NTNU
Norwegian University of
Science and Technology

Searching for, and identifying Protein Information in the Literature

Espen Klæboe

Master of Science in Informatics
Submission date: June 2010
Supervisor: Herindrasana Ramampiaro, IDI
Co-supervisor: Trond Aalberg, IDI

Norwegian University of Science and Technology
Department of Computer and Information Science

Abstract

As research papers grow in volume and in quantity, there is still to this day, a hassle to locate desired articles based on specific protein names and/or Protein-Protein-Interactions. This is due to the everlasting problem of extracting protein names and Protein-Protein-Interactions from bio-medical papers and articles. The goal of this thesis was to investigate an approach that suggests the use of the Lucene framework for storing and indexing different articles found in bio-medical databases and being able to effciently identify protein names and possible interactions that exist in them. The system, dubbed MasterPPI, locates protein names and Protein-Protein-Interaction keywords with the help of two dictionaries, and when these are found and labeled, determins a Protein-Protein-Interaction if a specific interaction-keyword is present in a sentence, between to protein names.

When tested against the test collection from the IAS subtask in the BioCreAtIvE2 challenge; the prototype system achieved a f-score of 0.34, showing that the system has potential, but needs a great deal of work.

Acknowledgements

First of all, I would like to thank my supporting and helpful family and especially my mother Berit; who, when I rant on about what I do, keeps on encouraging me but still has no clue to what I'm talking about; and to my cousin Henriette who took the time to proof-read a thesis from a field she only has a vague understanding of.

I would also thank my supervisor Heri Ramampiaro for helping me choose this thesis and offering advice in an hour of need from across the pond.

And last but not least, a *big* thanks to all the guys at Sule: Jørgen, Anders, Sindre, Johan and Kristian[1]. I could literally not have done this without all of our relevant discussions and ball-play when deadline was approaching.

[1]and Trond and Magnus, who sometimes showed up

Contents

List of Figures

List of Tables

Chapter 1

Introduction

This chapter will serve as an introduction to the thesis, by introducing the problem at hand, the motivation for attempting a solution, a short preview of the skeleton for the proposed solution and an overview of the thesis in general.

1.1 Motivation

As research papers grow in volume and in quantity, there is still to this day, a hassle to locate desired articles based on a specific protein name and/or PPI (Protein-Protein-Interactions). Having grown up with the internet, and seen how e.g google has evolved with its scholar[1] program, this statement does come across as a little suprising. As an example (to show the size of the scope), consider MEDLINE; the biggest collection of bio-medical papers available. This database consist (when this is written) of almost 18 million records, gathered from various publications [NLM, 2010]. The database is accessed through an engine called PubMed[2] that adds another 2 million records from other sources [NLM, 2010]. Untrained users of this system are sometimes annoyed with the large number of results returned by simple, straightforward queries, suggesting a system with to high recall but with low precision[3]. There exist many collections that try to muster up all the protein names that are mentioned in different bio-medical papers and provide useful tools for querying them, e.g SwissProt[4] (which is manually annotated and reviewed) and TrEMBL[5] (which is automatically annotated and not re-

[1]`http://scholar.google.com` - Stand on the shoulders of giants
[2]`http://www.ncbi.nlm.nih.gov/pubmed/`
[3]Precision and recall is covered in section 2.1.2
[4]`http://au.expasy.org/sprot/`
[5]`http://www.ebi.ac.uk/uniprot/`

viewed). And there exist others who try to unify these “smaller ones” into one big collection, like UniProt[6] who is a combination of the former two. These may also have connection or refrences to articles that can be found via PubMed.

When investegating the “hard-to-believe” statement above, one discover the interesting but fundamental problem of extracting protein names and PPIs from the papers and articles that are published out there[7] in some digital form. This is complicated with the fact that bio-medical texts are not the same as your regular paper or article. A collection of texts that combinds bio-medical terms with natural English produces challenges given by problems like e.g high ambiguity, since many protein names share lexical representation with common English words (e.g. by, an, for, can) [Krauthammer and Nenadic, 2004]. There is also the problem with unstructured texts and the non-existence of an uniform way of naming the different proteins or describing the interactions that may occur between them. As stated, this makes the extraction complicated, and the return-sets from queries into existing systems are not as accurate as one would like [Monsen, 2007].

So the main challenge is to identify the proteins and the interactions, store and index them, and make this available to the masses[8] to query in ways that they are used to (i.e easy way). Finding a way to streamline these tasks (or problems) in in an efficient, but traditional way is the main motivation behind this thesis.

1.2 Problem Specification

This thesis will, with the help of traditional information retrieval techniques, try to tackle the ongoing problem of identifying and categorising protein names and PPI’s in bio-medical texts (as outlined in section 1.1). The problem also consist of storing these terms (ie. names and interactions) efficient and indexing them so that one can recognise and perform a query on them at a later stage.

[6]`http://www.uniprot.org`

[7]The world in general.

[8]The general public who do not have Ph.D’s in medicine, computer science or both.

1.3 Solution Outline

The solution presented will be a Java based system for recognising, storing, indexing and searching for protein names and PPIs in bio-medical literature. The approach will take inspiration from ideas found in [Ramampiaro et al., 2007] and [Egorov et al., 2004], utilising LingPipe[9] for parsing MEDLINE-articles and Lucene[10] for storing and indexing the terms produced. The idea for the approach can be found in 4 and the implementation will be presented in chapter 5.

1.4 Thesis Structure

This thesis will be structured in the following manner:

Chapter 2: This chapter will provide the reader with the background knowledge that will/can be important for understanding the different procedures used for solving the task at hand. Basic concepts of bioinformatics, information retrieval and text mining will be presented.

Chapter 3: This chapter will present interesting and related work that is ongoing or have been done in the past.

Chapter 4: This chapter will present the reader with the idea for the approach and the thoughts behind it.

Chapter 5: This chapter will present, in depth, the implementation of the approach.

Chapter 6: This chapter will show how the system performed in a controlled test environment.

Chapter 7: This chapter will sum up this thesis, give a conclusion and present visions of future for the idea that was presented in chapter 4.

[9] `http://alias-i.com/lingpipe/`
[10] `http://lucene.apache.org/`

Chapter 2

Preliminary knowledge

This chapter will present the reader with knowledge and context that will be relevant for understanding different concepts, and techniques which is used later on. This will include fields from information retrieval, biology, bio-informatics and text mining (with biological texts as a special case).

2.1 Information Retrieval

The term IR (Information Retrieval) can mean a lot of things, and only a small explanation is given here. Basically it encompasses all that has to do with a given user's need to find some sort of information. With that said, as an academic field of study it is defined by [Manning et al., 2008] as:

> *. . . finding material (usually documents) of an unstructured nature (usually text) that satisfies an information need from within large collections (usually stored on computers).*

When Manning and his colleagues talk about "unstructured data", they mean data that has no clear, semantically overt, computer-understandable structure.

A key term to consider in the definition above is *document*. In our common world, the term means a textual record, but in the world of IR it can be so much more. [Buckland, 1997] defines the following points for when an object is a document, or how it can become one:

- **There is materiality**: Physical objects and physical signs only.
- **There is intentionality**: It is intended that the object can be treated as evidence.

- **The objects have to be processed**: They have to be made into documents.
- **There is a phenomenological position**: The object is perceived to be a document.

This list is an attempt to summarise the works of Suzanne Bret, who in 1951 published a manifesto on the nature of documentation [Bret, 1951]. This starts with the assertion that "A document is evidence in support of a fact". Examples of objects that are considered documents and that is regulary prone to IR is books, movies, sound files, articles and webpages. The goal of IR is to return these documents to a user in a quick and easy way without him or her knowing the location of the document in advance.

One important thing to note however is the difference between the terms IR (information retrieval) and DR (data retrieval). The latter only determines that a document, in a collection of documents, contains the keywords specified in a query provided by a user. An example of this is a relational database[1] which also has a clear computer-understandable structure. This may be to narrow for the user, as he or she might want relevant information to the keywords stated. This is what IR is trying to accomplish. And it aims to do this from mostly natural language. It must therefore extract syntactic and semantic information from the query-text provided, and then rank the documents that are returned, based on relevance compared to the query.

2.1.1 Information Retrieval Models

The IR-literature (e.g [Manning et al., 2008]) defines three models as traditional for the IR domain; the Boolean, the vector and the probabilistic model. A short description of the three will be given here.

The Boolean Model

This model is based on set theory and Boolean algebra, and sees each document as set of words. One can then pose any query in the form of a Boolean expression; that is combining terms with the operators *AND*, *OR* and *NOT*. The query terms are then considered to be present in the index or not at all (1 or 0), thus concluding that a document is relevant or not. As a simple example (from [Manning et al., 2008]; consider the query "Antony *AND* Caesar *NOT* Calpurnia" on a collection of Shakespears's finest works. This

[1]A database that stores information based on common characteristics into a specific group, making it easier to organise, update and maintain.

will look for any of Shakepears plays that will contain both Antony and Caesar, but not Calpurnia. Resulting in an answer-set consiting of "Antony and Cleopatra" and "Macbeth" amongst others.

The Vector Model

An illustration of this model can be seen in Figure 2.1. Here, a set of documents is represented as vectors in a common vector space. It differs from the Boolean model in that one now can give the index-terms non-binary weights. From this, one can then calculate the degree of similarity between a given query and a set of documents in a collection, based on the correlation between the two vectors (i.e the *cosine similarity*). What we then get is a ranking of the document based on its similarity with the query.

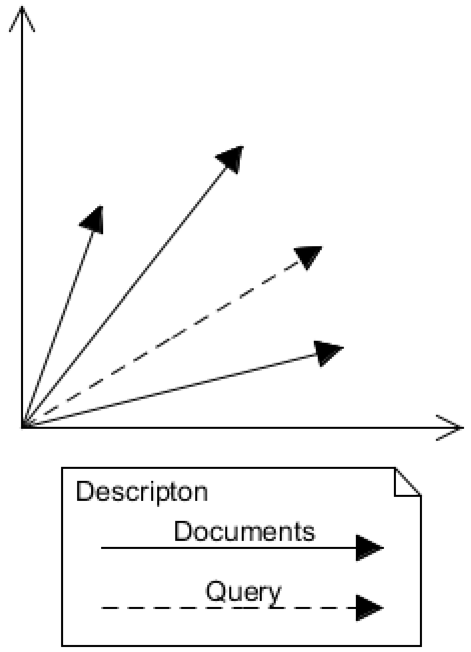

Figure 2.1: An illustration of the vector space model

The weighing of the index-terms are quantified, often by the *tf-idf-term-weighing*-scheme. This is based on the term frequency and the inverse document frequency. In short, the term frequency is how often a term is referenced in a document, and the document frequency is how many documents in a collection that contains a given term. By inversing it, we get a boost on all the terms that are rare and a lower score on a terms that occur more frequently. For a more elaborate explanation, please refer to [Manning et al., 2008].

The Probabilistic Model

In this model we assume that there exist a perfect return set, R, that contains exactly the relevant document(s) for a given query. The term weights are all binary, and we have to initially guess the return set R. The model will then assign to each document a probability-ratio, which gives an indication of the relevance that the document has to a given query divided, by the probability of it not being relevant. The ratio produced is then used as a measure for the document's similarity to the given query, and the result set is then ranked in increasing order based on this ratio.

2.1.2 Evaluation

When designing an IR-system, one would like to be able to test how good it actually operates when it is being used. This is not just a matter of observing how well the system retrieves documents (based on a query), but also how happy the user is with the result. Funny enough, developers and users have different views on the notion of quality, but in the end it is the user that we are trying to please. The key word here is *user happiness* [Manning et al., 2008], and we can derive from this that the goal of IR evaluation is to measure how well the system satisfies the user's *need for information.*

Test Collections

A test collection is a controlled set of documents, that are given a binary classification as either relevant or not relevant to a query. [Manning et al., 2008] defines three things that a collection like this should consist of:

1. A document collection
2. A test suite of information needs, expressible as queries
3. A set of relevance judgments, standardly a binary assessment of either *relevant* or *non-relevant* for each query-document pair

As mentioned above, the evaluation (and thus the relevance) is judged by the information need, and *not* the query. This means that a document is not necessarily relevant just because it contains all the keywords specified in the given query. It is all about what the user is really after. The misunderstanding of this distiction is often done in practice, because the information need is not overt [Manning et al., 2008].

