ABSTRACT

Title of Document: AUTOMATED KEYWORD EXTRACTION FROM BIO-MEDICAL LITERATURE WITH CONCENTRATION ON ANTIBIOTIC RESISTANCE

Maya Zuhl, MS, 2009

Directed By: Dr. Mihai Pop, Computer Science

The explosive growth of bio-medical literature makes it increasingly difficult and time consuming to keep up with newly discovered and published information. The extraction of knowledge from papers is critical in enabling computational analysis of biological data. In the last decade, tremendous effort has been put into development of automated and semi-automated tools for knowledge discovery and extraction from text, as an alternative to monotonous and time-consuming manual processing. This thesis research was focused on determining whether minor human supervision can improve the process of automated bio-medical text annotation. One of the main outcomes of this study is a tool that requires minimal effort and time from scientists to reach high precision in semi-automated annotation. The task we targeted is the extraction of keywords related to antibiotic resistance in bacteria. The tool is based on a machine learning algorithm that is retrained several times to achieve the best accuracy.

AUTOMATED KEYWORD EXTRACTION FROM BIO-MEDICAL LITERATURE WITH CONCENTRATION ON ANTIBIOTIC RESISTANCE.

By

Maya Zuhl

Thesis submitted to the Faculty of the Graduate School of the
University of Maryland, College Park, in partial fulfillment
of the requirements for the degree of
Master of Science
2009

Advisory Committee:
Dr. Mihai Pop, Chair
Dr. Louiqa Raschid
Dr. Vibha Sazawal

Dedication

This work is dedicated to my loving family: my most supportive parents Michael and Irina and my darling caring husband Dennis.

Acknowledgements

This thesis research could never have been accomplished without the kind support and valuable contribution of many people.
First of all, I would like to thank my academic advisor Dr. Mihai Pop. He guided me through this research without pressure but providing the needed support. He was very open to discuss every idea and was always there to help me resolve any issue I had. His belief in my capabilities and kind encouragement has been a great reinforcement. It has been a privilege to work with Dr. Pop.
Second, I would like to express my appreciation to the members of my thesis committee Dr. Louiqa Raschid and Dr. Vibha Sazawal. Their interest in my research and valuable input has been of great help in my research.
Third, I am very grateful to Dr. John Moult and his entire research group for their moral support, interest in my research, different point of view on this study, and valuable feedback. My special thanks to Dr. Lipika Ray; our conversations have been of great comfort and assistance.
Finally, I would like to thank my family, whose support and encouragement has been of utmost importance for me. I would not have been able to finish the research and writing without the constant faith and unconditional love my husband Dennis Shats, parents Dr. Michael and Irina Zuhl, and sister Jennifer provided.

Table of Contents

List of Tables

List of Figures

Chapter 1: Introduction and Background

1.1 Motivation

The last few decades have witnessed an exponential growth of available information in various fields. One of the areas that particularly flourished due to technological advancements is biology (molecular biology). The ability to sequence DNA/RNA quickly and reasonably cheaply has allowed biological scientists to generate large amounts of information in many different research areas. Such data are available for further research in the form of online databases and articles/papers. Every year, millions of articles are published and biological databases grow at an exponential rate, as seen, for example, in Figure 1.

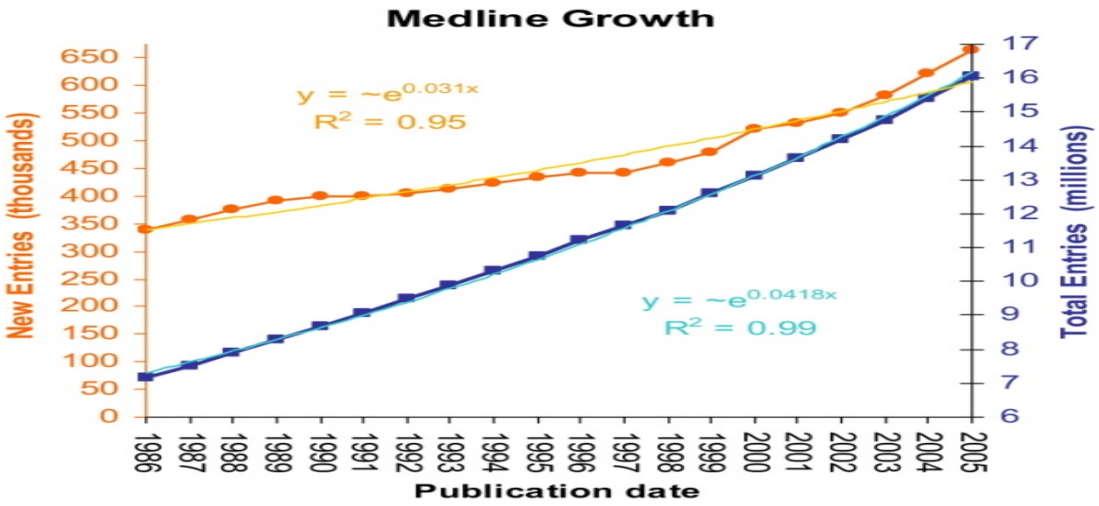

Figure 1: Medline Growth Rate (Cohen, 2006)

Due to the aforementioned growth in published information, it has become very difficult for scientists to follow all the new research and discoveries.

A researcher would need to read hundreds and maybe even thousands of papers if he or she wants to have all the recent information on a specific topic. Thus, numerous publicly available databases have been established providing the information from published literature in centralized and easier to understand format (Z. Lacroix, 2004). However, these databases are not the complete answer.

First, not all the data from published papers is aggregated into specific databases. Some of the information still remains hidden in a "paper-bound" form within published articles.

Second, many of the specialized databases are manually curated. In other words, some scientists who are experts in the field still need to sit and read large amounts of literature and manually insert the relevant information into the database. This process is expensive and time consuming, highlighting the need to develop new tools which will eliminate or speed up manual curation (L. Hirschman, 2002).

Two main research areas are concerned with development of such tools: Natural Language Processing (NLP) and Data Mining. One of the main objectives of NLP is to convert readable human language into a formal representation that is easier to handle within computer programs. NLP is used as the first step in automated data mining from published texts. Natural Language Processing is outside of the scope of this research, even though we use NLP techniques as a preliminary step to prepare text documents for further processing. The next section describes Data Mining and how it is used to retrieve information from text.

1.2 Data Mining

The general objective of data mining is retrieving useful, non-trivial information from the abundance of published data available in various forms, including published literature, databases, or various data streams (such as video or sensor data). Data mining software relies on the application of sophisticated algorithms and machine learning techniques. Data mining, as a whole, is a multidisciplinary field based on areas of mathematics and computer science such as statistics, pattern recognition, artificial intelligence, and data visualization. In this particular work, we are going to concentrate on retrieving information from a set of published papers which are publicly available from PubMed Central (U.S. National Institute of Health (NIH) , 2009). The NIH mandates that all publications resulting from NIH-funded research must be deposited in PubMed Central, making this a valuable and comprehensive resource for our work. The necessity of developing new techniques for data mining stems from the abundance and complexity of the available data. The last couple of decades have seen the flourishing of data-mining research in many directions, including classification and prediction of concepts and classes, pattern recognition in the data, cluster analysis, and trend and evolution analysis, to name just a few.
Data mining faces numerous challenges and must address issues such as efficiency and scalability, handling noise uncertainty, incompleteness of data, the incorporation of constraints, expert knowledge, background knowledge within the data mining process, and mining diverse and heterogeneous types of data. (Tan, 1999), (Kriegel, 2007). As we will describe later, several of these issues were addressed as part of our research.

The knowledge discovery process (U. Fayyad, 1996) consists of several key steps which have been followed during our work:

1. Gaining familiarity with the application domain and identifying the ultimate goal of the application.
2. Creating a target data set.
3. Data cleaning and preprocessing.
4. Data reduction and transformation: identifying useful features, standardization of the data, and reducing dimensionality of the data.
5. Identifying which particular method (such as classification, clustering or association) is the most suitable to our goal.
6. Choosing the mining algorithm(s) appropriate for the method chosen in the previous step.
7. Data mining: search for patterns of interest, classification, clustering, etc.
8. Evaluation of results and knowledge presentation: visualization, transformation, removing redundant patterns, etc.
9. Use of discovered knowledge (U. Fayyad, 1996).

Our study followed these stages paying particular attention to phases five and six. The following section describes in details which data mining methods and which algorithms we chose for our research and the reasons for those choices.

1.3 Machine Learning Algorithms

It is worth taking a more detailed look at stage six from the previous section – choosing mining algorithm. There are numerous machine learning algorithms

available, each tailored for a specific set of tasks. In general we can identify the following categories for machine learning algorithms:

- Classifiers – Given a training set, a classifier is supposed to categorize terms/concepts/keywords/entities into an appropriate class as learned from the training set. Classification can follow simple Boolean logic (whether an entity falls into a class of our interest or not). Classification can also involve many different classes. Some well known examples of machine learning classifiers are: C4.5 (Quinlan, 1996), Support Vector Machines (SVM) (Burges, 1998) and Neural Networks.
- Clustering algorithms – Grouping of similar elements into appropriate sets (P.S. Bradley, 1999). The main difference between this category and the previous one is that we usually do not know in advance how many sets or clusters there exist in reality. Some commonly-used clustering algorithms are: K-means (K. Alsabti, 1998), hierarchical clustering (Ying Zhao, 2005) and self-organizing maps (Kohonen, 1998).

