaning, as several other types of numerical elements must be taken into account: scalar values, ranges, variables, functions, complex expressions, spelled-out numbers, arithmetic symbols and enumerations. The following section presents further details about these numerical expressions.

3.3.2 Types of numerical expressions

Scalar value: a numeric expression composed by a single value. This value may be an integer or a decimal (using one of several possible notations for decimal and thousands separators). Although scalar values may also be represented in scientific notation, but these might be seen as particular cases of complex expressions.

```
For ALT, the ULN is taken to be 36 units/litre and for TBL is 21
mol/litre.
```

Example 3.2: Scalar numerical values in scientific article [Papastathopoulos and Tawn, 2012].

Range: an expression which defines a range of values. There are several possible notations, either mathematical, such as $0.9 < x < 1.1$, $[0.9, 1.1]$ or 1 ± 0.1, or textual ones, such as *from 0.9 to 1.1*, *over 0.9*, *at least 0.9*, *up to 0.9*, etc.

```
On the other hand, the entropy profiles between ~50 km and ~100
km are remarkably similar between all models.
```

Example 3.3: Numerical range in scientific article [Dolence et al., 2012].

Complex expression: A more or less complex mathematical expression (in which it may be difficult to separate the numerical expression from the parameter name and the unit of measurement). This includes values for non-scalar variables (e.g. vectors) and expressions featuring Greek characters, mathematical symbols, super/sub-script and other non-trivial text.

Example 3.4: Complex numerical expression in scientific article [].

Spelled-out: numbers spelled-out as words, such as *twenty four* or *one third*.

```
We derive the expansion properties of MCs in the outer helio-
sphere from one to five astronomical units to compare them with
those in the inner heliosphere.
```

Example 3.5: Spelled-out numbers in scientific article [Gulisano et al., 2012].

Enumerations: Multiple numerical values for the same parameter (e.g. representing multiple measurements, or values used in distinct experimental runs) are often presented as a list or enumeration in the text.

```
For instance, for a typical picture of 1.6×1.6kpx the tools
would take approximatively 0.69, 1.22 and 3.0 seconds,
respectively.
```

Example 3.6: Numerical parameter enumeration in scientific article [Geissmann, 2012].

3.3.3 Recognition

Given all the distinct numerical expression forms and shapes, the task of recognizing them among normal text is not trivially solved. We have implemented a Perl module, `Text::Math::NumExp` which implements several of the methods described in this section.

a) Spelled-out numbers

Spelled-out numbers are different from expressions composed by numbers and symbols, and as such can be recognized using a distinct approach. The task of recognizing numbers spelled-out in English can be performed using several availeble Perl modules: `Lingua::EN::Words2Nums` [Hess, J., 2001], `Lingua::EN::Numericalize` [Calder, E., 2003] and `Lingua::EN::FindNumber` [Cozens, S., 2003]. The latter is used in `Text::Math::NumExp`, and provides a regular expression for recognizing spelled-out numbers, and a `numify` method which converts them to numerical format.

b) Numerical expressions

Our early attempts to detect numerical expressions using well-defined regular expressions built based on the structure of numerical expressions (see Section 3.3.1) failed due to all the different formats (or lack of) used in scientific articles, and the noise introduced by (often consecutive) file format conversions. This method achieved a good *precision* score, but a very low *recall* (for the definition of *precision* and *recall* please refer to Section 2.6).

As such, we decided to implement in `Text::Math::NumExp` a technique which searches the text for segments containing digits and which do not seem normal text. The procedure my be roughly translated to:

1. Find one or more consecutive digits within the text. These define an anchor point for the *numerical segment*.
2. Repeat until nothing new is added to the segment:
   (a) Extend the segment to any immediately surrounding non-space characters.
   (b) If a term containing digits is available to either side of the segment, separated only by white space or a small word (3 or less characters), include it.

This method allows to detect very unorthodox numerical expressions, but, as usual in information retrieval tools, such improvements in the *recall* come at the expense of a deterioration on the *precision*. To minimize this effect, numerical segments are submitted to several normalization steps devised to ensure a common mathematical notation – multiplication sign, exponentiation format, etc – and false positive results are filtered out by verifying them

against specific features – for example, segments composed by a number between brackets (e.g. `[12]`) are often used within scientific articles to represent citations.

3.3.4 Parsing Numerical Expressions

A numerical expression cannot really be perceived as a number until it is solved and its true value determined. Knowing the numerical value of an expression is what allows us use it and operate over it as a number – compare it, sort it, etc.

In order to determine the value of a numerical expression, it is necessary to parse the expression, and then try to solve it. A first approach to the parsing and evaluation of numerical expressions has been implemented in `Text::Math::NumExp` by taking advantage of the `eval` and `looks_like_number` function.

`eval` is a Perl built-in function which evaluates a given string as if it were an expression and then returns its result. `looks_like_number` is a low-level routine used internally by Perl to test if the content of Scalar Value (SV) is or looks like a number. This routine is available in the Perl public API [Okamoto et al., 1997], and exposed as a regular function by the module `Scalar::Util` [Barr, G., 1999].

The `solve` function implemented in `Text::Math::NumExp` accepts a string and normalizes it, converting common mathematical patterns to their equivalent in the Perl language (e.g. `3.5x10^6` is converted to `3.5*10**6`). The normalized string is passed to `eval`, and the result is analyzed with `looks_like_number`.

3.4 Units of measure

3.4.1 What is a unit of measure?

The value of a parameter is truly meaningful only when the corresponding unit of measure is known. For instance, to know that the duration of a given procedure is 5 is not very helpful, as it may be years, minutes or microseconds. Usually, in scientific papers, values are indeed accompanied by units; nevertheless, cases where there is no unit or the unit is to be derived from context occur frequently.

Composed units: Units formed by two or more base units. These units may be written unabbreviated, such as *meters per second*, or by using their symbols to write mathematical expressions, like in ms^{-2}, in which case it may be recognized using similar approaches to the ones used to recognize complex mathematical expressions.

```
The most contrasting manifestation of this effect should be
observed in the range of the hard radiation fluxes from 10^10 to
10^12 ergs/(cm^2·sec).
```

Example 3.7: Composed units in scientific article [Ulyanov et al., 2012].

Multiples and fractions: Often, a value is expressed as a multiple or fraction of a given unit. In these cases, a unit prefix is indicated. When a unit is written unabbreviated, this prefix is a name (for instance, *kilometer* or *microsecond*); when the unit is represented by its symbols, prefixes are also abbreviated (for example, *km* or μs).

```
(...) the activity after 1 year of cooling is of the order of 10
mSv/hr, which greatly exceeds the safety limit for radiation
workers (20 mSv/yr).
```

Example 3.8: Different unit multiples in scientific article [Back et al., 2012].

Non standard units: The International System of Units (SI) defines seven base units, and several derived units [Goldman and Bell, 1986]. Despite this list being useful, there is a great amount of units used which are not part of the SI system. Some of them are used for convenience in spite of the existence of equivalent SI units (for example, an *ångström* is equal to 10^{-10} *meters* or 0.1 *nanometers*)).

Other units are so specific for a given field that they are not usually thought of as units in a traditional way (for example, *cells per millilitre* or *floating-point operations per second*).

```
The OCR of the myoblasts dropped from 1.04±0.19 fmol/min/cell at
the surface to 0.0 fmol/min/cell at 2000 µm into the gel.
```

Example 3.9: Non-standard units in scientific article [Davis et al., 2007].

3.4.2 Recognition

The process of recognizing units of measurement presents characteristics similar to both numerical expression and parameter names recognition: on the one hand, units are formed by specific terms which can be extracted from a dictionary or ontology; on the other, they are frequently composed using mathematical rules (division, multiplication, exponentiation) and their respective multitude of notations.

The domain of units of measurement has been the focus of several available ontologies, which synthesize the knowledge in a structured way, making them adequate for being used in software systems. A list with some of the most relevant ones and their description is available in Section 2.5.2. These generic

ontologies provide information on the most common or standard units of measurement; however, as stated previously, some units are specific for a given domain, and are not likely to be featured in these ontologies. Similar to the case of parameter names, this problem can be addressed by using domain-specific ontologies developed and curated by experts.

Both generic purpose or user-provided ontologies for units of measurement provide, for each unit of measure:

ID: A code to identify the unit on the ontology.

Unit: The canonical or most common form of the unit.

Synonyms: Other representations of the same unit, including abbreviated forms using the units symbols.

The unit and its synonyms can be loaded by the application and used to define search patterns. However, the similarities that units present with mathematical expressions mean that the same unit may have several possible representations (e.g. m/s^2 and $m \cdot s^{-2}$), and the one used in a given scientific article may not be included even in the unit's synonyms. To minimize the effects of this problem, units loaded from ontologies can be submitted to a simplification process – for example, by removing all the non-letter characters – and then fuzzy matched in the text. Once again, while this technique potentially decreases the precision values, it should be compensated by a larger increase in the recall.

3.5 Detecting relations

As described in Section 3.3.2, numerical parameters can be extracted only after mining parameter names, numerical expressions and units of measurement, and requires the inferring of connections between these elements. The problem of extracting numerical parameters is thus an instance of the wider challenge of relation extraction, a common and widely addressed task natural language processing (see Section 2.4).

Numerical parameters can be extracted using the concept of neighborhood or by searching the text for some common patterns.

3.5.1 Neighborhood-based extraction

Elements have a higher probability of being related the closer they are to each other within a text . More specifically, a parameter name, a numerical expression and a unit of measure are more likely to belong to the same numerical parameter if they are found in the same paragraph or, even better, in the same sentence.

As such, paragraph and sentence delimitation are crucial in the process of identifying numerical parameters. Descriptions of the implementation of Perl modules for such tasks are presented in Sections 4.5.1.2 and 4.5.1.3.

The extraction of numerical parameters is then performed by searching for parameter names, numerical expressions and units of measure present in the same segment of text (either paragraph or sentence). Whenever multiple instances of these elements are present in a given segment, there is the need to determine which ones belong to which numerical parameter. Here the order and the relative disposition of the elements can be used to disambiguate. Additionally, the ontology provided by the user may contain information relating each parameter name with the possible unit (or units) they might be represented in.

3.5.2 Patterns-based extraction

Searching for previously defined patterns is one of the traditional methods for relationship extraction 2.4.1.

Chapter 4

A framework for the extraction of numerical parameters from scientific literature

In this chapter is presented `T-Eng`, a modular text mining framework for performing the extraction of numerical parameters and other relevant entities from scientific texts. It is implemented as a Perl distribution, `Text::T-Eng`[1], and comprises modules for building a customized workflow from the document importing, annotation, analysis and results visualization.

4.1 System overview

Figure 4.1 presents a diagram of T-Eng's arquitecture, completed with the external objects.

4.2 Internal objects

Internal object are all objects that are involved in processes/tasks within the T-Eng operation. They result from processing external files or performing calculations on other internal objects, and can be provided as input for further processing, exported to external files or used to generate visualization objects. Internal objects have been structured/modeled according to requirements of

[1] Due to the fact that `T-Eng` is implemented in Perl, its nomenclature follows the Perl's specification and tradition. As such, `T-Eng`'s base module is `Text::T-Eng`, and all the other modules are directly or indirectly contained in this workspace (for example, documents are represented by the `Text::T-Eng::Document` object). In order to simplify the description of modules, throughout this chapter, the `Text::T-Eng` namespace will frequently be abbreviated to `TE` (and as such, `Text::T-Eng::Document` will be referred to as `TE::Document`, for example).