Here follows a list of some of the most standard test collections for IR evaluation:

Cranfield: This collection is considered as the pioneering test collection, dating back to the late 1950s. Assembled in the United Kingdom, it consist of 1938 abstracts of aerodynamics journals, a set of 225 queries and relevance judgements of the query-documents pairs [Manning et al., 2008].

TREC: The Text REtrieval Conference is a series of workshops that aim to support and encourage research within the IR community. This is done by providing the infrastructure necessary for large-scale evaluation of text retrieval methodologies and to increase the speed of lab-to-product transfer of technology. It consists of 1.89 million documents and 450 information needs[Manning et al., 2008], and was started in 1992 by NIST (The US National Institute of Standards and Technology). It is now co-sponsored by NIST and the US. Department of Defense [NIST, 2008].

GOV2: This is the largest collection of webpages easily availble for research purposes, and consists of 25 million unique pages. Also evaluated by NIST it is nevertheless still 2 orders of magnitude smaller than the document collections indexed by the biggest web search companies (e.g google, yahoo, etc.) [Manning et al., 2008].

As of this writing there exist no public de-facto test collection for evaluating medical IR systems, though some TREC collections do come close. It has a “Genomics track” where the goal is to study the retrieval of genomic data, though not just gene sequences but also supporting documentation, e.g research papers, lab reports, etc. It also has a “Chemical Track” where the goal is to develop and evaluate technology for large scale search in chemistry-related documents, including academic papers and patents, to better meet the needs of professional searchers, specifically patent searchers and chemists. But these are, in lack of a better term, to narrow since they only cover parts of what one can find in the bio-medical literature (i.e MEDLINE).

[Hersh et al., 1994] built their own collection from MEDLINE abstracts when evaluating their automated IR system (dubbed Saphire). It contained

75 queries and 2,334 citations. Closer to home[2], the lack of a specific Swedish collection made Karin Friberg Heppin start the work on MedEval [Heppin, 2008] in 2008. The BioCreAtIvE (Critical Assessment of Information Extraction systems in Biology) challenge evaluation is perhaps the closest one would get to a specific medical test collection. It consists of a community-wide effort for evaluating text mining and information extraction systems applied to the biological domain, and will be one of the test collections used to evalutate the approach this thesis presents.

Precision and Recall

When it comes to evaluate a IR system with the use of the set-based metrics, the most frequently used is the precision and recall measure. These two will evaluate the quality of an unordered set of retrieved documents, and can give a good measure of how good the user's information need is met.

- **Precision:** Defined as

  > ... *the fraction of retrieved documents that are relevant*,
  >
  > [Manning et al., 2008]

  precision is the the relationship between the number of relevant document retrieved and the total number of retrieved documents.

- **Recall:** Defined as

  > ... *the fraction of relevant documents that are retrieved*,
  >
  > [Manning et al., 2008]

  recall is the relationship between the number of relevant documents retrieved and the total number of relevant documents in the collection.

As illustrated in figure 2.2, where $|R|$ is the relevant sub-set of documents to an information need I, and where $|A|$ is the returned answer-set, we see that the relevant part of the answer-documents, $|Ra|$, is located in the intersection between these to sets. The formulas are pretty straight forward, and can be seen in equations 2.1 and 2.2.

$$\text{Precision} = \frac{|R_a|}{|A|} \tag{2.1}$$

$$\text{Recall} = \frac{|R_a|}{|R|} \tag{2.2}$$

[2]Norway

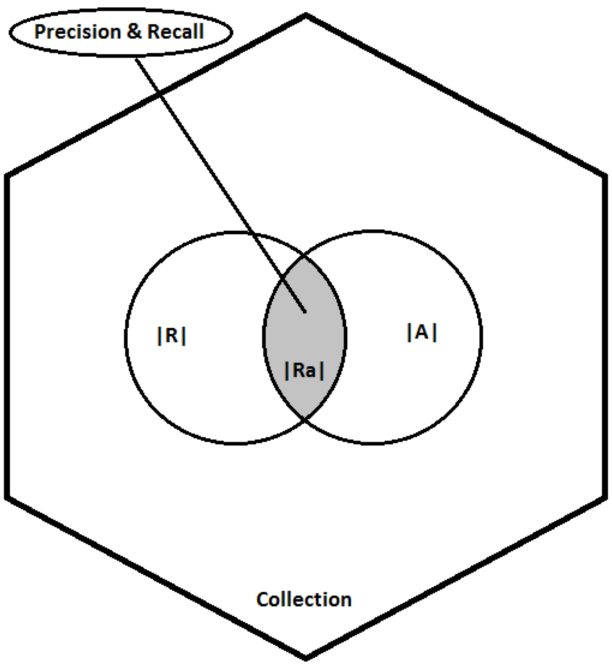

Figure 2.2: An illustration of Precision and Recall

One can deduse from this that there is an inverse relationship between precision and recall, being that one can reduce one metric to increase the other. For example; one could always increase the recall score with just introducing more documents in the return/answer-set, though this could increase the number of irrelevant documents and thus decrease the precision. These scores however are not usually put in isolation like this. Either they are combined into one score called the f-measure (F1- or F-score for short), or they are compared to a fixed measure of its counterpart. The f-measure is a weighted harmonic between precision and recall, and can be seen in equation 2.3.

$$f_{measure} = \frac{2 \cdot Precision \cdot Recall}{(Precision + Recall)} \tag{2.3}$$

2.2 Biology

Biology is a vast subject consisting of several subtopics, divisions and disciplines. Giving a full account for it here is way beyond the scope of this thesis. But in its broadest sense we can say that biology is the study of living organisms, including their structure, function, growth, origin, evolution, distribution, and taxonomy. Simply put; the study of life [IES, 2010, TXT, 2010].

The subdivisions of biology is often specialised desciplines on their own [WSMNS, 2008] and can be seen, generalised , in the following list:

- **Molecular biology**: Studies the complex interactions of systems of biological molecules.
- **Biochemistry**: Examines the rudimentary chemistry of life.
- **Physiology**: Examines the physical and chemical functions of the tissues, organs, and organ systems of an organism.
- **Cellular biology**: Examines the basic building block of all life, the cell.
- **Ecology**: Examines how various organisms interrelate with their environment.

2.2.1 Proteins

For random events to produce even a single protein would seem a stunning improbability - like a whirldwind spinning through a junkyard an leaving behind a fully assembled jumbo jet.

Fred Hoyle

Somehow along the way, scientists in the disciplines listed above will come across the study of this peculiar entity. Proteins are one of the four macromolecules[3] that all organismes exist of and are, in short, very complex entities. They are also highly improbable, as stated by the English astronomer Fred Hoyle. To make a protein, one would need to assemble amino acids in a particular order, like one would assemble letters to form a word. There are 22 naturally occuring amino acids known on Earth that have been discovered so far, and only twenty of them are necessary to "produce" a human. Given that the English alphabet only consists of 27 letters, one would think that this wouldn't be to difficult. The challenge arises from the fact that, when the longest English word (who, ironically or perhaps rather conveniently, is a name for a protein) coined in a major English dictonary is "just" 45 letters long[4], a regular "protein-word" consists of several hundred "amino acid-letters". To illustrate, consider this example from [Bryson, 2003]: The word *collagen* is a 8 letter word and is a common type of protein. But in contrast, it is also a 1055 amino-acid-letter "protein-word". In other words, to make it, you would have to arrange 1055 amino acids in precisely the right order. Even scaling it down to a protein word of just 200 amino-acid-letters, the probability of them all to just *conveniently* assemble in the right sequence is 1 to 10^{260}, or in other words 1 to a number larger than the number of all the atoms in the known universe.

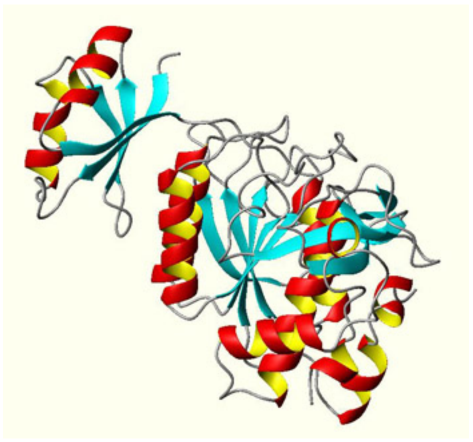

Figure 2.3: An example of what a folded protein could look like.

[3]The other three being: Nucleic acids, carbohydrates, and lipids

[4]Pneumonoultramicroscopicsilicovolcanoconiosis [DICT, 2010]

To complicate matters even more, a protein is not just a linear chain of amino acids. They turn, twist and fold into very unique shapes, as can be seen in figure 2.3. Since the shape it folds into will determine how a protein functions, it is the subject of intense study, more precisely on four levels: Primary, secondary, tertiary and quaternary. The primary structure is the basic amino sequence, which defines its native conformation. The secondary structure is the basic three-dimensional form of the protein (and the amino acids), and the tertiary structure describes specific atomic positions in this three-dimensional space. Lastly, the quaternary structure involves assembling or co-assembling sub-units that have already folded. Being able to pre-determine how a protein (or a protein complex) will fold based on a amino-sequence (or a protein-sequence) is one of the biggest open problems in science today [Nair, 2007].

Proteins are typically divided into categories according to their functions, which can be summerised ([ProteinCrystallography.org, 2007]) as:

Enzymes: Proteins that work as catalysts in chemical and bio-chemical reactions. Responsible for all metabolic reactions in the living cells. Well known examples are: DNA[5]- and RNA[6]-polymerases.

Hormones: These work as regulators for many processes in an organism. A commonly known hormone is insulin who regulates the blood sugar (or more specifically the cells ability to take up glucose from the blood). Many hormones are predecessors of peptide hormones, such as endorphine who works as a “natural pain reliever” for the body.

Transport: Group of proteins that transports or stores ions and/or some other chemical compound. The most commonly known examples are hemogloboin and myogloboin who transports oxygen in the bloodstream.

Defense: Proteins that are involved with the immune responses of the organism. I.e neutralisation of foreign molecules like bacteries and viruses. More commonly reffered to as antibodies or immunoglobulin.

Structural: Proteins tasked with the operation of maintaining the structure of biological components, like cells, tissue and fibers. Take for example keratin which is important in the formation of nails and hair.

[5]Deoxyribonucleic acid
[6]Ribonucleic acid

Motion/Motoric: Responsible for converting chemical energy into mechanical energy. Most famous are Actin and Myosin who are responsible for muscular motion (muscular contraction).

Receptors: A family of proteins which are responsible for signal detection and conversion into an other type of signal. An example of a protein from this family is Rhodopsin, a light detecting protein.

Signaling: These proteins work as a form of catalyst (enzyme) in a signaling translation process. Usually by changing some attribute of itself in the presence of some signaling molecules. Other types of proteins can interact with the receptors directly (usually smaller ones).

Storage: Proteins containing energy, ready for release during metabolism. Almost all proteins can be seen as a source of energy and building block(s) for other organisms.

2.2.2 Protein-Protein-Interaction

The interactions of proteins is a wast scientific field and is of central importance to every process in a cell. It involves the direct-contact or long-range interaction between two or more proteins, whom can be of the same (homo) or of different type(s) (hetero). When the interactions happen in a non-covalent[7] way, the proteins form what is called a protein complex, which is a more of long term bond than just an interaction. By direct-contact it is intuitively implied physical contact, but long-range interactions is a little more ambiguous. These will refer to higher levels of relations between proteins, and can be very different (as stated by [Rivas and de Luis, 2004]) in the way they occur:

- Inclusion in multiprotein complexes,
- Common cellular compartements,
- Same signaling pathway,
- Same metabolic pathway,
- Co-expression,
- Genetic co-regulation,
- Co-evolution

PPI's can also be seen as a series of events within the cell (and even the organism), in that they can (and will) form a chain of events that affects the

[7]Any relatively weak chemical bond that does not involve an intimate sharing of electrons. [Berk et al., 2000]

cell (and organism). This can be expressed as a directed graph, and is called (in terms of proteomics[8]) a PPI-network where an simple interaction is the basic unit. This kind of network is called an *interactome*[9], and though not exclusively used about PPI-networks, it is the most common. The study of interactomes is called interactomics and deals with both the interaction and the consequences it leads to.

