MUTAIMPACT MINER: A GENE-MUTATION IMPACT EXTRACTOR AND CURATOR FOR BIOMEDICAL LITERATURE

by

SAGAR SANTOSH TARKHADKAR

(Under the Direction of Krzysztof J. Kochut)

ABSTRACT

Manual curation of knowledge from biomedical literature is both expensive and time consuming. Scientific publications in biomedicine have an enormous amount of valuable information on gene mutations and their impacts, which is significant in addressing multiple research problems. In this thesis, we have developed a text mining system for extracting and curating mutation impacts from full text scientific documents. The objective of this system is to populate biomedical knowledge-bases with accurate knowledge regarding mutation impacts, in a semi-automated way. We have used a number of Natural Language Processing tasks in developing this system. Furthermore, a curation module allows the scientists to decide if the mutation impact information is suitable to be included to the knowledge base, hence eliminating the possibility of adding incorrect data. Our prototype system has been used in the Protein Kinase domain, but can be adapted to work in other domains, in the future.

INDEX WORDS: text mining, NLP, semantic web, ontology, bioinformatics

MUTAIMPACT MINER: A GENE-MUTATION IMPACT EXTRACTOR AND CURATOR FOR BIOMEDICAL LITERATURE

by

SAGAR SANTOSH TARKHADKAR

BE, University of Mumbai, India, 2011

A Thesis Submitted to the Graduate Faculty of The University of Georgia in Partial Fulfillment of the Requirements for the Degree

MASTER OF SCIENCE

ATHENS, GEORGIA

2013

© 2013

Sagar Santosh Tarkhadkar

All Rights Reserved

MUTAIMPACT MINER: A GENE-MUTATION IMPACT EXTRACTOR AND CURATOR BIOMEDICAL LITERATURE

by

SAGAR SANTOSH TARKHADKAR

Major Professor: Kzysztof J. Kochut

Committee: Khaled Rasheed
Ismailcem Budak Arpinar

Electronic Version Approved:

Maureen Grasso
Dean of the Graduate School
The University of Georgia
December 2013

DEDICATION

I dedicate this work to my parents and my grandmother, who provided me constant support and motivation and continuously pushed me to better myself.

ACKNOWLEDGEMENTS

I would like thank Dr. Kochut for his guidance, not only in this project, but also in my overall graduate academic career. I would also like to thank Dr. Natarajan Kannan from the Department of Biochemistry and Molecular Biology and the Institute of Bioinformatics at UGA for his valuable input and knowledge in Life Sciences. I also appreciate the help of Daniel McSkimming, a doctoral student in the Institute of Bioinformatics at UGA for his help in evaluating my work. Also thanks to my friends here at UGA for all the joyous moments.

TABLE OF CONTENTS

Page

LIST OF TABLES

Page

LIST OF FIGURES

Page

CHAPTER 1

INTRODUCTION

A lot of research is carried out in the field of biomedicine, bioinformatics and other similar fields by researchers with the objective of answering many of nature's difficult questions. Majority of the study results in these fields are published in documents and journals and other forms of scientific literature. One of the major research problems is extraction of *important* or *interesting* information from the literature, which might help scientists in research in their respective field of study. As an example, manual curation of biomedical literature is time consuming. To gain useful information about even a single gene mutation, curators have to sift through hundreds or even thousands of scientific articles, study them in detail and understand what the articles intend to convey. Not only that, but once some information is gained, it is not entirely sure how valuable the information is to be added to the respective knowledge base. A knowledge base is a special kind of database for knowledge management. Most of the knowledge bases are dedicated to a specific domain, like for example, protein kinases. Some of the examples of knowledge bases in the field of biomedicine include UniProt [22] and COSMIC [20]. A few other examples are biomedical ontologies such as Gene Ontology [14] and ProKino [15, 53]. For most of the biomedical text extraction systems, the extracted knowledge is curated and stored in a knowledge base. This knowledge can be later used for a number of purposes, such as data analysis and hypothesis formulation and testing by querying the knowledge base. Hence, it is very important to populate the knowledge base only with information relevant to that domain.

In this work, we have developed a text mining system that analyses full text biomedical documents and extracts impact of gene mutations of interest. To avoid incorrect or non-relevant information to be processed further, the system also provides a curation module to help scientists in assessing the quality of the extracted information. The perfect automated biomedical text mining system is yet to be developed since extraction of knowledge base from biomedical natural language text is an error prone process. Hence, some level of human expertise and judgment is needed, which ensures that information added to the knowledge base is accurate. The advantage of developing this system is the reduced amount of research time, which directly impacts the effort involved in this project. The system uses information from an ontology to find specific terms from full text and later uses a Natural Language Processing evaluation module to find relationships between these terms. The results, consisting of the extracted mutation, impact word, impact statement and the impact type, are displayed for a curator with options to decide which of the results can be classified as important and accurate.

This thesis is organized as follows:

Chapter 2 includes background information on text mining, natural language processing, ontologies and ontology population, the ProKino ontology and curation. Chapter 3 includes a review of related work. Chapter 4 includes the objectives of the system, text pre-processing, NLP tasks like tagging, parsing and grammatical dependencies and curation. Chapter 5 includes the actual implementation of the system including the system architecture. Chapter 6 includes results and evaluation. Finally, chapter 7 concludes this work and discusses future work.

CHAPTER 2

BACKGROUND

2.1 Text Mining

Text mining refers to the uncovering of unknown knowledge that can be found in text or in a group of text documents [1]. There are two other areas that are similar to text mining: Information Extraction (IE) and Information Retrieval (IR). Even though at the first instance, all three methods look towards having similar objectives, they are quite different from a specific application point of view. The objective of IR is to return documents that match a query either fully or partially. IE refers to the extraction of "structured data and predetermined relationships" that matches the interest of a user [1]. Text mining takes into account more difficult obstacles to conquer, such as dealing with and making sense of unstructured text from various sources (most likely of the same topic) and recognizing textual patterns. In text mining, hidden facts are discovered from unstructured, natural language text. The whole idea of text mining is to get the gist of the whole document, whether general or specific, depends on the needs of the user. Projects that involve text mining need to consider many issues with respect to natural language, which we have discussed in detail in the Natural Language Processing subchapter.

When retrieving information, stop words are typically removed since they occur in high frequencies. Stop words are words that are filtered out prior to, or after, the processing of text. Many stop words can affect a keyword search negatively and return undesired results. However, text mining applications may need to utilize stop words in order to semantically make sense of a sentence or phrase [1]. Stemming is another technique that is used to reduce a word to its root,

increasing similarity comparisons between words [1]. This technique can also have an unfavorable effect on the semantics of the text.

Another issue is Word Sense Disambiguation (WSD) [1]. A word that has more than one meaning can exist in a body of text, and the sense of the word, whether it is a noun, verb adjective, etc. should be detected or decided using two types of disambiguation. Supervised disambiguation involves the aid of a dictionary or thesaurus, whereas in unsupervised disambiguation, the possible senses of a word are unknown [1].

Tagging text using Extensible Markup Language (XML) [2] or some mark-up language aids the text mining process greatly. It allows the text to be more structured, and the meaning of sentences, phrases, and words can be discovered and used more easily than if such mark-up of text were not present. Within this text, part-of-speech (PoS) [3] tags should be present if the researcher wishes to gain the best results out of his/her project. These PoS tags are used to identify nouns, verbs, adjectives, adverbs, etc. such that even larger parts of the text can be identified, such as verb phrases and prepositional phrases. Also, some groups of words exist as to convey a single idea. If the group of words is to be separated, each individual word of that group takes on a whole new meaning that does not relate to the meaning of the words when grouped together. A collocation refers to two or more words that are grouped together to convey a single concept or idea [1].

Tokenization allows text to be split into different parts, such as documents, sections, paragraphs, sentences, phrases, and words [1]. When analyzing data, this act allows the researcher to more easily locate the text that is being dealt with. Tokenization is not an easy process because the delimiters used may not necessarily be a standard used for all the documents the researcher has to work with [1]. For instance, an empty line may not separate all paragraphs; one text may do so

while another may not. Typically, all sentences end in some form of punctuation. However, abbreviations containing a period, for instance, throw a wrench in the sentence splitting process. In some rare cases, you might have noun modifiers which have certain symbols that can be wrongly interpreted by the system as punctuation.

In the biomedical domain, the rate of publications is quite high, and scientists working in this area need systems to aid them in searching and sifting through information in numerous scientific articles. Even if the researchers do find interesting articles, it takes a lot of time to study and analyze every single one of them and get important information. Although many systems have been created in the past to aid researchers in this task, there are systems still being created today with a slightly different goal in mind. Many of the early biomedical text mining systems used Natural Language Processing (NLP) [4] techniques to understand the text grammatically [5].

There are many types and definitions of text mining. The strictest definition of a text mining system is one that returns knowledge that cannot be found explicitly in text [5]. Another definition of a text mining system is one that extracts information from text or takes into account the prerequisites necessary for extracting information [5]. Some earlier systems aimed to extract information from text by first recognizing named entities (often called Named Entity Recognition) and then realizing relationships between those entities [5]. Once these relationships are established, an ontology-like form exists such that inferred relationships between entities can be made known without the relationship being visible in the original text. The driving factor behind text mining systems is that they should focus on user needs. The needs or requirements of the user must drive the technology behind text mining systems.

Many of the early text mining systems focused on NER. Entities that are proper nouns in text are identified or recognized, and then the system may attempt to classify the entities based on predefined categories [5]. The NER task most often leads to relationship extraction between entities as a second step. These relationships are discovered using statistical approaches that capture how two entities are related when co-existing in the same sentence, for example [5]. In most systems that perform this task, PoS tagging and NLP techniques are used [5]. A few years ago, elaborate parsing methods and syntactic structures were computationally expensive as compared to today [5]. These types of systems have performed quite well, using methodologies like machine learning and NLP [5]. However, parsing raw text and programming a computer to understand the text grammatically requires the researcher to be proficient in computer science, artificial intelligence, and linguistics. Although such systems are still being researched and created, there are also systems that do not include such ideas but rather focus on the core needs of the user.

A type of text mining that is perhaps consistent with the strictest definition is text summarization [5]. There are three basic types of summaries: indicative, informative, and targeted [5]. Indicative summaries help the user decide whether or not (s) he would be interested in reading the underlying document [5]. Informative summaries are intended to deliver content of the document to the user [5]. Targeted summaries seek to fulfill a user's request, which is usually expressed as a query [5]. The last type of summary can be very difficult to generate even when dealing with one specific domain [5]. It may involve assigning sentences in a document to predefined categories and determining how relevant the sentences are to the categories [5]. The most relevant sentences are included in the final generated summary.

Question answering systems aim to provide short answers to questions posed by users [5]. Most often the answers contain links to supporting documents from which the answers are derived [5]. In this manner, the system can be evaluated by allowing users to decide how closely the answer matches the supporting documents. Initially, question-answering systems were targeted towards open domain applications, but more recently they have become directed towards restricted domains - first the clinical domain and later genomics [5]. And even though the clinical domain questions have been widely recognized, question answering specifically for medicine is a relatively new field.

Literature-based discovery is another form of text mining, which involves inferring new ideas or hypotheses from literature that are not obvious from human reading [5]. More work has been done in the past as researchers today find NLP techniques to be computationally too costly for practical, user-friendly systems. Full parsing of text is the most demanding while co-occurrence of terms, though less computation is needed, may still produce promising results [5]. In either case, such systems will need to use some form of NLP due to the nature of the desired results: information not present in the raw text is discovered computationally. These systems incorporate such concepts as NER and relationship discovery. Most of these systems probably use co-occurrence since words appearing within the same context are likely to be related [5]. Words may belong to the same category (strong relationship), same sentence, same paragraph, or same document (weak relationship). Instead of performing NLP on text and risking mistakes from the computer (language is not perfect), the user can decide, based on words appearing within the same context, if the text surrounding the words represents the answer or outcome sought after. The key to evaluating the usefulness of a text mining system lies in compiling feedback from users. What one user perceives as important information may turn out to be bogus for another.