Our ultimate task is classification and annotation of large collections of scientific papers. Thus, we focused on algorithms from the classifier category. Many machine learning algorithms in that category are known to perform well under a variety of conditions. The accuracy of the algorithms depends upon the characteristics of the specific task. We decided to use an SVM-based framework that has been previously reported to perform well on large data-sets (A. Statnikov, 2005).

Another reason for choosing SVM as our primary machine learning algorithm is our previous extensive and positive experience with this method. We have previously used SVMs in a study aimed at understanding the relationship between Single

Nucleotide Polymorphisms (SNPs) and human disease (P. Yue, 2006). In this study we automatically extracted SNP and disease information from bio-medical literature and public databases, and used this information to create a new resource that combines all the extracted information with the relevant data-sources. SVMs were used in this project to both perform literature mining and to determine the strength of the contribution of a particular SNP to disease.

1.4 Support Vector Machine (SVM) Algorithm

Support Vector Machine algorithm is based upon Statistical Learning Theory developed by V. Vapnik (Vapnik, 1999). SVM non-linearly maps the input space into a multi-dimensional feature space using mathematical functions known as kernels. SVM performs classification by constructing an N-dimensional hyperplane that optimally separates the data into two categories – positive and negative examples. First, SVM is trained on a labeled data set in order to find such optimal hyperplane. Then the trained model can be applied to the data set we want to classify. Each data point is mapped into feature space determining which side of the separating plane it lies on. Much of the SVM's power comes from its criterion for selecting a separating plane when many candidate planes exist: the SVM chooses the plane that maintains a maximum margin from any point in the training set. According to the statistical learning theory, the choice of the maximum margin hyperplane for some classes of well-behaved data will lead to maximal generalization when predicting the classification of previously unseen examples (Vapnik, 1999). The SVM algorithm can also be extended to cope with noise in the training set and with multiple classes.

1.4 Hypothesis

Since Data Mining is such a broad research area, we have decided to focus on a more specific topic. One problem of particular interest and difficulty is increasing the accuracy of the retrieved data. This has been a long-standing area of research; however, it turns out to be a difficult problem. It has been long known that automated tools for information retrieval are virtually never 100% accurate. On the other hand, manual data mining (especially from text) is usually quite accurate. That is one of the reasons that public biological databases are usually curated manually (examples: HGMD (P. D. Stenson et al, 2003), KEGG (M. Kanehisa, 2008), OMIM (OMIM, 2009)).

However, as it has also been mentioned earlier, the abundance of data is such that manual review significantly slows down the process of information retrieval (which gets us back to the problem of being overwhelmed with the data and starving for knowledge). To resolve this issue a hypothesis has been proposed that minor human supervision can dramatically improve the accuracy of the automated retrieval process. We investigated this hypothesis as part of this thesis and our results indicate that manual supervision can, indeed, improve the results. We specifically focused on mining bio-medical text to retrieve information related to antibiotic resistance. However, the system has been built in such a way that it can be easily adjusted for any specific topic of interest, even outside of biological research.

1.5 Antibiotic Resistance

The discovery of antibiotics has been one of the greatest breakthroughs in the 20th century. However, bacteria resistant to antibiotic treatments have become a substantial problem almost immediately after the introduction of said antibiotics. As the authors Dan Anderson and Bruce Levin write in their article from 1999 "Within 50 years, the number of species and strains of pathogenic and commensal bacteria resistant to antibiotics and the number of antibiotics to which they are resistant has increased virtually monotonically world-wide" (Levin, 1999). Thus, antibiotic resistance has become a serious and world-wide threat to the treatment of infectious diseases (see e.g. (Conly, 2002)..

In today's world antibiotics are used in many areas such as cattle disease treatment and the agrifood industry (Conly, 2002), thus increasing selective pressure towards antibiotic–resistant organisms even more. The emergence of antibiotic resistance due to increased antibiotic usage is well documented (Andersson, 2003, 2006). Furthermore, this problem is here to stay as there is little evidence that reduced reliance on antibiotics can reverse the trend.

<u>Molecular mechanism of antibiotic drugs resistance:</u>

Antibiotics are produced from naturally occurring substances such as penicillin-producing fungus and from synthetic substances such as sulfa drugs. Widely used groups of antibiotics, both natural and synthetic, generally have four targets in bacterial cells (Wright, 2005):

- Bacterial cell wall biosynthesis
- Bacterial protein biosynthesis
- DNA replication and repair

- Folate coenzyme biosynthesis

The development of antibiotic resistance is due to two key mechanisms: intrinsic and acquired. Some bacteria had antibiotic resistance genes even before the substantial introduction of antibiotics into environment in the recent 50 years (Martinez, 2008). Such types of bacteria are said to be intrinsically resistant to antibiotics. Other major forms of antibiotic resistance are acquired by Horizontal Gene Transfer (HGT), and mutations (Martinez et al., 2009). These types of bacteria have acquired the ability to resist antibiotics only recently due to selective pressure of antibiotics used in therapeutic settings (Martinez, 2008).

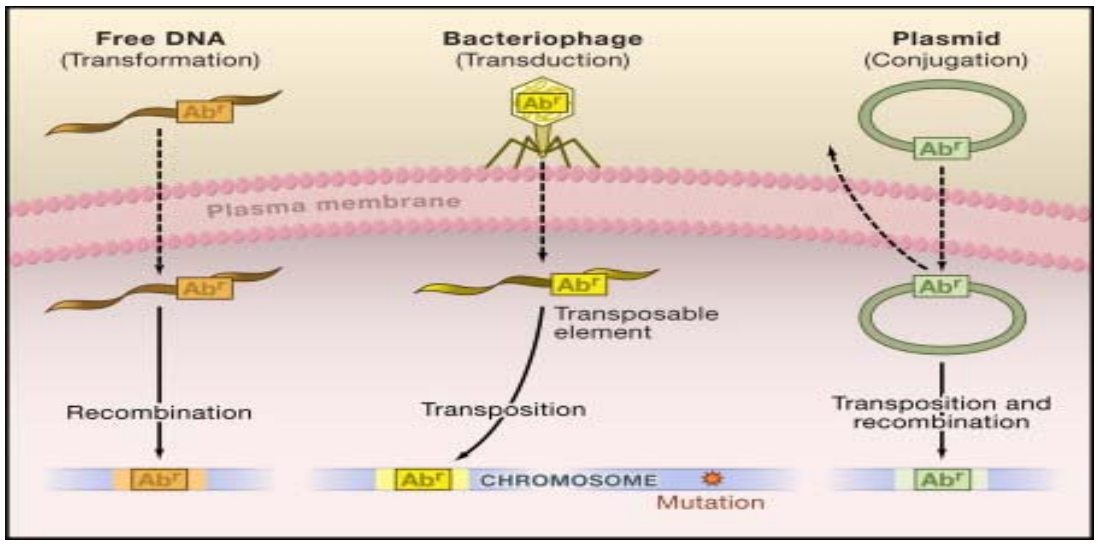

Figure 2: Antimicrobial resistance molecular mechanism
Acquisition of Antibiotic Resistance Bacteria can become antibiotic resistant (Ab^r) by mutation of the target gene in the chromosome. They can acquire foreign genetic material by incorporating free DNA segments into their chromosome (transformation). Genes are also transferred following infection by bacteriophage (transduction) and through plasmids and conjugative transposons during conjugation. The general term transposable element has been used to designate (1) an insertion sequence, (2) composite (compound), complex, and conjugative transposon, (3) transposing bacteriophage, or (4) integron. (Levy, 2007)

Bacteria use three major mechanisms to resist the antibiotic treatment: (i) destruction of the antibiotic by bacterial enzyme; (ii) change of the antibiotic target in the bacteria in such way that its susceptibility is lowered; and (iii) pump out the antibiotics using transmembrane efflux pumps, ensuring that the concentration of antibiotics stays below the toxic threshold in the bacteria cell (Wright, 2005).

Better understanding these resistance mechanisms, finding additional targets, and the development of new molecules will facilitate the discovery and development of new antibiotics that should be able to fight currently antibiotic-resistant bacteria, and more importantly, the development of new drugs that bacteria cannot develop resistance against. Thus, it is of utmost importance to have the related information available to the scientific community in an easily accessible and understandable way. The development of an Antibiotic Resistance Genes Database (ARDB) is one such source of information (Liu and Pop, 2009). This database unifies most of the publicly available information on antibiotic resistance. According to ARDB developers, most of the important information regarding antibiotic resistance and related data is "mostly paper-bound". It was necessary to read and analyze several books and hundreds of journal articles in order to compile and validate the data for the ARDB database. Retrieval of the data from published literature has been done manually. Therefore, it is quite clear that much of the time and effort could be saved if a semi-automated data retrieval tool has been available. One of the major aims of this research is to develop such tool to facilitate information retrieval for further versions of the ARDB database in a much faster and easier way.

1.6 Research Outline

We would like to define our specific task for this research and how the resulting software might be used. We are going to build a tool that will perform automated annotation of bio-medical text related to antibiotic resistance. We will use two types of data sets – semi-automatically annotated and automatically annotated to train our machine learning algorithm and test our aforementioned hypothesis. The semi-automatically set of documents will first be annotated using the software. Then a

person will review the results of automated annotations and edit them according to his/her expert knowledge. After training the SVM on these two different data sets, we will apply the resulting models on a testing data set. We will compare the resulting annotations and conclude whether or not we were able to prove our hypothesis.
In the grand scheme, we hope this system will reduce the time and effort required for documents annotations and databases compilation based on these annotations.