For the same reason one could not list up in a paper every protein that exist, due to the sheer number of instances, one can neither list up all the the interactions or associations that occur between them. One of the biggest databases for PPI's is the Database for Interacting Proteins (DIP) which has, as of this writing, a collection of 69171 unique identified PPI's between 21891 proteins [DIP, 2010]. Even so, a more general list of interactions/associations can be found in [Rivas and de Luis, 2004]:

1. Co-interacting protein; defined as physical contact:
    - **Permanent interaction**: Proteins forming a stable protein complex that carries out a biomolecular role (structual or functional). These proteins are protein sub-units of the complex and they work together.
    - **Transient interaction**: Proteins that come together in certain cellular states to undertake a biomolecular function.
2. Correlated proteins; defined as proteins that are involved in the same biomolecular activity but do not physically interact:
    - **Metabolic correlation**: Proteins involved in the same metabolic pathway. These are mostly enzymes.
    - **Genetic correlation**: Proteins that are encoded by co-expressed or co-regulated genes. These could be called operontype proteins.
3. Co-located proteins; defined as proteins that work in the same cellular compartment:
    - **Soluble location**: Proteins placed in the same cellular soluble space.
    - **Membrane location**: Proteins placed in the same cellular membrane.

[8]The branch of genetics that studies the full set of proteins encoded by a genome.

[9]Defined as: "A complete set of macromolecular interactions (physical and genetic)." [NAT, 2004]

2.3 Bio-informatics

As defined in [Nair, 2007], bio-informatics (and indeed computational biology) is:

...the application of computer science and allied technologies to answer the questions of biologists about the mysteries of life.

Bio-informatics is a relatively new field, and the term was coined as late as 1979 by Paulien Hogeweg for the study of informatic processes in biotic systems. The constant flood of data from biology is in need of fast but thorough analysises, and to achieve this it makes sense to make use of computers to do the heavy work. The scope of bio-informatics encompasses in large (but is not limited to) computational and statistical techniques, algorithms, creation and advancements of databases, and, of course, theory to solve different problems arising from analysis of biological data, be it formal or practical.

One thing [Nair, 2007] emphasises is the difference between bioinformatics and computational biology, even though they have the same definition. They are both Computers + Biology, but the difference arises from what comes first in the equation. In short:

$$\textit{Bioinformatics} = \textit{Biology} + \textit{Computers} \tag{2.4}$$

and

$$\textit{Computational Biology} = \textit{Computers} + \textit{Biology} \tag{2.5}$$

So a biologist who use and develop computational tools to answer questions of biology is a bio-informatician, but a computer scientist who develop theories and techniques for such tools is a computational biologist.

2.4 Text mining

Text mining is a process on the intersection of the fields computational linguistics, IR and NLP (Natural Language Processing), and is defined in [Mooney and Nahm, 2005] as:

...the application of data mining techniques to automated discovery of useful or interesting knowledge from unstructured text.

Simplified we can say that text mining is a sub-field of data mining, where we spesify that the data pool where we want to extract information from, consist of textual data. Data mining (or knowledge discovery) as an own term is defined by [Frawley et al., 1992] as:

. . . the nontrivial extraction of implicit, previously unknown, and potentially useful information from data.

[Kamber and Han, 2006] notes that data mining can be viewed as a result of the natural evolution of information technology, but that the term is somewhat a misnomer. One is not mining for data, but knowledge/information *from* data. Thus a more appropriate name would have been "knowledge mining" or indeed "knowledge discovery" as used by [Frawley et al., 1992], but these are considered to be to long and tricky. All in all, one can summerise the process of text mining into a sequence of steps (shown in figure 2.4) that are briefly explained here:

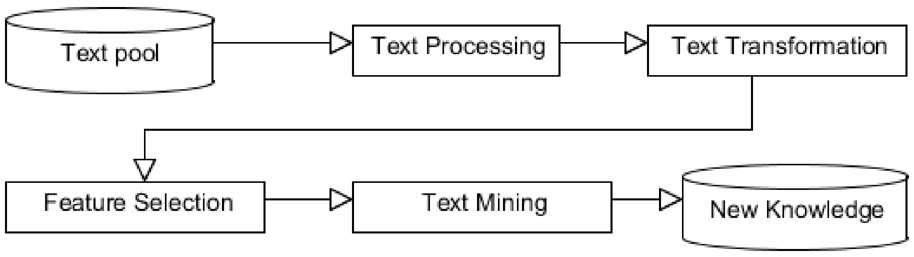

Figure 2.4: The process of text mining

One would start off with **pre-processing** the input text(s) from the data pool. This will structure the text with the help of semantic/syntactic analysis. Here, each word will be labeled with the corresponding part of speech[10], in what is called POS(Part-of-Speech)-tagging, and a parser may be used to obtain a representation of the gramatical structure of the input.

The next step would be **text transformation**. This could be seen as a part of the pre-processing step, but to emphasise the differences it gets its own description. Here the text(s) is divided into single words, who are then processed induvidually. Sub-tasks of this step is to remove all *stop words*[11] and the use of *stemming*[12].

The third step, here dubbed **feature selection**, is where one defines the features one would like to obtain from the mining by reducing the dimensionality. Typical sub-tasks here are term-analysis and named-entity recognition.

[10]Noun, adjective, verb

[11]Words that occur frequently, but does not have much meaning. E.g and, the, of, it, a.

[12]Reduce a word to its root. E.g Horses $\rightarrow$ Horse

With this, one can obtain terms that consist of several words, and exclude words/terms that are deemed not relevant.

Lastly, we have the the actual **text mining**. The goal here is to identify patterns in the result uncovered in the previous step, and thus uncover potentially new knowledge.

2.4.1 Biological text mining

Given the huge growth in digital medical literature, there has been an incresing interest in the use of text mining solutions for searching through these publications. One doesn't need to look any further than the MEDLINE database, as mentioned before. There are though, several differences and indeed challenges when mining in biological texts versus regular texts (e.g news stories, books, etc.).

The main one, that have already been adressed above, is the task of identifing biological entities in the texts (mainly protein names). Since there is no definitive standard in how one should name or adress a given protein, there now exists multiple variants of the same term and new ones are made up every day [Krauthammer and Nenadic, 2004]. All of them need to be condsidered when deciding which terms to extract. There is also an increased problem with ambiguity, since regular English words and biomedical-specific words can be the same but have different meanings, and gene symbols may also corrrespond to disease names.

Some of the strategies that so far have been developed to tackle these challengens are ad-hoc rule-based, machine learning and dictionary-based approaches/techniques. There also exist hybrids of one or more of these. As before, a short summary of the three will be given here:

Ad-hoc rule-based

A rule-based approach for machine learning when talking about biological text mining will make use of pre-exististing knowledge about how the biological literature is structured. The knowledge is specified through a series of rules (often "hand-written") that are derived from experts which combine surface clues[13] with syntactic/semantical properties. As one can deduse, the successfulness of this approach relies on how well defined the rules are. But it is a system that is easy to support, expand and adjust. As one discover new knowledge about the structure it is easy to just add a new rule.

[13]E.g alphanumerical word-composition, presence of special symbols, and capitalisation.

Machine learning

The machine learning approach can be seen as a automated version of the rule-based approach, as one is still interested in rules that describes properties and so on. The difference lies in that this knowledge-annotation is now automated, and the system will learn the rules from expert-annoted training-sets by the use of various statistical algorithms.

Dictionary-based

With this approach, one will use a pre-compiled list of terms (e.g protein names) when indentifying occuerences in a text. The approach will usually make use of sub-string matching when comparing possible matches with the dictionary list. Intuitively, one can see that the performance of this approach is closely related to the quality of the dictionary, in the same way the performance of machine learning and rule-based approaches is related to the rules they employ. All in all, tests show that a dictionary-approach outperform the rule-based apporaches on accuracy [Egorov et al., 2004].

Chapter 3

Previous and/or related work

What should be obvious from the previous chapter, is that these fields of science are very popular and that extensive research is being put down in the development of new and better methods and systems. In this chapter the reader will be introduced to some of these systems that are closley related to the work that this thesis is presenting.

3.1 ProtScan

ProtScan, outlined in [Egorov et al., 2004], is a dictionary-based approach that utilises carefully constructed dictionaries of mammalian protein names to indentify protein names in Medline abstracs. This dictionary was based on the LocusLink[1] database, enriched with names and symbols from GenBank[2], GoldenPath and HUGO[3]. The system was implemented i C, and achieved a 98% precision score and a 88% recall score when applied to Medline abstracts. Though noting that development and maintenance of these comprehensive protein name dictionaries is a nontrivial task, since new genes are discovered every day, the authors argue that this also applies to rule-based and machine learning techniques, since they require expert work for creation of rules and manual tagging of corpuses. They also point out that once the inital dictionary is constructed, its maintenance and updating will be a much simpler task and will require less human interaction.

[1]`http://www.nih.gov/science/models/rat/resources/locuslink.html`
[2]`http://www.ncbi.nlm.nih.gov/genbank/`
[3]`http://www.hugo-international.org/`

3.2 Yapex

The Yapax system is a rule-based protein name tagger which is outlined in [Franzén et al., 2002]. The proposed algorithm may be seen as a collection of seven steps, where the first four are tasked with lexical analysis of single word tokens. Step five and six are tasked with a syntactic analysis of noun phrases and the lexical categories derived from in the previous steps, and the seventh and last step will utilise the syntactic information to indentify new single- and multi-word protein names. The evalutation of the system showed a f-score[4] of 82.9% under "sloppy" conditions and 67.1% during strict conditions.

3.3 Textpresso

Texpresso is a text processing system (presented in [Müller et al., 2004]) that splits biological papers into sentences, and these sentences again into words and/or phrases. Each of these words and/or phrases are then labeled, using XML (eXtensible Markup Language), accordingly to the lexicon of an ontolgy they have constructed. An ontolgy is catalog of types or categories, devised for discussing a domain of interest. The ontolgy constructed for this system conists of 33 different categories for the terms or phrases. Then the sentences are indexed with respect to the different labels the ontolgy has "given" a word or a phrase. Focusing on 3800 full-text articles and 16000 abstacts on the topic of Caenorhabditis elegans (a free-living, transparent roundworm, about 1mm in length), the authors report of a precision-score for abstracts of 52% and a recall-score of 45%.

3.4 PubMed

Pubmed[5] is the offical search-tool used to access the tons of data located in the Medline database. It presentes the user with some rather usefull features, like MeSH[6] term support. Knowing how to use these terms really helps narrow down the result-set, but the downside is that it has a steep learning curve. The system also sports 'Clinical Queries' and 'Systematic Reviews' options that one can use to indentify more relevant studies by applying special

[4]Don't remember F-Scores? See section 2.1.2 and equation 2.3

[5]`http://www.ncbi.nlm.nih.gov/pubmed/`

[6]Medical Subject Headings - a comprehensive controlled vocabulary for the purpose of indexing journal articles and books in the life sciences. See `http://www.nlm.nih.gov/mesh/`

filters to every search. New and untrained users of the system are sometimes annoyed with the large and wide number of results returned by simple queries. As of October 2009, PubMed got a new interface that encourages simple search formulations in contrast to the complex ones that were sported before.

3.5 ProtIR

The ProtIR is a proposed prototype developed for the IAS subtask in the BioCreAtIvE2 challenge for text mining and information extraction, and can be viewed in [Ramampiaro et al., 2007]. The system is partitioned into seven modules where the main ones are dubbed M1-5. The two other last modules, dubbed MA and MB, are support modules for M4. The main modules are, in respective order, responsible for tokenization, indexing, term categorisation, evidence score calculation and relevance classification. In order to recognise proteins and PPI's the system used a pre-compiled list of protein/gene-symbols and names, and a list of PPI keywords. Evaluation based on a test collection from BioCreAtIvE2 produced a f-score of 68.2%

3.6 Other

In her PhD thesis[7], Barbara Rosario presents several steps and algorithms that extract semantics from biomedical texts, using statistical machine learning techniques. The thesis focuses on indetification of entities and/or roles, and also the relationship among them (i.e PPI's). She reports of F-Scores slightly above and below 70% for all her dynamic models when applying a smoothing factor.

A dictionary approach, dubbed iMasterThesis, is proposed by Magnus Skuland in [Skuland, 2005]. Here the protein name dictionary is realised as a multi-level tree structure of hash-tables, and tried to facilitate a more flexible and relaxed matching scheme than previous approaches. Evaluated against a golden standard of 101 expert-annotated Medline abstracts, the solution achieved a 10% recall score and a precision score of 14%.

[7]See [Rosario, 2005].