The results provided by text mining systems are somewhat subjective, and users can give their opinions (feedback) on how useful and successful such systems are at fulfilling their needs. The use of ontologies facilitates better user interaction within these types of systems.

2.2 Ontologies

An ontology can be defined as specification of a conceptualization [6]. The word ontology originates from philosophy and is the study of the "nature of being, existence, or reality" [7, 8]. It deals with which entities exist, how they are related to each other, or how they can be grouped together. An entity, as defined in philosophy, is usually a noun or abstract noun.

Ontology in information science, however, takes on a whole new meaning. While philosophers discuss the idea of ontology, computer and information scientists create ontologies and use them in software systems. It can be even as simple as creating a class like say 'Student' to represent a student entity. The creation of classes has long been a staple of object oriented programming in computer science. Since the rise of object oriented programming, many software systems contain entities or abstract nouns, but the relationship between those objects is unknown and is difficult to represent. There is a better way to represent a system containing many entities, nouns, or abstract nouns without creating objects in memory.

An ontology is a representation of entities and their relationships with each other. In a computer, a serialization of an ontology can be represented as a file in auxiliary memory. The file representation can be in various formats, including Resource Description Framework (RDF) [9] and Web Ontology Language (OWL) [10]. The World Wide Web Consortium (W3C) coined the term "Semantic Web," which refers to technologies designed to help people "create data stores on the web, build vocabularies, and write rules for handling data" [11]. RDF provides the ability

to publish and link data, while OWL aims to improve on the capabilities of RDF by giving data additional meaning [11]. The W3C has created an RDF-specific query language named SPARQL Protocol and RDF Query Language (SPARQL) that makes it possible to query an ontology using triple pattern query syntax [11]. At least one resource reference in a SPARQL query is a variable, and upon execution, the SPARQL engine returns the resources for all triples that match the query pattern [11]. Inference comprises procedures to infer or realize new relationships and connections based on existing data and a vocabulary, such as a set of rules [11].

The primary focus of rules is to define mechanisms for discovering new relationships based on existing ones [11]. The RDF data model represents statements in XML syntax about resources in the form subject-predicate-object, which are also known as triples [9]. The subject is the resource, and the predicate represents the relationship between the subject and some object. A collection of RDF statements also represents a directed graph of nodes and edges, where the entities represent nodes and relationships represent edges [9]. OWL is primarily used in applications that need to "process the content of information" rather than simply displaying information [10]. It promotes better "machine interpretability" of content than XML and RDF since it contains additional syntactic vocabulary along with formal syntax [10]. The OWL vocabulary can represent relations between classes, cardinality, equality, richer property types, characteristics of properties, and enumerated classes [10]. OWL is available in three sublanguages, namely OWL Lite, OWL DL, and OWL Full [10]. OWL Lite is the least complex language that can represent a "classification hierarchy and simple constraints", such as providing cardinality constraints of only 0 or 1 [10]. OWL DL provides a highly expressive vocabulary, yet it still maintains "computational completeness and decidability" [10]. OWL Full is only to be used by those who want full expressiveness of OWL with no "computational guarantees"

[10]. OWL Full is an extension of RDF, but OWL DL and OWL Lite are "extensions of a restricted view of RDF" such that every OWL document is an RDF document, and every RDF document is an OWL Full document [10]. A triple in an ontology represents two entities connected by a relationship, where one entity is the subject and the other is the object; the relationship is commonly referred to as the predicate. Alternatively, a triple can also be represented as a directed graph edge, where each entity is a node. A relationship between two entities in an ontology can be visualized in Figure 1.

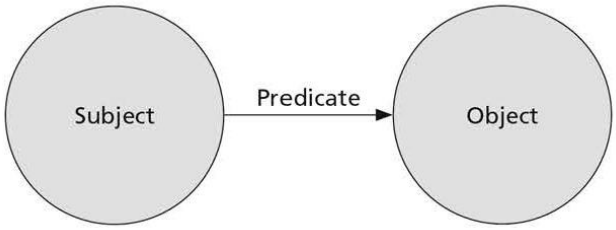


Figure 1. Graphical representation of a triple

The subject is commonly referred to as the primary entity of interest. The object, as you may have guessed, is commonly referred to as the range, which is specific to the subject.
In comparison to the definition of an ontology, a "specification of a conceptualization," [6] the subject is the concept of interest, and all objects related to a subject represent the specification. As an example, let us have the subject *T790M*, a mutation, associated with a protein property say, *kinase*. With the two entities *T790M* and *kinase*, a relationship between them would be helpful. Consider the relationship *associatedWith* to represent the relationship between *T790M* and *kinase*. This can be used to show that *T790M* mutation is associated with the property *kinase*. The subject node is labeled *T790M* , the predicate edge is labeled *associatedWith*, and the object node is labeled *kinase*.

Today, ontologies are being often used within the biological sciences domain. New terms and concepts are being created or realized at a rapid rate, and scientists need a way of keeping track of these concepts, new and old, and how they are related to each other. Since ontologies are both persistent and show relationships between entities, they are a good solution for use in applications and systems that help biological scientists perform research more easily and more quickly than if their research was performed manually. The Open Biological and Biomedical Ontologies (OBO) is a "collaborative experiment involving developers of science based ontologies who are establishing a set of principles for ontology development with the goal of creating a suite of orthogonal interoperable reference ontologies in the biomedical domain" [12]. The OBO Foundry is the collaborative organization responsible for maintaining these biomedical ontologies, and they currently have 102 ontologies available for use. BioPortal [13] is another organization that maintains biomedical ontologies, and it currently hosts 366 ontologies. An example of a biological ontology hosted by both the OBO Foundry and BioPortal is the Gene Ontology (GO) [14]. The aim of the GO ontology is to provide a "controlled vocabulary that can be applied to all eukaryotes" as an ongoing process as the "knowledge of gene and protein roles in cells is accumulating and changing" [14]. Since the vocabulary is quite large, the authors have separated it into three ontologies, namely biological process, molecular function, and cellular component [14]. For our system, we have utilized an ontology that includes the vocabulary of Protein Kinases. Its name is ProKinO [15, 53] which we have discussed in detail later.

2.3 Natural Language Processing

Natural language processing in the simplest terms means enabling computers to derive meaning from human or natural language input [16]. NLP researchers aim to gather knowledge on how humans understand and use language so that appropriate tools and techniques can be developed to make computer systems understand and manipulate natural languages to perform the desired tasks [4]. An NLP system may begin at the word level, determining its parts of speech or morphological structure, and then might move on to the sentence level to determine its grammar, and then to the context of the overall environment [4]. NLP consists of many components. Each component differs from the other, based on the problem statement, specific corpora for evaluating the component and some metrics to evaluate the overall system. Natural language understanding [4, 16] converts chunks of text into first order logic, which becomes easier for the computers to understand and handle. Optical character recognition [16] analyzes images and extracts corresponding text from it. As discussed earlier, question answering is another NLP task that aims to answer users' specific questions [5]. Depending on the questions there can be closed domain answers, specific to a topic, or open domain answers which can be as broad as encompassing the entire knowledge base. Sentiment analysis and opinion mining [17] are other tasks that perform subjective analysis on a set of documents to retrieve opinions and trends of public opinion in the social media. Parts-of-speech tagging [5] is a task that assigns each word of the sentence a specific tag based on its parts of speech, namely noun, adjective, verb and so on. This is generally necessary in an NLP system where analysis starts from the word level to generate context of the sentence. Parsing [5, 16] is performed on PoS tags to generate parse trees for the sentence (sentence-level analysis) with, each parse tree having multiple grammar analyses. A specific grammar analysis conforms to a specific topic or need of the user. For this

reason, some level of grammatical dependency evaluation of the tree is necessary. Another methodology, although not used in NLP, called relationship extraction tries to find implicit and explicit semantic relationship between entities [18]. As mentioned earlier, which of these NLP tasks are to be used, depends on the problem statement and the corpus available.

2.4 ProKinO

ProKinO is the Protein Kinases Ontology that "serves as a shared vocabulary to leverage knowledge that can be used in various useful applications in the protein kinase domain" [15]. The specific research interest of protein kinases has existing data from many heterogeneous sources, but ProKinO exists as a solution to integrate this data, form a conceptualization in the form of an ontology, and represent the relationships between entities in the protein kinase domain [15]. Both the user and the computer take interest in these relationships between concepts; the computer is able to process the relationships as applications see fit, and the user can understand the literal meaning of relationships between two objects. The database resources used for gathering protein kinase knowledge include the interactive kinase database KinBase [19], the Catalogue of Somatic Mutations in Cancer (COSMIC) [20], the Protein Family Database (Pfam) [21], the Universal Protein Resource (UniProt) [22], and the Protein Data Bank (PDB) [23]. KinBase contains a classified hierarchy of kinases divided into groups, families, and subfamilies that help evaluate "kinase function and growth by comparison of related kinases" [15]. COSMIC is a database that hosts knowledge about somatic acquired mutations that relate to human cancers, and it contains information including publications, samples, and mutations [15]. Pfam is a database for conserved protein families that consists of "multiple sequence alignments and profile Hidden Markov Models (HMMs)" [15].

Pfam HMM is regarded as a great source for identifying domains within proteins, which helps to better understand the function of proteins [15]. Each Pfam HMM includes a protein family or domain [15]. UniProt is a "comprehensive catalogue of protein sequences and functional annotations" that aids scientists in analyzing proteins of interest by interconnecting and storing "information from voluminous and disparate sources" [15]. PDB is a source of "three-dimensional structures of macromolecular complexes of proteins, nucleic acids, and other biological molecules" that can help scientists understand the role of structures in human health, disease, and drug development.

First, protein kinase data is fetched from KinBase and ProKinO is populated with this data. The information in the ontology further becomes the basis for acquiring and parsing the data from other sources of knowledge [15, 53]. Mutation knowledge about protein kinases comes from COSMIC and provides information including "mutation location, mutation type, tissue type, cancer type, and literature reference" [15]. From PDB, information is acquired that consists of the PDB ID, three dimensional coordinates, and structure abstracts. UniProt provides knowledge about functional features of protein kinase genes, such as "modified residue, signal peptide, topological domain, cellular location, and tissue specificity" as well as database cross references [15]. Once all data is acquired, the knowledge is integrated into ProKinO through an automatic population process. This knowledge includes protein kinase genes and their species, their corresponding groups, families and subfamilies, synonyms, and chromosomal position [15].

This paper talks about an application that uses mutation and mutation impact knowledge from the ProKino ontology. Instead of having to sift and analyze full text research documents to gain information, the user can use knowledge from the ProKino to quickly get only the important

impact statements and their pronouncements from the document. Later on, the user can select which statements he wants via a curator.

2.5 Ontology Population using text mining and NLP

There can be two kinds of systems in relation to ontologies: Ontology-based and Ontology-driven. In the first method, some information in the ontology is used solely as lookup information to perform some task. As this paper will later describe, we use mutation information in the ProKino ontology to find mutation and impact patterns from full text. Ontology-driven systems make a detailed use of the ontology to drive or constrain some process [25]. Even though ontologies serve as a great resource for domain knowledge and concepts, the actual population of ontologies is a time consuming, error prone process. Methodologies developed in the fields of NLP and information extraction provide techniques for automating the enrichment of ontology from free-text documents [24].

For example, word sense disambiguation, co-reference resolution and discourse reasoning are some of the NLP tasks that may require an understanding of complex relationships between concepts [24]. For example if *President Obama* and *President of the United States* are two words that refer to the same entity in a document, then a co-reference relationship can be established between them in the corresponding ontology. One approach to facilitating the ontology population process is to use informatics tools to accelerate the interactions among domain experts and ontologists necessary to the ontology development process. An important recent development is the National Center for Biomedical Ontology's BioPotal. BioPortal enables the biomedical community to find, comment on, and contribute to biomedical ontologies, thereby

facilitating interactions among ontology users and developers to increase the value of the ontologies [26].