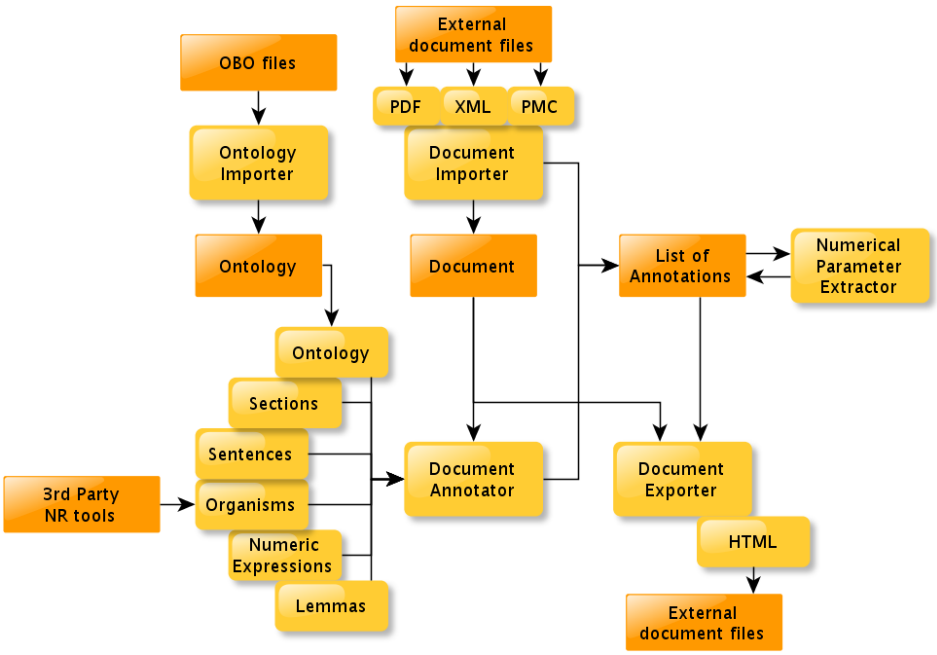


Figure 4.1: Arquitecture of T-Eng.

core NLP processes. Communication with processes or resources outside T-ENG is granted by specialized wrappers. This is important to guarantee T-Eng interoperability and cross-domain application.

The main internal objects used by T-Eng are *documents*, *ontologies* and *annotations*.

4.2.1 Document

The `TE::Document` object represents a scientific article, including all its contents and metadata. Namely, a document object includes the following fields:

Document ID: A code which identifies the document and its source. The specific format of the identifier code may depend on the specific source, but generally follows the form *source:id* (for example, `PMID:16332320` identifies a document whose PubMed identifier is 16332320).

Authors: A list containing the names of the document's authors.

Title: The title of the document.

Full text: The full text of the document, in simple text format.

URL: The URL where the document can be obtained (if applicable).

Keywords: A list containing the document's keywords.

Subject: The domain or subject of the document.

Date: The document's publication date.

This object is created when a document is imported from an external source (for more details on document importation please refer to Section 4.3).

4.2.2 Ontology

The `TE::Ontology` object is used to represent and manipulate domain-specific knowledge. Domain-specific knowledge refers to parameter designations (common name and associated synonyms), and the corresponding units of measurement. Additionally, it may include information considered of interest but not directly related to parameter mentioning (e.g. the designation of experimental methods).

`TE::Ontology` objects contain the following attributes:

Ontology ID: A code which identifies the ontology.

Terms: A list containing the terms of the ontology as `TE::Ontology::Term` objects.

URL: The URL where the ontology is accessible (if any).

Description: A small description of the ontology's content.

Date: The ontology's creation date (if known).

Version: The ontology's version (if any).

Each element contained in the **Terms** is itself a `TE::Ontology::Term` complex object which contains the following fields:

Name: The main or most common name for the term.

ID: An identifier code indexing the term in the ontology.

Synonyms: A list of synonyms (alternative names or designations) for the same term.

4.2.3 Annotation

Singular annotations are produced when documents are imported, and every time a tool operates over them. They represent data inferred through any of the processing steps, and that may later be used to extract knowledge about the document. `TE::Annotation` is the object which allows to use them internally, and comprises the following attributes:

Start: An integer value indicating the offset (in characters) of the first character of the text which the annotation is concerned with.

End: An integer value indicating the offset (in characters) of the last character of the text which the annotation is concerned with.

Text: The portion of text being annotated.

Rank: An integer value used for solving annotation conflicts (when there are two conflicting annotations concerning the same portion of text).

Source: An indicator of the origin of the annotation (which ontology or annotating tool generated the annotation)

There are several types of annotations, divided in two main subclasses:

Structure-related annotations: these annotations are related to structural elements, and provide the context awareness needed by some of the processors to fine tune their results. In this category are included section, sentence, paragraph and token annotations.

Content-related annotations: these annotations are related to the contents of the document. This category includes annotations generated upon the recognition of numerical expressions, ontology terms, units of measure and domain-specific entities (e.g. organisms or genes).

Details on the existing types of each class of annotations are provided in Section 4.5. Additionally, there is a class *TE::ListOfAnnotations*, whose instances contain lists of annotations and which implements methods for querying, searching and filtering multiple annotations.

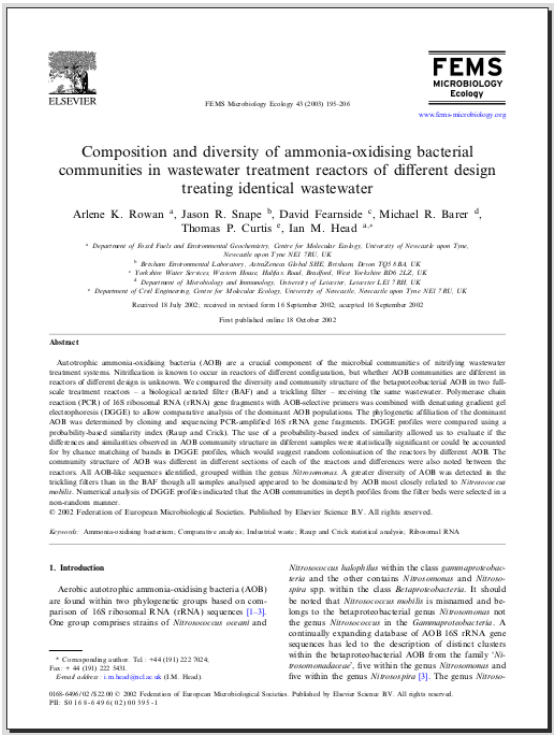
4.3 Document importation

Most text processing tools (such as stemmers, taggers, NER recognizers, etc) are only able to process plain text files, which means that documents in other formats need to be converted before being processed. Depending on the file format and the conversion tool used, additional metadata can also be extracted.

Tools which are capable of handling non-plain text documents usually come bundled with conversion libraries or third party applications, which are used to obtain the plain text documents. This is also the case of T-Eng, which includes in the `TE::DocImporter` namespace importers for different formats which extract the full text and the available metadata to build a document internal object.

4.3.1 PDF

The Portable Document Format (PDF) is a file format developed by Adobe Systems. Initially developed as a proprietary format, it has evolved into an open standard [Adobe Systems Inc., 2008]. In 2010, Adobe claimed that more that 200 million PDF documents had been posted on the web [Adobe Systems Inc., 2010], which makes it one of the main formats for document exchange, including scientific articles. The format itself has evolved and is now capable of supporting both vector and raster graphics, forms, attachments and other advanced features. Figure 4.2 presents an example of a PDF file.



ELSEVIER

FEMS Microbiology Ecology 43 (2003) 195–206

FEMS MICROBIOLOGY Ecology

www.fems-microbiology.org

Composition and diversity of ammonia-oxidising bacterial communities in wastewater treatment reactors of different design treating identical wastewater

Arlene K. Rowan [a], Jason R. Snape [b], David Fearnside [c], Michael R. Barer [d], Thomas P. Curtis [e], Ian M. Head [a,*]

[a] Department of Fossil Fuels and Environmental Geochemistry, Centre for Molecular Ecology, University of Newcastle upon Tyne, Newcastle upon Tyne NE1 7RU, UK
[b] Brixham Environmental Laboratory, AstraZeneca Global SHE, Brixham, Devon TQ5 8BA, UK
[c] Yorkshire Water Services, Western House, Halifax Road, Bradford, West Yorkshire BD6 2LZ, UK
[d] Department of Microbiology and Immunology, University of Leicester, Leicester LE1 7RH, UK
[e] Department of Civil Engineering, Centre for Molecular Ecology, University of Newcastle, Newcastle upon Tyne NE1 7RU, UK

Received 18 July 2002; received in revised form 16 September 2002; accepted 16 September 2002

First published online 18 October 2002

Abstract

Autotrophic ammonia-oxidising bacteria (AOB) are a crucial component of the microbial communities of nitrifying wastewater treatment systems. Nitrification is known to occur in reactors of different configuration, but whether AOB communities are different in reactors of different design is unknown. We compared the diversity and community structure of the betaproteobacterial AOB in two full-scale treatment reactors – a biological aerated filter (BAF) and a trickling filter – receiving the same wastewater. Polymerase chain reaction (PCR) of 16S ribosomal RNA (rRNA) gene fragments with AOB-selective primers was combined with denaturing gradient gel electrophoresis (DGGE) to allow comparative analysis of the dominant AOB populations. The phylogenetic affiliation of the dominant AOB was determined by cloning and sequencing PCR-amplified 16S rRNA gene fragments. DGGE profiles were compared using a probability-based similarity index (Ruap and Crick). The use of a probability-based index of similarity allowed us to evaluate if the differences and similarities observed in AOB community structure in different samples were statistically significant or could be accounted for by chance matching of bands in DGGE profiles, which would suggest random colonisation of the reactors by different AOB. The community structure of AOB was different in different sections of each of the reactors and differences were also noted between the reactors. All AOB-like sequences identified, grouped within the genus *Nitrosomonas*. A greater diversity of AOB was detected in the trickling filters than in the BAF though all samples analysed appeared to be dominated by AOB most closely related to *Nitrosocuc mobilis*. Numerical analysis of DGGE profiles indicated that the AOB communities in depth profiles from the filter beds were selected in a non-random manner.

Keywords: Ammonia-oxidising bacteria; Comparative analysis; Industrial waste; Ruap and Crick statistical analysis; Ribosomal RNA

1. Introduction

Aerobic autotrophic ammonia-oxidising bacteria (AOB) are found within two phylogenetic groups based on comparison of 16S ribosomal RNA (rRNA) sequences [1–3]. One group comprises strains of *Nitrosococcus oceani* and *Nitrosococcus halophilus* within the class *gammaproteobacteria* and the other contains *Nitrosomonas* and *Nitrosospira* spp. within the class *Betaproteobacteria*. It should be noted that *Nitrosococcus mobilis* is misnamed and belongs to the betaproteobacterial genus *Nitrosomonas* not the genus *Nitrosococcus* in the *Gammaproteobacteria*. A continually expanding database of AOB 16S rRNA gene sequences has led to the description of distinct clusters within the betaproteobacterial AOB from the family '*Nitrosomonadaceae*', five within the genus *Nitrosomonas* and five within the genus *Nitrosospira* [3]. The genus *Nitroso-*

* Corresponding author. Tel.: +44 (191) 222 7024; Fax: +44 (191) 222 5431.
E-mail address: i.m.head@ncl.ac.uk (I.M. Head).

0168-6496/02 $22.00 © 2002 Federation of European Microbiological Societies. Published by Elsevier Science B.V. All rights reserved.
PII: S0168-6496(02)00395-1

Figure 4.2: Example of scientific article in PDF format.

There are several tools available which perform the conversion of PDF files to plain text. The main difficulty while performing such conversion is that in PDF documents there is a poor notion of document structure – text elements are positioned in pages through graphical directives which specify the horizontal and vertical offset, and the font to be used. As such, most conversion tools have a hard time determining the order of the textual elements, and are easily confused by multiple columns, headers and footers, etc.