Chapter 4

Own Approach

This chapter will give the reader an overview of the approach this thesis is taking; highlighting choices, thoughts, the reasoning behind them and introduce the system that emerged from it, dubbed *MasterPPI*.

4.1 The idea and the reasoning behind it

As previously mentioned, MasterPPI will take inspiration from the systems dubbed ProtScan and ProtIR, which is described in [Egorov et al., 2004] and [Ramampiaro et al., 2007] respectfully (and briefly outlined in section 3.1 and 3.5 in this thesis). The idea is to use a version of the "assembly line"-like architecture from ProtIR combined with the use of the several-dictionaries-type of approach from ProtScan (though not constructed in the same way). The ProtScan dictionaries required a lot of manual work; like removing all entries that were purely numerical or just under a length of 1 and moving entries that had a length of 2-4. MasterPPI will try to simplify this the whole process, primarily by automating a task or let Lucene[1] handle it . The use of different workstations in the assembly line (where one encloses a work task in its own module) will make it easy to take out and change/improve specific parts without having to rewrite the entire system, thus supporting a software architecture quality known as modifiability [Bass et al., 2007]. An overall illustration of the proposed system can be seen in figure 4.1, and an in depht explaination of the modules can be found in sections 4.2 through 4.6.

MasterPPI's first difference from ProtIR is the re-location of term-filtering (i.e stopwords) from the catergorization module to the tokenizer and indexer module. Further on, MasterPPI will not use a weighing-scheme when finding

[1]More on Lucene in section 5.1.1

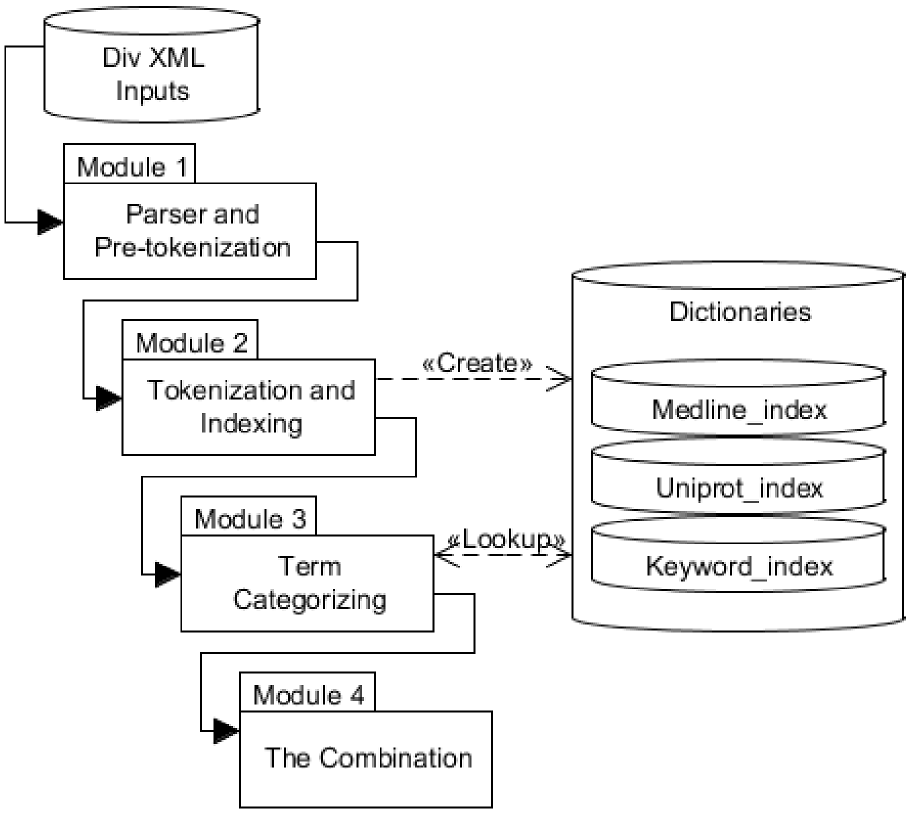

Figure 4.1: Illustration of the Assembly line approach

PPI's and not have the same base for the dictionaries it constructs. ProtIR used the NTNU annotation database for their recognision of names, and it was theorised that this database was not complete (or up to date) enough and that this made the system not as good as it could have been. ProtScan used LokusLink for its dictionaries that has since then been succeded by EntrezGene, so though their results were good at the time, the database is outdated (EntrezGene may not be of course). Based on this, the choice for MasterPPI will be UniProt which is a collapsed database of two other[2] big ones, and offer it free for download in the XML-format. More info on XML can be found in section 5.1.2. The MEDLINE articles from the PubMed database will also be indexed before they are term-categorized, thus providing easy access to its terms and their position/offsets.

ProtScan used a combination of two separate dictionaries, one being curated and the other being non-curated, when trying to identify protein names in a text. Further on, the system processed one sentence at a time, scanning them for a sequence of tokens that could be a "protein-word", and then tried to match these with entries in the dictionaries. ProtIR applied weights to terms based on their probable relevance, and then calculated the probabillity of the existens of a PPI in a given text. MasterPPI will also work on a sentence-level, first determining if a token is a protein name, interaction-keyword or just a random word, and then label it accordingly with their possitions in the sentence. Too achieve this, the tokens will be checked against two dictionaries that have been compiled from the biggest, relevant and most recently updated databases on the web. The first dictionary will be for protein names and the second for interaction-keywords and both dictonaries will be realised as Lucene indices. This also goes for the MEDLINE index. Having access to information on positions and offsets from the MEDLINE-Lucene-index, the system can then check the positions of the different occuring tokens, determining an PPI using the naive approach that an interaction in a sentence needs at least one interaction-keyword and at least two protein names. Further on, the position of the keyword should be between the two proteins.

4.2 The XML-inputs

MasterPPI will take three collections of inputs (illustrated in figure 4.2; The first will be a collection of MEDLINE "articles" constructed in the same structure that PubMed offers from their site. These are not full articles, but consists of the articles PMID (PubMed IDentification), the title, abstract,

[2]Though MasterPPI will only use the part that is manually annotated and reviewed

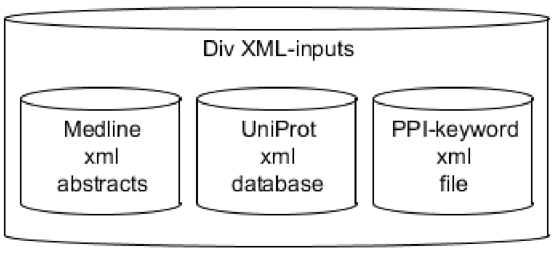

Figure 4.2: Illustration of the XML-files that works as input

an several other metadata-entries (publishing date, journal, volume, pages, etc). The second input is a database dump from UniProt, also in the XML format, and the third is a XML file compiled from the table of keywords presented in [Temkin and Gilder, 2003].

4.3 Module 1 - The Parsers and Pre-Token-Processing

This module (illustrated in figure 4.3) will parse the different xml-abstracts, -database and -keywordlist into useable strings which again will be processed according to the heuristic rules provided in [Jiang and Zhai, 2007]. These six rules are briefly summerized here:

1. Replace the following characters with spaces (whitespaces): ! ” # $ % < = > ? @ \ |
2. Remove the following characters if they are followed by a space: .:;,
3. Remove the following pairs of brackets if the open bracket is preceeded by a space and the close bracket is followed by a space: ()[]
4. Remove the single quotation mark if it is preceeded or followed by a space: ’
5. Remove ’s and ’t if they are followed by a space
6. Remove slash / if it is followed by a space

The parsing of the Medline files will be done with a special framework distributed by the LingPipe project. This framework gives direct access to

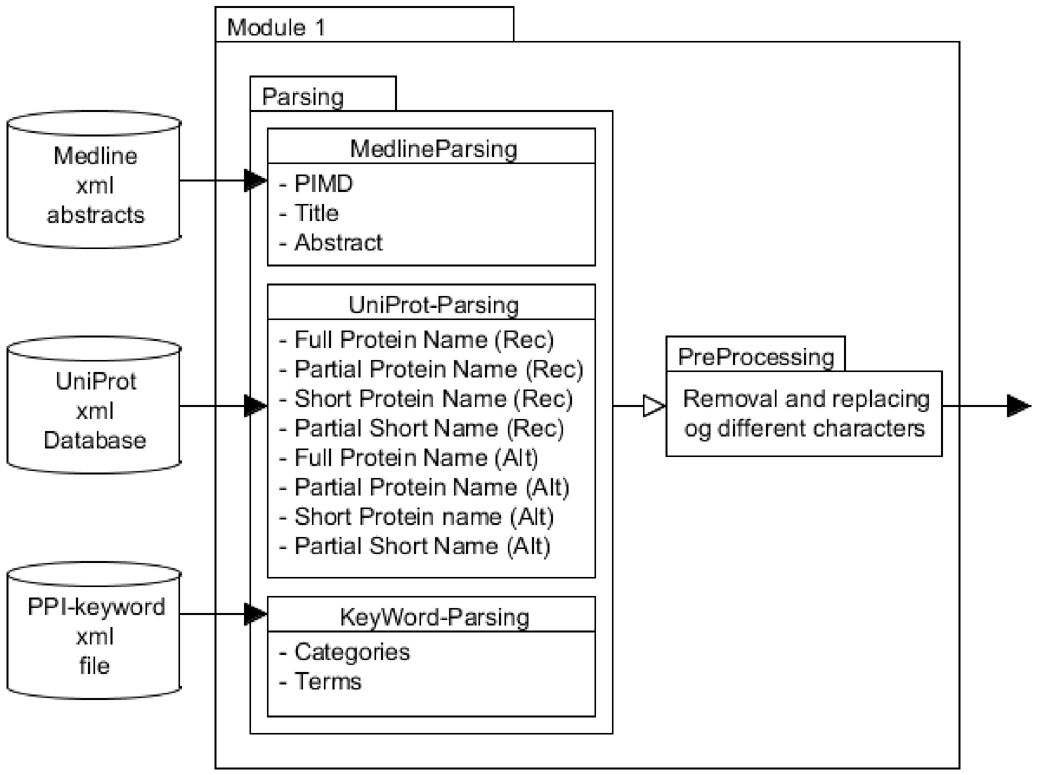
Module 1
Parsing
Medline xml abstracts
MedlineParsing
- PIMD
- Title
- Abstract
UniProt xml Database
UniProt-Parsing
- Full Protein Name (Rec)
- Partial Protein Name (Rec)
- Short Protein Name (Rec)
- Partial Short Name (Rec)
- Full Protein Name (Alt)
- Partial Protein Name (Alt)
- Short Protein name (Alt)
- Partial Short Name (Alt)
PPI-keyword xml file
KeyWord-Parsing
- Categories
- Terms
PreProcessing
Removal and replacing og different characters

Figure 4.3: Illustration of Module 1

elements like the PMID of the document, the title-text and the abstract-text. For the UniProt database a custom built SAX-parser has to be constructed, and the same for the keywordlist. All that is gathered from this database is a proteins full name, and (if it exists) its alternative name. When this is done the processed data is sent to the next module for tokenization and indexing.

4.4 Module 2 - The Tokenizer and indexer

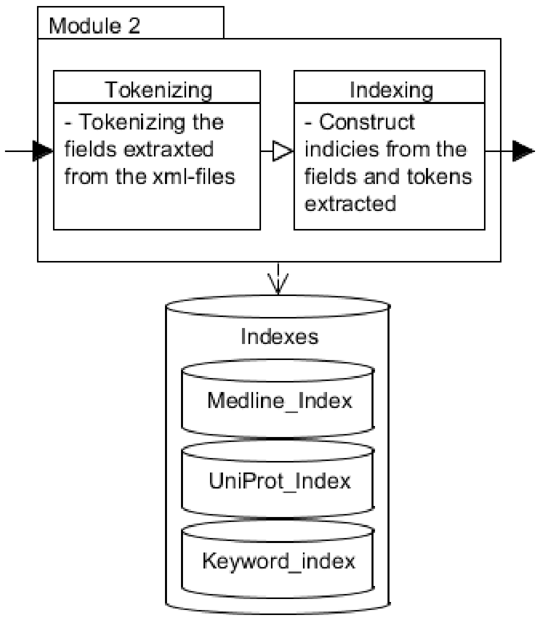
Module 2
Tokenizing
- Tokenizing the fields extraxted from the xml-files
Indexing
- Construct indicies from the fields and tokens extracted
Indexes
Medline_Index
UniProt_Index
Keyword_index

Figure 4.4: Illustration of Module 2

The tokenizer and indexer are placed in the same module (illustrated in figure 4.4) because both will be utilising the Lucene library. The tokenizer part will consist of a special-written Lucene Analyser that will deal with the process of dividing a text-string into stand-alone tokens and running these through one or more filters. The splitting of tokens will occur when the tokenizer encounters a whitespace. The filters will take care of the post-tokenizing work, like making all characters lowercase and the removal of stop words. The regular list of words that are defined as stop words will be extended with a "special case"-list of words often found in bio-medical literature, and provided by PubMed (see Appendix A). By removing these in this module (in contrast to ProtIR, who did it in the term categorization module), the generation-process of the index, and the index itself, will be much more efficient and accurate. Since the stopword list is defined in lowercase, it is crucial that the tokens are made lowercase *before* applying the stopwords-filter. [Jiang and Zhai, 2007] recommends not using a stopword-filter because they could not determin if it helps or not. The decision to include it here it therefore: “Why not”. As long as it *might* help, and it does not decrease overall system performance then why not try it. Of course, this is just as good an argument for leaving it out, but this system will include it.