Linguistic rules describing the relationships between terms in the text can also be used to identify conceptual relationships within the ontology. The most common symbolic approach is to use lexico-syntactic pattern (LSP) matching (Hearst [27]). LSPs are surface relational markers that exist in a natural language. Another approach used is the statistical approach that uses large corpus of data and utilize different linguistic principles for statistical measurements to extract semantic information [24]. Statistical methods can be divided into clustering methods and machine learning methods. The first technique is based on some similarity measure whereas the second one attempts to treat the extraction process as a classification process [24].

There exist few issues with regards to such automatic or even semi-automatic methods of ontology enrichment which need to be addressed before the full potential of such systems can be reached. Even then, a few of these techniques have been experimented, because many of the features such as pattern matching used in these linguistic approaches are quite prevalent in biomedical documents. For example, some entities like ‘mutations’ have a specific pattern which can be easily retrieved using regular expressions. Another example is that biomedical literature is filled with compound nouns i.e. nouns enhanced by adding modifiers to the existing terms. Systems that work on extracting information based on these linguistic features can be designed and can be evaluated on a corpus of biomedical data. Also, the biomedical field has well-developed knowledge and lexical resources such as existing ontologies/terminologies, domain-specific corpora, and general dictionaries that are necessary for knowledge extraction [24]. However, as mentioned earlier, there are a few issues with regards to using these techniques to make the perfect automated system for ontology population and enrichment. It is an error- prone

process with the potential to corrupt an already established rich knowledge base. Hence, some level of human interaction is needed while populating these ontology. This is where the process of curation comes in.

2.6 Curation

Curation is the process of collection and review of data .Although text mining shows considerable promise as a tool for supporting the curation of biomedical text, there is little concrete evidence as to its effectiveness [28].

Curating biomedical data into knowledge bases is a laborious task requiring considerable expertise. The general consensus is that text mining and NLP methods can make this task easier and less time consuming. However, some of these methods suffer the limitation of applicability due to the requirement of manual acquisition and codification of lexical knowledge for each domain [24]. Some statistical methods cannot provide linguistic insight of their own. Hence the more practical approach is to involve some level of human expertise in the curation process. For example, certain relationships between entities in an ontology might be further refined based on suggestions given by a human curator. There can also arise another scenario in which current tools of information extraction may produce noisy results [29], but even these results might be useful from the perception of the human expert. Such systems should have the ability to rank these noisy results on the basis of their relevance.

The task of creating the perfect automated information extraction system for biomedical literature is something which will take a lot of years to achieve, given the current circumstances. However, a practical system combined with human involvement is something that can provide a similar level of effectiveness, is achievable at a low cost.

CHAPTER 3

RELATED WORK

3.1 Mutation Finder

There exists many text mining and information extraction systems, however, we will focus only on those that use ontologies or are related to the biomedical domain. Mutation Finder [30] is an open source, high performance extraction system that extractions mutation mentions from full text. The baseline extraction system is rule based, which refer to rules described in Horn et al. [31]. Mutation Finder builds on this system and uses approximately 700 regular expressions [32] to find mentions of point mutations from full text. The main improvement of this system over previous mutation extraction systems is the automation of pattern generation. This allows for less commonly used formats for mutations to be matched directly which consequently leads to improved recall over prior systems, and maintains high precision [31].

Another major advantage of this system is its usability. Mutation Finder has implementations in Python, Perl [33] and Java and can act as a standalone application or can be integrated in another system. In addition to this, a high quality gold standard data set has been made available. However, the objective of Mutation Finder is limited to extraction of only point mutations. The objective of our system is not only to extract mutations, but its impacts and try to analyze these impact sentences. Even though, Mutation Finder is fast and has a near perfect precision, there is always the chance of false positives as always is the case with biomedical extraction systems. One such example is mention of other entities, such as genes, proteins or cell lines, whose names look similar to mutation mentions. For example, MutationFinder would mistakenly extract the

gene name L23A and the cell line T98G, as mutation mentions [30]. Also the system does not use any knowledge base or ontology to find relationships between the extracted entities. Overall, Mutation Finder is an excellent tool for quick extraction of point mutations, and works well when integrated with another extraction system. We initially planned to use this tool for our system to find mutation mentions, but since the objective of our system is to find information for mutations present in ProKino only, it was not needed. However, Mutation Finder can be used in the future when the plan to expand the system to find all possible mutations is implemented.

3.2 Textpresso

Textpresso [34, 35] is a text mining system that utilizes ontologies. The Textpresso ontology is specific to the domain of interest, depending on the researcher utilizing the search engine. Textpresso builds its ontology from several databases, but Textpresso also has its own database because it has a curation feature that allows users to manually enter scientific data and literature into the Textpresso database. This feature ensures that "quality control of the data is done to the highest degree, based on human expertise." [34]

Textpresso's ontology is somewhat specialized in that it contains Perl [33] regular expressions that match different forms of words. For example, the regular expressions are designed to match forms of a word that contain "ing" or "ed" appended to the root form of the word. This feature is particularly useful because it will return results that contain at least the root forms of keywords. Textpresso indexes the sentences of all documents in a collection to be queried. In addition to indexing sentences, Textpresso tags each sentence and constructs an XML version of each sentence that can potentially be processed using NLP. However, Textpresso does not currently use the marked up XML form of the documents.

Textpresso's interface includes many useful features. The user can specify whether to match the constructed query to sentences or whether to match the query against an entire document. Matching a query against an entire document suggests a Boolean keyword search while matching a query against sentences within a document focuses more on concept searching. The user may also determine whether the document matching should include the abstract, title, body, or all sections of the documents. The list of documents matching the query is sorted according to the number of occurrences of matches in the documents such that the most relevant document will be on top of the list. When using Textpresso at first, forming the initial request involves a bit of a learning curve. The user must be very familiar with the domain represented by Textpresso (many different domains may be represented) such that terms and concepts of that domain must be somewhat known by the user before searching for documents containing those terms and concepts. First, the user has the option of searching for keywords by typing them into a search field; this is similar to the way a user interacts with a search engine. Obviously, searching in this manner requires the user to know what (s)he is searching for before executing the search query. Next, the user may choose from a list of categories and subcategories, and one of the categories contains relationships present in the ontology. Additionally, all categories from the GO ontology are available for searching [34]. Once chosen categories are added to the query, entities from the ontology belonging to those categories are included in the document search. While Textpresso gives the user flexibility beyond the simple keyword search and uses categories to include terms the user may not think of, the system lacks a user-friendly interface and request formulation process that is ideal for all levels of users. Rather, the system best suits those who are highly skilled in the domain. If a beginner to intermediate user plans to use the system, (s) he probably does not have the vocabulary set of a professor. After utilizing the category search, the result set

will contain many matches that are undesirable. Textpresso includes a filter for including or excluding certain terms, but students using the system may not be comfortable using the filter because it is almost like manually forming the query. On the other hand, an expert will be able to form a better query since (s) he has a wide vocabulary set and will know particular terms and phrases to include or exclude. Textpresso is a great idea, but its user interface and ease of use can be drastically improved.

3.3 Open Mutation Miner

Open Mutation Miner [36] is an ontology-based text mining system that extracts and analyzes mutation impact information from full text articles. It is ontology based because it validates the information found against an OWL-DL [10] ontology. The newly found information is then used to populate the same ontology for further processing.

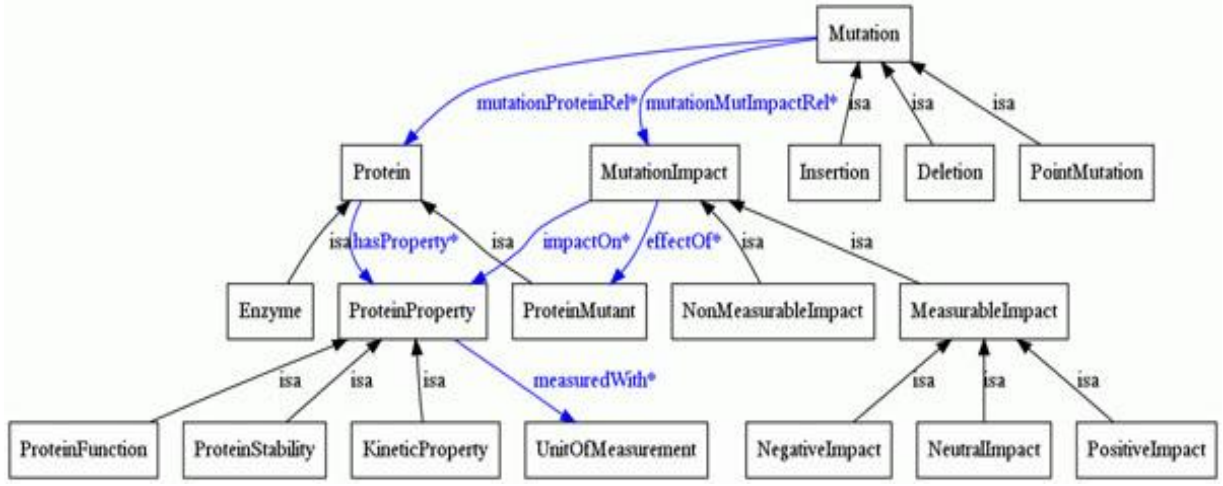


Figure 2. OMM impact ontology

As seen in Figure 2, the ontology is made up of several classes. Some of the important classes are: Mutation, which means an alteration to a gene [36], MutationImpact which denotes what

impact the mutation had and ProteinProperty, which is a class for protein properties. The main objective of this system is to find out mutation impacts and their effects on the protein properties and to what degree the effect is.

The system can be divided into three parts, namely the Mutation extraction component, Protein properties extraction component and the Impact extraction component [36]. The Mutation extraction component makes use of Mutation Finder [30] to extract point mutations. The protein properties are expressed in RDF format and detected through gazetteering [36]. The impact extraction component uses the OMM Impact ontology that segregates 130 impact words into three sub-classes namely, Positive, Negative and Neutral impacts.

Basically, the system selects sentences that consist of these three pieces of information and grounds the impacts to the specific protein property in addition to the degree of the impact. On some experimentation with the system, it was found that even though the system was efficient enough to find good information; it did not exactly address the main objective of our research problem. Our system is specific to only mutations and their impacts and not protein properties. Also for this level of focused text mining, we decided to use NLP for a better analysis of the impact sentences. Basically, OMM performs pattern matching for mutations and mutation impacts. One issue of not using NLP for a deeper analysis can be observed in the case, in which, a positive impact word is preceded by a negation. Consider the example '*T790M does not activate the kinase*'. In this case, OMM will accurately identify the mutation word *T790M* and the impact word *activate*. However, the system will annotate the mutation impact as a positive impact since it belongs to the Positive Impact class, when in reality it can be seen that the orientation of the statement is negative due to the word *not* preceding the impact word.

Another problem with OMM was that the extracted data was not curated by human experts; hence the chance of the ontology getting corrupted with false information is very high. As a result, we decided to develop a simple, albeit effective curation page so that experts can analyze and decide which information can be added to ProKino without corrupting it.

As compared to these three systems, the performance of MutaImpact Miner is better since it addresses every single issue of each system. As compared to the Mutation Finder, our system performs NLP on a sentence-level which provides a better understanding of the extracted information as a whole. The system has a better ease of use as compared to Textpresso and has a relatively user friendly interface. Also, since our system uses NLP, the annotations made for the extracted mutation impacts are more accurate than OMM and are better improved by the curation page that involves human involvement.

CHAPTER 4

MUTAIMPACT MINER

4.1 Motivation and Objectives

The primary objectives and motives of our system is to aid researchers and scientists extract mutation and mutation impact mentions from full text biomedical documents, so that this extracted impact knowledge can be added to the suitable knowledge base. For example UniProt [22] has some amount of information regarding some mutation impacts, followed by the paper references. To make that single observation, manual curators have to study multiple documents, understand the mutation impact information mentioned in these documents and summarize the information before making that observation. Obviously, this process is time consuming and expensive.