Proposed Workflow

1. A scientist retrieves a set of documents related to his/her research interest (antibiotic resistance in our case).
2. He/she then applies our software to automatically annotate his/her documents.
3. He/she reviews these annotations and corrects them as needed.
4. The reviewed and corrected set of documents is added to a training set for machine learning, so that the next application of our software will be more accurate and complete.

Based on our hypothesis, we hope that every additional manual review will improve the system's accuracy up to the point where a scientist will need to put as little effort as possible into document annotation. If automated annotation becomes sufficiently accurate, it will require much less time and effort to populate specialized databases with the data from texts.

Chapter 2: GATE development environment

The hypothesis for this research has been that a semi-automated annotation system will be more accurate and flexible than a fully automated one. To prove this hypothesis, we built a tool to perform semi-automated data retrieval and annotation. We had to narrow down the topic for information retrieval so that we could construct an appropriate data set and had a good performance benchmark. Thus, we decided to concentrate on pursuing the topic of drug-resistant bacteria, especially since such a tool will help continue populating the aforementioned ARDB database at a much faster rate.

Since the main purpose of this research was to prove the hypothesis and not to develop completely new annotation algorithms and data mining tools, we decided to use an off-the-shelf product. The tool requirements were as follows:

- The tool should perform text analysis and annotation
- The tool should be available for download
- The code should be open source

There were additional parameters that were not required, but would provide distinct advantage to us by having them. For example, it would be helpful if some other researchers already used the tool in similar tasks successfully. It would also be beneficial if tool developers would be available to provide support. Finally, it would be beneficial if the tool were written in a programming language known to us, such as Java, C, C++, or Perl. The following section discusses some of the tools that we have found and their conformation to the requirements.

2. 1 Review of available Text Mining software

The following is the list of tools that were chosen as candidates for our research. We describe each tool in a short summary and explain which requirements satisfied or not satisfied.

ABNER - A Biomedical Named Entity Recognizer (Settles, 2005) – ABNER is a Java-based software tool for bio-medical texts analysis. A machine learning algorithm – linear-chain conditional random fields (CRFs) - is used in ABNER for entity annotation. ABNER also has a simple user interface and Java API. This tool was one of the main candidates for this research; however, it turned out to be quite difficult to customize it for our needs. ABNER only finds general entities such as *"Protein", "DNA", "Cell Line"*, etc. We faced numerous problems with ABNER installation and customization, thus it was decided to look for a more flexible and easily adjustable alternative. In addition, the performance of ABNER was not very impressive. The following figure summarizes the precision and recall for various entities learned by ABNER:

NLPBA model. Five entities trained on 18,546 sentences, evaluated on 3,856.

Entity	Recall	Precision
Protein	77.8	68.1
DNA	63.1	67.2
RNA	61.9	61.3
Cell Line	58.2	53.9
Cell Type	65.6	79.8
Overall	72.0	69.1

BioCreative model. One entity trained on 7,500 sentences, evaluated on 2,500.

Entity	Recall	Precision
Protein	65.9	74.5

Figure 3: ABNER accuracy measurements (Settles, 2005)

Textpresso - an Ontology-Based Information Retrieval and Extraction System for Biological Literature (Muller HM, 2004) – Textpresso is a text-mining system for scientific literature. The authors of Textpresso claim that their tool can not only perform keyword search but can also find and classify biological concepts (such as gene, allele, cell, or phenotype) and classify two related objects (e.g., associations, regulations, etc.). Textpresso's main programming language is Perl. Textpresso marks up text to produce an e**X**tensible **M**arkup **L**anguage (XML) document, which is then processed using an ontology. The ontology was compiled using regular expressions and word counts. Several search engines have been compiled using Textpresso technology for different organisms such as *C. elegans*, Rat, Arabidopsis, Zebrafish, etc.

Even though Textpresso seems to satisfy all our needs, unfortunately it was not available for downloading at the time we performed the literature/tool survey (autumn 2008). Moreover, its official web-site has some problems and is often not available at all. Thus, this tool could not be used for our research.

CBioC – Collaborative Bio Curation (Baral, 2007) – is a web-browser extension that facilitates collaboration over automated bio-medical data extraction from texts. CBioC developers claim that the data is initially extracted automatically from the texts so that it can then be reviewed and verified by collaborating scientists. CBioC uses Natural Language Processing (NLP) tools IntEx (Ahmed CT, 2005) for the initial extraction of protein-protein interactions, gene-disease relations, and gene-bioprocess relations. Unfortunately, CBioC does not perform the exact task required for our research. Its performance highly depends on extensive collaboration which is not a goal of our study. Moreover, its source code is not available for modification and customization for our needs. Thus, this tool could not be used in our study.

POSBIOTM/W – Bio Text Minining System Workbench (Kim Kyungduk, 2005) - POSBIOTM/W is a workbench for machine-learning oriented biomedical text mining system. It is comprised of four components: the Managing tool, the Named Entity recognition tool, the Event Extraction Tool, and the Annotation Tool. We would be interested in the latter two components. According to POSBIOTM/W developers, the Event Extraction component is based on a modified WHISK (Soderland, 1999) machine-learning algorithm. Unfortunately, there is not much detailed information regarding this tool, nor does it seem available for download. Thus we could not use it in our research.

FACTA – Finding Associated Concepts with Text Analysis (Yoshimasa Tsuruoka, 2008) – FACTA is a text mining tool which is supposed to help discover associations

between biomedical concepts mentioned in MEDLINE articles. It is a C++ based tool which uses word/concept indexing system to retrieve relevant data from the abstracts. As with the previous tool there was not much detailed information about this software and the algorithms it uses. It is also not available for download at the time of literature/tools review for this study.

KEX – Knowledge Extraction tool (Ken-ichiro Fukuda et al, 1998) – KEX was listed as a similar tool to ABNER. It is a protein name annotation tool based on PROPER (PROtien Proper-noun Extraction Rules) written in C and Perl. Unlike the previous two applications, this one is downloadable. However, we could not find any publication describing which concepts and algorithms have been used for this tool. According to the limited documentation available, this seems to be more an entity recognition tool rather than annotation tool. Thus we decided not to use this application in our research.

Finally, we have found a tool that satisfied all our requirements and offered additional features. General Architecture for Text Engineering (GATE) is a language processing tool and development environment; it is very flexible and easily downloadable. We have found numerous successful projects using GATE and performing tasks in similar field of study. Examples of other biology-related information extraction tools based on GATE are as follows:

- **Parallel IE,** *Merck KGaA, Darmstadt, Germany* - Information Extraction on a Linux cluster for bio-medical text mining and indexing.
- **Medical Informatics,** *University of Pittsburgh, USA* - Annotating surgical pathology reports using UMLS.
- **Medical Informatics,** *Institute for Medical Informatics and Biometry, University of Rostock, Germany* - Analyzing MEDLINE abstracts to extract

causal functional relations, which are essential for the construction of genetic networks, as a step towards characterization of diseases.

- **BioRat,** *University College, London, U.K.* (Corney, 2004) - A general-purpose information extraction tool designed to be used by biologists to data-mine text from journals. It has been successfully applied to protein-protein interaction discovery and more projects are underway in several other areas. It uses GATE at its core, while also providing tools to design new templates, edit gazetteers and to download full-length papers from the web. The software is available for academic use, and is part of an ongoing research project.
- **InESBi,** *Institute for Medical Informatics and Biometry, University of Rostock, Germany* - the information extraction for this structural biology project is aimed at the 'material and method' part of the structural biology publications. The purpose of this project is to populate a database. Some of the pieces of information for the database are retrieved from structured files named PDB. The material and method used for experiments are not in PDB files. Thus, the intent is to extract that information from the text of the publications.

In addition to the stated above advantages, GATE has a very extensive user manual and user support system where GATE developers and experts answer user questions. The following section describes GATE framework in details.

2.2 GATE framework

Gate framework has been developed in the department of computer science at the University of Sheffield, UK. According to GATE developers this is “a framework and graphical development environment which enables users to develop and deploy

language engineering components and resources in a robust fashion." (Humphreys., 1996) This development environment consists of different modules that deal with various aspects of language processing. As mentioned earlier, GATE has been chosen because of its versatility, availability of source code, well-designed personalization, and an ability to extend it in various ways. GATE framework consists of reusable and extensible Language Engineering modules. The main language used for GATE is Java with XML configuration files, in some cases a Java based rules language – Java Annotations Pattern Engine (JAPE) is used. Additional modules can be developed in any language of choice, however Java interface is necessary for the new module incorporation.

GATE consists of three main types of components: Language Resources (LR), Processing Resources (PR), and Visual Resources (VR).

- Language Resources represent entities that are related to general language (GATE can be used to analyze texts in different languages) such as ontology, lexicons, and corpora.
- Processing Resources are mainly processing modules such as parsers, generators, machine learning algorithms, and transducers.
- Visual Resources are mainly related to visual representation, GUI, and editing modules. (H. Cunningham, 2002)

All these resources are available for editing and extending. The following section describes in more details which modules have been changed and which remained untouched.

Specifics:

We used A Nearly-New Information Extraction system (ANNIE) (H. Cunningham, 2002) module as a skeleton for our project implementation. The classic version of

ANNIE includes the following modules: Tokenizer, Gazeteer, Sentence Splitter, Part of Speech Tagger, OrthoMatcher, and Transducer.