The indexing process is the core of the Lucene library and promises scalable, high-performance indexing with a size of roughly 20-30% of the text-size. This index will provide a list of term-frequency vectors that can be used in (a) later module(s), for acquiring token offsets, positions and frequencies. Details on how Lucene works will be covered in the next chapter (see section 5.1.1.Lucene).

4.5 Module 3 - The Term Categorization

The idea for this module (illustrated in figure 4.5) is to put each term extracted from the MEDLINE-index into one of two pre-defined categories: *Protein name* and *PPI-keyword*. Further more, a term categorized as a protein name, will also get labeled accordingly if it is a full name or a part of a name (protein names can consist of multiple words). These possible-part-of-name tokens will, if the offsets between the end of one and the start of another is 1, combined and then checked again against the fullname index. When it comes to recognising different terms, there was in the previous module compiled a set of dictionaries from different sources, consistent with the different categories, for the the terms to be compared against. The potential protein names will be checked against the dictionary compiled from UniProt, whereas the potential interaction-keywords will be checked against the dictionary compiled from a list of words provided by [Temkin and Gilder, 2003][3]. Tokens that do not get a match in neither will just simply be discarded as not important.

If possible, the PPI-keyword should be labeled with the direction of the interaction that has been identified, since this can be used later for specifying which protein is the “interacter” in a PPI. Seen as a usefull feature, but not an essential one, this will not be implemented in MasterPPI but could be an improvement for the future. If none full-name entites have been found in the

[3]The XML file can be viewed in Appendix B

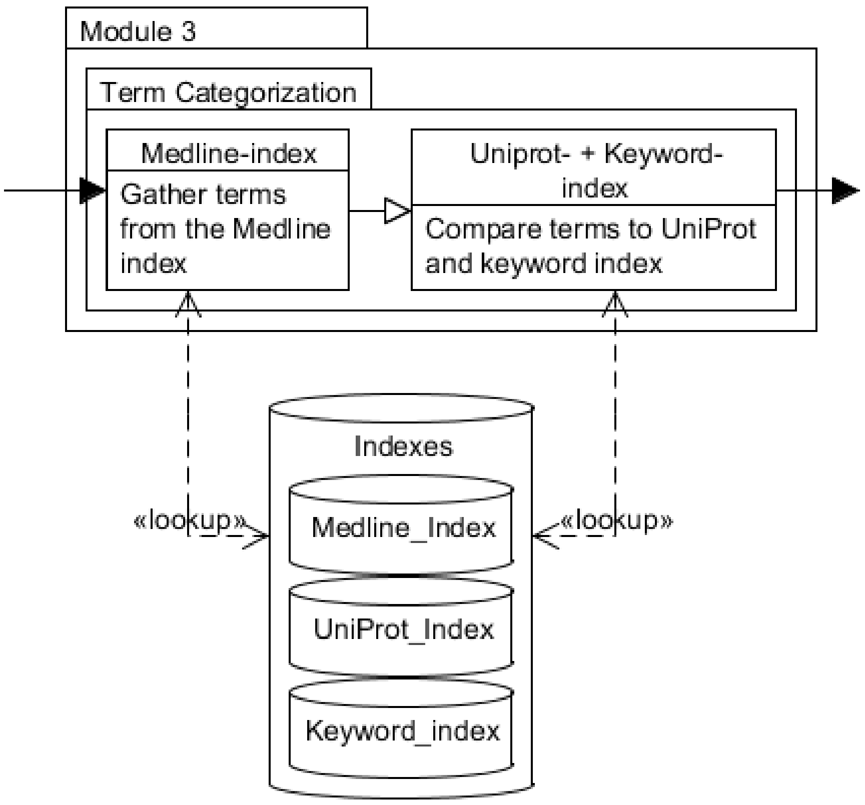

Figure 4.5: Illustration of Module 3

end, the list with potensial-short-name entities will become the primary list in a last-resort-strategy. In essence, the result will then be that an article (at least the PMID) will have two pointers, first to a list of protein name-tags and second to a list of keyword-tags.

4.6 Module 4 - The Evaluation

The last module (as seen in illustration 4.6) is basically where it all comes together. It takes all the entities tagged in the previous modul and compare them based on their offsets and position. For a given MEDLINE article representet by its PMID, its task is to check each interaction-keyword located in that article with all the protein names identified. If a keyword is found to have its position between two protein names, an interaction is presumed and MasterPPI will generate an output marking the article as one that has a PPI occurence. One should note that a keyword does not have to be *just* between the protein names, but can occur between other words as well, as long as their outer parts are protein names. The example sentence

> "Secondly, {***some protein name***} is essential for the ***activation*** of the first compartment-specific transcription factor {***some protein name***} in the prespore."

illustrates this. Here we see the keyword "activation" occur between two protein names, and MasterPPI would therefore label this as a comfirmed interaction.

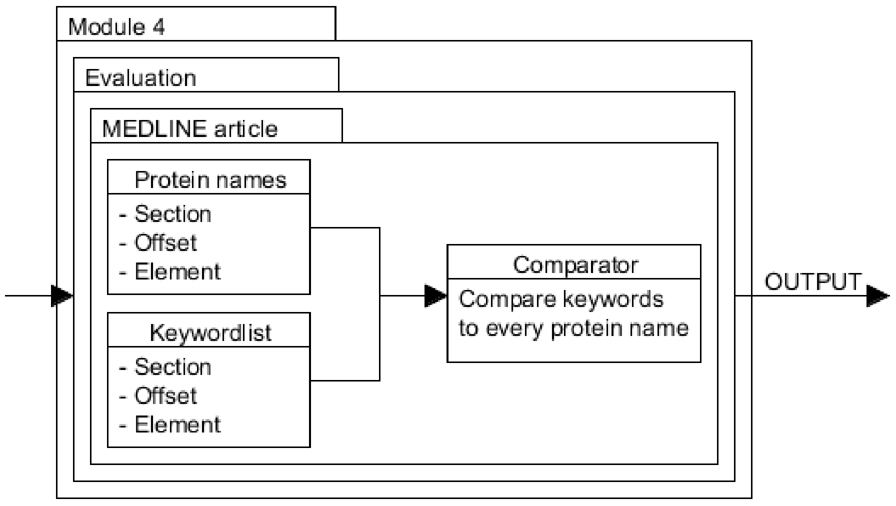

Figure 4.6: Illustration of Module 4

Chapter 5

Implementation

This chapter will elaborate on, and explain the choice of, the technologies that have been used in the making of MasterPPI. It will also dive into implementation details in the different classes and the system as a whole. Finally, two class diagrams are provided, to illustrate the connections and dependencies among the different classes in the system.

5.1 Technologies used

To be able to make any system, one may have to adopt a wide spectrum of different technologies, formats and applications. Here follows a short summary of answers to the "what" and "why" questions surrounding the use and choice for the adoptations in the development of this system.

5.1.1 Java

The system presented in this thesis has been achieved through the use of Java 1.6[1]. This is an object-oriented, cross-platform, programming language developed by James Gosling at Sun Microsystems in 1995, and is now maintained and distributed by Sun Microsystems (now a part of the Oracle Corporation[2]). One of greatest strengths of Java is its cross-platform ability. This is achieved by compiling the application to byte code, which is designed to run on any JVM (Java Virtuel Machine). Thus if a system has a JVM-implementation, any Java application can run on that system.

[1]`http://www.sun.com/java/`
[2]`http://www.oracle.com/`

Eclipse

Eclipse[3] is a free, open-source IDE (Integrated Development Environment), written in Java and distributed under the Eclipse Public License[4]. The system is primarily ment for the development of Java applications, but can with the help of an extensive plug-in system be extended to support other languages (C, C++, Python, etc.) and tasks (UML modeling, tex teditor, etc.), making it a powerful tool and ally when developing applications.

LingPipe

The LingPipe Project[5] offers a range of free, open-source libraries for linguistic analysis of the human language, and is also written in Java. The use of it in this thesis is limited to their parser of Medline abstracts, but it offers loads of more techniques that are relevant for the field of bio-informatics.

Lucene

Another library written[6] in Java, Lucene[7] is a free, open-source information retrieval engine that offers full text indexing and several searching options from a variety of text formats (PDF's, HTML, Word) given that the text can be extracted. The core of the Lucene architechture is the notion of *documents* and *fields*. One or several fields are added to a document where they work as categories for different tokens. The document will work as a record, and can be an actual document, a row from a database or something else entirely. This is all up to the programmer and what he or she is indexing. When a document is complete, it is added to an index along with an *Analyzer*. This entity determins the rules for how a token is made and how the index will be searched when queried.

Luke

Luke is a simple application that provides a GUI (Graphical User Interface) for viewing and manipulating the constructed Lucene index. One can browse a document, the fields constructed for it, the terms they consist of, and also provide the means to query the index with the help of different pre-compiled Lucene analysers.

[3]`http://www.eclipse.org/`
[4]`http://www.eclipse.org/legal/epl-v10.html`
[5]`http://alias-i.com/lingpipe/`
[6]But also portet to Delphi, Perl, C#, C++, Python, Ruby and PHP
[7]`http://lucene.apache.org/`

5.1.2 XML

XML is a mark-up language, derived from SGML (Standard Generalized Markup Language), produced by W3C[8] and designed to meet the challenges of large-scale electronic publishing by defining a set of rules for the encoding of the document. The advantage of XML is that a document will have a logical layout and have clear accesspoints to different data-entries. XML dates back to the 14th of November, 1996, when the first draft of the specification was published. The idea is to contain information in so-called *elements* that have a clear start- and end-points. For the XML-element

```
<name id='13'>Rincewind</name>
```

from some big XML-file which perhaps stores names from the Discworld-books, the start-point, or start-tag would be

```
<name id='13'>
```

where the id-part is that elements attribute, and the id's value is 13. Following this, the end-point or closing-tag, will be

```
</name>
```

encompassing the elements value, in this case the name Rincewind. A full XML-file can be seen in appendix B.

XML-parsing

When using XML-files with a programming language, it is necessery to transform them into strings and/or objects that the language can handle. This is achieved by parsing, and when it comes to this there is two dominant techniques to choose from: DOM[9] and SAX[10].

The DOM (Document Object Model) is, like the name implies, a way of representing the document and elements in HTML-, XML- and XHTML-files as objects. It also provides means for accessing these objects and tools for the manipulation of them. The model views the XML-file as a tree structure, with the root-element in the file being the root-node, parent-elements be parent-nodes and so on. With this representation any node/element can be

[8]http://www.w3.org/XML/
[9]http://www.w3.org/DOM/
[10]http://www.saxproject.org/

reached at any given time. The downside is that the whole tree has to reside in main memory. With files ranging from a couple of hundred megabytes to 5 gigabytes (for the UniProt database), this is an approach that will not work on a regular computer.

With the SAX (Simple API for XML) approach, the XML-tree is not directly mapped to a data structure as with DOM, but instead every element is viewed as a stream of events. The events correspond with when the parser encounters new elements as it traverses the document. It is up to the programmer to capture these events and make something useful of them. It is superior to the DOM in both memory and traversing, but instead of having access directly to elements, one would have to wait for the parser to encounter them and fire an event.

For the system proposed in this thesis, there will be several xml-files of different size, and some of them will be small enough (i.e the keyword list) for using the DOM approach. But other files, such as the UniProt database, will require the use of SAX.