The solution to this problem is a system that inputs full text documents and produces mutation impact results for the experts or researchers to analyze and curate. Another advantage of such a system is that if it does not return any result, it can be concluded that the particular article had no significant information regarding mutation impacts and can be overlooked the next time. As mentioned earlier, as it is always the case with text extraction systems, there is a chance for false positives. In this case, the simple curator module allows the experts to get rid of the false data, thereby keeping the knowledge base enriched and non-corrupted.

In contrast to Open Mutation Miner [36], which is more general based and attempts to locate information regarding genes, mutations and protein properties, our system is specifically dedicated to mutations and impact information from ProKino. Furthermore, this system uses

NLP tasks for a deeper analysis of the results, which improves the system by filtering out unwanted results. The results can be further filtered out with the help of a human expert, who can select which information is important enough, on the system's simple curator page. Also, once the results are curated, they are added to a persistent storage to prevent addition of duplicate results in case they are encountered again.

In the system presented here, the user spends more time in studying and curating each result than the amount of time spent for the actual extraction of impact statements from full text documents. To aid the user , the system also provides an option of displaying eiter the full text of the paper or only the paragraph context with only the impact results highlighted, so that the user can view the sentences before or after the result, prior to making his decision whether to select that particular impact statement or not. Overall, the whole motivation behind this system is to help the user in studying only 'important' statements and making decisions, thus speeding up the whole process. Figure 3 shows the system components which are described in detail in the following subsections.

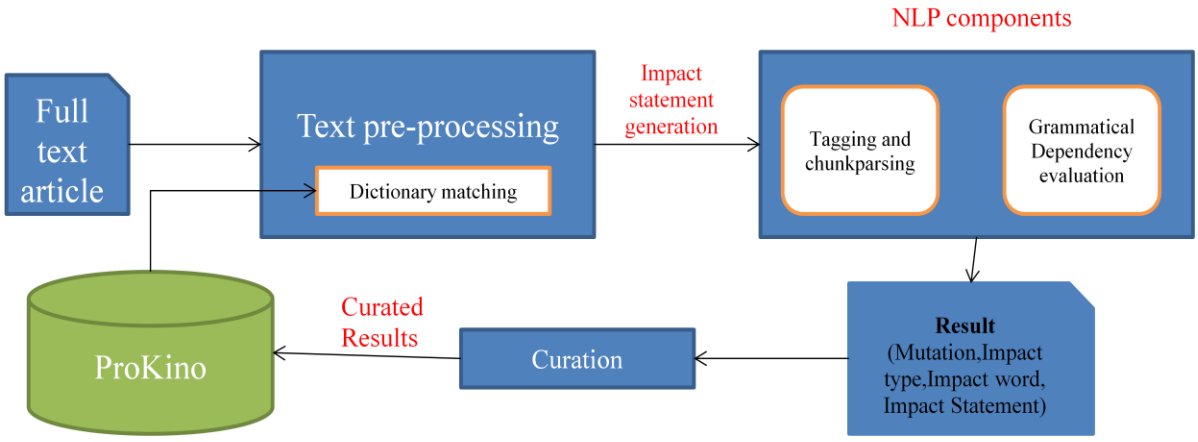


Figure 3. System components

4.2 Text pre-processing

In any information extraction system, pre-processing is an important part , since the characters, words and sentences identified at this stage , are passed on to the further stages of the system for processing to components like taggers and parsers. Pre-processing is essentially the act of converting raw text or a sequence of digital bits into a proper format of linguistically-meaningful pieces.

Documents in biomedical literature available via free access are often provided in Portable Document Format (PDF) [40]. As a result, our system is designed in such a way that it accepts a PDF document and converts it into a chunk of full text. Since PDF documents have their own different format of displaying content, there may arise problems once they are converted to text and directly forwarded to the NLP components of the system. One of the most common problems is the structure of the converted document which can be without any organization or format. Hence, some level of pre-processing is needed to clean up the structure and make it suitable to be passed on to further processing.

Speed is also an important issue as too much time should not be spent by the system in just pre-processing. Hence, the PDF-to-Text converter module that the system implements and which we have described in detail later in the implementation, performs conversion efficiently, including the formatting of the structure.

For the formatting, the system inputs the chunk of full text and uses a sentence boundary detector [37] to detect sentences and arrange them in a proper line-by-line structure for the entire text. Since the NLP components process the text, sentence-by-sentence, this structure best suits the needs for the system.

4.3 Impact Statements Generation

Our system depends on finding mutation mentions and their impacts from the text. Hence, before the NLP process actually takes place, it is better to filter out the noisy results and forward only interesting statements to the succeeding components. These sentences are referred to as impact statements. Basically, an impact statement is a line or pattern of text that contains a mutation reference as well as an impact word. For example, '***T790M*** *mutation* ***activates*** *the kinase*' is an impact statement, where *T790M* is the mutation and *activates* is the impact word. The system extracts only these kinds of statements in pre-processing and forwards it to the NLP components for further analysis.

This is where the ProKino ontology plays a role. Our system retrieves all mutations from the Mutation class in the ontology. As mentioned earlier, this system can be expanded to consider all mutations in the future and Mutation Finder [30] can be used to extract all mutation references from text for this purpose. In addition to mutations, the system also uses a dictionary of nearly 150 impact words compiled manually from biomedical literature. Creating this list initially was a laborious task as there is no existing dictionary of impact words readily available. However, we have added later as more papers were analyzed. The impact words have been divided into two classes namely, Positive and Negative Impact classes, which are akin to the Impact classes in the OMM impact ontology. Positive impacts can be classified as words having positive effects like "activates", "increases", "higher" and so on. Negative impacts are words such as "inhibits", "lowers", "abrogates", "reduces" etc. Initially, the system was tested with only these two classes in mind. On further experimentation we found out that there might arise a case in which the impact pronouncement is neutral i.e. in "no effect" for the system to decide. Hence, we added one more class termed as 'Neutral Impact'. The logic behind this was there can be some cases of

mutations that might not have a determined impact as positive or negative, but can still be classified as an impact statement. Some examples of neutral impacts can be “same”, “similar” and “apparent”. A few examples of these different classes are later recorded in the evaluation section.

The system performs basic dictionary matching on each sentence to check whether it contains both the mutation and the impact word, and adds it to a file of impact statements which is then forwarded to the NLP components.

4.4 Tagging and Parsing

The first NLP tasks that our system performs is tagging and parsing. Our system’s algorithm is based on determining the linguistic meaning of each word of the sentence and finding the grammatical dependencies between them. For this reason, the grammatical structure of the sentence needs to be determined.

Part-of-speech tagging is the task of identifying the parts of speech for each word of the sentence such as noun, adjective, adverb, verb, preposition and so on [4]. Consider the following example:

“*T790M mutation activates the kinase 5-fold as compared with the WT enzyme*.”

After passing this sentence through the tagger, the output is as follows:

T790M/CD mutation/NN activates/VBZ the/DT kinase/NN 5-fold/JJ as/IN compared/VBN with/IN the/DT WT/NNP enzyme/NN ./.

As seen, each word of the sentence is assigned its part-of-speech label. *T790M* is labeled by CD (numeral), *mutation* by NN (noun, singular), *activates* by VBZ (present tense verb) and so on. In most of the NLP systems that use tagging, the labels or POS tags are based on the University of Pennsylvania (Penn) Treebank Tag-set [38].

The output of the tagger [52] is then passed to the parser, which generates the parse tree for that sentence [4]. The parser that we use is a chunk parser [46]. Depending on the grammar of the sentence, the parse tree can be analyzed to find its linguistic meaning. It should be noted that one parse tree can have multiple possible analyses and it is important to identify which analysis suits our needs the most.For the example discussed above, following is the parse tree in Figure 4 which we will analyze in the next step.

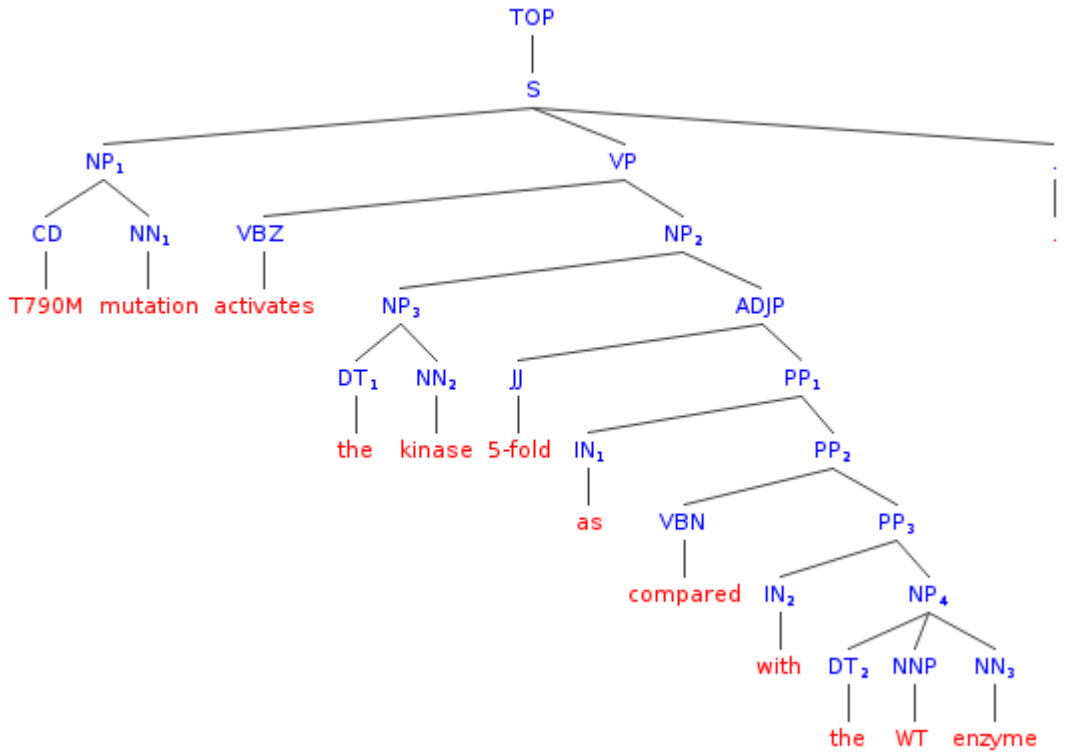


Figure 4. Parse tree example

4.5 Grammatical Dependency Evaluation

Since each parse tree may have multiple analyses, it is very important to determine which grammatical analysis of the parse tree best satisfies the condition for it to be classified as important information. The grammatical dependency evaluation algorithm performs this task. It

helps the system determine the grammatical relationships between different words of the impact sentence. In our interest, the concepts we are interested in are ‘mutations’ and ‘mutation impacts’. The evaluation algorithm thus, helps the system to find out whether there exists any relation between these two concepts. This process allows the system to filter out a lot of noise and come up with only a few interesting impact statements which the curator can then analyze further.

We have decided to use the Stanford NLP group core package [54], which can be used in any text extraction system. However, the only piece of interest from this package was the Stanford typed dependency API [41]. This API helps to analyze a parse tree by determining the relations between two words in a sentence. Basically, it identifies dependency between two words in the form of a triple: name of the relation, governor and dependent. The Stanford dependency manual [42] provides an in-depth look at each of the relations the system assigns to a triple. Many of the relations provide representations for identifying grammatical structures like abbreviations, exclamations, question-answer words and so on. We do not consider such relations. In our system, we are interested in studying only the following relations in Table 1.

Table 1. Dependencies

Name of relation (dependency)	**Description**
Nsubj (nominal subject)	noun phrase which is the syntactic subject of a clause
Agent(agent)	complement of a passive verb
Neg(negation modifier)	relation between a negation word and the word it modifies
Nn(noun compound modifier)	noun that serves to modify the head noun
Num(numeric modifier)	any number phrase that serves to modify the meaning of the noun
Dep (dependent)	General dependency showing words are connected in the parse tree

The algorithm for the evaluation module is straightforward and depends on three cases. By this stage, the system already knows what the mutation and impact word is in the impact statement, thanks to the lookups it matched to in the Impact Statement generation stage. The algorithm revolves around the concept of the impact word as the 'governor' of the statement.