The following flow chart is schematics of ANNIE components pipeline:

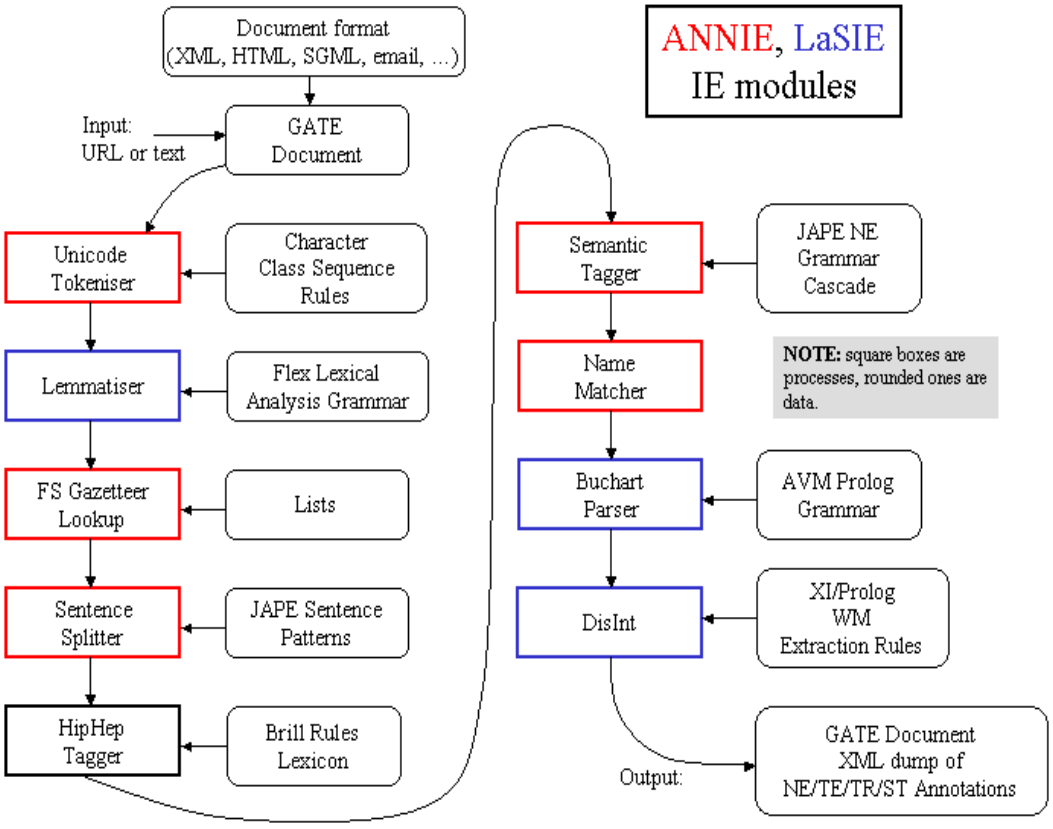

Figure 4: ANNIE modules pipeline (Humphreys., 1996)

The following is a brief description of each sub module and our changes/additions where applicable.

Document Reset: The purpose of this module is to strip off the annotations and get the document back to its original state that it was in before any processing has been applied to it. In other words, this module reinitializes the document to its original state. There have been no changes to this module.

English Tokenizer: This module splits the text into simple tokens such as numbers, words, punctuation marks, etc. This module remained unchanged, as its functionality is necessary in full range and without changes for subsequent pipeline.

Gazetteer: Gazetteer is a collection of lists with various names such as personal names, locations names, organization names, etc. A gazetteer list contains one entry per row. In addition there is "lists.def file", which contains all the gazetteer lists we need for the specific application. Each entry in "lists.def" file is a gazetteer list filename as well as a major type and minor type of the list where applicable. The following modules in pipeline will use the assigned types for annotation of the keywords. We have added a few gazetteer lists which we compiled using ARDB database data (Pop, 2008). The following is an example of "lists.def" file:

```
abbreviations.lst:stop
adbc.lst:adbc
airports.lst:location:airport
antibiotic_names.lst:medicine:antibiotic_name
ARclass.lst:antibiotic_resistance_class
bacteria_names.lst:bacteria_names
```

Figure 5: Gazetteer configuration file

In this example, "antibiotic_names.lst" gazetteer list is of major type *medicine* and of minor type *antibiotic_name*. It is not necessary to define minor type but in certain cases it is helpful for further annotations.

We have added the following gazetteer lists and assigned them the following major/minor types:

antibiotic_names.lst : medicine : antibiotic_name

ARclass.lst : antibiotic_resistance_class

bacteria_names.lst : bacteria_names

go_terms.lst :go_terms

kegg_drug_names.lst : medicine : kegg_drug_name

taxon_names.lst : taxon_names

Figure 6: Gazetteer lists we added

Sentence Splitter: This module splits the text into sentences. In other words, it assigns an annotation "*Sentence*" to each appropriate piece of text by using a cascade of finite-state transducers. This module is required for the following "tagger" module. Sentence Splitter distinguishes sentence-marking full stop from other kinds of markings with the help of a gazetteer list of abbreviations.

Part of Speech (POS) Tagger: The tagger module produces a part-of-speech tag as an annotation for each word in the text. The algorithm is based on the Brill tagger (Hepple, 2000).The tagger uses a default lexicon and rule set (the result of training on a large corpus taken from the Wall Street Journal).

Transducer: This module assigns semantic annotations to words. It consists of a default set of JAPE based rules which can be modified or more rules can be added. The default annotation types the transducer assigns include "*Money*", "*Location*" "*Organization*", etc. This module has been modified and more rules were added. The additional rules assign the following annotation tags: DrugName, Resistance, GOterm, taxonname, and several more.

Orthographic Coreference (OrthoMatcher): The Orthomatcher module adds identity relations between named entities found by the semantic tagger, in order to perform

co-reference. It does not find new named entities as such, but it may assign a type to an unclassified proper name, using the type of a matching name.

The matching rules are only invoked if the names being compared are both of the same type, i.e. both already tagged as (say) organizations, or if one of them is classified as 'unknown'. This prevents a previously classified name from being re-categorized. In addition to aforementioned modules, we used the "learning" module necessary for training and applying SVM.

The annotations which we receive as the result of ANNIE pipeline application are organized in directed acyclic graphs. Annotations may be considered as the arcs in the graph; they have a start Node and an end Node, an ID, a type, and a Feature Map. Nodes have pointers into the source document as character offsets. GATE uses XML schemas to define different annotation types. The following is an example of such a schema which defines Date annotation types:

```
<?xml version="1.0"?>
<schema xmlns="http://www.w3.org/2000/10/XMLSchema">
      <!-- XSchema deffinition for token-->
      <element name="Date">
        <complexType>
          <attribute name="kind"  use="optional">
              <simpleType>
                <restriction base="string">
                    <enumeration value="date"/>
                    <enumeration value="time"/>
                    <enumeration value="dateTime"/>

                </restriction>
            </simpleType>
      </attribute>
  </complexType>
</element>
</schema>
```

Figure 7: XML schema example

In addition to the ANNIE built-in pipeline of modules we have also used a GATE based Support Vector Machine (SVM) tool (Y. Li, 2005). In general, the SVM used in GATE is very similar to the classical SVM algorithm introduced earlier, with a few modifications. The main difference is that the GATE-based SVM uses "uneven margins parameter". This means that unlike the case of classical SVM, we do not need to have a training set which contains the exact same number of positive examples as negative examples. Since we are dealing with text classification and more specifically, classification of keywords, it is obvious that the number of positive examples will be much smaller in the text than negative ones if we consider each word in text as an example for the SVM. Clearly, there will be only a few words of type Drug Name in comparison to the total number of words in one paper. Thus there is a need to use an uneven margins parameter which is a ratio of the negative margin to the positive margin. In our specific implementation we used a margin value of 0.66.

For the purpose of text annotation GATE-SVM (Y. Li, 2005) SVM classification is applied for each word in the text. Specifically, each word is regarded as a separate instance either belonging or not belonging to one of the annotation classes (such as Drug Name in our case). The feature vector for SVM is composed of linguistic features of the words. For example capitalization information of the word, part-of-speech tag, token kind, or named entity according to ANNIE's rule-based transducer. Thus, when customizing SVM to detect the keywords for our study, it is very important to identify features unique to those keywords. In addition, the SVM input vector takes into account features of words preceding and following the current word of interest. GATE developers call this a *window size*. For example, in our implementation we used a window size 5, which means that SVM input vector is a

combination of the features of the current word and those of its 10 (five from each side) neighboring words.

After SVM has been applied, additional post processing is applied. The output of SVM classifier is transferred into a probability value via the Sigmoid function $s(x) = 1/(1 + exp(-\beta x))$ where β is set to 2. At the end of post processing, the probabilities for all possible annotation tags are reviewed. The entity under consideration is assigned an annotation tag only if the combined probability is higher than 0.25

The following chapter discusses in greater detail the implementation of our project.

Chapter 3: Implementation

3.1 Data Set

There are many different web based collections of bio-medical papers. The largest and most popular one is National Library of Medicine (NLM) PubMed. PubMed usually includes only the title and the abstract of a published paper. We wanted to include the whole paper texts, since we believe that most of the relevant information is within the body of the paper rather than in its title and/or abstract. Thus, a free digital archive of biomedical and life sciences journal literature (PubMed Central (PMC) (U.S. National Institute of Health (NIH) , 2009) has been used to generate the data set for our research. We have chosen papers from *Annals of Clinical Microbiology and Antimicrobials* and *Molecular Microbiology* journals for our data set, since these journals often publish papers on our area of interest – antibiotic resistance. The data set for the experiments consists of 43 papers. Our set of negative examples consists of 20 papers which are related to microbiology and diseases, however, they do not mention antibiotic resistance. The positive set consists of 23 papers which are related to antibiotic resistance research. A total of 16 papers have been randomly chosen and separated into three sets. The first set includes arbitrary chosen 3 positive and 3 negative example papers – this is our testing set for most of the experiments. The next set of 10 positive papers was first annotated automatically and then reviewed manually – this is the main training set for the experiments.