Examples of PDF conversion tools are `pdftotext`, based on Xpdf [Noonburg, 2001], and Apache Tika [Mattmann and Zitting, 2011, Noonburg, 2001]. An in-depth review of several PDF to plain text converters can be found in [Robinson, 2001].

T-Eng wraps two external tools for importing documents in the module `TE::DocImporter::PDF`:

- `pdftotext`, to perform the conversion of the document to plain text (an output example can be found in Figure 4.3).
- `dumppdf.py`, a tool bundled with PDFMiner [Shinyama, 2010] which extracts (if available) a table of contents from the document metadata (an output example can be found in Figure 4.4).

```
FEMS Microbiology Ecology 43 (2003) 195^206

www.fems-microbiology.org

Composition and diversity of ammonia-oxidising bacterial
communities in wastewater treatment reactors of di¡erent design
treating identical wastewater
Arlene K. Rowan a , Jason R. Snape b , David Fearnside c , Michael R. Barer d ,
Thomas P. Curtis e , Ian M. Head a;Ã
a

Department of Fossil Fuels and Environmental Geochemistry, Centre for Molecular
Ecology, University of Newcastle upon Tyne,
Newcastle upon Tyne NE1 7RU, UK
b
Brixham Environmental Laboratory, AstraZeneca Global SHE, Brixham, Devon TQ5
8BA, UK
c
Yorkshire Water Services, Western House, Halifax Road, Bradford, West Yorkshire
BD6 2LZ, UK
d
Department of Microbiology and Immunology, University of Leicester, Leicester
LE1 7RH, UK
e
Department of Civil Engineering, Centre for Molecular Ecology, University of
Newcastle, Newcastle upon Tyne NE1 7RU, UK
Received 18 July 2002; received in revised form 16 September 2002; accepted 16
September 2002
First published online 18 October 2002

Abstract
Autotrophic ammonia-oxidising bacteria (AOB) are a crucial component of the
microbial communities of nitrifying wastewater
treatment systems. Nitrification is known to occur in reactors of different
configuration, but whether AOB communities are different in
```

Figure 4.3: Example of scientific article after conversion with `pdftotext`.

4.3.2 PubMed

PubMed is a free bibliographic repository for scientific documents covering Life Sciences and biomedical topics which provides access mainly to the Medical Literature Analysis and Retrieval System Online (MEDLINE) [Greenhalgh, 1997] bibliographic database.

PubMed Central (PMC)[2] is a subset of PubMed comprised only by full-text scientific literature. Articles from PMC are freely available both in PDF and NXML format – the latter being a XML schema defined by the National Library of Medicine Journal Archiving and Interchange DTD [Rosenblum, 2010]. An example of an article in NXML format is presented in Figure 4.5.

PubMed is the most frequently used source of articles for test and evaluation of biomedical text mining applications. Among other supporting features, it makes available tools for bulk retrieval and specialized search filters. `T-Eng` includes a PubMed importer module, `TE::DocImporter::PubMed`, which given a list of PubMed IDs fetches the corresponding articles in NXML

[2]PubMed Central – `http://www.ncbi.nlm.nih.gov/pmc`

```
<outlines>
  <outline level="1" title="Composition and diversity of
       ammonia-oxidising bacterial communities in wastewater
       treatment reactors of different design ...">
    <dest>
      <list size="3">
        <ref id="247"/>
        <literal>FitH</literal>
        <number>1670</number>
      </list>
    </dest>
    <pageno>0</pageno>
  </outline>
  <outline level="2" title="Introduction">
    <dest>
      <list size="3">
        <ref id="247"/>
        <literal>FitH</literal>
        <number>10</number>
      </list>
    </dest>
    <pageno>0</pageno>
  </outline>
  <outline level="2" title="Materials and methods">
    <dest>
      <list size="3">
        <ref id="1"/>
        <literal>FitH</literal>
        <number>1670</number>
      </list>
    </dest>
    <pageno>1</pageno>
  </outline>
  <outline level="3" title="Wastewater treatment plant &#40;WWTP&#41;">
    <dest>
      <list size="3">
        <ref id="1"/>
        <literal>FitH</literal>
        <number>1670</number>
      </list>
    </dest>
    <pageno>1</pageno>
```

Figure 4.4: Outline of article extracted with `dumppdf.py`.

```
<?xml version="1.0"?>
<!DOCTYPE article PUBLIC "-//NLM//DTD Journal Archiving and Interchange
DTD v3.0 20080202//EN" "archivearticle3.dtd">
<article xmlns:xlink="http://www.w3.org/1999/xlink" xmlns:mml="http://
www.w3.org/1998/Math/MathML" article-type="research-article">
  <?DTDIdentifier.IdentifierValue article.dtd?>
  <?DTDIdentifier.IdentifierType system?>
  <?SourceDTD.DTDName article.dtd?>
  <?SourceDTD.Version 1.0?>
  <?ConverterInfo.XSLTName bmc2nlmx2.xsl?>
  <?ConverterInfo.Version 2?>
  <front>
    <journal-meta>
      <journal-id journal-id-type="nlm-ta">BMC Bioinformatics</journal-id>
      <journal-title-group>
        <journal-title>BMC Bioinformatics</journal-title>
      </journal-title-group>
      <issn pub-type="epub">1471-2105</issn>
      <publisher>
        <publisher-name>BioMed Central</publisher-name>
      </publisher>
    </journal-meta>
    <article-meta>
      <article-id pub-id-type="pmc">2922198</article-id>
      <article-id pub-id-type="publisher-id">1471-2105-11-408</article-id>
      <article-id pub-id-type="pmid">20678237</article-id>
      <article-id pub-id-type="doi">10.1186/1471-2105-11-408</article-id>
      <article-categories>
        <subj-group subj-group-type="heading">
          <subject>Research Article</subject>
        </subj-group>
      </article-categories>
      <title-group>
        <article-title>Comparison study of microarray meta-analysis
methods</article-title>
```

Figure 4.5: Excerpt of article in NXML format.

format, which is then structurally processed to extract the full text of the article (or the abstract if the full text version is not available) and its metadata.

4.3.3 Plain text

There are many other formats available for writing, publishing and sharing text documents, some used more than others. Given the impossibility of implementing importers for each single format, `T-Eng` has a plain text importer, `TE::DocImporter:::PlainText`. This means that any text document can

be imported, as long as it is converted to plain text beforehand.

This conversion approach is considered a last resource since no metadata or information about structure can be easily and fully extracted from plain text documents. This loss of information has to be minimized through the use of specific information extraction modules on the plain text.

4.4 Ontology importation

Declarative objects such as ontologies open the possibility to discuss subjects with people who have no programming experience, allowing them to directly contribute with their knowledge to the development of the software, or to autonomously configure the system accordingly to their needs.

Currently, `T-Eng` imports ontologies in the OBO format, making use of `OBO::Parser::OBOParser`, a file parser for OBO ontologies [Antezana, E., 2006] on the `TE::OntoImporter` namespace. T-Eng is able to import two types of ontologies:

- third-party developed ontologies for the recognition of generally used units of measurement.

- user-provided ontologies for the recognition of domain-specific units of measurement, parameter names and domain entities.

4.4.1 Ontologies for units of measurement

`T-Eng` makes use of third-party developed ontologies of units of measurement for the recognition of generic, or most common, units in the documents. Currently, `T-Eng` imports the OBO Foundry Units of Measurement ontology [Smith et al., 2007]; however, it is easily extended for importing other ontologies, the only requirement for the new importers being that they should implement the `TE::OntoImporter::GenericUnits` role.

The importation process consists of reading the ontology source file, taking into account its inner structure, and process it structurally creating a list of ontology `TE::Ontology::Term` objects, which will ultimately give origin to the `TE::Ontology` object.

```
[Term]
id: UO:0000024
name: nanogram
def: "A mass unit which is equal to one thousandth of one
        millionth of a gram  or 10^[-9] g." [UOC:GVG]
subset: unit_slim
synonym: "ng" EXACT []
is_a: UO:0000002 ! mass unit
[Term]
id: UO:0000043
name: femtomole
def: "A substance unit equal to 10^[-15] mol." [NIST:NIST
"http://physics.nist.gov/cuu/Units/"]
subset: unit_slim
synonym: "fmol" EXACT []
is_a: UO:0000006 ! substance unit
```

Example 4.1: Excerpt of the OBO Foundry Units of Measurement ontology.

4.4.2 T-Eng domain-specific ontologies

Generic ontologies provide a way of recognizing the standard or the most commonly used units of measurement, but often scientific fields make use of their own specific units, which may include non-standard multiples or sub-multiples of standard units (see Section 3.4.1), or units representing domain-specific quantities (for example, *copies*/L). `T-Eng` solves this question by allowing the user to provide his own domain-specific ontologies, which are imported using the `TE::OntoImporter::OBOFile` module. These ontologies is typically created with the help of a domain expert, and contains not only the most common units used within that domain, but also the corresponding parameter names and other relevant terms. An example of a domain-specific ontology is provided in Chapter 5 in Example 5.1.

4.5 Document annotation

The annotation of a document is the process by which additional metadata is added to a document, often linked to specific elements within the document. Typically, there are two ways of linking annotations to a element in the text:

- inserting predefined marks on the boundaries of the segments being annotated, marking the beginning and the end of the annotation and adding other relevant information (e.g. identifying the biological species or cross-referencing an external data source).

- maintaining the annotations in a standalone file which in addition to the relevant information, contains the offsets of the beginning and end characters of the annotation.

The latter form of annotation – also known as standoff annotation – does not require the modification of the original document and makes it easier to cope with the possible overlap of different types and sources of annotations. This is the form used in `T-Eng`, allowing the documents to be preserved all the way since they initial importation.

In `T-Eng`, annotations may be added to a document from the moment it is imported. Annotation modules must implement the `TE::DocAnnotator` role, which simply requires them to provide an `annotate` method which takes a `TE::Document` as input, performs whatever steps it needs to create the annotations and inserts them in a `TE::ListOfAnnotations` which is returned within the original `TE::Document` object.

Section 4.2.3 presented the annotation types divided into Structure and Content; annotation subclasses ocasionally present some additional attributes or methods related to their specific type. The remaining of this section presents a description of the annotator modules currently included in T-Eng, including the respective output objects.

4.5.1 Structure

Structure annotators are responsible for extracting information regarding the document's structure, generating annotations which are required to implement the `TE::Annotation::Structure` role.

4.5.1.1 Sections

Determining the structural hierarchy of a document, as well as delimiting the boundaries of each section, serves a multitude of purposes:

- automatic generation of tables of contents
- mining specific sections for specific content
- sectioned visual representation

The section annotation process is closely related to the document importation because while some document formats allow to extract some representation of the document structure, others do not. As a result of this, T-Eng implements several different methods for detecting sections, aimed at XML documents, PDF documents, and plain text documents.

Regardless of their origin, annotations concerned with the documents' sections are represented using a `TE::Annotation::Section` object. Besides the attributes inherited from its super classes, these objects include two additional ones:

Section level: An integer value representing the nesting level of the section (subsections have higher values than main sections).

Section title: Represents the title of the section.

a) XML extraction

The NLM Journal Publishing Tag Set DTD [Rosenblum, 2010] used to represent PubMed articles and mentioned in Section 4.3, includes `<sec>` elements which are used to represent sections. `<sec>` elements can be nested, which allows to represent different levels of sections.

While traversing the XML document to extract other metadata, T-Eng generates section annotations whenever it finds a `<sec>` element.

b) PDF outline

PDF documents are occasionally generated with an embedded table of contents (also known as *outline*). T-Eng extracts this outline (if present) using `dumppdf.py` (see Section 4.3), and uses this information to find the section boundaries within the document, generating section annotations in the process.

c) Plain text section patterns

When a document is imported in plain text format, or even if it was imported from a PDF file but it did not have an embedded outline, information about the document structure has to be extracted from the document's text.