In the first check, the system determines whether the impact word is preceded by a negation or not. This can be found out by the ***negation*** relation as described in the table. In the second check, the system tries to find out whether the mutation word is the direct subject of the impact word. This can be determined by the ***nominal subject*** relation. However, there can be a case of passive voice, albeit few and far between. This case has also been handled by the ***agent*** relation that has a similar meaning as of the ***subject*** relation, but in passive voice. In some cases, the mutation is not the direct subject of the impact, but is connected to the impact by some intermediate noun phrase. This is the third check that the system performs, to find the intermediate noun phrase, after which it is determined whether the intermediate word is connected to the mutation word in the tree. The connection between the intermediate word and the mutation word can be determined by the ***noun compound modifier***, the ***numeric modifier*** or simply the ***dependent*** relations, and if it exists, then the system can safely say that there exists an indirect relation between the impact word and the mutation word. This process can be better explained with the help of an example. Consider the following impact statement, "*T790M mutation does not activate WT EGFR*". The system identifies this statement as a negative impact statement and lists *T790M* as the mutation and *activate* as the impact word. Let us take a look at the graphical representation of this statement in Figure 5 with the point of view of the evaluation algorithm.

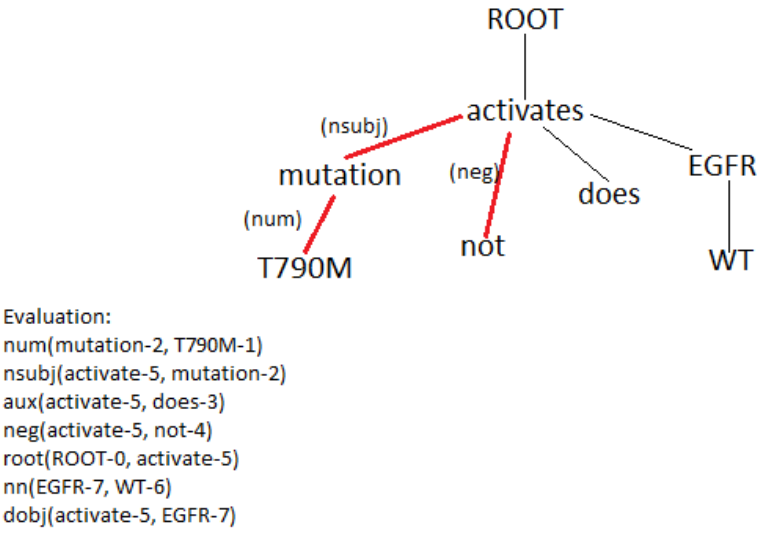


Figure 5. Graphical representation of impact statement (dependency evaluation algorithm)

The algorithm first checks whether the impact word *activate* is preceded by a negation, which it does and hence labels the impact as a negative impact, even if *activate* is a positive word. Secondly, the system tries to find out the subject of the impact word, which in this case is not directly the mutation mention *T790M*, but the intermediate word *mutation*. Lastly, the system checks whether *T790M* and *mutation* are connected and as seen in the example, they are represented by the relation of numeric modifier. Since *T790M* is indirectly related to the impact word *activate*, the system classifies this as an important impact statement which is then displayed on the curator page. On the curator page, if the human curator expert agrees with this result, the impact statement is added to the ontology.

The dependency evaluation algorithm takes into account unigrams or single tokens and finds dependencies between them. The system depends on this simple and straightforward algorithm for filtering out statements that are incomplete, grammatically incorrect and more importantly, do not pass the algorithm selection process. The algorithm selection process ensures that only

good quality impact statements are extracted whereas the rest are discarded. Since the text extraction process is susceptible to false positives, the problem of filtering out these false positives is handled by the curation process, thanks to the human expert involvement.

4.6 Curation

The system after applying the dependency evaluation algorithm, displays the results to the user on a simple curation page. The user is generally an expert in the biomedical domain who can understand the impact statements generated by the system. In addition to his/her expertise, the system also provides a way to view the entire content of the full text document along with the highlighted impact statements. Also, the system provides the user an option to download and view the original PDF in case (s) he wants to read it before curating the results.

Overall, the involvement of human expertise means the results can be carefully curated and added to ProKino. Also, every time a new PDF file is to be processed, a check is done to see whether the extracted impact statement is already present in our knowledge base so that duplicate results can be avoided. In this case, the statement is only displayed to the user, but not available for curation or selection. This process helps to maintain the integrity of ProKino.

Furthermore, the system provides the user different options of downloading the annotation files that consist of the statements and other information selected by the user. The annotation file consists of triples that contain information like mutation, impact type, impact statements, references etc. selected by the user. An example of such an annotation file will be discussed in chapter 5.

Overall, the main objective of the curation stage is to aid the user in selecting only important impact statements so that knowledge in the ontology is enriched and false data is avoided.

CHAPTER 5

IMPLEMENTATION

5.1 Introduction

MutaImpact Miner has been implemented as a Java [43] web application with several components. First, the application can input biomedical articles in the PDF format via two options. The first option is by entering the PubMed reference for the specific mutation that fetches the paper from the remote server. However, this option does not function every time due to a few issues. The first issue is that some of the papers are not available for open access and demand a certain cost. Hence it is difficult to obtain those PDFs for free. Another issue is that each of the science journals has its own variable framework of retrieving a PDF document from the website. Hence, there is no general way in which the system can handle this variability and obtain the paper from all journal websites. Hence, another option has been provided to manually upload the PDF to be processed from local disk. This is a relatively simple method, as it also allows the user to keep a record of the PDF on the local disk and the interface is easy enough for him/her to upload the PDF file.

Once the PDF file has been obtained, a Java class performs the initial task of conversion to text and formatting of the textual content. This process is fairly simple and takes about a second or less, depending on the number of pages and content. For PDF-to-text conversion, we decided to use a Java library provided by Snowtide Informatics [44] which we describe in detail in later sections. A sentence boundary detector [37] has been used to detect individual sentences so that

the chunk of converted text can be formatted in a specific sentence-wise structure, suitable for our NLP processing.

The Jena framework [45] has been used for loading the ProKino ontology into memory. These libraries are crucial for retrieving mutation information existing within the ontology. In addition to this, three other lookup dictionaries consisting of manually compiled positive, negative and neutral impact words are prepared and loaded into the system. Using basic pattern matching, an intermediate document consisting of impact statements is generated and input to the NLP components.

We have conducted many experiments to decide which tagger and parser to use. Speed and accuracy were obviously the two factors to be considered. The problem here is that most of the existing parsers are slow and not suitable for processing hundreds of documents. Hence, we decided to use the CFG chunk parser [46, 52] which is fairly fast as well as accurate as compared to the other existing parsers. Details of this parser are mentioned in the later sections. A separate Java module has been implemented to use this tagger and parser and forward each parse tree to the dependency evaluation module.

The dependency evaluation module uses of Stanford dependency Java API [41] that inputs one parse tree at a time and provides a dependency representation we earlier saw in Figure 5. Since our algorithm does not need to analyze each relation and triple and only needs to take care of 6 relations as mentioned in Table 1, implementing the algorithm was fairly easy as compared to designing it. Also, in biomedical literature, around 80-85% of impact statements have similar specific linguistic pattern, so it is relatively simpler to analyze them if similar patterns emerge.

Figure 6 shows how an impact result is displayed in the curator after the NLP processing takes place.

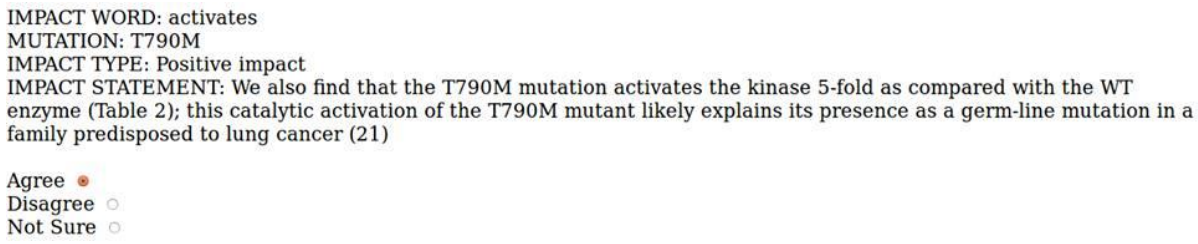


Figure 6. Example of an impact statement on curator page with pronouncement

As seen, every impact statement can be added to the knowledge base by simply selecting "Agree" and submitting the results. To help the user, the content of the full text can be viewed with only the impact statements highlighted as seen in Figure 7.

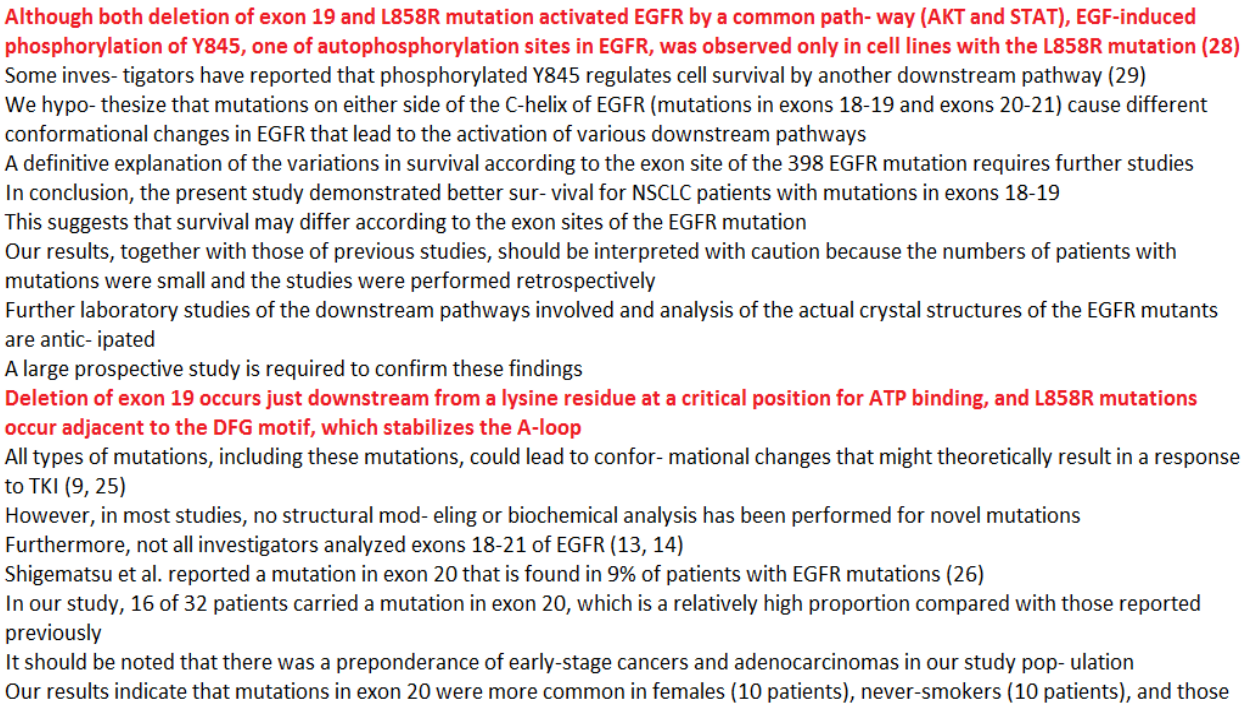


Figure 7. Content of full text document with impact statements highlighted

This helps the user to view the content preceding and succeeding the impact statement thus helping him make a decision whether to add the statement or not. In addition, once a statement has been added and if the same statement is encountered in some other PDF, the user is already informed about the added statement to prevent duplicates, thus saving time. One such example is shown in Figure 8. In addition to this, the user can also view the original PDF of the document while making a decision. Figure 9 shows the entire curator page with a popup window for the paragraph in context and another popup showing the original PDF.