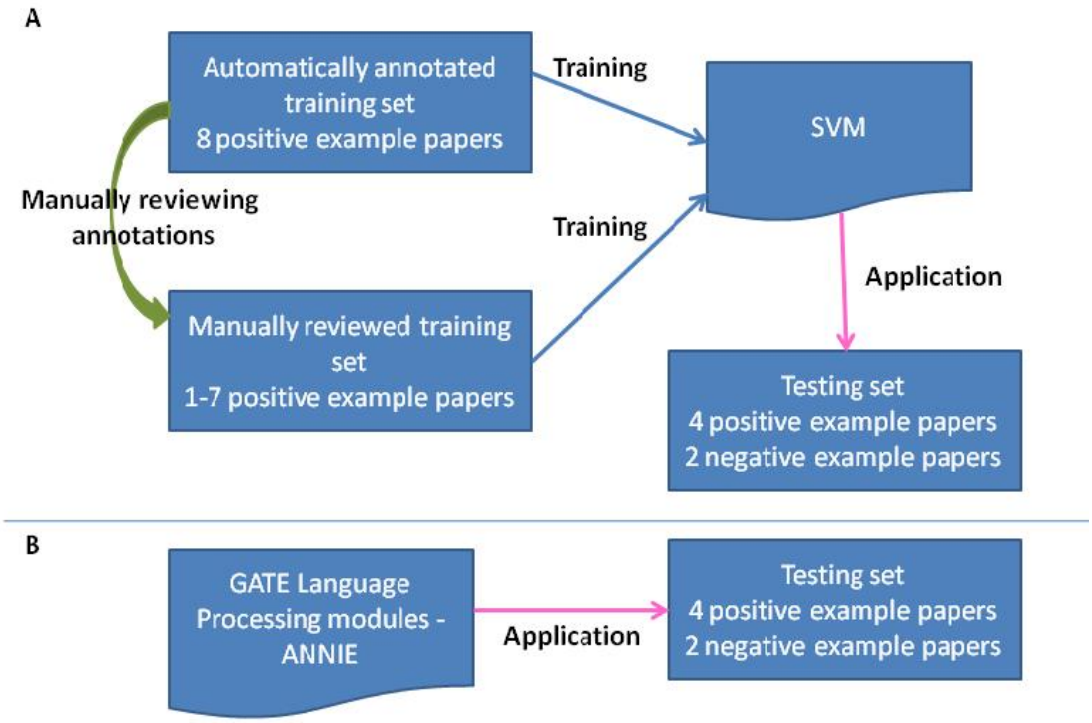

Figure 8: Data Sets diagram
(A) Displays the two types of training sets we prepared to train SVM (B) We used the same testing set for experiments not involving SVM, so that our measurements would be compatible.

3.2 Customizing GATE configuration files

GATE configuration files are usually XML schemas used to define various entities such as Annotation Type, SVM parameters, processing resources parameters, etc. An example of such configuration file for SVM parameters can be found in Appendix.

3.3 Jape Rules

JAPE allows you to recognize regular expressions in annotations on documents. However, GATE's model of annotation is based on graphs, meaning much more complicated data structures than simple strings. Thus, the result of regular expression applications is not always deterministic.

The following is an example of a few rules we have developed for identification of keyword *resistance* in various forms:

```
Phase:      Resistance
Input:  Lookup Token
Options: control = appelt

Rule: Resistance1
Priority: 20
// resistance to ...
(
 {Token.string == "resistance"}
 {Token.string == "to"}
 ( {Lookup.majorType == medicine})
)
:resistanceto
-->
 :resistanceto.ResistanceTo = {kind = "resistance", rule =
"Resistance1"}

Rule: Resistance2
Priority: 20
// ...-resistant
(
 ( {Lookup.majorType == medicine})
 {Token.string == "-"}
 {Token.string == "resistant"}
)
:resistance_
-->
 :resistance_.ResistanceDash = {kind = "resistance", rule =
"Resistance2"}

Rule: Resistance3
Priority: 15
```

Figure 9: JAPE rules example

3.4 SVM

Figure 17 (appendix) is an example configuration file used for our implementation. In this example we trained SVM to distinguish “DrugName” labeled keywords whose unique linguistic feature is “rule”. We have given this feature to this type of keywords using one of our JAPE rules. However, when applying the SVM learned module, there is no need to use that rule. SVM is able to classify the keywords with very high precision.

3.5 Work flow

The following diagrams show our work flow for this project.

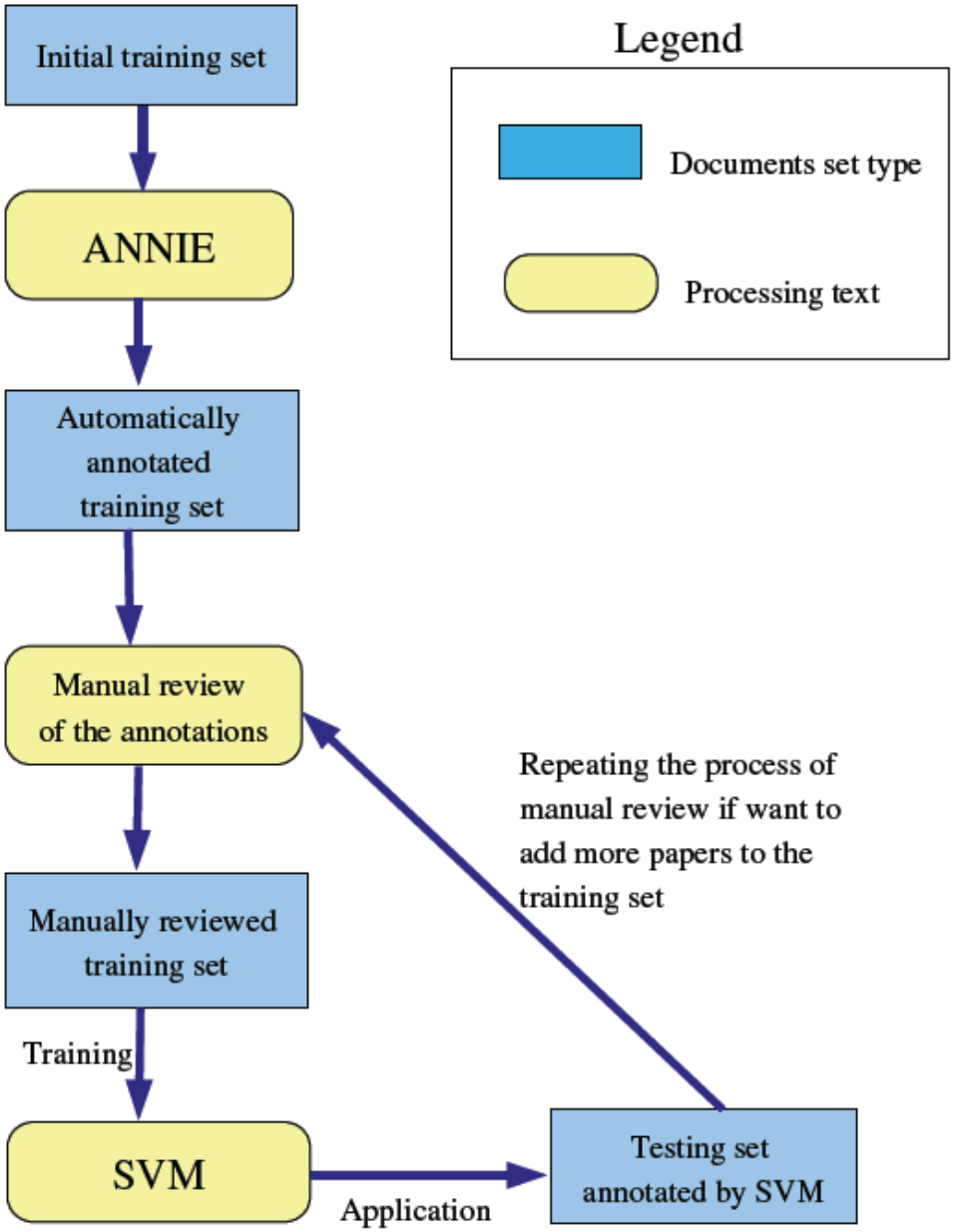

Figure 10: Workflow diagram

First we developed and configured some of the built-in modules supplied with the ANNIE pipeline. Afterwards, we selected papers for training and testing sets. Finally, we performed numerous experiments with semi-automatic text annotation with and without machine learning.

The following figures are snapshots of GATE graphical user interface with examples of what processing looks like.