The basic structure of a scientific paper is often summarized as IMRAD: introduction, methods, results and discussion [Peh and Ng, 2008]. This often transpires to the section names: papers with sections named Introduction, Methods, Results and Discussion are a common occurrence.

In order to compile a list of common section names in scientific articles in the biomedical domains, the PMC-OAS articles were used to extract section names. From the 362,129 documents analyzed, 3,242,975 section titles were extracted. After normalizing and aggregating the identical results, 1,896,193 distinct section names were obtained, which were brought down to 12,341 after removing titles with less than 10 occurrences. The remaining titles were integrated in T-Eng's section annotator. Table 4.1 presents the ten most occurring section titles and the number of occurrences in PMC-OAS.

Not all article sections, however, belong to a well-defined set, and often sections or subsections have a name related to the subject being discussed. In order to cover these cases, T-Eng incorporates an additional module for article-specific section pattern recognition. Section names are searched using patterns which take into account elements such as empty lines, line length and the presence or lack of punctuation to detect section titles.

[3]The number of occurrences for *acknowledgements* is actually higher due to 2,577 occurrences of the alternative spelling *acknowledgments*.

Table 4.1: Number of occurrences of each normalized section title in PMC-OAS.

Occurrences	Section Title
326,742	results
232,545	discussion
203,159	introduction
190,648	background
176,259	methods
153,491	conclusion
118,189	conclusions
117,473	materials and methods
85,836	authors' contributions
83,276	acknowledgements[3]

4.5.1.2 Paragraphs

In T-Eng, determining paragraph boundaries is important for relationship extraction, because the probability of two elements being related is much smaller if the elements are mentioned in distinct paragraphs. Additionally, just like determining section boundaries, it helps to improve the visual representation of the document.r

Paragraph delimitation is a task common to many other text mining activities, and paragraphs have been the subject of study before [Szanser, 1973, Weissberg, 1984]. Nevertheless, when developing T-Eng we felt the need for a highly configurable tool, and given the lack of even a proper paragraph delimitation Perl module, we decided to create `Text::Paragraph::Splitter`, a Perl module which given a text determines it's paragraph boundaries.

`Text::Paragraph::Splitter` acts based on the principle that a paragraph can be spotted by detecting the presence of several clues. These clues can rarely be found all together; however, finding several of them at a given place within a text often is enough to identify a paragraph with a reasonable degree of confidence.

A specific weight is attributed to each type of clue, and the total sum of the weights of the clues at a given point of the text must reach a minimum threshold to consider that point as a paragraph break. Example 4.2 presents an illustrative paragraph with clues annotated. The clues are explained as follows:

1. lines ending in one of `?!.` usually indicate an end of sentence, which have a higher probability of being followed by a paragraph than lines with no punctuation at the end (zero weight) or with one of `,;:` (negative weight).

2. short lines are often found preceding paragraph breaks.

3. blank lines are commonly used to break paragraphs.

4. paragraphs are frequently started with indented lines.

5. lines beginning with an uppercase character.

```
1 (...) commodi harum nostrum deserunt facere adipisci optio
2 occaecati similique.[1][2]
3 [3]
4 [4][5]Sed inventore expedita nihil nemo impedit. Exercitationem
5 omnis atnihil aspernatur rerum et sed. Praesentium possimus
6 enim consectetur (...)
```

Example 4.2: Paragraph with annotated clues.

`Text::Paragraph::Splitter` starts by extracting several metrics, such as the average line length, the total number of indented lines, and the number of blank gaps (one or more blank consecutive lines). These measurements are affected by the following parameters:

short line length: maximum length of a line to still be considered as *short*. Can be specified in total number of characters or as a percentage of the average line length. Defaults to 50 characters.

indentation length: minimum length of white space needed on the start of a line to consider it *indented*. Defaults to 2 characters.

tab width: width of a tab character (used for indentation calculation). Defaults to 4 characters.

The weight of each type of clue can also be changed, as well as the total minimum threshold. After processing the text and determining the paragraph boundaries, `Text::Paragraph::Splitter` can return a list of paragraphs, a list of paragraph boundaries offsets, or the original text with the paragraphs annotated in a XML style.

T-Eng comprises a module, `TE::DocAnnotator::Paragraphs`, which is responsible for using the output of `Text::Paragraph::Splitter` to generate `TE::Annotation::Paragraph` annotations.

4.5.1.3 Sentences

Sentence splitting is a well-known problem, common to many tasks within the text processing domain. There are several Perl modules available for this

task; however, they all have in common the fact that they return a modified version of the original text: either by introducing artificial blank lines in the original text, or by returning some kind of list where each element is a sentence.

T-Eng's relies on standoff annotation (see Section 4.5), which requires the original text to remain unchanged. This means that annotations must include the start and end offsets of the text they refer to (in this case, the start and end of each sentence). Given the lack of a module with such a feature, it was decided to adapt an existing module to make possible the standoff annotation of sentence boundaries.

The chosen module was `Lingua::EN::Sentence` [Yona, S., 2001]. This module presents the advantage of being rather small (about 200 lines of code, excluding documentation) but also quite complete: it can be given lists of acronyms to be taken into account in the sentence splitting, and the splitting process is performed by sweeping the text multiple times to search naively for sentence breaks, removing the false ones, and adjusting offsets taking into account punctuation and other features. This module has been used before in biomedical text mining tasks [Lourenço et al., 2011, Krallinger et al., 2011].

The adaptation of this module, named `Lingua::EN::Sentence::Offsets`, was initially developed with the main goal of providing similar results (that is, given a similar text, split it in the same places as `Lingua::EN::Sentence`) but returning a list of boundary offsets instead of a list of sentences.

On T-Eng, the module responsible for dealing with sentence splitting is called `TE::DocAnnotator::Sentences`, and uses the structure outputted by `Lingua::EN::Sentence::Offsets` to produce `TE::Annotation::Sentence` annotation objects.

4.5.1.4 Tokens

The problem of text tokenization is very identical to sentence splitting: it is also a common task in text processing and there are many tools available to perform it [He and Kayaalp, 2006], but they return lists of tokens instead of their offsets.

A similar approach to sentence splitting was taken: an existing tool was chosen, analyzed and adapted to fit our needs. In this case, we chose a tokenizer script written by Josh Schroeder and bundled with a pack of tools available in the EuroParl website[4] [Koehn, 2005]. The resulting module was named `Lingua::EN::Tokenizer::Offsets`.

Similarly to the case of paragraphs and sentences, this module is used by T-Eng, in the `TE::DocAnnotator::Tokens` module, which transform its output in `TE::Annotation::Token` annotations.

[4]Europarl Parallel Corpus, Download section – `http://www.statmt.org/europarl/v7/tools.tgz`

4.5.2 Content

Content annotators are responsible for extracting information regarding the content of the document, generating annotations which are required to implement the `TE::Annotation::Content` role.

4.5.2.1 Numerical expressions

The extraction of numerical expressions has been covered in Chapter 3, which described the implementation of `Text::Math::NumExp`. This module is used in T-Eng's module `TE::DocAnnotator::NumExp` to produce annotations of the `TE::Annotation::NumExp` subclass, which comprise an additional **value** field to hold the value of the numerical expression (if any) determined by `Text::Math::NumExp`.

4.5.2.2 Domain ontology

The domain ontology described in Section contains information about parameter names, units of measurement and domain entities. As such, there is a T-Eng module, `TE::DocAnnotator::DomainOntology`, which is capable of generating several types of annotations. This module is responsible for using a domain ontology represented by a `TE::Ontology` object to generate `TE::Annotation::Unit`, `TE::Annotation::Parameter` and `TE::Annotation::Entity` objects when it recognizes units of measurement, parameter names or domain entities, respectively Annotations produced by this module present the following additional fields:

Ontology ID: A code which identifies the ontology responsible for this annotation.

Term ID: A code which identifies this term in the source ontology.

Name: The common name of this term.

4.5.2.3 Generic units of measurement

In addition to `TE::DocAnnotator::DomainOntology`, another module is capable of generating `TE::Annotation::Unit` objects: the `TE::DocAnnotator::GenericUnits` module. This module handles the generic units ontologies, represented in `TE::Ontology` objects. Annotations produced by this module include the same additional fields as the ones produced by `TE::DocAnnotator::DomainOntology`.

4.5.2.4 Third-party tools example: Linnaeus

Distinct domains have distinct needs in terms of annotated entities. Often, tools already exist to address those needs. In T-Eng, these third-party tools

can be wrapped in a module implementing the `TE::DocAnnotator` role, allowing T-Eng to be customized to any given domain.

In biomedical text mining, for example, there is often the need to extract mentions to specific organisms. As such, T-Eng includes a wrapper module built on top of Linnaeus, an open-source software tool for recognizing and normalizing species names which uses a dictionary-based approach implemented as a finite-state automaton [Gerner et al., 2010].

Linnaeus uses the NCBI Taxonomy [Wheeler et al., 2007, Benson et al., 1997] for the normalization of species names, and although is comes bundled with several dictionary files it allows the use of custom-made files as well.

Example 4.3 present an excerpt of the results of using Linnaeus to process a scientific article from the ecotechnologies domain. The first line provides the name of each column; each following line is a distinct entry containing:

1. the recognized organism unique NCBI taxonomic identifier;
2. the document where the organism was identified;
3. the start and end offsets of the recognized text;
4. the text containing the recognized organism.

```
#entity             #document  #start  #end  #text
species:ncbi:42354  paper.txt   1184   1208  Nitrosomonas oligotropha
species:ncbi:915    paper.txt   1367   1388  Nitrosomonas europaea
species:ncbi:51642  paper.txt   1389   1410  Nitrosococcus mobilis
species:ncbi:42354  paper.txt   4390   4414  Nitrosomonas oligotropha
species:ncbi:915    paper.txt   4527   4548  Nitrosomonas europaea
```

Example 4.3: Excerpt of *Linnaeus*' results for a scientific article [Limpiyakorn et al., 2006].

T-Eng's `TE::DocAnnotator::Linnaeus` processes the documents text using Linnaeus and parses the resulting output, generating `TE::Annotation::Organism` objects.

4.6 Document and annotation exportation

After the annotation and post-processing phases, the generated data must be passed on to the users. There are two main reasons for exporting data out of T-Eng:

- data will be fed as input to other tools, which will use it to perform further processing or mining not covered by T-Eng.

- data is ready to present to be presented to the end-user, e.g. biologists.

These two different uses for the data generated by T-Eng have distinct requirements, and as such the exporting format must present distinct features. Data exported to be further processed must present the following characteristics:

Formal language: in order to be easily processed by software, data must be written in some formal language rather than natural language.

Structured format: regardless of the specific format used, it must be easily to process by a structural processor.

Complete/exhaustive: In order to use techniques such as data mining, information must be as complete as possible. Besides, computers can easily analyze large amounts of data.

Examples of formats which can be used are comma-separated values (CSV) or XML schemas. Data exported for visualization, on the other hand, must present the following characteristics:

Readable language: users often find it easier to understand something if it is written in natural language than in a formal one.

Intuitive: data must be presented in intuitive formats, such as tables and graphs, to make it easy to assimilate.

Partial/relevant: more often than not, only the most relevant results should be presented (top occurrences, most significant values, etc) to avoid overloading the users with data.

Examples of formats for data visualization include HTML files, graphs and figures.

4.6.1 HTML

T-Eng includes a module, `DocExporter::HTML`, which is able to export documents and their respective annotations in HTML format. The result of exporting a document with this module is original text with the annotated elements color highlighted.

This transformation is performed using Template Toolkit, an all-Perl fast, flexible, powerful and extensible template processing system [Chamberlain et al., 2011], which provides an easy way to process template files, filling in embedded variable references with their equivalent values. Despite the fact that it can be used to process any kind of text documents, it is most often used for generating static and dynamic web content.