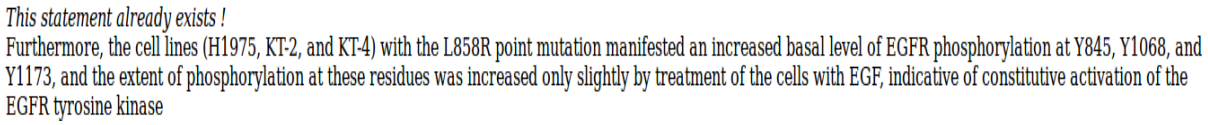


Figure 8. Duplicate statement encountered, user notified in curator phase

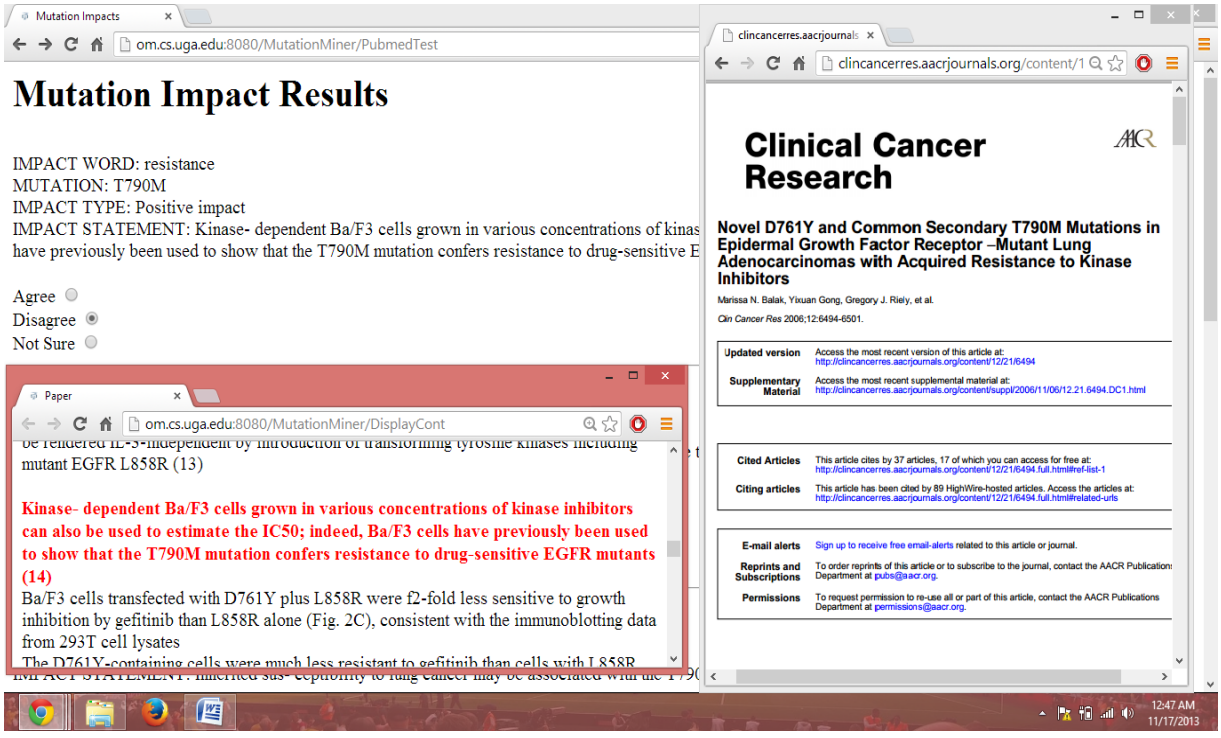


Figure 9. Curator page showing the available information

Once the results have been submitted, the system also gives the user an option to download an annotation file for every PDF in case (s) he wants to keep a separate record of it. The format of the annotation file and the information to be contained in it depends on the user. The system offers two default formats for the annotation file. Figure 10 shows one such default format of triples that consists of information like mutation, impact type, impact word, impact statements in addition to the PubMed reference.

"L858R" "increased|Positive impact" "Studies of cells expressing L858R revealed increased gefitinib sensitivity in vitro 4" <http://www.ncbi.nlm.nih.gov/pubmed/18021415>
"L858R" "activating|Positive impact" "One of the most common activating mutations, L858R in exon 21 was identified" <http://www.ncbi.nlm.nih.gov/pubmed/18021415>
"L858R" "hindrance|Negative impact" "L858R mutation results in steric hindrance of binding of gefitinib to the ATP kinase" <http://www.ncbi.nlm.nih.gov/pubmed/18021415>
"L858R" "resistance|Positive impact" "In addition, L858R has been implicated in drug resistance" <http://www.ncbi.nlm.nih.gov/pubmed/18021415>

Figure 10. Annotation file example

In some cases, the user may not need all the information but might select only some specific fragments of the extracted knowledge which (s) he wants to add to the knowledge base. In that case, the system also provides an option for the user to select only specific parts of the information. For example, the user can create an annotation file that consists of triples in the form of “Mutation” “Impact Word” “Impact Statement”.

The overall system needs to be accessible from a web browser and so a Java servlet container was created to parse requests and responses, to and from the web browser.

5.2 Architecture

The architecture of the system is described as a web application with the front-end accessible from a web browser**.** For the pre-processing and impact statement generation stage, both the ProKino ontology and the dictionaries should exist on the server. As stated earlier, the PDF documents can be retrieved either directly from the PubMed journal websites or via the upload from local disk functionality. The intermediate Impact Statement document is passed on to the NLP components of tagging, parsing and dependency evaluation, which should exist and work on the application server. Once the NLP processing is done, the results are sent in the HTML response to the browser. Only the selected results are added back to the knowledge base, whereas rest all is discarded in the curation stage. In addition to this, download functionality is also provided on the browser to obtain the selected results for the specific paper, which can be saved on disk.

The architecture of the system can be represented in two parts. The first part to some extent does the same work as Open Mutation Miner that is pattern matching and finding mutation mentions and impact words. The second part focuses more on accuracy, where the NLP components are applied. Figure 11 shows the pre-processing part and Figure 12 represents the NLP evaluation and curation process.

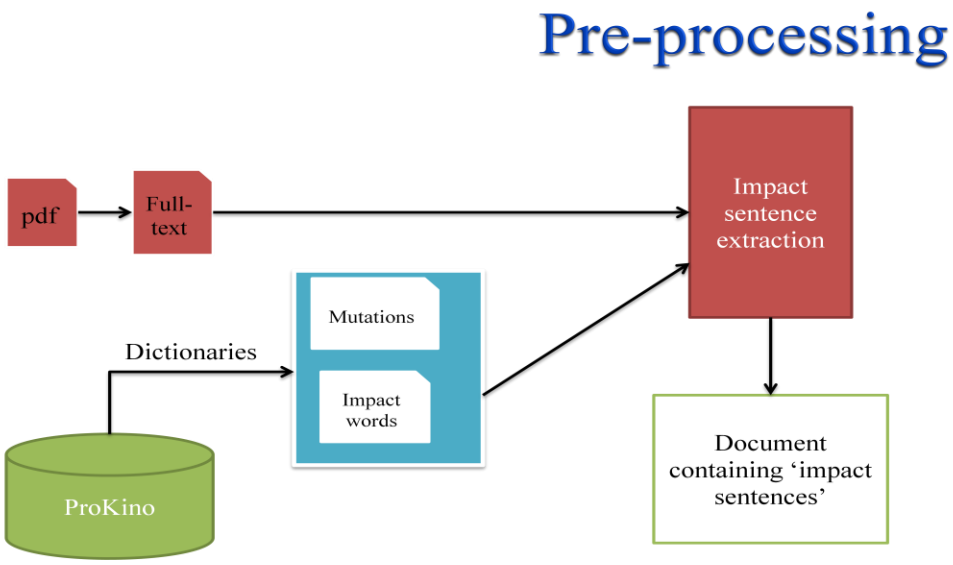


Figure 11. Pre-processing

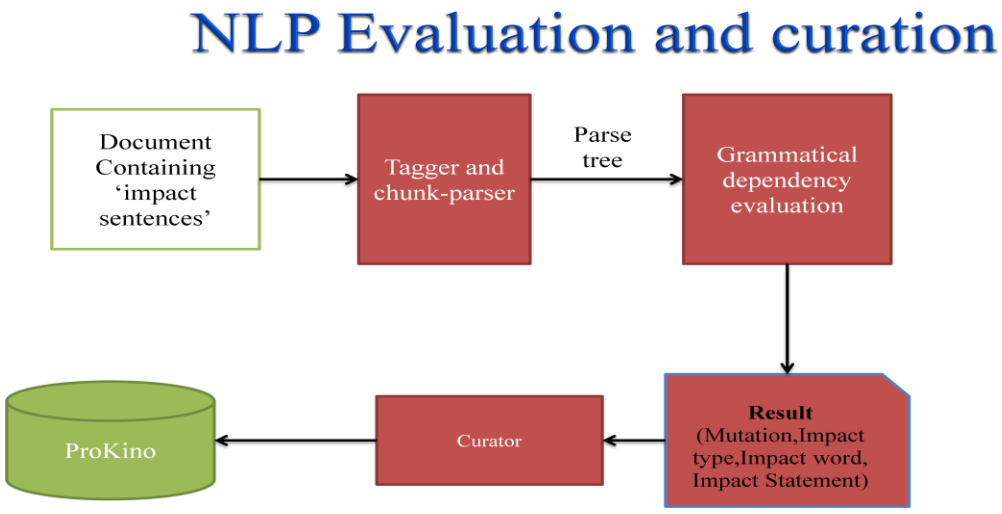


Figure 12. NLP evaluation and curation

5.3 Tools and technologies used

Jboss 7.0 [47] was used for deploying the application and Servlet 3.0 technology [50] was used for browser request and response functions. In the beginning, Apache Tomcat [48] was used, but it proved to be slightly more troublesome to use with the Servlet 3.0. In the end, JBoss was the selected. For the front end development, basic HTML and some functions from JQuery [49] were used.

The system uses a PDF to text converter provided by Snowtide Informatics [44] that is packaged as a Java API . The main reason for selecting this particular package was its speed as it is a fast converter available for PDF extraction. As of now it runs in a single thread, but a multi threaded option is available in a commercial version as well.

For formatting purposes, the system uses Apache's Open NLP sentence boundary detector [37]. The tagger [52] and the CFG parser used [46] have been implemented at the University of Tokyo, Japan for chunk parsing of textual content. The parser has an accuracy of around 85 % but can process around 71 sentences per second. The implementation is available as a command line tool for UNIX and Windows and can be integrated into the system by making process runtime calls.

For the dependency evaluation process, Stanford Core NLP's grammatical dependency API [42] has been used. Stanford NLP group has their own tagger, parser and other NLP components but they are relatively slow and do not meet the objectives of a focused text mining system as ours. Hence, we have decided to only use its dependency API and tweak it to implement the algorithm for selection of impact statements, as we have earlier described in Chapter 4. The results and the curation page display are again, developed using simple HTML.

CHAPTER 6

EXPERIMENTS AND EVALUATION

ProKino has data about over 71,000 kinase mutations. For our experiments, we decided to use the 20 most common mutations having PubMed references. Overall the experiments were performed on a set of 2591 full text PDF documents. the As mentioned earlier, each full text document from a reference is passed on to the pre-processing stage and the NLP component stage after which it provides results on the curation page.

Evaluating a system such as MutaImpact Miner is not easy. There is no easy way to gauge how the browser or a result set satisfies the user's objectives. Furthermore, there is no easy way to form a set of objectives for the system to satisfy because doing so would produce very subjective results. In addition to this, how every user analyses the impact statements and deems the information important depends on his/her expertise and needs. The actual speed of processing is not an issue and does not give much of a measure, since every PDF document takes around 1 second or less to extract the list of interesting impact statements. Furthermore, the speed of the curation stage is more dependent on how much time the user takes to study and select the sentences (s)he needs. First, the f-score of the parser used is 85 % but high accuracy is not the main objective. We decided to trade some level of accuracy for speed and it worked well, since the parser can parse around 71 sentences per second.