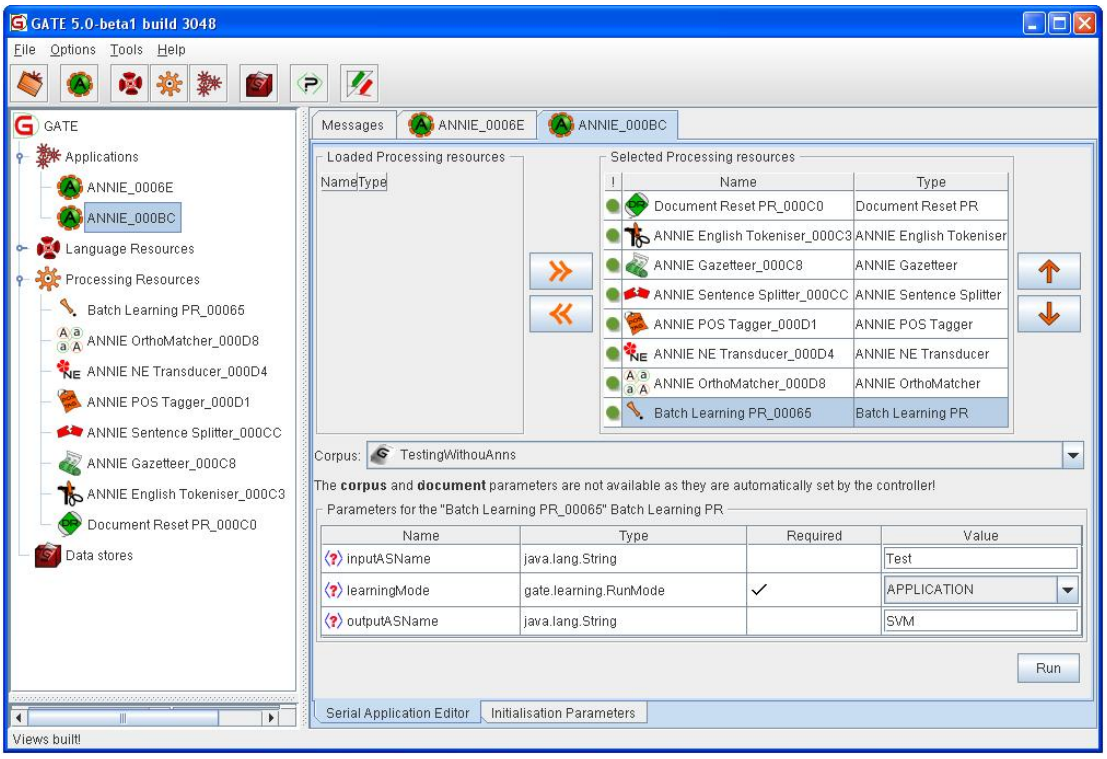

Figure 11: GATE user interface example

Figure 11 shows the pipeline consisting of language processing modules (ANNIE) and a machine learning module at the end of the pipeline. As you might notice we can select processing resources for a particular run. Currently all loaded processing modules are selected – this is why there is nothing left on the left side of the main screen. The left pane also displays which processing resources are currently loaded, as well as the application pipelines (left upper side of the menu – "Applications").

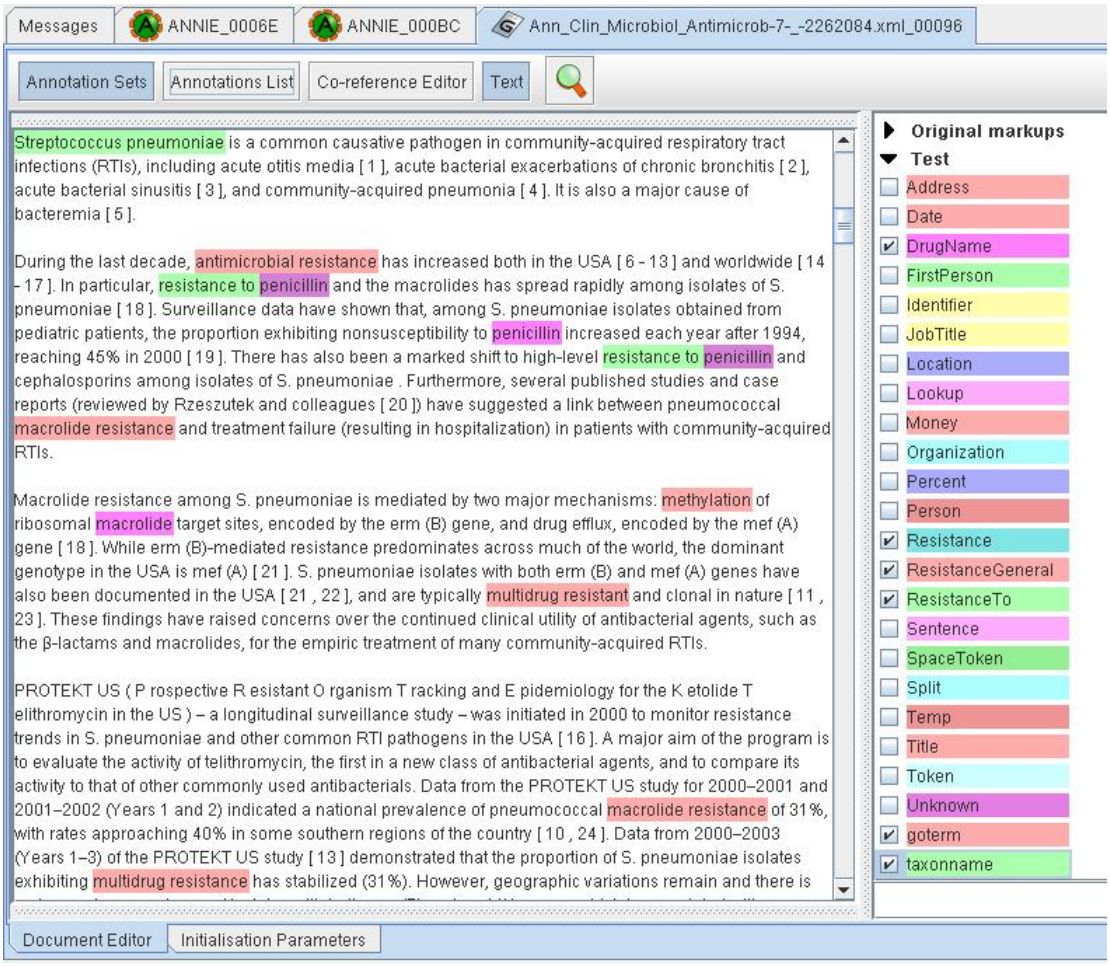

Figure 12: Automated annotation using GATE example

This figure displays what a processed document looks like. The right pane contains all the possible annotation categories found in this document. Putting a check mark in the right pane causes the chosen annotations to be highlighted with the corresponding color.

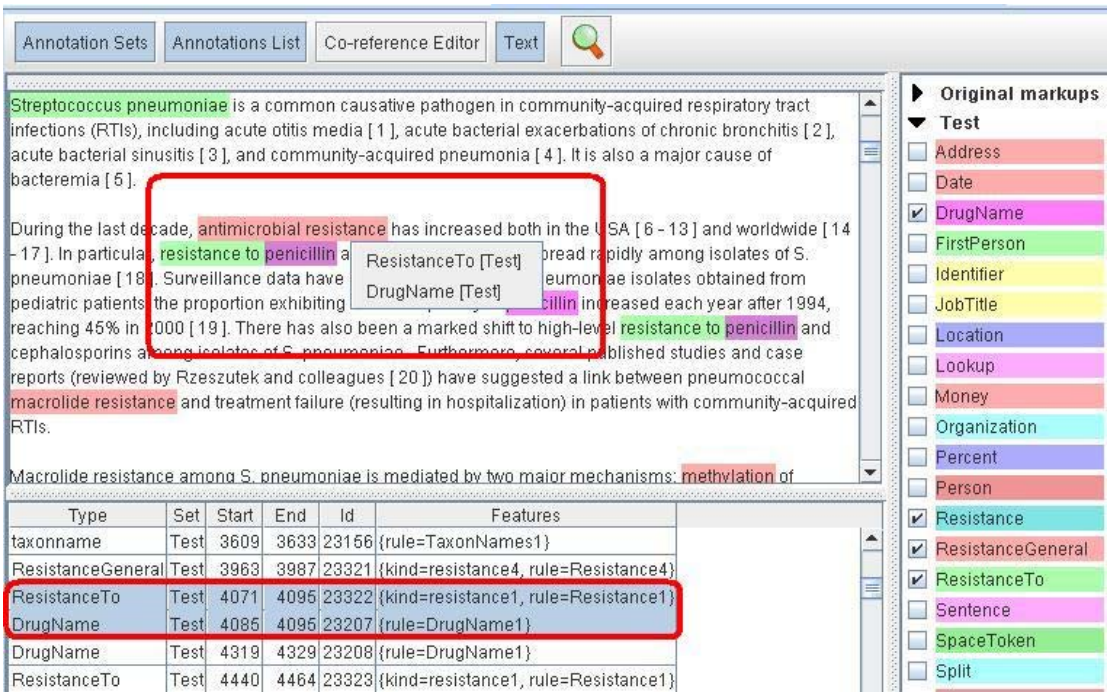

Figure 13: GATE multiple annotations example

Figure 13 shows that we can have multiple annotations on the same word – in this case word "penicillin" highlighted with purple color inside the box with red margins. This keyword is annotated with both *DrugName* and *ResistanceTo* labels. Additional information can be gathered from the lower left pane in the red-margin box. This pane displays the Type of annotation, its beginning and end, its NodeId, and features with their values. For example the annotation of type *ResistanceTo* has two features 'kind' and 'rule' with corresponding values of 'resistance1' and 'Resistance1'. The features and their values have been assigned in the example JAPE rule mentioned in "JAPE rules" section.