5.2 Implementation details

MasterPPI conists of 11 java classes; with 4 i module 1, 2 in module 2, 3 in module 3 and 2 in module 4 respectfully. Here is a quick run-through of their functions and their responsibilities.

5.2.1 Module 1

MedlineParseImplementation.java

Handles the parsing of MEDLINE articles, with the help of LingPipe. The LingPipe parser is basically an extended SAX-parser, tailored for the XML-structure that MEDLINE provides its database in. The values extracted from the article will be the PMID (for identification), its title and its abstract. These are put in a Lucene Field object as part of making it ready for indexing. An article will thus consist of 3 fields; the first for the PMID, the second for the tokenized and analyzed title, and the third for the tokenized and analyzed abstract. The last two will each also have a Term-Field-Vector object made at index time, providing info on position and offsets of the terms.

PPIKeyWordParser.java

A very simple SAX-parser that takes the XML-file provided in appendix B as input and makes it ready for the indexer. When the parser encounters an end-tag of a word, the value of that elementet is extracted and made ready for indexing. Each keyword will be put in a Field object, consistent with its category. This will therefore give 15 fields in the index.

UniProtXMLParser.java

A SAX-parser that takes an XML-version of the UniProt database as input and extracts the full recommended name, short name, alternative name and the alternative short name. A full version of the name, as well as a tokenized version is made ready for indexing bringing the Field-count up to eight when finished.

NonFuncCharRemoval.java

Makes use of the 5 rules seen in section 4.3 to pre-process sentences before they go to tokenizing and indexing. Each rule uses regular expressions to deal with the task at hand. As can be seen in this example of rule 1:

```java
//Replaces any of the chars !"#$%&*<=>?@\| with a whitespace
private String rule1(String workingText) {
    return workingText.replaceAll("[\\!\"#\\$%&\\*<=>\\?\\@\\\\\\|]", " ");
}
```

Regular expression, or regex for short, gives concise and flexible ways of matching strings of text, such as induvidual letters, characters, phrases and/or syntax.

5.2.2 Module 2

CustomAnalyzer.java

This class extends the abstract Lucene Analyzer class for custom tailoring how filters are applied and how tokenization is performed. The stopwords filter will use the array seen in A and there is also a lower-case filter that is applied to a whitespacetokenizer.

CustomIndexer.java

A simple class that is tasked with handeling the Lucene IndexWriter instance, who writes the field objects to the index. Its output is the complete Lucene index.

5.2.3 Module 3

Entity.java

A datastructure for representing a potential important entity in a text. It details which section of the text the entity was found (title or abstract), its offsets and the element token it self. The class implements the Comparable interface, so that entities can be sorted and compared when put in an array(list) or a set(map).

PreQueryProcess.java

The Lucene search engine is a little picky when it comes to how a query is structured. An example being that if a parenthesis that have no "mate" (i.e have noe closing bracket or opening one) exists in a query, this will produce an error. To deal with this, the static PreQueryProcess class takes a potential query as its input and formates it in a way that Lucene will accept. Since this process can lead to a query-string that is Lucene-valid, but may not make any sense, e.g a lot of whitespaces and numbers or just some random characters put together, the class also sports a check function to see if the string is valid for searching.

TermCategorizer.java

With its 234 lines of code, the term-categorizer class is the biggest, and most essential in the whole system. MasterPPI relies on the identification of names and keywords, and both of these tasks are located here.

It starts of with extracting a document (a document is, in this case, a representation of a MEDLINE-article) and its terms from the MEDLINE-index through a IndexReader object that is part of the Lucene library. So for each field in a given document it extracts the terms and gives them to an IndexSearcher, as a query object. When constructing the queries for the searchers the PreQueryProcess class is used. If a term gets a match in the PPIKeyword-index, an Entity object is constructed with parameters gathered from the index, like its field and value, and its offset acquired from the Term-Field-Vector object of the given term. This Entity object is then stored in an Arraylist that is put in a TreeMap with the articles PMID as its key.

The protein name search works in a similar way, but is a bit more complex. The start is the same, where potential protein name tokens are identified and put in its respective Arraylist as Entity objects, the same way as with the keyword-search. After this though, it tries all of these as full protein names, before trying to combine tokens that have positions right next to each other. In other words, if the list contains two potential protein name -terms, where one of them occurs right after the other in the text, the system combines them into one entity and tries that in the full name search.

From this one would ideally get a list of perfectly matched protein names that can be put in a map with the PMID of the article as the key. This is not always the case for different reasons (some listed in section 6.3), so as a last desperate resort, if no protein names have been comfirmed, the potential list as a whole is included instead. This will not produce a good result, but a bad result is better than no result at all.

5.2.4 Module 4

Evaluation.java

This class will take the two maps generated by the term-categorizer and an Arraylist of PMIDs. It will then acquire the keyword-Entity list from the keyword map based on the PMIDs, and for each entry compare its offset to all of the protein entries in the corresponding protein name list. If a keywords start-offset comes after an end-offset from a protein name, and its end-offset comes before another protein names start-offset, the system will label the PMID as one where an interaction occurs, and the result is passed to the Output class.

Output.java

A class for writing the result from the evaluation to a plain text-file.

5.3 Class diagrams

As seen in following two class diagrams, the system kan be viewed as two entities; pre-indexing and post-indexing. In part 1 (figure 5.1) the pre-indexing is detailed showing the creation of the indices, and i part 2 (figure 5.2) is outlined showing the use of the indices. The system is flexible enough that these two can be combined with ease into one application, but for the sake of testing during development, it was divided into two parts.

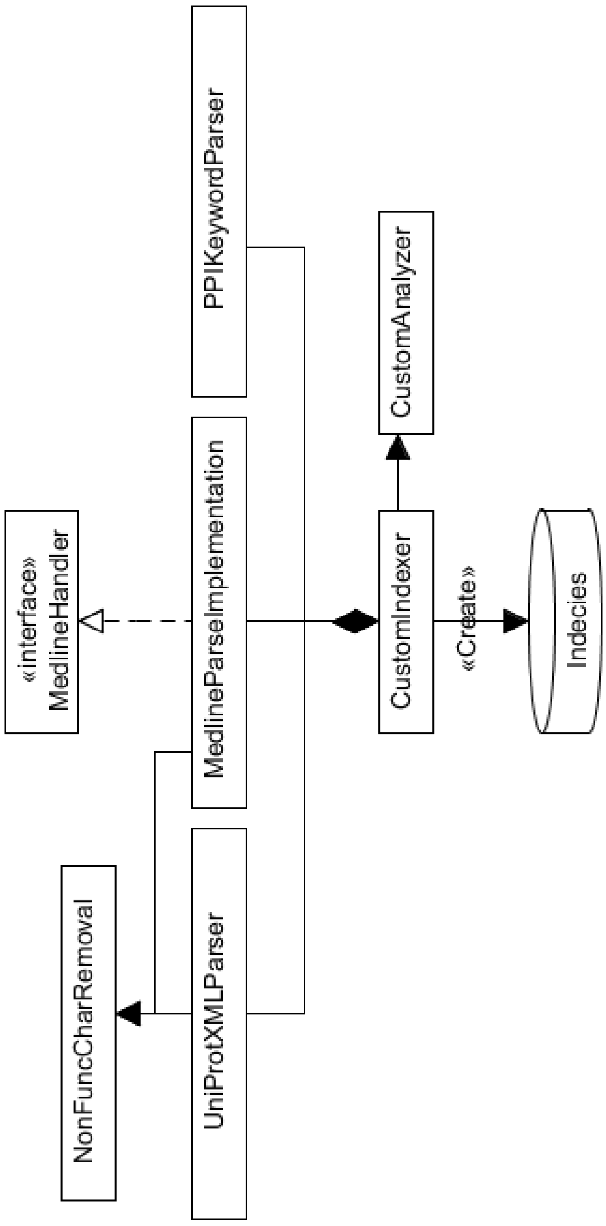

Figure 5.1: Class diagram Part 1

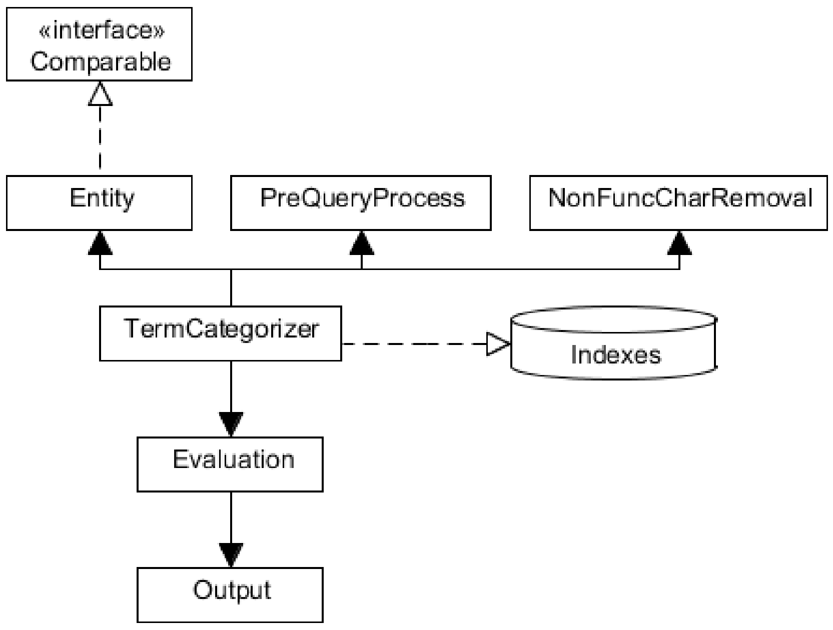

Figure 5.2: Class diagram Part 2

Chapter 6

Evaluation

This chapter will elaborate on the three evaluation methods the system has been tested against, and present and discuss the results acquired from these tests.

6.1 Evaluation Methods

Direct comparison of PPI/protein name extraction methods is difficult because some methods distinguish between proteins, genes and interactions and others do not. Further on, there is a wide variation in both the type and size of the test-sets used by each group to evaluate their methods. This problem har been adressed in [Pyysalo et al., 2008] and the authors have proposed a standard that, though still in beta, shows a lot of promise. The test-collection consists of five different, derived, variants of corpora, including AIMed[1], BioInfer[2], HPRD50[3], IEPA[4] and LLL[5]. From these there is compiled a training-set, a test-set with answerfiles and an evaluation-script. This script takes one of the pre-noted answer files, a compiled answer file from the system being tested and then produces an output that gives a precision-, recall-, F- and accuracy-score for the predictions the system being tested has made.

A downside with this standard is that it focuses solely on the interaction and not the proteins it occurs between. In fact they have blanced them all out, switched out each character in the protein name with underscores so

[1] `ftp://ftp.cs.utexas.edu/pub/mooney/bio-data/`

[2] `http://mars.cs.utu.fi/BioInfer/`

[3] `http://www.bio.ifi.lmu.de/publications/RelEx/`

[4] `http://class.ee.iastate.edu/berleant/s/IEPA.htm`

[5] `http://genome.jouy.inra.fr/texte/LLLchallenge/`

that the offsets still will be correct. This is problematic when it comes to the system this thesis presents, as it relies on confirming protein name occuranses and their offsets, *as well* as the PPI-keyword in a sentence or text. To deal with this, the test-collection made for the Yapex system (described in chapter 3.2) was introduced[6] as an evaluation method for detecting protein names. This collection consists of a training-set and a test-set, the former containing 99 MEDLINE-abstracts and the latter containing 101 and both annotated by domain experts connected to the Yapex project.

To be able to see how MasterPPI performed in a complete way, a third test-was introduced. This time from the IAS subtask in the BioCreAtIvE2 challenge that was also used to evaluate the ProtIR system (See chapter 3.5). Much like the Yapax-set, this features a set of annoteded MEDLINE-articles and a boolean answer-set (i.e if an article contains interactions or not).

The reason for using three different collections like this, is to be able to spot which part of the system that is lacking, thus getting a hint of where efforts can be put in to improve the system.

6.2 Results

MasterPPI was designed to work on MEDLINE-articles extracted directly from the MEDLINE database, using the LingPipe MEDLINE-parser. These test-sets, though all representing MEDLINE articles (excluding the PPI test-set, which consists of single sentences), where all structured in their own way, thus providing the need for three different parsers, evaluators and in the PPI test-sets case, a special term-categorizer. These were implemented as close to the original idea as possible. Thoughts and comments on the different results will be given in Section 6.3, and examples of the structure of the input documents can be seen in the following sections.