`TE::DocExporter::HTML` replaces the annotated elements by the correspondent HTML code provided by Template Toolkit, which also provides a default style-sheet containing the visual formatting options. Due to the flexibility provided by Template Toolkit, styling options can be completely changed, which results in a HTML document easily integrated in any framework. An example of a document exported to HTML format is presented in Chapter 5 in Figure 5.1.

4.6.2 Annotations

`PostProc::Parameters`, T-Eng's module responsible for finding numerical parameters, acts based on the annotations generated by other modules, which are passed directly in their internal object representation. Third party tools however, need to be integrated (via a wrapper module integrated in T-Eng), or else the annotations have to be provided in other format.

The `TE::Annotation` and `TE::ListOfAnnotations` classes includes methods to export their objects to Javascript Object Notation (JSON) [Crockford, 2006] and YAML [Ben-Kiki et al., 2008], data interchange formats characterized by their simplicity and readability. Example 5.2, in Section 5.3, presents an example of annotations exported in JSON format.

4.7 Web interface

T-Eng pluggable modules and extension mechanisms allow the construction of a customized processing system, supplying text mining developers parts which they can assemble to adapt the framework to a given domain and workflow.

This modularity, however, may sometimes be lost on another type of end-users, which require only a "black box" system, capable of accepting their articles as input and outputting the results after performing a default chain of actions, and do not possess the knowledge or the need to change the internal mechanisms of T-Eng.

For the latter users, two T-Eng's web interfaces were developed: one allows to implement a domain-specific base of knowledge, allowing the navigation within a database of articles and annotations; the other implements an annotation service which allows the submission of articles and domain ontologies for processing. These interfaces are implemented as Dancer applications , and use the same object structures as the command-line interface.

4.7.1 Data storage

T-Eng is implemented using Moose, a " postmodern object system for Perl 5" [Styn, 2011, Moose Cabal, 2012], which allows to write Perl code using

the Object-Oriented (OO) paradigm. Moose provides inheritance and role extension mechanisms which, despite being extremely useful at runtime, add complexity when it comes to data persistence. In fact, most traditional relational databases are not inherently prepared to deal with object inheritance, which makes the mapping between Moose objects and table rows a little bit difficult. This problem has been debated previously [Buneman and Atkinson, 1986]; possible solutions include adding complexity to the relational database (creating additional tables to reflect the inheritance mechanisms) or using a database with inheritance support – possibly a non-relational database [Stonebraker, 2010, Seeger and Ultra-Large-Sites, 2009].

ElasticSeach is a distributed search server built on top of Apache Lucene [Jakarta, 2004] which supports schema-less indexing and advanced search features [Global, 2012, Infochimps, 2012]. `Elastic::Model` [Gormley, C., 2012] is a Perl distribution which provides a "NoSQL document store with full text search for Moose objects using ElasticSearch as a backend".

4.7.2 Annotation service

This interface consists of a web-based platform for annotating user-uploaded papers. Users are able to create their own set of articles by filling in a form which asks them for the following data:

Uploader name: Allows to identify who uploaded this set of articles.

Set name: Identifies the set, allowing the user to easily manage the annotation of several distinct sets.

Set domain: Identifies the domain to which the articles belong to.

Additionally, the user is asked to upload the articles he wishes to annotate, in PDF format. Alternatively, the articles' PubMed identifier may be provided, and they will be automatically retrieved by the web application. Users may also upload ontology files, in the OBO format, to be used as domain ontologies in the annotation process. Figure 4.6 presents the article upload form and Figure 4.7 presents the PubMed identifier version.

Once the set information has been provided, articles are fetched and annotated in a background job, built using `TheSchwartz`, a Perl modules which implements a reliable job queue stored in a database [Six Apart Ltd., 2007]. The results of the processing are made available through the other interface, the base of knowledge.

4.7.3 Base of knowledge

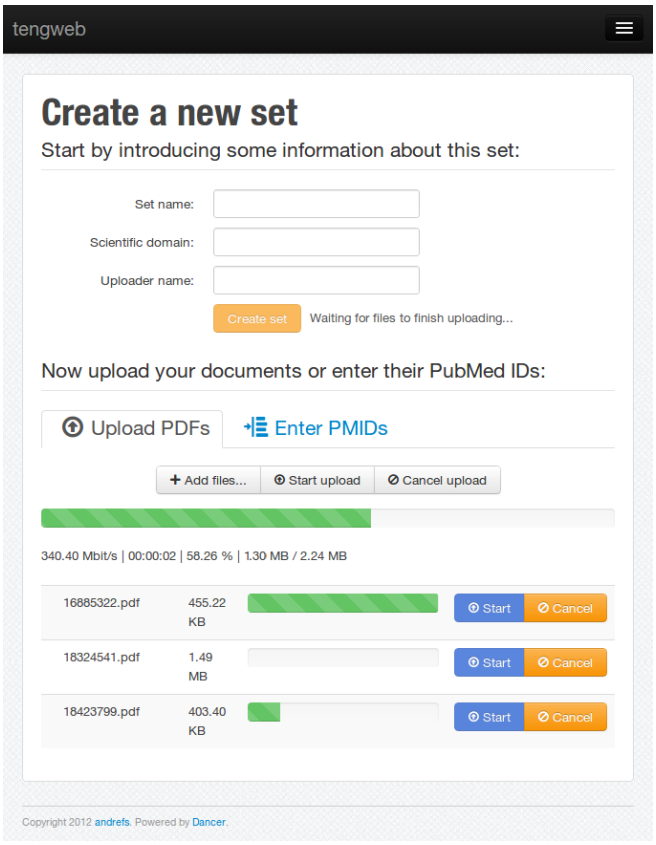


Figure 4.6: PDF files upload form in T-Eng's web interface.

Figure 4.7: PubMed ID in T-Eng's web interface.

Chapter 5

A case study: the ecotechnologies domain

This chapter reports the application of the tools previously mentioned in the ecotechnology domain. It starts by describing the field and contextualizing its need for automatic extraction of parameters, and proceeds to describe the creation of an ontology for wastewater parameters. Then the application of T-Eng to the most relevant articles related with wastewater treatment is described, completed with a discussion about the results obtained.

5.1 The ecotechnologies domain

The field of ecotechnologies for wastewater treatment is being challenged by a gap of communication between wastewater scientists and engineers concerning the way to integrate the knowledge about microbial ecology acquired by the first ones in the design and optimization of the work that the latest perform in Waste Water Treatment Plants (WWTPs).

5.2 Buiding a wastewater ontology

Chapters 3 and 4 mentioned the use of domain-specific ontologies, to be provided by users, which allowed to adapt the system to a given field of studies by providing common parameter names, units of measure and relevant entities within that domain. For this case study, an ontology was designed, populated and curated by an expert. Example 5.1 presents two examples of entries contained in the wastewater ontology, and Table 5.3 summarizes the total number of distinct elements in each class of elements.

- the first entry classifies activated sludge as a domain entity and indicates nitrifying activated sludge as a (narrower) synonym;

Table 5.1: Number of entries for each type of entity in the wastewater ontology.

Type of entity	Number of entries
Biological entities	24
Units of measure	6
Variable names	6

- the second entry refers `cells/l` as a unit of measure, listing `cells(-1)`, `cells-1` and `copies/l` as equivalent synonyms.

This vocabulary, in addition to being used to generate the code responsible for annotating parameter names and units of measure in the document, and detecting entities which help determining the document's relevance, is also a code-independent platform to assist in the interchange of information between researchers and engineers.

```
[Term]
id: ID:0000007
name: activated sludge
synonym: "nitrifying activated sludge" NARROW []
is_a: ID:0000026 ! domain_entities
```

```
[Term]
id: ID:0000014
name: cells/l
synonym: "cells(-1)" RELATED []
synonym: "cells-1" RELATED []
synonym: "copies/l" RELATED []
is_a: ID:0000002 ! units_of_measure
```

Example 5.1: Excerpt of the wastewater ontology.

5.3 Processing the most relevant articles

In order to test the described tool, a set of more than fifty scientific articles from the wastewater domain were selected by an expert, based on the importance and diversity of the parameters previously referred. The articles were fetched in PDF format, given the interest in processing their full text and their unavailability in other formats.

Taking advantage of the modularity of T-Eng, a small Perl script was written which implemented the workflow necessary to import both the documents and the wastewater ontology, annotate the documents, and perform the neighborhood-based numerical parameter extraction. The script's output consisted of all the generated annotations (see Example 5.2) and the numerical parameters extracted (Example **??**).

```
[
   {
      "sentence" : [7678,7796], "segment" : [7762,7772],
      "type"     : "numexp",    "text"    : "750 and 45",
   },{
      "sentence" : [7678,7796], "segment" : [7773,7775],
      "type"     : "unit",      "text"    : "mg",
   },{
      "sentence" : [7678,7796], "segment" : [7776,7781],
      "type"     : "unit",      "text"    : "liter",
   }
]
```

Example 5.2: Excerpt of generated annotations in JSON format.

A similar workflow was also tested using the web-based annotation service described in Section 4.7.2. The same papers were uploaded, also in PDF format, as well as the wastewater ontology. The workflow resulted in HTML versions of the papers with highlighted elements (Figure 5.1) and tables presenting a summary of the results (Figure 5.2), which were made available using the functionalities presented in Section 4.7.3.

14682578 - Molecular assessment of ammonia- and nitrite-oxidizing bacteria in full-scale activated sludge wastewater treatment plants.

Nitrification was assessed in two full-scale wastewater treatment plants (WWTPs) over time using molecular methods. Both WWTPs employed a complete-mix suspended growth, aerobic activated sludge process (with biomass recycle) for combined carbon and nitrogen treatment. However, one facility treated primarily municipal wastewater while the other only industrial wastewater. Real time PCR assays were developed to determine copy numbers for total 16S rDNA (a measure of biomass content), the amoA gene (a measure of ammonia-oxidizers), and the Nitrospira 16S rDNA gene (a measure of nitrite-oxidizers) in mixed liquor samples. In both the municipal and industrial WWTP samples, total 16S rDNA values were approximately 2-9*10^13 copies/L and Nitrospira 16S rDNA values were 2-4*10^10 copies/L. amoA gene concentrations averaged 1.73*10^9 copies/L (municipal) and 1.06*10^10 copies/L (industrial), however, assays for two distinct ammonia oxidizing bacteria

Figure 5.1: Example of article abstract with highlighted elements.

Figure 5.2: Example of table with numerical parameters extracted from a document.

5.4 Results

Processing the previously mentioned set of articles generated a vast amount of annotations whose analysis provides useful insights about the articles' contents. Table 5.2 presents the total number of content annotations generated

for the whole set, as well as average numbers per article. Figure 5.3 presents the top organims mentioned in the articles.

Table 5.2: Occurrences of of annotations in the wastewater set (total and average per article).

Type of annotation	**Total occurrences**	**Average occurrences**
Sentences	19,525	336.6
Organisms	1,191	20.5
Numerical expressions	29,983	516.9
Terms from WW ontology	14,879	256.5
Units of measure	23,522	405.5

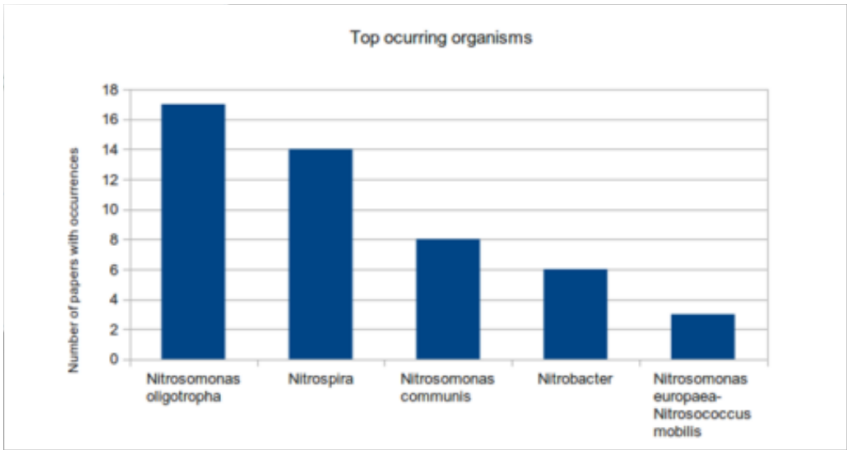


Figure 5.3: Occurrency of organisms in wastewater treatment articles.