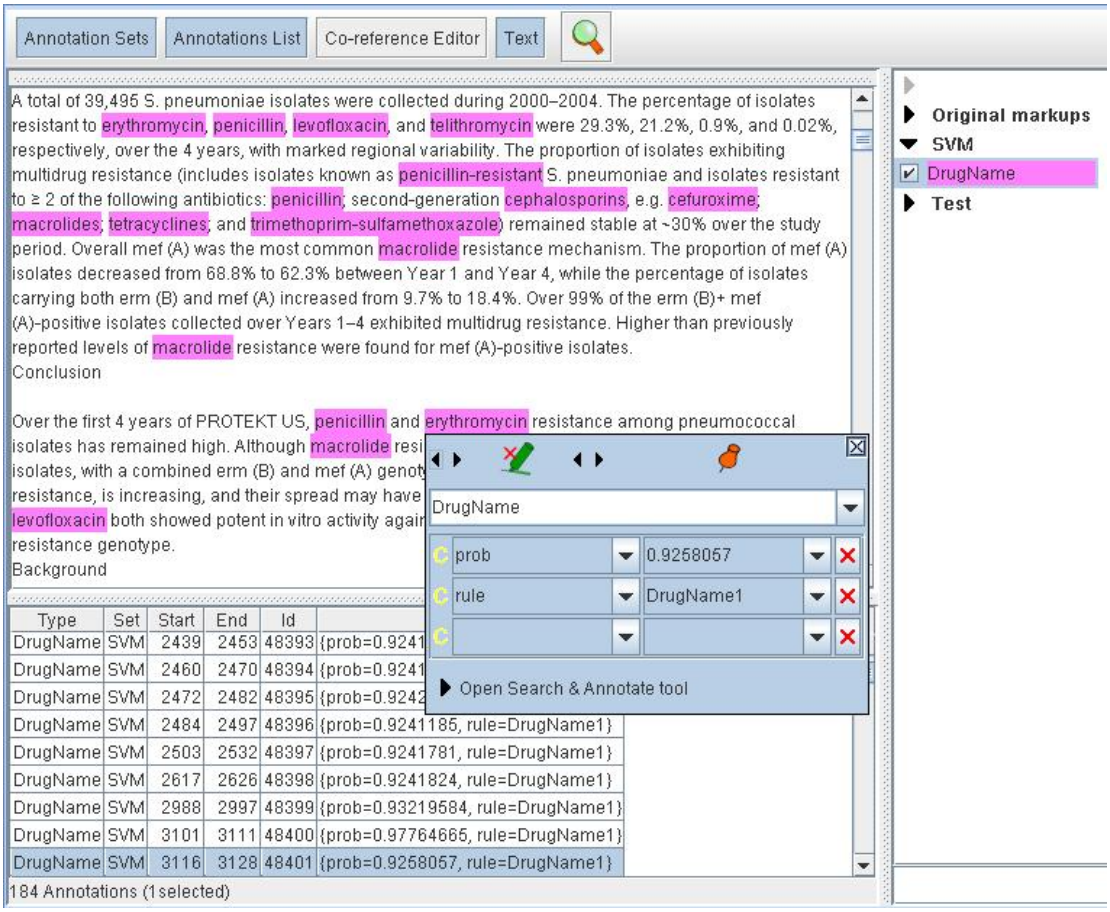

Figure 14: GATE SVM application example

Figure 14 displays the resulting annotation after application of machine learning – SVM. The right pane displays now three sets of annotation types since we defined a separate set for SVM output annotations called “SVM”. The dialog box in the figure is the usual dialog box that opens upon mouse-over. The only addition is the “prob” feature which is output by the SVM. In this case SVM assigned 0.93 probability that erythromycin is a drug name. Additional SVM-assigned probabilities for the highlighted keywords could be seen in the lower pane.

We performed numerous experiments with the described application. The following chapter presents the results of the experiments we performed to prove our hypothesis.

Chapter 4: Results

For our experiments we have used precision and recall metrics as a widely accepted way to measure the accuracy of information retrieval applications (N. J. Belkin, 1992). The results have been calculated according to the following formulas:

$$\text{Precision} = \frac{tp}{tp + fp}$$

$$\text{Recall} = \frac{tp}{tp + fn}$$

Automatic annotation without using of SVM

Our first experiment was to measure the accuracy of completely automated text annotation based on machine learning algorithms. The following table summarizes the results we received.

Keyword labels	True Positive	False Negative	False Positive	Precision	Recall
DrugNames	341	411	16	96%	45%
Resistance	55	201	0	100%	21%
Taxon	243	223	2	99%	52%
Total for all three categories of keywords	639	835	18	97%	43%

Table 1: Automated annotation experiment summary

The graph in Figure 15 presents precision and recall for each annotation type as well as the total for all three types. This graph might help to better understand the trends.

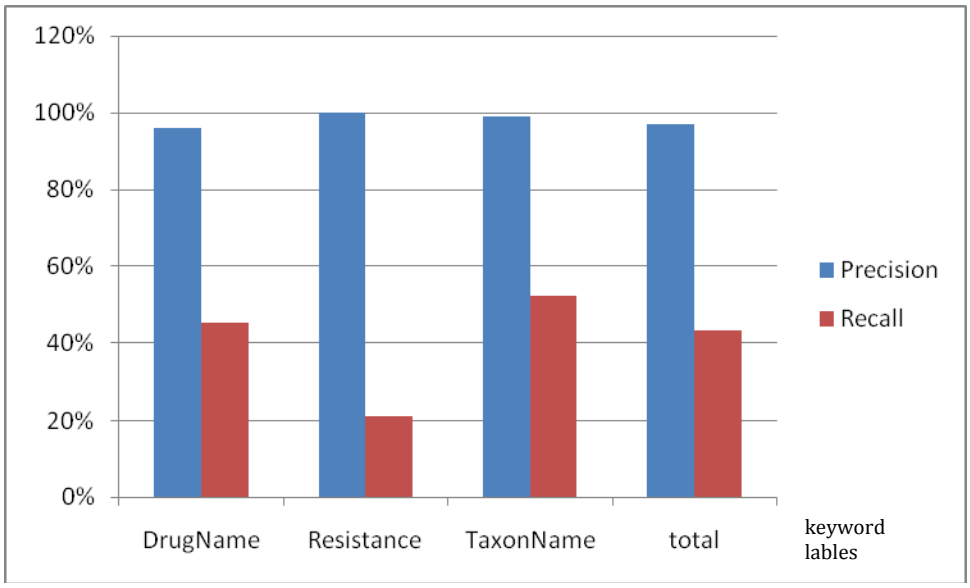

Figure 15: Precision and Recall Diagram for automated annotation experiment

Results of using SVM trained on automatically annotated set of papers (no manual review)

We have trained an SVM on a set of documents that has only been automatically annotated without human intervension. The training set was six papers, and the testing set also consisted of six papers. The results are as follows:

Total true positive keywords found	**150**
Total missed keywords (false negative)	**115**
Precision	**93%**
Recall	**57%**

Table 2: Automated annotation using SVM experiment summary

Annotation using SVM trained on manually reviewed documents.

This experiment had two purposes. First, we wanted to show how the performance of machine learning improves when documents used for training are manually reviewed. Second, we wanted to measure how much effort is enough to improve the performance (recall in this case)

# of papers in training set	True Positive	False Negative	False Positive	Precision	Recall
1	160	68	5	97%	70%
3	201	67	28	88%	75%
5	243	38	30	89%	86%
7	243	39	8	96%	86%

Table 3: Manually supervised annotation using SVM experiment summary

Diagram of precision and recall for annotation using manually reviewed papers for SVM training is included for better understanding of the trends in precision/recall changes:

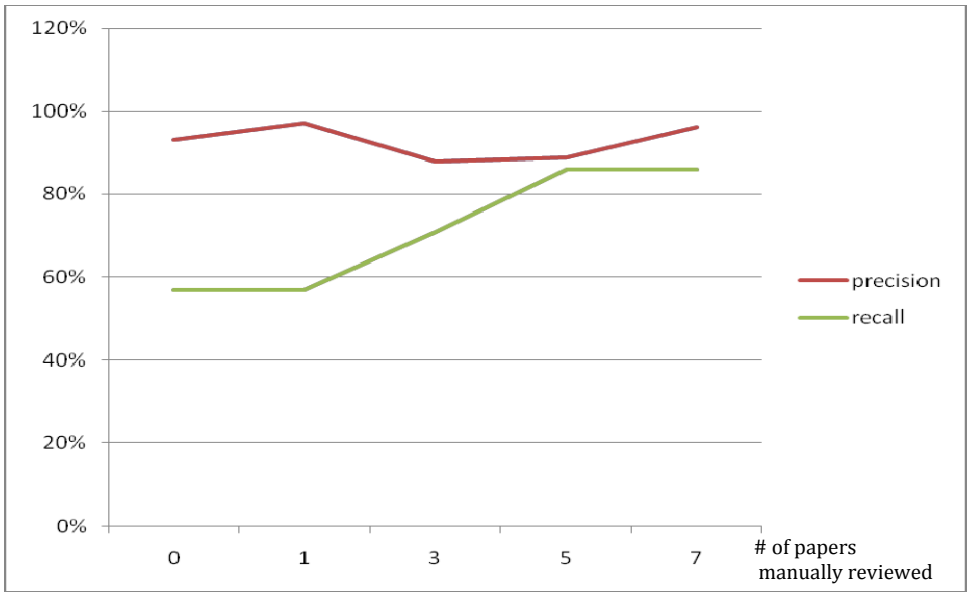

Figure 16: Precision and Recall graph for semi-automated annotation using SVM

Chapter 5: Discussion and Conclusion

5.1 Discussion

It is clear from the first glance at the results, that total accuracy of the annotations is much higher when manual review is used. Thus, we can immediately say that our hypothesis has been proved correct at least on the example of this specific application.