6.2.1 The Yapex Test collection

The system was tested on the 101 expert-annotated MEDLINE-articles in the test-part of the collection. In these articles there is identified 368 protein names in the title section, and 1528 in the abstract section, making a total of 1896 names in the whole collection. An example of how an article in the test collection is represented can be seen here in XML, with the abstract shortened for readability purposes:

[6] `http://www.sics.se/humle/projects/prothalt/`

```
<PubmedArticle>
  <MedlineID>21256213</MedlineID>
  <PMID>11357136</PMID>
  <ArticleTitle><Protname>LDL-receptor-related ↩
     → protein 6</Protname> is a receptor for↩
     → Dickkopf proteins.</ArticleTitle>
  <AbstractText><Protname>Wnt glycoproteins</ ↩
     →Protname> have been implicated in ↩
     →diverse processes during embryonic ↩
     →patterning in metazoa. They signal ↩
     →through frizzled-type seven-↩
     →transmembrane-domain receptors to ↩
     →stabilize     <Protname>beta-catenin</↩
     →Protname>. (...)
</PubmedArticle>
```

To clarify; these are unique names in the title and abstract field of an article. This means that a protein name can occur many times over the span of the different articles, twice in a single article and only once in a specific field. The reason for making this distinction is that when one is looking for a protein name like this, one does only need to find it once to connect the name and the document. To get more analytical data, the article was split into the two fields; the title field and the abstract field respectfully. This can later be used with weighing schemes that for example weigh an occurence in a title higher than an occurence in the abstract (see chapter 7.2 for future improvements)

With this setup, the system achieved a precicion-score of 0.08002, a recall-score of 0.47644 and thus a f-score of 0.13703.

6.2.2 The PPI Test collection

This test will evaluate how well a system is at recognizing PPI's on a sentence-based level, utilising the position and offsets of protein names and PPI-keywords in the text. A corpora is divided into documents, which will have multiple sentences. These sentences may have protein name entities occur in them, tagged with an offset, type (what kind of entity that occurs here; protein, protein compex, etc) and represented in the sentence with underscores. A sentence may also have a number of pairs that the system is tasked with checking. A pair will consist of two entities and the goal is to see if there is an interaction between these two. An example of this setup can be seen here:

```xml
<corpus id="AIMed">
  <document id="AIMed.d3" origId="11781834">
    <sentence id="AIMed.d3.s32" seqId="s32" text="The induced expression of ___, similar to ___, produces a senescent-like phenotype.">
      <entity charOffset="26-28" id="AIMed.d3.s32.e0" seqId="e54" type="protein" />
      <entity charOffset="42-44" id="AIMed.d3.s32.e1" seqId="e55" type="protein" />
      <pair e1="AIMed.d3.s32.e0" e2="AIMed.d3.s32.e1" id="AIMed.d3.s32.p0" />
    </sentence>
  </document>
</corpus>
```

Corpora	Precicion	Recall	F-Score	Accuracy	# of Pairs
AIMed	0.261	0.351	0.299	0.713	1095
HPRD50	0.526	0.385	0.444	0.643	70
IEPA	0.500	0.446	0.472	0.588	136
LLL	0.286	0.067	0.108	0.459	61
Overall Average	0.393	0.312	0.331	0.601	-

Table 6.1: Summary of the results from the PPI evaluation test collection

In table 6.1 the results from the different corpora evaluations have been listed, together with an overall average score. The BioInfer corpora was ommited from the evaluation due to inconsistency in the notation from the the other corpora. The problem was found in several entity elements, like this one:

```xml
<entity charOffset="121-127,141-150" id="BioInfer.d211.s0.e7" origId="e.492.10" type="Individual_protein" />
```

As one can see, the charOffset attribute has listed four offset points, but on manual inspection of position in the sentence the second one does not make any sense, often appearing in the middle of another word. Minus the pairs from the BioInfer corpora, the evaluation testet 1362 different pairs of possible PPIs.

6.2.3 The BioCreative Test collection

The last collection, which demands both recognition of protein names and the interactions between them, will give an indication of how MasterPPI performes as a complete system. The test-set given to the system as input consists of 677 unique MEDLINE articles, like the two collections before it; with PMID's, titles and abstracts. An example a representation of an article can be seen here, again with the abstract shortened:

```
<ENTRY>
  <CURATION_RELEVANCE>
  NONE
  </CURATION_RELEVANCE>
  <PPI_DATABASE>
  NONE
  </PPI_DATABASE>
  <PMID>
  16413544
  </PMID>
  <TITLE>
  Induction of apoptosis by p110C requires mitochondrial translocation of
  the proapoptotic BCL-2 family member BAD.
  </TITLE>
  <SOURCE>
  NONE
  </SOURCE>
  <ABSTRACT>
  p110C, a 50-kDa isoform of the PITSLRE kinase family, was demonstrated to play an important role in cell apoptosis. However, how p110C exactly promotes apoptosis is unclear. Our previous study showed that p110C    interacted with p21-activated kinase 1 (PAK1), an important kinase of the proapoptotic BCL-2 family member BAD, and evidently inhibited its kinase activity. (...)
  </ABSTRACT>
</ENTRY>
```

The answer-set in it self conists of just the boolean answer to the question if an article contains an interaction or not; represented by a P (positive) or a N (negative) after the articles PMID like this: 16272158 P or 16272159 N.

The following task was then to compare the answer-set file to the result file produced by MasterPPI and see if the predictions for an article matched. Of the 667 possible articles found in the test-set, MasterPPI matched 330 of them correct. Numbers from the evaluation are shown in table 6.2.

	Positive	Negative	Sum of articles
Answer-Set	338	339	667
Result-Set	175	502	667

Table 6.2: Analytical data from the BioCreative2 test collection

Of the 175 articles that MasterPPI labeled as positive hits; 83 were a match in the answer set, and of the 502 negative hits; 247 were correct. This gives MasterPPI, when it comes to finding an article where a PPI occurs, a precision of 0.4971 and a recall of 0.2574. This combined gives a F-Score of 0.3392.

6.3 Discussion and Comments

As suggested by the result from the Yapex test with its 8% precision-score, the system lacks an ability to correctly identify most of the protein names in a random bio-medical text. When manually trying terms and tokens[7] in the search engine found on the UniProt website, a lot of them got a direct hit in the Tremble database but not in the SwissProt part of UniProt. In other words, aleast some of the failed terms would be matched if the UniProt_index consisted of the whole database and not just the part that is manually annotated. One can therefore argue that this approach could be much more successful, finding more protein names and thus more possible interactions, with an expanded dictionary. It should also be noted that searching in only abstracts and titles, in contrast to the full text, forces the system to work on a form of text that may have an alternative writing style than the rest of the text. This comes from the fact that authors, in an abstract, are forced to compress a lot of information into just one page, thus having to make sacrifices and compromises on how to structure it. This case is reflected in [Müller et al., 2004], where the developers of Textpresso reported an improvment of almost 50% in recall when working on the full text, rather then the abstract.

[7]Tokens in this case being a collection of terms put together

Representing the MEDLINE articles as a Lucene index still seems like a good idea, since one gets a lot of information; like position, offset and frequencies for free when using it. This was very useful in the development of MasterPPI, but we also see potential uses when taking different approaches (e.g the use of weights and applying boost-factors). For the protein name dictionary, the use of Lucene gave a rather small index (266 megabytes), from a rather large database (several gigabytes), and gave the system an easy way of determin the presence of a protein name (given that the database is up to date, and complete) by the means of a simple query. We can therfore conclude that this is an efficient and easy way to set up, update and store a dictionary. Using a Lucene index as a dictionary for the keywords, may seem a little overkill in retrospect, given that the index takes up approximately the same size (8 kilobytes) as the XML-file it was constructed from on a harddisk. The reason for doing it this way, was to standarize the representation of how the different queries should look like, but this could just as easily have been represented as a Map or an ArrayList data structure directly within the system, saving Input/Output time. By large we will argue that the use of Lucene was a good idea.

There were not performed any tests with this system to see what kind of impact the removal of non-functional characters had, but a manually inspection of what was inserted in the index and queries looked promising, and we will put our trust in what Jing Jiang and Cheng Xiang Zhai reports in their studies[8].

When excluding the protein name identification part of the the system, and just looking at the PPI interaction, one sees that with just the naive approach of looking for PPI keywords between positively identified protein names gets an average precision-score of $\approx 39\%$; suggesting that it has some promise, but does not come close to the top systems described in chapter 3. Due note that a direct comparison with these would be invalid, since none of them were evaluated with the same test collection(s) as MasterPPI. On manually inspecting the sentences that were part of the PPI test-set, it was revealed that an interaction keyword is not bound exclusively to appear *between* the protein names, but sometimes can occur just before or just after, like in this sentence from the AIMed coropa:

Interaction of plant ________ with mammalian _____.

It can also provide false positives when occuring between two names, but have a negative spin on the sentence; e.g saying that this/that protein does *not* interact with this/that protein. This suggests that a more advanced NLP

[8][Jiang and Zhai, 2007]

approach, where such things can be detected and tagged, could yield better results.

Chapter 7

Conclusion and future work

This chapter will present a summary and a conclusion of the thesis to the reader, as well as thoughts on how to possible improve the system in future incarnations.

7.1 Summary and Conclusion

The goal for this thesis was to develop means that could identify and locate protein names and possible PPIs in a bio-medical text, store and index them and thus making them searchable. This thesis presents a system, dubbed MasterPPI, that can take an arbitrary number of MEDLINE articles, index and store them efficiently, and determine which proteins and what possible PPI's an article incompasses. The system was tested on three different test collections; the Yapex protein name test collection, the PPI test collection and the BioCreative2 IAS test collection, where the latter produced a f-score of approximatly 33%. The main factor for bringing the overall score down was identified to be the systems inability to identify complete protein names, due to an incomplete dictionary.

The findings show that representing both the MEDLINE articles and the dictionaries as Lucene indexes was an efficient and easy way of approaching the the problem at hand, but that the quality of the dictionary and the PPI search-method in itself was lacking. All in all we see promise in the structure of the data, but not so much in the rather naive identification method that was used.

7.2 Future improvements

As noted, including the Tremble part of the UniProt collection in the dictionary will greatly improve the ability of the system for recognising protein names. There is also nothing that stands in the way of including other databases like EntrezGene, though a potential problem is that one would get an overlap of names, making the index unnecessary big and complex. One would also have to develop a new parser, but this is just nitpicking.

Further on, the method for locating PPIs needs improvment. This can be done with different approaches; like extending the current method with more rules, use a weighing scheme, or use another NLP approach. We are confident that either way, incorporating it into MasterPPI should not be to much of a hassle. There is also the possibility of indexing a PPI database like DIP, construct queries from the MEDLINE articles in much the same way as was done in this approach, and see if they exist there. This will help to narrow down the real PPIs and exclude false positives, so one could see what was a real interaction and what was just blind luck.

On the system as a whole, an introduction of a GUI would not go amiss, and with this combining the two parts shown i figures 5.1 and 5.2 in to a complete system. There could also be introduced a new index that goes the other way around, linking protein names and PPIs to a specific article.