Manual inspection of the results obtained have shown that the performance of the recognition of units of measure is affected by two main problems, one caused by the generic units ontologies, and the other by the recognition process.

Generic unit ontologies contain, by definition, units from several distinct domains. This means that T-Eng will attempt to match all units in the generic ontology, regardlessly of the domain of the article. Due to the fact that unit symbols are often conflicting with other words (for example, most of the letters from the alphabet are used as symbols for a given unit), there are a considerable amount of false positive matches, decreasing substantially the precision of the recognizer.

The unit recognizer uses the elements from the ontologies to generate recognition patterns, simplifying the unit names and synonyms to allow a more relaxed matching (see Section 3.4.2). However, even using this approach, several unit mentions remain unrecognized, mainly due to questions related to spacing and line breaks and UTF-8 characters. The existence of these false negative results decreases the precision of the recognizer.

From the results obtained, it is also possible to detect a high frequency of partial matches. Example 5.3 presents a sentence extracted from an article, where all the elements of interest have been manually emphasized. Table 5.3, on the other hand, provides a list of the annotations generated by T-Eng for the same sentence, where it is possible to observe that the numbers belonging to numerical ranges have been recognized as independent elements, and only part of the `copies per nanogram DNA` has been identified.

```
1 Real-time PCR results demonstrated that the levels of AOB amoA
2 varied from 2.9×103 to 2.3×105 copies per nanogram DNA, greatly
3 (about 60 times) higher than those of AOA, which ranged from
4 1.7×102 to 3.8×103 copies per nanogram DNA.
```

Example 5.3: Sentence from an article from the wastewater domain [Jin et al., 2010], and correspondent annotations generated.

Table 5.3: Sample of annotations generated in the processing of a sentence (Example 5.3).

Sentence start	**Sentence end**	**Segment start**	**Segment end**	**Type**	**Text**
991	1222	1001	1004	Term	PCR
991	1222	1045	1048	Term	AOB
991	1222	1066	1073	NumExp	2.9×103
991	1222	1077	1084	NumExp	2.3×105
991	1222	1096	1104	Unit	nanogram
991	1222	1118	1134	NumExp	(about 60 times)
991	1222	1156	1159	Term	AOA
991	1222	1179	1186	NumExp	1.7×102
991	1222	1190	1197	NumExp	3.8×103
991	1222	1209	1217	Unit	nanogram

Chapter 6

Conclusions and future work

This chapter presents a summary of the work performed in the context of this dissertation, establishing some conclusions about the usefulness and practical utility of the tools developed and the nature of the problems tackled.

Additionally, future work possibilities are discussed, including specific features and methods which could be added to the existing tools.

6.1 Conclusions

The modules developed are publicly available on CPAN[1]. Releasing the tools as open source software required additional work in order to guarantee that the tools could be used and installed in other computers, and that the community standards were followed. It also means that they are continuously being maintained and improved, as more feature requests and bug reports are being issued.

6.2 Future Work

All the methods and tools described and implemented in the context of this dissertation can be improved and extended. This section lists some features which are planned already or under consideration.

[1] Comprehensive Perl Archive Network – `http://cpan.org`

References

Adobe Systems Inc. ISO 32000-1:2008 – Document management – Portable document format – Part 1: PDF 1.7. Standard, ISO/IEC, 1. July 2008. URL `http://wwwimages.adobe.com/www.adobe.com/content/dam/Adobe/en/devnet/pdf/pdfs/PDF32000_2008.pdf`. **Cited** on pages 10 and 39.

Adobe Systems Inc. Portable document format (pdf) - a history of trust. http://web.archive.org/web/20100825004119/http://www.adobe.com/pdf/about/history/, 2010. **Cited** on pages 10 and 39.

S. Agarwal and H. Yu. Automatically classifying sentences in full-text biomedical articles into introduction, methods, results and discussion. *Bioinformatics*, 25 (23):3174, 2009. **Cited** on page 12.

E. Allen, D. Chase, V. Luchangco, J.W. Maessen, and G.L. Steele Jr. Object-oriented units of measurement. *ACM SIGPLAN Notices*, 39(10):384–403, 2004. **Cited** on page 20.

U. Ammon. *The dominance of English as a language of science: Effects on other languages and language communities*, volume 84. De Gruyter Mouton, 2001. **Cited** on page 1.

M.E. Anders and D.P. Evans. Comparison of pubmed and google scholar literature searches. *Respiratory care*, 55(5):578–583, 2010. **Cited** on page 9.

Antezana, E. OBO::Parser::OBOParser Perl module, 2006. Retrieved in October, 2012 from `https://metacpan.org/module/OBO::Parser::OBOParser`. **Cited** on page 42.

G. Antoniou and F. Harmelen. Web ontology language: Owl. *Handbook on ontologies*, pages 91–110, 2009. **Cited** on page 17.

Apache. Apache solr. http://lucene.apache.org/solr. **Cited** on page 8.

N. Bach and S. Badaskar. A review of relation extraction. *Literature review for language and statistics II*, 2007. **Cited** on pages 17 and 18.

John J. Back, Chris Densham, Rob Edgecock, and Gersende Prior. Particle production and energy deposition studies for the neutrino factory target station, 2012. **Cited** on page 32.

B. Baldwin and B. Carpenter. LingPipe. http://alias-i. com/lingpipe (accessed June 8, 2010). **Cited** on page 18.

A. Barbosa-Silva, T.G. Soldatos, I.L.F. Magalhaes, G.A. Pavlopoulos, J.F. Fontaine, M.A. Andrade-Navarro, R. Schneider, and J.M. Ortega. LAITOR- Literature Assistant for Identification of Terms co-Occurrences and Relationships. *BMC bioinformatics*, 11(1):70, 2010. **Cited** on page 18.

Barr, G. Scalar::Util Perl module, 1999. Retrieved in October, 2012 from `http://metacpan.org/module/Scalar::Util`. **Cited** on page 31.

C.R. Batchelor and P.T. Corbett. Semantic enrichment of journal articles using chemical named entity recognition. In *Proceedings of the 45th Annual Meeting of the ACL on Interactive Poster and Demonstration Sessions*, pages 45–48. Association for Computational Linguistics, 2007. **Cited** on page 15.

O. Ben-Kiki, C. Evans, and I. dot Net. Yaml 1.2 specification, 2008. **Cited** on page 52.

D.A. Benson, M.S. Boguski, D.J. Lipman, and J. Ostell. Genbank. *Nucleic acids research*, 25(1):1–6, 1997. **Cited** on pages 15 and 50.

Diego Berrueta and Luis Polo. Measurement units ontology. `http://forge.morfeo-project.org/wiki_en/index.php/Measurement_Units_Ontology`, 2012. **Cited** on page 19.

K. Bontcheva, V. Tablan, D. Maynard, and H. Cunningham. Evolving gate to meet new challenges in language engineering. *Natural Language Engineering*, 10(3-4): 349–373, 2004. **Cited** on pages 15 and 18.

T. Bray, J. Paoli, C.M. Sperberg-McQueen, E. Maler, F. Yergeau, et al. Extensible markup language (xml) 1.0, 2000. **Cited** on page 11.

T.M. Breuel. The ocropus open source ocr system. In *Proceedings IS&T/SPIE 20th Annual Symposium*, volume 2008. Citeseer, 2008. **Cited** on page 11.

P. Buneman and M. Atkinson. Inheritance and persistence in database programming languages. In *ACM SIGMOD Record*, volume 15, pages 4–15. ACM, 1986. **Cited** on page 53.

Calder, E. Lingua::EN::Numericalize Perl module, 2003. Retrieved in October, 2012 from `https://metacpan.org/module/Lingua::EN::Numericalize`. **Cited** on page 30.

N.J. Cecchino. Google scholar. *Journal of the Medical Library Association: JMLA*, 98(4):320, 2010. **Cited** on page 9.

D. Chamberlain, D. Cross, and A. Wardley. *Perl Template Toolkit*. O'Reilly Media, 2011. **Cited** on page 51.

S. Chaudhuri, K. Ganjam, V. Ganti, and R. Motwani. Robust and efficient fuzzy match for online data cleaning. In *Proceedings of the 2003 ACM SIGMOD international conference on Management of data*, pages 313–324. ACM, 2003. **Cited** on page 16.

Google Code. Issue 109 on google ajax apis: Please add google scholar to the api, including # of citations and bibtex reference. http://code.google.com/p/google-ajax-apis/issues/detail?id=109, February 2012. **Cited** on page 9.

A.M. Cohen and W.R. Hersh. A survey of current work in biomedical text mining. *Briefings in Bioinformatics*, 6(1):57, 2005. **Cited** on pages 13 and 18.

P. Corbett and P. Murray-Rust. High-throughput identification of chemistry in life science texts. *Computational Life Sciences II*, pages 107–118, 2006. **Cited** on pages 15 and 19.

O. Corcho and A. Gómez-Pérez. A roadmap to ontology specification languages. *Knowledge Engineering and Knowledge Management Methods, Models, and Tools*, pages 80–96, 2000. **Cited** on pages 16 and 17.

Cozens, S. Lingua::EN::FindNumber Perl module, 2003. Retrieved in October, 2012 from `http://metacpan.org/module/Lingua::EN::FindNumber`. **Cited** on page 30.

D. Crockford. RFC 4267 – The application/json media type for javascript object notation (JSON). 2006. **Cited** on page 52.

H. Cunningham, D. Maynard, K. Bontcheva, and V. Tablan. A framework and graphical development environment for robust nlp tools and applications. In *Proceedings of 40th Annual Meeting of the Association for Computational Linguistics (ACL2002)*, pages 168–175, 2002. **Cited** on pages 15 and 18.

B.H. Davis, T. Schroeder, P.S. Yarmolenko, F. Guilak, M.W. Dewhirst, and D.A. Taylor. An in vitro system to evaluate the effects of ischemia on survival of cells used for cell therapy. *Annals of biomedical engineering*, 35(8):1414–1424, 2007. **Cited** on page 32.

J.E. Dayhoff and J.M. DeLeo. Artificial neural networks. *Cancer*, 91(S8):1615–1635, 2001. **Cited** on page 20.

C.A.B. de Mello and R.D. Lins. A comparative study on ocr tools. In *Vision Interface'99 Conference*, pages 224–231. Citeseer, 1999. **Cited** on page 11.

Joshua C. Dolence, Adam Burrows, Jeremiah W. Murphy, and Jason Nordhaus. Dimensional dependence of the hydrodynamics of core-collapse supernovae, 2012. **Cited** on page 29.

Elsevier. Scirus white paper: How scirus works. Technical report, August 2004. **Cited** on page 8.

Laura M. Felter. Google scholar, scirus, and the scholarly search revolution. *Searcher*, 13(2):43 – 48, 2005. ISSN 10704795. URL `http://search.ebscohost.com/login.aspx?direct=true&db=a9h&AN=16014815&site=ehost-live&scope=site`. **Cited** on page 9.

W.B. Frakes. Stemming algorithms. *Information Retrieval Data Structures and Algorithms*, pages 131–160, 1992. **Cited** on page 16.

K. Fundel, R. Kuffner, and R. Zimmer. RelEx–relation extraction using dependency parse trees. *Bioinformatics*, 23(3):365, 2007. **Cited** on page 18.