Furthermore, we decided to evaluate the system based on the experiments performed. An offline processing of these 2591 full text documents produced interesting results which gave us an idea about our case study. The results of the experiments are shown in Table 2.

Table 2. Evaluation Statistics

Statistic	**Result**
Total number of statements	12703
Number of impact statements selected by the system	8750
Number of positive impact statements	6479 (74.04%)
Number of negative impact statements	1091 (12.46%)
Number of neutral impact statements	103 (1.22%)
Number of statements with no outcome	1075 (12.28%)

From the statistics, it can be seen that the NLP component discarded around 31.11 % of statements. It should be noted that the NLP components discard incomplete or grammatically incorrect statements. More importantly, the system also discards statements that do not satisfy the dependency evaluation algorithm. From the remaining 8750 statements left, it can be seen that positive impact statements have the largest percentage amongst all other pronouncements. This can be attributed to the fact that majority of the results from the literature reviewed, includes positive pronouncement about mutation impacts, especially with words such as “activates”, “increases” , “activating” and so on. A low percentage of neutral impact statements were to be expected for the fact that there were hardly any important neutral impact words that can be matched against the literature. The 12% of statements are those whose outcome cannot be determined. There can arise some cases in literature where there is a mutation mention and occurrence of some impact, but due to factors such as lack of clear linguistic meaning or context, the system cannot determine the pronouncement.

Next, we involved an expert from the UGA Institute of Bioinformatics to evaluate the system. For this purpose, a random sample of around 150 papers (5.7 % of the total dataset) was selected that consisted of information on the 20 most common mutations. The results obtained by the

expert were compared with the overall results of the system for these specific 150 papers and are shown in Table 3.

Table 3. Human evaluator's evaluation statistics

Statement Type	**Number**	**Total**	**Accuracy (%)**
Positive impact	138	186	74.20
Negative impact	20	25	80
Neutral impact	2	3	67
No outcome ("Not sure" option)	56	70	80
Total	**216**	**284**	**76.05**

The accuracy of 76 % can be attributed to the fact that the accuracy for positive statements is lower than expected. We believe that this situation occurred due to two reasons. The first reason has been explained earlier, i.e. accuracy of our parser is not perfect (85%) and we are trading accuracy for speed. The second reason is that there were some impact words which do not exactly convey any important information from the point of view of biomedicine. We decided to eliminate such words ("favorable", "identical", "detectable", "drops", etc.) from our impact word dictionary. This experiment when performed again resulted in 213 sentences to be selected out of 267 which increased the overall accuracy to 79.78 %. The results are shown in Table 4.

Table 4. Improved evaluation statistics

Statement Type	**Number**	**Total**	**Accuracy (%)**
Positive impact	142	177	80.22
Negative impact	18	22	81.81
Neutral impact	2	3	67
No outcome ("Not sure" option)	51	65	78.46
Total	**213**	**267**	**79.78**

Furthermore, we decided to evaluate the system based on the quality of statements extracted. In this section, we will discuss how the system accurately analyzed three different statements selected from literature. In Figure 5, we had already presented one example of how the system evaluates a statement and we will use the same terminology for the upcoming examples.

For our first example, let us consider the statement *"The structural modeling suggests that T790M can abrogate the binding of gefitinib"*. As we discussed earlier, this sentence after passing through the pre-processing stage, goes to the tagging and parsing stage, the output of which is as seen in Figure 13.

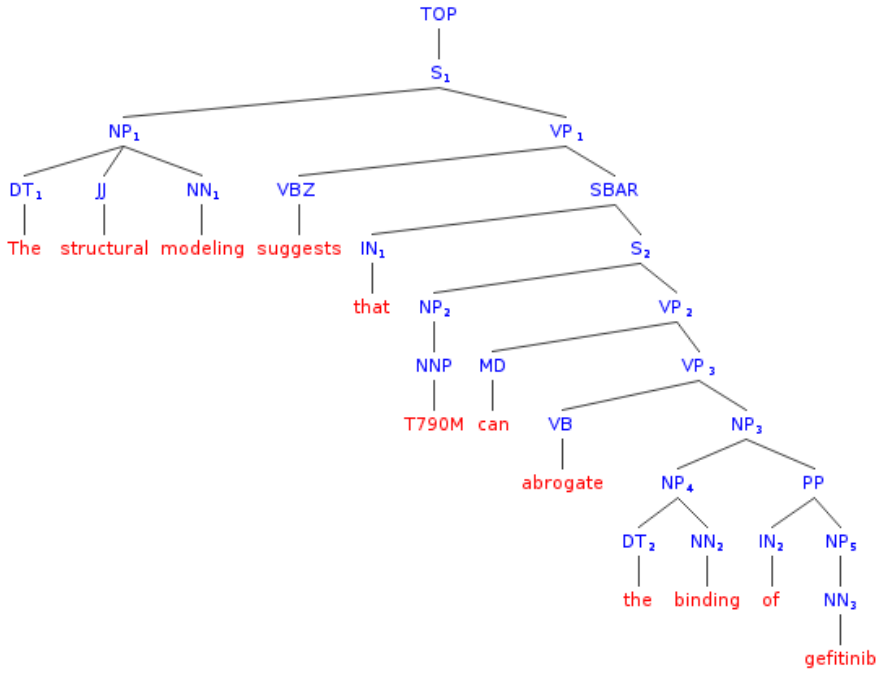
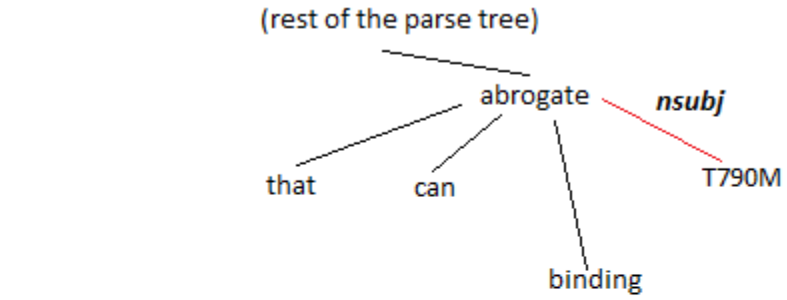


Figure 13. Parse Tree for example 1

The system passes this parse tree to the dependency evaluation algorithm that will evaluate it and find grammatical dependencies between tokens. It will first find the "governor" of this statement, which in this case is the impact word *abrogate* and try to find out if it has any interesting relation

with our mutation in question *T790M*. Figure 14 shows the analysis of a part of this tree under interest.



Evaluation:

mark(abrogate-8, that-5)
nsubj(abrogate-8, T790M-6)
aux(abrogate-8, can-7)
dobj(abrogate-8, binding-10)

Figure 14. Graphical representation of example 1 in dependency evaluation stage

As seen, the system finds out that the mutation *T790M* is the direct subject of impact word *abrogate* denoted by relation ***nsubj***. *abrogate* is classified as a negative impact word in our dictionary and hence the system accurately outputs this statement as an impact statement with a negative impact pronouncement, along with the specific mutation *T790M* and the impact word *abrogate*.

Generally, a majority of our results are composed of straightforward statements such as this and hence the system can evaluate them accurately. However, there can be rare cases in which the statement might be depicted in a passive voice, but our algorithm has the ability to evaluate them as well. Consider our next example, *"The phosporylation is increased by E17K mutation"*. The parse tree of this statement is as shown in Figure 15.

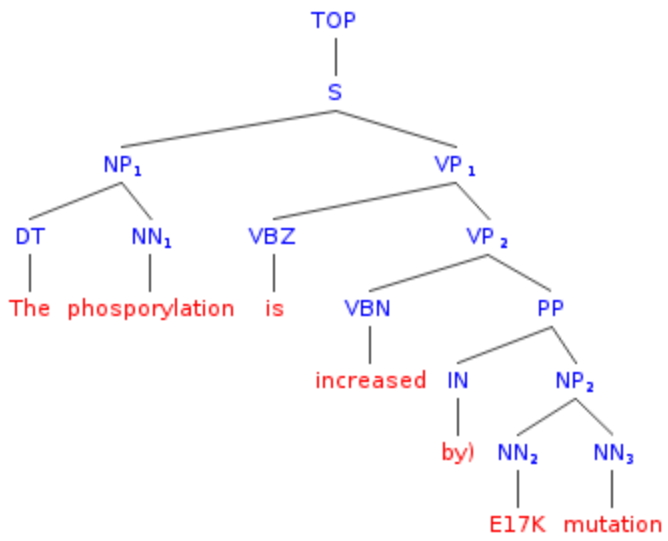
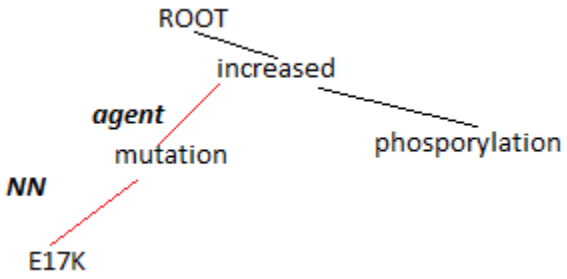


Figure 15. Parse tree for example 2.

Our algorithm finds the governor of this statement which in this case is *increased* and attempts to find a suitable relation to our mutation *E17K* as seen in Figure 16.



Evaluation:
nsubjpass(increased-4, phosporylation-2)
root(ROOT-0, increased-4)
nn(mutation-7, E17K-6)
agent(increased-4, mutation-7)

Figure 16. Graphical representation of example 2 in dependency evaluation stage

The algorithm detects that even if the mutation is not a direct subject of the impact word, it still has a relation identified by ***agent***. This particular relation is used to find the complement of a verb, exclusively in the passive voice. The word *mutation* is the agent of *increased* and since there is a relation ***nn*** (noun compound modifier), between *mutation* and *E17K*, we can say that there is an indirect relation between the impact word and the mutation in the parse tree. The system recognizes this and outputs this statement as a positive impact statement with *E17K* as the mutation and *increased* as a positive impact word.

For our third example, consider the case in which the statement cannot be classified as an impact statement and is disregarded by the system. Consider example 3 *"We have found some occurrence of T790M in patients"*. The parse tree of this statement is represented in Figure 17.

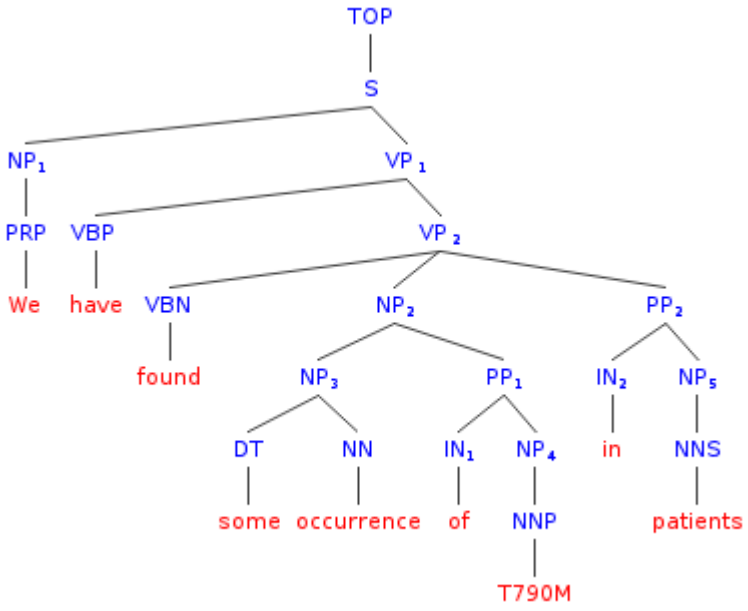


Figure 17. Parse tree for example 3

First, there is no impact word in this statement, and hence the statement will be discarded at the first instance itself. In addition to this, the algorithm cannot determine any specific grammatical

dependency with respect to our mutation *T790M* or the governor *occurrence* and hence it does not select this statement as an impact statement. Overall, the main focus of the system is on speed, but the accuracy has been addressed due to the NLP component of the system. Furthermore, the integrity of ProKino is also maintained thanks to the curation stage.