Appendices

Appendix A

```
String[] STOPWORDS = {
"a", "about", "again", "all", "almost",
"also", "although", "always", "among",
"an", "and", "another", "any", "are",
"as", "at", "be", "because", "been",
"before", "being", "between","both",
"but", "by", "can", "could", "did",
"do", "does", "done", "due", "during",
"each", "either", "enough", "especially",
"etc", "for", "found", "from", "further",
"had", "has", "have", "having", "here",
"how", "however", "i", "if", "in",
"into", "is", "it", "its", "itself",
"just", "kg", "km", "made", "mainly",
"make", "many", "may", "mg", "might", "ml",
"mm", "most", "mostly", "must", "nearly",
"neither", "no", "nor", "obtained", "of",
"often", "on", "or", "our", "overall",
"perhaps", "proteins", "pmid", "quite",
"rather", "really", "regarding", "seem",
"seen", "several", "should", "show",
"showed", "shown", "shows", "significantly",
"since", "so", "some", "such", "than",
"that", "the", "their", "theirs", "them",
"then", "there", "therefore", "these",
"they", "this", "those", "through",
"thus", "to", "upon", "use", "used",
"using", "various", "very", "was", "we",
"were", "what", "when", "which", "while",
"with", "within", "without", "would"
};
```

Appendix B

```xml
<?xml version='1.0' encoding='UTF-8'?>
<root>
    <keywordlist>
        <category id="Activate">
            <word>accumulate</word>
            <word>accumulated</word>
            <word>accumulates</word>
            <word>accumulation</word>
            <word>activate</word>
            <word>activated</word>
            <word>activates</word>
            <word>activation</word>
            <word>elevate</word>
            <word>elevated</word>
            <word>elevates</word>
            <word>elevation</word>
            <word>hasten</word>
            <word>hastened</word>
            <word>hastenes</word>
            <word>incite</word>
            <word>incited</word>
            <word>incites</word>
            <word>increase</word>
            <word>increased</word>
            <word>increases</word>
            <word>induce</word>
            <word>induced</word>
            <word>induces</word>
            <word>induction</word>
            <word>initiate</word>
            <word>initiated</word>
            <word>initiates</word>
            <word>promote</word>
            <word>promoted</word>
            <word>promotes</word>
            <word>stimulate</word>
            <word>stimulated</word>
            <word>stimulates</word>
            <word>stimulation</word>
            <word>transactivate</word>
            <word>transactivated</word>
            <word>transactivates</word>
            <word>transactivation</word>
            <word>up-regulate</word>
            <word>up-regulated</word>
            <word>up-regulates</word>
            <word>up-regulator</word>
            <word>up-regulation</word>
            <word>upregulate</word>
            <word>upregulated</word>
            <word>upregulates</word>
            <word>upregulator</word>
        </category>
        <category id="Association">
            <word>associate</word>
            <word>associated</word>
            <word>associates</word>
            <word>association</word>
        </category>
        <category id="Attach">
            <word>add</word>
            <word>adds</word>
            <word>addition</word>
            <word>bind</word>
            <word>binds</word>
            <word>bound</word>
            <word>catalyze</word>
            <word>catalyzed</word>
            <word>catalyzes</word>
            <word>complex</word>
        </category>
        <category id="Break Bond">
            <word>cleave</word>
            <word>cleaved</word>
```

```xml
            <word>cleaves</word>
            <word>demethylate</word>
            <word>demethylated</word>
            <word>demethylates</word>
            <word>demethylatation</word>
            <word>dephosphorylate</word>
            <word>dephosphorylated</word>
            <word>dephosphorylates</word>
            <word>dephosphorylatation</word>
            <word>sever</word>
            <word>severe</word>
            <word>severed</word>
            <word>severes</word>
        </category>
        <category id="Cause">
            <word>influence</word>
            <word>influences</word>
            <word>influenced</word>
        </category>
        <category id="Contain">
            <word>contain</word>
            <word>contained</word>
            <word>contains</word>
        </category>
        <category id="Create Bond">
            <word>methylate</word>
            <word>methylated</word>
            <word>methylates</word>
            <word>methylation</word>
            <word>phosphorylate</word>
            <word>phosphorylated</word>
            <word>phosphorylates</word>
            <word>phosphorylation</word>
        </category>
        <category id="Generate">
            <word>express</word>
            <word>expressed</word>
            <word>expresses</word>
            <word>expression</word>
            <word>overexpress</word>
            <word>overexpressed</word>
            <word>overexpresses</word>
            <word>overexpression</word>
            <word>produce</word>
            <word>produced</word>
            <word>produces</word>
            <word>production</word>
        </category>
        <category id="Inactivate">
            <word>block</word>
            <word>blocks</word>
            <word>blocked</word>
            <word>decrease</word>
            <word>decreased</word>
            <word>decreases</word>
            <word>decreasing</word>
            <word>deplete</word>
            <word>depleted</word>
            <word>depleting</word>
            <word>depletes</word>
            <word>depletion</word>
            <word>down-regulate</word>
            <word>down-regulated</word>
            <word>down-regulates</word>
            <word>down-regulation</word>
            <word>downregulate</word>
            <word>downregulated</word>
            <word>downregulates</word>
            <word>downregulation</word>
            <word>impair</word>
            <word>impairs</word>
            <word>impaired</word>
            <word>inactivate</word>
            <word>inactivated</word>
            <word>inactivates</word>
            <word>inactivation</word>
            <word>inhibit</word>
            <word>inhibits</word>
            <word>inhibited</word>
            <word>inhibition</word>
            <word>reduce</word>
            <word>reduced</word>
            <word>reduces</word>
            <word>reduction</word>
            <word>repress</word>
            <word>represses</word>
            <word>repression</word>
            <word>repressed</word>
            <word>supress</word>
            <word>supressed</word>
            <word>supresses</word>
            <word>supression</word>
        </category>
        <category id="Modify">
            <word>modify</word>
            <word>modified</word>
            <word>modification</word>
        </category>
        <category id="Process">
            <word>apoptosis</word>
            <word>myogenisis</word>
        </category>
        <category id="React">
            <word>interact</word>
            <word>interacts</word>
            <word>interacted</word>
            <word>interaction</word>
            <word>react</word>
            <word>reacts</word>
            <word>reacted</word>
            <word>reaction</word>
        </category>
        <category id="Release">
            <word>disassemble</word>
            <word>disassembles</word>
            <word>disassembled</word>
            <word>discharge</word>
            <word>discharged</word>
            <word>discharges</word>
        </category>
        <category id="Signal">
            <word>signal</word>
            <word>signals</word>
            <word>signaled</word>
            <word>signalled</word>
            <word>signales</word>
            <word>mediate</word>
            <word>mediates</word>
            <word>mediated</word>
```

```xml
            <word>modulate</word>
            <word>modulated</word>
            <word>modulates</word>
            <word>modulation</word>
            <word>participate</word>
            <word>participated</word>
            <word>participates</word>
            <word>participation</word>
            <word>regulate</word>
            <word>regulates</word>
            <word>regulated</word>
            <word>regulation</word>
            <word>autoregulate</word>
            <word>autoregulates</word>
            <word>autoregulated</word>
            <word>autoregulation</word>
            <word>auto-regulate</word>
            <word>auto-regulates</word>
            <word>auto-regulated</word>
            <word>auto-regulation</word>
        </category>
        <category id="Substitute">
            <word>replace</word>
            <word>replaces</word>
            <word>replaced</word>
            <word>substitute</word>
            <word>substituted</word>
            <word>substitutes</word>
            <word>substitution</word>
        </category>
    </keywordlist>
</root>
```

Bibliography

[Bass et al., 2007] Bass, L., Clements, P., and Kazman, R. (2007). *Software Architecture in Practice.* SEI Series. Addision Wesly, second edition edition.

[Berk et al., 2000] Berk, A., Zipursky, L., Matsudaira, P., Baltimore, D., and Darnell, J. (2000). *Molecular Cell Biology.* W. H. Freeman, 4th edition. Glossary.

[Bret, 1951] Bret, S. (1951). *Qu'est-ce que la documentation.*

[Bryson, 2003] Bryson, B. (2003). *A short history of nearly everything.* Broadway Books.

[Buckland, 1997] Buckland, M. K. (1997). *What is a "document"?*

[DICT, 2010] DICT (2010). *Pneumonoultramicroscopicsilicovolcanoconiosis.* `http://dictionary.reference.com/browse/Pneumonoultramicroscopicsilicovolcanoconiosis`. dictionary.com, based on Random House Dictionary.

[DIP, 2010] DIP (2010). *DIP Statistics.* `http://dip.doe-mbi.ucla.edu/dip/Stat.cgi`. Database statistics for the Database of Interacting Proteins.

[Egorov et al., 2004] Egorov, S., Yuryev, A., and Daraselia, N. (2004). *A Simple and Practical Dictionary-based Approach for Identification of Proteins in Medline Abstracts. Journal of the American Medical Informatics Association*, 11(3):174–127.

[Franzén et al., 2002] Franzén, K., Eriksson, G., Olsson, F., Asker, L., Lindén, P., and Cöster, J. (2002). *Protein names and how to find them. International Journal of Medical Informatics*, 67(1-3):49 – 61.

[Frawley et al., 1992] Frawley, W. J., Piatetsky-Shapiro, G., and Matheus, C. J. (1992). Knowledge Discovery in Databases: An Overview. *AI Magazine*, 13(3).

[Heppin, 2008] Heppin, K. F. (2008). *MedEval - The Construction of a Swedish Medical Test Collection.*

[Hersh et al., 1994] Hersh, W. R., Hickam, D. H., Haynes, R. B., and McKibbon, K. A. (1994). *A Performance and Failure Analysis of SAPHIRE with a MEDLINE Test Collection. The Journal of the American Medical Informatics Association*, 1:51–60.

[IES, 2010] IES (2010). *Biology - The science of life.* `http://www.intstudy.com/articles/wc266a04.htm`. The International Education Site.

[Jiang and Zhai, 2007] Jiang, J. and Zhai, C. X. (2007). *An empircal study of tokenization strategies for biomedical information retrieval. Information Retrieval*, 10(4-5):341–363.

[Kamber and Han, 2006] Kamber, M. and Han, J. (2006). *Data mining: concepts and techniques.* Morgan Kaufmann, second edition.

[Krauthammer and Nenadic, 2004] Krauthammer, M. and Nenadic, G. (2004). *Term identification in the biomedical literature. Journal of Biomedical Informatics*, 37(6):512 – 526. Named Entity Recognition in Biomedicine.

[Manning et al., 2008] Manning, C. D., Raghaven, P., and Schütze, H. (2008). *Introduction to Information Retrieval.* Cambridge University Press.

[Müller et al., 2004] Müller, H.-M., Kenny, E. E., and Sternberg, P. W. (2004). *Textpresso: An Ontology-Based Information Retrieval and Extraction System for Biological Literature. PLoS Biol*, 2(11):e309.

[Monsen, 2007] Monsen, T. H. (2007). *Googler gener.* `http://www.forskning.no/artikler/2007/mai/1179818465.89`.

[Mooney and Nahm, 2005] Mooney, R. J. and Nahm, U. Y. (2005). Text mining with information extraction. `http://www.cs.utexas.edu/users/ml/papers/discotex-melm-03.pdf`.

[Nair, 2007] Nair, A. S. (2007). *Computational Biolgy and Bioinformatics: A gentle overview.* `http://bit.ly/b40ZU4`.

[NAT, 2004] NAT (2004). *Glossary.* `http://www.nature.com/nrg/journal/v5/n7/glossary/nrg1383_glossary.html`. Nature.com - ”The worlds best science and medicine on your desktop”.

[NIST, 2008] NIST (2008). *Trec overview.* `http://trec.nist.gov/overview.html`. National Institute of Standards and Technology.

[NLM, 2010] NLM (2010). *Data, News and Update Information.* `http://www.nlm.nih.gov/bsd/revup/revup_pub.html`. National Library of Medicine.

[ProteinCrystallography.org, 2007] ProteinCrystallography.org (2007). . `http://proteincrystallography.org/protein/`. A web project started in January 2007 by Dr. Sergey Ruzheinikov, University of Sheffield.

[Pyysalo et al., 2008] Pyysalo, S., Sætre, R., Tsjuii, J., and Salakoski, T. (2008). *Why Biomedical Relation Extraction Results are Incomparable and What to do about it.*

[Ramampiaro et al., 2007] Ramampiaro, H., Chen, Y. H., Lægrid, A., and Sætre, R. (2007). *ProtIR prototype: abstract relevance for Protein-Protein Interaction in BioCreAtIvE2 Challenge, PPI-IAS subtask.*

[Rivas and de Luis, 2004] Rivas, J. D. L. and de Luis, A. (2004). *Interactome data and databases: different types of protein interaction. Comparative and Functional Genomics*, 5(2):173–178.

[Rosario, 2005] Rosario, B. (2005). *Extraction of semantic relations from bioscience text.* PhD thesis, Berkeley, University of California.

[Skuland, 2005] Skuland, M. (2005). *Identification of biomedical entities from Medline abstracts using a dictionary-based approach.* Master’s thesis, NTNU.

[Temkin and Gilder, 2003] Temkin, J. M. and Gilder, M. R. (2003). *Extraction of protein interaction information from unstructured text using a context-free grammar. Bioinformatics*, 19(16):2046–53.

[TXT, 2010] TXT (2010). *Aquarena Wetlands Project: Glossary of Terms.* `http://www.bio.txstate.edu/~wetlands/Glossary/glossary.html`. Texas State University.

[WSMNS, 2008] WSMNS (2008). *Life Science.* `http://community.weber.edu/sciencemuseum/pages/life_main.asp`. Weber State Museum of Natural Science.