Quentin Geissmann. OpenCFU, a New Free and Open-Source Software to Count Cell Colonies and Other Circular Objects, 2012. **Cited** on page 29.

M. Gerner, G. Nenadic, and C. Bergman. Linnaeus: A species name identification system for biomedical literature. *BMC bioinformatics*, 11(1):85, 2010. **Cited** on pages 15 and 50.

GetCITED. Getcited home page. `www.getcited.org` (accessed February 15 2012), 2012. **Cited** on page 8.

D. Giustini and E. Barsky. A look at google scholar, pubmed, and scirus: comparisons and recommendations. *Journal of the Canadian Health Libraries Association*, 26(3):85–89, 2005. **Cited** on page 9.

TNR Global. Flexible analytics for large data sets using elasticsearch. Technical report, 2012. **Cited** on page 53.

Andreas Gohr. Linux ocr software comparison. `http://www.splitbrain.org/blog/2010-06/15-linux_ocr_software_comparison`, June 2010. **Cited** on page 11.

D.T. Goldman and RJ Bell. International system of units (SI). *NASA STI/Recon Technical Report N*, 87:20444, 1986. **Cited** on page 32.

Gormley, C. Elastic::Search Perl module, 2012. Retrieved in October, 2012 from `https://metacpan.org/module/Elastic::Model`. **Cited** on page 53.

T. Greenhalgh. How to read a paper: the medline database. *Bmj*, 315(7101):180–183, 1997. **Cited** on pages 9 and 40.

T.R. Gruber and G.R. Olsen. An ontology for engineering mathematics. In *Fourth International Conference on Principles of Knowledge Representation and Reasoning*, pages 258–269. Morgan Kaufmann, 1994. **Cited** on page 20.

A.M. Gulisano, P. Demoulin, S. Dasso, and L. Rodriguez. Expansion of magnetic clouds in the outer heliosphere. *Astronomy & Astrophysics*, 543, 2012. **Cited** on page 29.

R.W. Hamming. Error detecting and error correcting codes. *Bell System Technical Journal*, 29(2):147–160, 1950. **Cited** on page 16.

Y. He and M. Kayaalp. A comparison of 13 tokenizers on medline. *Bethesda, MD: The Lister Hill National Center for Biomedical Communications*, 2006. **Cited** on page 48.

S. Heinen, B. Thielen, and D. Schomburg. Kid-an algorithm for fast and efficient text mining used to automatically generate a database containing kinetic information of enzymes. *BMC bioinformatics*, 11(1):375, 2010. **Cited** on pages xi, 21 and 23.

Hess, J. Lingua::EN::Words2Nums Perl module, 2001. Retrieved in October, 2012 from `http://metacpan.org/module/Lingua::EN::Words2Nums`. **Cited** on page 30.

K.M. Hettne, A.J. Williams, E.M. van Mulligen, J. Kleinjans, V. Tkachenko, and J.A. Kors. Automatic vs. manual curation of a multi-source chemical dictionary: the impact on text mining. 2010. **Cited** on page 14.

Infochimps. The power of elasticsearch. Technical report, 2012. **Cited** on page 53.

ISO. Information technology – Open Document Format for Office Applications (OpenDocument) v1.0 ISO/IEC 26300:2006. Technical report, ISO/IEC, 2006. **Cited** on page 11.

P. Jacsó. Google scholar: the pros and the cons. *Online Information Review*, 29(2): 208–214, 2005. **Cited** on page 9.

P. Jacsó. Google scholar revisited. *Online information review*, 32(1):102–114, 2008. **Cited** on page 9.

A. Jakarta. Apache lucene-a high-performance, full-featured text search engine library, 2004. **Cited** on page 53.

D.M. Jessop, S.E. Adams, E.L. Willighagen, L. Hawizy, and P. Murray-Rust. Oscar4: a flexible architecture for chemical text-mining. *Journal of Cheminformatics*, 3: 41, 2011. **Cited** on page 15.

T. Jin, T. Zhang, and Q. Yan. Characterization and quantification of ammonia-oxidizing archaea (aoa) and bacteria (aob) in a nitrogen-removing reactor using t-rflp and qpcr. *Applied microbiology and biotechnology*, 87(3):1167–1176, 2010. **Cited** on page 59.

R.B. Kaplan. English–the accidental language of science. *The Dominance of English as a Language of Science: Effects on Other Languages and Language Communities. Berlin*, pages 3–26, 2001. **Cited** on page 1.

G. Klyne and J.J. Carroll. Resource description framework (rdf): Concepts and abstract syntax. *Changes*, 10(February):1–20, 2004. **Cited** on page 17.

P. Koehn. Europarl: A parallel corpus for statistical machine translation. In *MT summit*, volume 5, 2005. **Cited** on page 48.

W. Kraaij and R. Pohlmann. Viewing stemming as recall enhancement. In *Proceedings of the 19th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 40–48. ACM, 1996. **Cited** on page 16.

M. Krallinger, M. Vazquez, F. Leitner, D. Salgado, A. Chatr-aryamontri, A. Winter, L. Perfetto, L. Briganti, L. Licata, M. Iannuccelli, et al. The protein-protein interaction tasks of biocreative iii: classification/ranking of articles and linking bio-ontology concepts to full text. *BMC bioinformatics*, 12(Suppl 8):S3, 2011. **Cited** on page 48.

W. Kuhn. A functional ontology of observation and measurement. *GeoSpatial Semantics*, pages 26–43, 2009. **Cited** on page 20.

U. Leser and J. Hakenberg. What makes a gene name? Named entity recognition in the biomedical literature. *Briefings in Bioinformatics*, 6(4):357, 2005. **Cited** on page 14.

V.I. Levenshtein. Binary codes capable of correcting deletions, insertions, and reversals. In *Soviet physics doklady*, volume 10, pages 707–710, 1966. **Cited** on page 16.

H. Li, I. Councill, W.C. Lee, and C.L. Giles. Citeseerx: an architecture and web service design for an academic document search engine. In *Proceedings of the 15th international conference on World Wide Web*, pages 883–884. ACM, 2006a. **Cited** on page 8.

H. Li, I.G. Councill, L. Bolelli, D. Zhou, Y. Song, W.C. Lee, A. Sivasubramaniam, and C.L. Giles. Citeseerx - a scalable autonomous scientific digital library. *InfoScale'06*, 2006b. **Cited** on page 8.

Priit Lilleleht. How to extract text from images: a comparison of 10 free ocr tools. http://www.freewaregenius.com/2011/11/01/how-to-extract-text-from-images-a-comparison-of-free-ocr-tools/, November 2011. **Cited** on page 11.

T. Limpiyakorn, Y. Shinohara, F. Kurisu, and O. Yagi. Communities of ammonia-oxidizing bacteria in activated sludge of various sewage treatment plants in tokyo. *FEMS microbiology ecology*, 54(2):205–217, 2006. **Cited** on page 50.

A. Lourenço, M. Conover, A. Wong, A. Nematzadeh, F. Pan, H. Shatkay, and L.M. Rocha. A linear classifier based on entity recognition tools and a statistical approach to method extraction in the protein-protein interaction literature. *BMC bioinformatics*, 12(Suppl 8):S12, 2011. **Cited** on page 48.

A. Mansouri, L.S. Affendey, and A. Mamat. Named entity recognition approaches. *IJCSNS*, 8(2):339, 2008. **Cited** on page 14.

Chris A. Mattmann and Jukka L. Zitting. *Tika in Action*. Manning Publications Co., 1st edition, 2011. URL `http://www.manning.com/mattmann/`. **Cited** on pages 12 and 39.

J.R. McEntyre, S. Ananiadou, S. Andrews, W.J. Black, R. Boulderstone, P. Buttery, D. Chaplin, S. Chevuru, N. Cobley, L.A. Coleman, et al. Ukpmc: a full text article resource for the life sciences. *Nucleic acids research*, 39(suppl 1):D58, 2011. **Cited** on page 8.

G. McKiernan. E-profile: Scirus: for scientific information only. *Library Hi Tech News*, 22(3):18–25, 2005. **Cited** on page 8.

N. Medeiros. Introducing scirus: Elsevier's shot at the title. *OCLC Systems & Services*, 18(3):121–124, 2002. **Cited** on page 8.

Moose Cabal. Moose Perl distribution, 2012. Retrieved in October, 2012 from `http://metacpan.org/module/Moose`. **Cited** on page 52.

H. Müller and F. Mancuso. Identification and analysis of co-occurrence networks with NetCutter. *PLoS ONE*, 3(9), 2008. **Cited** on page 18.

D. Nadeau and S. Sekine. A survey of named entity recognition and classification. *Lingvisticae Investigationes*, 30(1):3–26, 2007. **Cited** on page 14.

G. Navarro. A guided tour to approximate string matching. *ACM computing surveys (CSUR)*, 33(1):31–88, 2001. **Cited** on page 16.

NCBI. PMC Open Access Subset. http://www.ncbi.nlm.nih.gov/pmc/tools/openftlist/, 2012a. **Cited** on page 10.

NCBI. Pubmed central open archives service. http://www.ncbi.nlm.nih.gov/pmc/tools/oai/, 2012b. **Cited** on page 10.

D. Noonburg. xpdf: A c++ library for accessing pdf, 2001. **Cited** on pages 12 and 39.

G.S. Novak Jr. Conversion of units of measurement. *Software Engineering, IEEE Transactions on*, 21(8):651–661, 1995. **Cited** on page 23.

OBO Phenotype Group. PATO – Phenotypic Quality Ontology. `http://obofoundry.org/wiki/index.php/PATO:About`, 2006. **Cited** on page 19.

Jeff Okamoto, Dean Roehrich, Malcolm Beattie, Andreas Koenig, Paul Hudson, Ilya Zakharevich, Paul Marquess, Neil Bowers, Matthew Green, Tim Bunce, Spider Boardman, Ulrich Pfeifer, Stephen McCamant, and Gurusamy Sarathy. perlapi - autogenerated documentation for the perl public API, 1997. Retrieved in October, 2012 from `http://perldoc.perl.org/perlapi.html#SV-Body-Allocation`. **Cited** on page 31.

D.L. Olson and D. Delen. *Advanced data mining techniques*. Springer Verlag, 2008. ISBN 3540769161. **Cited** on page 20.

C.D. Paice. An evaluation method for stemming algorithms. In *Proceedings of the 17th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 42–50. Springer-Verlag New York, Inc., 1994. **Cited** on page 16.

J. Paoli, I. Valet-Harper, A. Farquhar, and I. Sebestyen. Ecma-376 office open xml file formats. *URL http://www. ecmainternational. org/publications/standards/Ecma-376. htm*, 2006. **Cited** on page 11.

Ioannis Papastathopoulos and Jonathan A. Tawn. Stochastic ordering under conditional modelling of extreme values: Drug-induced liver injury, 2012. **Cited** on page 29.

WC Peh and KH Ng. Basic structure and types of scientific papers. *Singapore medical journal*, 49(7):522, 2008. **Cited** on pages 12 and 45.

JRG Pulido, MAG Ruiz, R. Herrera, E. Cabello, S. Legrand, and D. Elliman. Ontology languages for the semantic web: A never completely updated review. *Knowledge-Based Systems*, 19(7):489–497, 2006. **Cited** on page 17.

R.H. Rand. Introduction to maxima. *Cornel University*, 25, 2005. **Cited** on page 22.

H. Rijgersberg, M. Wigham, and JL Top. How semantics can improve engineering processes: A case of units of measure and quantities. *Advanced Engineering Informatics*, 25(2):276–287, 2011. **Cited** on page 19.

H. Rijgersberg, M. van Assem, and J. Top. Ontology of units of measure and related concepts. *Semantic Web*, 2012. **Cited** on page 19.