CHAPTER 7

CONCLUSIONS AND FUTURE WORK

We have designed and implemented a prototype of a new system that provides users the ability to input full text biomedical documents, extract mutations and mutation impacts, and uses this knowledge to enrich an ontology. The novel idea in this process is the use of Natural Language Processing tasks that has converted a basic pattern matching module into a more efficient system that filters out low quality statements. The general consensus is that most of the important information of a paper belongs in its Abstract and Conclusion pages. However, our experimentations showed that there is a lot of information that can be processed from the full text of the article as well. In fact, the experiments we have performed show that many of the impact statements extracted were placed in the body of the full text of the document and not just the abstract and conclusion parts. Even though, retrieving full text PDF documents can be bit difficult in some cases, the system performs very well in doing so and even provides an upload from local disk functionality, in case the user wants to work offline and keep a record of the documents (s) he is analyzing.

In addition to this, as is always the need with information extraction systems, the curation stage involves human expertise and supervision to verify automatic findings. No longer does the user have to spend a long amount of time to study every document to find important information. The system will aid the user to mine out only the relevant text in a short amount of time and assist him/her in analyzing them.

There exists few other text mining systems in the field of biomedicine, although they have more of a broader perspective and hence NLP tasks are not needed for deeper analysis. The main motivation behind our MutaImpact Miner is focused text mining of mutation impacts and hence an NLP-driven system was our necessity.

Also, we selected the biomedicine domain as a case study since literature in biomedicine consists of standard specific patterns, which are similar and are easy to extract using pattern matching. In addition, the problem of time-consuming manual curation of biomedical text is common, which directly impacts the cost of the whole project. Our system ensures that the researchers focus more on the actual analysis of important information in the text, instead of trying to read the whole paper in order to locate pertinent fragments. Our user interface is also simple and offers clear instructions on how to use the system. The main objective here is that the user spends as much time as possible on the actual analysis of the impact statements, which can be done on a very user friendly curation page. Finally, the evaluation of the system shows that it achieves the objective of fast knowledge extraction from biomedical scientific articles. In addition to this, the improved accuracy as shown in Chapter 6 makes it an efficient system capable of extracting good quality impact statements. Finally, the curation page aids the researchers in selecting impact statements that are only suitable for ProKino.

Many improvements can be made to this system in the future. As of now, the system works on one article at a time. However in the future, it may be possible to not only retrieve the article of interest, but also other articles that have cited it in PubMed and evaluate them too.

Improvements to the curation stage can also be made by making it available to the cancer research community online. Collective problem solving via crowdsourcing [51] in the oncology community is something that the system can play a part in by allowing all interested parties to

jump in give their opinions on the impact statements, which can then be ranked based on some ranking or voting system.

Also, since this system uses basics of NLP components, it can be expanded to function not only in biomedicine but also in other domains of research. This new text mining and curation system opens up many opportunities for applications and research in the future.

REFERENCES

1. Sholom M. Weiss, N.I., Tong Zhang, Fred J. Damerau, *Text Mining: Predictive Methods for Analyzing Unstructured Information*. 2005, New York, NY: Springer. 237.
2. W3C. *Extensible Markup Language (XML).* 2011; Available from: http://www.w3.org/XML/.
3. Robert Dale, H.M., H.L. Somers, *Handbook of Natural Language Processing*. 2000, New York, NY: Marcel Dekker. 943.
4. Chowdhury, G.G., *Natural language processing*. Annual Review of Science and Technology. Vol. 37. 2003.
5. Pierre Zweigenbaum, D.D.-F., Hong Yu, Kevin B. Cohen, *Frontiers of biomedical text mining: current progress*. Briefings in Bioinformatics, 2007. 8(5): p. 358-375.
6. Gruber, T. *What is an Ontology?* 08/14/2010; Available from: http://www-ksl.stanford.edu/kst/what-is-an-ontology.html
7. S. Staab and R. Studer. *Handbook on Ontologies*. 2nd edition, Springer, 2009.
8. Abraham Silberschatz, A., Henry F. Korth, and S. Sudarshan. *Ontology (information science)*. JSON 195 XML 204 Web Ontology Language 216 Resource Description Framework 226, 16.
9. *W3C. RDF/XML Syntax Specification*. 2004; Available from: http://www.w3.org/TR/REC-rdf-syntax/.

10. *W3C. OWL Web Ontology Language Overview*. 2004; Available from: http://www.w3.org/TR/2004/REC-owl-features-20040210/.
11. Berners-Lee, Tim, James Hendler, and Ora Lassila. *The semantic web*. Scientific American 284.5 (2001): 28-37.
12. Smith B, A.M., Rosse C, Bard C, Bug W, Ceusters W, Goldberg L J, Eilbeck K, Ireland A, Mungall C J, The OBI Consortium, Leontis N, Rocca-Serra P, Ruttenberg A, Sansone S-A, Scheuermann R H, Shah N, Whetzel P L and Lewis S, *The OBO Foundry: coordinated evolution of ontologies to support biomedical data integration*. Nature Biotechnology, 2007. 25: p. 1251-1255.
13. Noy, N.F., Shah, N.H., Whetzel, P.L., Dai, B., Dorf, M., Griffith, N., Jonquet, C., Rubin, D.L., Storey, M.A., Chute, C.G., Musen, M.A., *BioPortal: ontologies and integrated data resources at the click of a mouse*. Nucleic Acids Research, 2009. 37.
14. Michael Ashburner, C.A.B., Judith A. Blake, David Botstein, Heather Butler, J. Michael Cherry, Allan P.Davis, Kara Dolinski, Selina S. Dwight, Janan T. Eppig, Midori A. Harris, David P. Hill, Laurie Issel-Tarver, Andrew Kasarskis, Suzanna Lewis, John C. Matese, Joel E. Richardson, Martin Ringwald, Gerald M. Rubin, Gavin Sherlock, *Gene Ontology: tool for the unification of biology*. Nat Genet, 2000.
15. Gosal, G., *ProKinO: Design and Development of Ontology on Protein Kinases*, in Computer Science Thesis. 2010, University of Georgia: Athens, GA.
16. Allen, James F. *Natural language processing*. (2003): 1218-1222.
17. Pang, Bo, and Lillian Lee. *Opinion mining and sentiment analysis*. Foundations and trends in information retrieval 2.1-2 (2008): 1-135.

18. Ramakrishnan, Cartic, Krys J. Kochut, and Amit P. Sheth. *A framework for schema-driven relationship discovery from unstructured text.* The Semantic Web-ISWC 2006. Springer Berlin Heidelberg, 2006. 583-596.

19. *The Kinase Database at Sugen/Salk*. Available from: http://kinase.com/kinbase/.

20. *Catalogue of Somatic Mutations in Cancer (COSMIC).* Available from: http://www.sanger.ac.uk/genetics/CGP/cosmic/.

21. *Protein Families Database (Pfam).* Available from: http://pfam.sanger.ac.uk/.

22. *The Universal Protein Resource (UniProt).* Available from: http://www.uniprot.org/.

23. *RCSB Protein Data Bank: An Information Portal to Biological Macromolecular Structures*. Available from: http://www.pdb.org/pdb/home/home.do.

24. Liu K, Hogan WR, Crowley RS. *Natural language processing methods and systems for biomedical ontology learning*. J Biomed Inform 44.1 (2011): 163-179.

25. Sophia Ananiadou, J.M., *Text Mining for Biology and Biomedicine*. 2006, Massachusetts: Artech House, Inc. 286.

26. N.F. Noy, N.H. Shah, P.L. Whetzel, B. Dai, M. Dorf, N. Griffith et al. *BioPortal: ontologies and integrated data resources at the click of a mouse Nucleic Acids* Res, 37 (2009), pp. W170–173

27. Hearst MA. *Automatic acquisition of hyponyms from large text corpora.* In: Proceedings of the 12th Conference on Computational Linguistics, 1992.

28. Alex, Beatrice, et al. *Assisted Curation: Does Text Mining Really Help?*. Pacific Symposium on Biocomputing. Vol. 13. 2008.

29. Rodriguez-Esteban R, Iossifov I, Rzhetsky A. *Imitating manual curation of text-mined facts in biomedicine*. PLoS Comput Biol 2006: e118.

30. Caporaso, J. Gregory, et al. *MutationFinder: a high-performance system for extracting point mutation mentions from text*. Bioinformatics 23.14 (2007): 1862-1865.

31. Horn F, et al. *Automated extraction of mutation data from the literature: application of MuteXt to G protein-coupled receptors and nuclear hormone receptors*, Bioinformatics, 2004.

32. Hopcroft, John E., Rajeev Motwani, and Jeffrey D. Ullman. *Automata theory, languages, and computation*. International Edition 24 (2006).

33. Wall, L. *Perl*. 1987; Available from: http://www.perl.org/.

34. Hans-Michael Muller, E.E.K., Paul W. Sternberg, *Textpresso: An Ontology-Based Information Retrieval and Extraction System for Biological Literature*. PLoS Biology, 2004. 2(11).

35. Hans-Michael Muller, A.R., Tracy K. Teal, Paul W. Sternberg, *Textpresso for Neuroscience: Searching the Full Text of Thousands of Neuroscience Research Papers*. Neuroinformatics. 6(3): p. 10.

36. N. Naderi, R. Witte, *Automated extraction and semantic analysis of mutation impacts from the biomedical literature.* BMC Genomics, 13.Suppl 4 (2012): S10.

37. *Apache OpenNLP*. Available at: http://opennlp.apache.org/documentation/1.5.3/manual/opennlp.html

38. *Penn tagset*, University of Pennsylvania Available at: http://www.americannationalcorpus.org/OANC/penn.html

39. Hopcroft, John E., Rajeev Motwani, and Jeffrey D. Ullman. *Automata theory, languages, and computation*. International Edition 24 (2006).

40. *Adobe Portable Document Format. 6* ed. 2006: Adobe Systems Incorporated.

41. Marie-Catherine de Marneffe, Christopher D. Manning, *The Stanford typed dependencies representation*, Coling 2008: Proceedings of the workshop on Cross-Framework and Cross-Domain Parser Evaluation, p.1-8, August 23-23, 2008, Manchester, United Kingdom

42. *Stanford typed dependency manual*. Available from: http://nlp.stanford.edu/software/stanford-dependencies.shtml

43. Oracle. Java Platform, Standard Edition 7: API Specification. 2011; Available from: http://download.oracle.com/javase/7/docs/api/.

44. *PDFTextStream* Available at: http://snowtide.com/#features

45. *Jena*. Jena - A Semantic Web Framework for Java. 2009; Available from: http://jena.sourceforge.net/.

46. Yoshimasa Tsuruoka and Jun'ichi Tsujii, *Chunk Parsing Revisited*. In Proceedings of the 9th International Workshop on Parsing Technologies (IWPT 2005), pp. 133-140.

47. *JBoss*. Available from: http://www.jboss.org.

48. *Apache Tomcat.* 2011; Available from: http://tomcat.apache.org/.

49. *jQuery*. 2010; Available from: http://jquery.com/.

50. *Servlet 3*.0; Available from: http://docs.jboss.org/resteasy/docs/1.0.0.GA/userguide/html/Asynchronous_HTTP_Request_Processing.html

51. Daren C. Brabham, *Crowdsourcing as a Model for Problem Solving : An Introduction and Cases* University of Utah, Convergence: The International Journal of Research into New Media Technologies 14.1 (2008): 75-90.

52. Tsuruoka Y, Tsujii Ji, *Bidirectional inference with the easiest-first strategy for tagging sequence data.* Proceedings of the conference on human language technology and empirical methods in natural language processing; 2005: Association for Computational Linguistics.

53. Gosal, Gurinder, Krys J. Kochut, and Natarajan Kannan. *ProKinO: An Ontology for Integrative Analysis of Protein Kinases in Cancer.* PloS one 6.12 (2011): e28782.

54. Stanford Natural Language Processing Group. *Stanford CoreNLP Tools*. Available at: http://nlp.stanford.edu/software/corenlp.shtml