C. J. Van Rijsbergen. *Information Retrieval*. Butterworth-Heinemann, Newton, MA, USA, 2nd edition, 1979. ISBN 0408709294. **Cited** on page 20.

N. Robinson. A Comparison of Utilities for converting from Postscript or Portable Document Format to Text. Technical report, CERN-OPEN-2001, 2001. **Cited** on pages 12 and 39.

Michel Rodriguez. XML::Twig Perl module, 2012. Retrieved in February, 2012 from `http://search.cpan.org/perldoc?XML::Twig`. **Cited** on page 12.

B.D. Rosenblum. NLM Journal Publishing DTD Flexibility: How and Why Applications of the NLM DTD Vary Based on Publisher-Specific Requirements. 2010. **Cited** on pages 11, 40 and 45.

A. Santos, R. Nogueira, and A. Lourenço. Applying a text mining framework to the extraction of numerical parameters from scientific literature in the biotechnology domain. *Advances in Distributed Computing and Artificial Intelligence Journal (ADCAIJ)*, 1:1–8, June 2012. **Cited** on page 27.

J. Schulenburg. Gocr: Open source character recognition. http://jocr.sourceforge.net, 2004. **Cited** on page 11.

S. Schulz, H. Stenzhorn, M. Boeker, and B. Smith. Strengths and limitations of formal ontologies in the biomedical domain. *Revista Electronica De Comunicacao, Informacao & Inovacao EM Saude*, 3(1):31, 2009. **Cited** on page 19.

M. Seeger and S. Ultra-Large-Sites. Key-value stores: a practical overview. *Computer Science and Media*, 2009. **Cited** on page 53.

B. Settles. ABNER: An open source tool for automatically tagging genes, proteins, and other entity names in text. *Bioinformatics*, 21(14):3191–3192, 2005. **Cited** on page 14.

Y. Shinyama. PDFMiner: Python PDF parser and analyzer, 2010. **Cited** on pages 13 and 40.

Alberto Simões. Text::RewriteRules Perl module, 2012a. Retrieved in February, 2012 from `http://search.cpan.org/perldoc?Text::RewriteRules`. **Cited** on page 22.

Alberto Simões. XML::DT Perl module, 2012b. Retrieved in February, 2012 from `http://search.cpan.org/perldoc?XML::DT`. **Cited** on page 12.

Six Apart Ltd. TheSchwartz Perl module, 2007. Retrieved in October, 2012 from `https://metacpan.org/module/TheSchwartz`. **Cited** on page 53.

B. Smith, M. Ashburner, C. Rosse, J. Bard, W. Bug, W. Ceusters, L.J. Goldberg, K. Eilbeck, A. Ireland, C.J. Mungall, et al. The obo foundry: coordinated evolution of ontologies to support biomedical data integration. *Nature biotechnology*, 25(11):1251–1255, 2007. **Cited** on pages 19 and 42.

R. Smith. An overview of the tesseract ocr engine. In *Document Analysis and Recognition, 2007. ICDAR 2007. Ninth International Conference on*, volume 2, pages 629–633. Ieee, 2007. **Cited** on page 11.

L.B. Sollaci and M.G. Pereira. The introduction, methods, results, and discussion (imrad) structure: a fifty-year survey. *Journal of the Medical Library Association*, 92(3):364, 2004. **Cited** on page 13.

M. Stonebraker. Sql databases v. nosql databases. *Communications of the ACM*, 53 (4):10–11, 2010. **Cited** on page 53.

H.V. Styn. Moose. *Linux Journal*, 2011(209):8, 2011. **Cited** on page 52.

P. Suber. Open access overview. *Exploring Open Access: A Practice Journal*, 1(1): 14, 2009. **Cited** on page 9.

A.J. Szanser. A study of the paragraph structure. *Statistical Methods in Linguistics*, 8(2):70–79, 1973. **Cited** on page 46.

J. Tamames and V. De Lorenzo. Envmine: A text-mining system for the automatic extraction of contextual information. *BMC bioinformatics*, 11(1):294, 2010. **Cited** on pages xi, 21 and 22.

O. M. Ulyanov, S. M. Andrievsky, V. F. Gopka, and A. V. Shavrina. Some evolutionary aspects of the binary stellar systems containing neutron star, 2012. **Cited** on page 32.

R.C. Weissberg. Given and new: Paragraph development models from scientific english. *Tesol Quarterly*, 18(3):485–500, 1984. **Cited** on page 46.

D.L. Wheeler, T. Barrett, D.A. Benson, S.H. Bryant, K. Canese, V. Chetvernin, D.M. Church, M. DiCuccio, R. Edgar, S. Federhen, et al. Database resources of the national center for biotechnology information. *Nucleic acids research*, 35 (suppl 1):D5–D12, 2007. **Cited** on pages 15 and 50.

World Wide Web Consortium. Ontology for units of measure. Retrieved February 20th, 2012 from `http://www.w3.org/2007/ont/unit`, 2007. **Cited** on pages 19 and 20.

Yona, S. Lingua::EN::Sentence Perl module, 2001. Retrieved in October, 2012 from `https://metacpan.org/module/Lingua::EN::Sentence`. **Cited** on page 48.

Appendix A

Software Documentation

This section includes documentation for the commands and modules developed Notice that these tools are being developed continuously: so, for up-to-date documentation check the current versions available at CPAN – `http://search.cpan.org`.

A.1 Software Installation

The Perl modules presented in this document are available at CPAN – `http://cpan.org`, and, as such, can be installed with any of the CPAN modules installation utilities – `cpan`, `cpanm`, `cpanp`, etc.

Alternatively, they can be manually downloaded from CPAN, compiled and installed. More information in included in the INSTALL file contained in the package of each tool.

A.1.1 Requirements

At the moment, these tools only work on Unix systems, and depend on:

- A modern version of Perl (above 5.8)
- Unix tools like grep, diff, sort, etc
- Several Perl modules, installable through CPAN.

A.2 Text::Math::NumExp

Text::Math::NumExp - Find numeric expressions in text.

VERSION

version 0.01_13

Synopsis

```
use Text::Math::NumExp;

my $text = "Light travels at 3x10[8] m/s."
norm_numexp($text);

# Returns:
# "Light travels at 3x10^8 m/s."

$text = "The program used for the amplification was as follows:
       5 min at 94°C, followed by 50 cycles consisting of 30s
       at 94°C, 30s at 62°C, and 30s at 72°C";

find_numexp($text);

# Returns:
# [ { length => 1, offset =>  54, text => 5,           value => 5     },
#   { length => 2, offset =>  63, text => 94,          value => 94    },
#   { length => 2, offset =>  81, text => 50,          value => 50    },
#   { length => 9, offset => 105, text => "30s at 94", value => undef },
#   { length => 9, offset => 119, text => "30s at 62", value => undef },
#   { length => 9, offset => 137, text => "30s at 72", value => undef },
# ]

$text = "One plus one equals two.";
find_numwords($text);

# Returns:
# [ { length => 3, offset =>  0, text => "One", value => 1 },
#   { length => 3, offset =>  9, text => "one", value => 1 },
#   { length => 3, offset => 20, text => "two", value => 2 },
# ]
```

Description

This module searches for numbers and numeric expressions in a text, including:

- numbers (e.g 30.000, 3.4, -20):

- spelled-out numbers (e.g. "one million",: "three")

- complex numeric expressions (e.g. 1.5x10^-5):

Text::Math::NumExp

Text::Math::NumExp - Find numeric expressions in text.

SUBROUTINES/METHODS

find_numexp

Finds numeric expressions in text.

find_numwords

Finds spelled-out numbers in text.

solve

Returns the value of a numerical expression. Retuns undef if expression is not solvable.

norm_numexp

Normalizes common numerical expression patterns (including Unicode characters).

AUTHOR

Andre Santos <andrefs@cpan.org>

Copyright AND LICENSE

A.3 Text::Paragraph::Splitter

Guesses the notation used for paragraphs to split them.

VERSION

version 0.01_02

METHODS

offsets

Finds paragraphs boundaries and returns a reference to an array of pairs, each consisting of a paragraph's first and last character offsets.

split

Split the paragraphs and returns them in an array reference.

annotate

Returns the original text with paragraphs annotated XML-style.

AUTHOR

André Santos <andrefs@cpan.org>

Copyright AND LICENSE

A.4 Lingua::EN::Sentence::Offsets

Finds sentence boundaries, and returns their offsets.

VERSION

version 0.01_06

Synopsis

```
1       use Lingua::EN::Sentence::Offsets qw/get_offsets get_sentences/;
2       my $offsets = get_offsets($text);     ## Get the offsets.
3       foreach my $o (@$offsets) {
4               my $start  = $o->[0];
5               my $length = $o->[1]-$o->[0];
6               my $sentence = substr($text,$start,$length)  ## Get a sentence.
7               # ...
8       }
9       ### or
10      my $sentences = get_sentences($text);
11      foreach my $sentence (@$sentences) {
12              ## do something with $sentence
13      }
```

METHODS

get_offsets

Takes text input and returns reference to array containin pairs of character offsets, corresponding to the sentences start and end positions.

get_sentences

Takes text input and splits it into sentences.

add_acronyms

user can add a list of acronyms/abbreviations.

get_acronyms

get defined list of acronyms.

set_acronyms

run over the predefined acronyms list with your own list.

remove_false_eos

split_unsplit_stuff

Finds additional split points in the middle of previously defined sentences.

adjust_offsets

Minor adjusts to offsets (leading/trailing whitespace, etc)

initial_offsets

First naive delimitation of sentences

offsets2sentences

Given a list of sentence boundaries offsets and a text, returns an array with the text split into sentences.

ACKNOWLEDGEMENTS

Based on the original module *Lingua::EN::Sentence*, from Shlomo Yona (SHLOMOY)

See also

Lingua::EN::Sentence, *Text::Sentence*

AUTHOR

Andre Santos <andrefs@cpan.org>

Copyright AND LICENSE

A.5 Lingua::EN::Tokenizer::Offsets

Finds word (token) boundaries, and returns their offsets.

VERSION

version 0.01_04

Synopsis

```
use Lingua::EN::Tokenizer::Offsets qw/token_offsets get_tokens/;

my $str <<END
Hey! Mr. Tambourine Man, play a song for me.
I'm not sleepy and there is no place I'm going to.
END

my $offsets = token_offsets($str);     ## Get the offsets.
foreach my $o (@$offsets) {
    my $start  = $o->[0];
    my $length = $o->[1]-$o->[0];

    my $token = substr($text,$start,$length)  ## Get a token.
    # ...
}

### or

my $tokens = get_tokens($str);
foreach my $token (@$tokens) {
    ## do something with $token
}
```

METHODS

tokenize($text)

Returns a tokenized version of $text (space-separated tokens).

$text can be a scalar or a scalar reference.

get_offsets($text)

Returns a reference to an array containin pairs of character offsets, corresponding to the start and end positions of tokens from $text.

$text can be a scalar or a scalar reference.

get_tokens($text)

Splits $text it into tokens, returning an array reference.

$text can be a scalar or a scalar reference.

adjust_offsets($text,$offsets)

Minor adjusts to offsets (leading/trailing whitespace, etc)

$text can be a scalar or a scalar reference.

initial_offsets($text)

First naive delimitation of tokens.

$text can be a scalar or a scalar reference.

offsets2tokens($text,$offsets)

Given a list of token boundaries offsets and a text, returns an array with the text split into tokens.

$text can be a scalar or a scalar reference.

ACKNOWLEDGEMENTS

Based on the original tokenizer written by Josh Schroeder and provided by Europarl http://www.statmt.org/europarl/.

See also

Lingua::EN::Sentence::Offsets, *Lingua::FreeLing3::Tokenizer*

AUTHOR

André Santos <andrefs@cpan.org>

Copyright AND LICENSE