Now we would like to discuss the results in some details. First of all, let us look at the results of automatic annotation without human interference. We can notice that while precision is very high, the recall is extremely low. This means that whichever keywords the automatic annotator finds, they are correct in most cases. However, the annotator finds only about half of all keywords that actually exist in the text. There might be a few reasons for such behavior. As it has been noted in chapters two and three, text annotation is done mostly based on gazetteer lists and JAPE rules. Our gazetteer lists and JAPE rules are very specific thus GATE is able to identify the

keywords correctly and the precision is high. However, we can assume that gazetteer list does not contain all the names of antibiotics or all types of bacteria. We can also assume that the rules are written in a too discriminative way, and do not consider some of the correct input. We have also found that some of the keywords are not annotated since they begin with a capital letter if appear in the beginning of a sentence. Most of these issues might have been fixed if we had prepared more extensive gazetteer lists. For example, we could include the various forms for bacteria names such as *Escherichia coli* and *E. Coli*. We could also add more rules to identify the keywords. However, such additions do not seem very efficient. We will have to spend a lot of time for extensive customization of the application for specific task, while just manually reviewing and adding missing annotations in a few papers could achieve almost the same result much quicker.

An additional advantage of using manual review and machine learning is the fact that whenever a new paper is published, we could add missing annotations based on that new paper without actually modifying the application code, or configuration files.

Secondly, let us take a look at the results of combined manual and automated annotation. As it has been mentioned earlier, the annotation was a two-fold process. First, the training set was annotated automatically. Second, it was reviewed manually, missing annotations have been added and incorrect annotations removed. The results shown in the previous chapter are what we got after applying the trained SVM model on a test set that has not been processed before in any way. It is clear from the results that precision still remains high (above 85%) while recall also grows significantly. In other words, the application is capable of recognizing most of the real keywords in the text correctly. This result proves that our hypothesis has been correct, and we can continue developing our tool fully using all the advantages of our approach.

Even though our precision remains high for semi-automated annotation, there is one point about Figure 16 which requires further discussion. We observe a slight reduction of precision for training sets consisting of three and five papers. After this small drop, the precision climbs back to the high value of ninty-six percent. Lower precision means that during those two experiments we found more false positive annotations than in other tests. We believe that this drop in precision is due to the nature of the papers chosen for the manual review. It is possible that some of the annotations added as a result of the manual review were a little vague or not specific enough. Thus, SVM included additional unrelated keywords to its classification. Probably, repeating such experiments with other randomly chosen papers and averaging the results will eliminate the small drop in precision and we will witness a steady rise of the precision.

Finally we would like to compare results of applying SVM which has been trained on automatically annotated set of papers versus SVM that has been trained on manually reviewed set of papers. It is easy to see that the precision remains high for both cases. However, recall for automated training data set consisting of six papers is as low as recall for manual annotated training data set consisting of only one paper. In other words, SVM trained on automatically annotated data set is unable to find almost half of the correct keywords, which again proves correctness of our hypothesis.

5.2 Conclusion

This thesis' main aim was to investigate the hypothesis that automated scientific papers annotations using machine learning can become more accurate with some minor human supervision. To research this problem a system performing automated annotation of documents has been built. Then human review was simulated and

machine learning algorithm was applied to a set of documents. The resulting experiments have shown that significant improvement can be achieved using only minor human supervision. Moreover, it is clear from these experiments that the machine learning algorithm (SVM) can be adjusted and trained according to user's personal preferences. Even though the system was built for a quite narrow topic of antibiotic resistance, it has been developed in such way and using such tools can be easily adjusted to any bio-medical or any other scientific topic. Thus, we claim that the main hypothesis of this work has been proven to be true. It is possible to significantly improve automated annotation system with only some minor human supervision.

The main motivation of our work was to make scientists' work less time consuming and more focused on creative research rather than performing mundane repetitive tasks. We believe that our experimental software achieves this goal well. It is not necessary anymore to read hundreds of papers to retrieve relevant information, but just a few papers on the topic of interest should be enough. Specifically in the case of this research a database ARDB can now be populated with additional information with no need to analyze all the published literature on the topic.

5.3 Our Contributions

- A set of annotated documents that can be used by anybody as a training set for machine learning, or any other purpose
- A set of rules which identify various types of keywords related to antibiotic resistance, but can be easily extended to identify any topic related to microbiology

- A trained SVM that can identify keywords in relation to antibiotic resistance. This SVM has been trained on manually annotated papers and thus has about 90% accuracy for the specific topic
- Definition of requirements and an outline for our data mining tool user interface which are presented in the next “Future Work” chapter

Chapter 6: Future Work

This research has been performed in the frame of a bigger project designed to help scientists analyze published literature faster and easier. As mentioned before the developed (or rather extended) software is easy to train for any topic of interest. Next stage would be to build an application that will enable scientist to analyze and retrieve information from massive amounts of materials. For this specific task there is a need to develop User Interface. The following is an outline for our proposed UI.

6.1 User Interface Outline

Before presenting the actual outline of the user interface it is important to state what are the requirements. **First and foremost requirement** is the intuitive and easy to use interface. Our target audience are researchers from biological, medical or pharmacological background who usually no little about text mining. Thus the interface must be easy to understand and use for people who are not experts in data/text mining. **Our second requirement** is that the interface will be efficient from time point of view. Our ultimate goal is to save time and effort to the researches. Therefore, having a slow loading or slow processing interface will contradict that goal. **The third requirement** is for the interface to be interactive and customizable. **The final requirement** is ability to be integrated with other tools like web browser or other data mining applications.

After defining the requirements we can proceed to the outline of the interface. Based on our fourth requirement for the ability to be integrated with other tools, our user interface should be written in a platform independent programming language such as Java. The users should be given an option to train a new SVM model, use the default one, and retrain SVM on newly annotated papers. We should give the users the ability

to save SVM models separately and to choose which trained SVM module they would prefer to use. There should be an option of loading a set of documents rather than each document separately. Each annotation should be highlighted clearly, and the color of the highlight should be customizable. For each word, concept or relation there should be an option to add/delete annotations as well as having multiple annotations. SVM is able to classify each keyword, concept or entity into different categories. If such multiple classifications occur, we should present the users with the top few choices, since the highest probability does not always mean the correct annotation. As we stated beforehand, the interface should be easy to use, interactive and customizable. Thus it would be a good idea to have different versions of the interface for different expertise level. In other words, we would want our application to be accessible to a novice, and inspiring for advanced user. This outline is a very high level and there is need to put more effort and perform further research to create a really stimulating, time and effort saving user interface for our text mining tool. The following section discusses additional areas for further research.

6.2 Additional Research

It has been mentioned earlier in the thesis that this particular research is a part of a bigger combined effort to improve and make more efficient the process of reading scientific literature. We would like to incorporate our tool with some additional tools to make reading of the literature more productive and creative process. For that reason it would be important to research some of the following issues.

We have been able to prove that minor supervision over automated annotation process improves precision and recall significantly. However, it is mostly recall that we were able to improve since the precision in both fully automated and semi-

automated processed has been very high. In other words our application does not assign many false positive annotations. Thus, when a researcher needs to manually annotate the paper for machine learning he or she will have to put in much more effort by having to look for missed annotation rather than deleting the wrong ones. It is not clear which way is easier and more intuitive. From the first glance, it seems that deleting wrong annotations might be faster. However, if the precision is low, and the annotator has to examine closely every other highlighted annotation, the process of manual review might become slow. Thus, it would be important to research which is preferable for more effective manual review – low precision or low recall. This issue could be investigated with experiments measuring time and accuracy needed to perform the manual annotation. In addition a survey might help to determine the better alternative.

An additional issue worth further research includes further development of the tool to incorporate more relations annotation and building an ontology based on the annotations. GATE environment provides modules that facilitate building new ontology from annotations or *vice a versa* use an existing ontology for more accurate annotations. The time frame for this study was limited, thus, we were not able to incorporate ontology-related issue into this research. However, it is our intention now to continue the development and experimentation in that direction.

Finally, we want to try to apply our tool to solving other biology related problems. It would be very interesting to investigate how easy it is to adjust our tool to other tasks with similar general direction but different specific question. For example, it has been mentioned in the introduction chapter that our additional area of interest is Single Nucleotide Polymorphisms (SNP) relation to common human diseases. There is an abundance of literature on this topic. However, the specific mechanisms of how these

point mutations cause diseases are yet to be discovered. We know there are many research groups who investigate this intriguing problem from different points of view. If we could combine all their research data and publications into one integrated database where the data will be easily viewable and retrievable, that would be a great help in advancing the solution for the problem. We believe that our tool is flexible and customizing it for disease SNPs or another similar problem should not require a lot of effort. Our intention is to use our data-mining tool for elucidation of various SNPs, antibiotic resistance, or any other bio-medically related research problems we find fascinating.

Appendices

Example of XML based configuration file for SVM:

```
<?xml version="1.0"?>
<ML-CONFIG>
  <VERBOSITY level="2"/>
  <FILTERING ratio="0.0" dis="far"/>
 <SURROUND value="false"/>

<PARAMETER name="thresholdProbabilityEntity" value="0.3"/>
<PARAMETER name="thresholdProbabilityBoundary" value="0.50"/>
<PARAMETER name="thresholdProbabilityClassification" value="0.5"/>
<IS-LABEL-UPDATABLE value="true"/>
<IS-NLPFEATURELIST-UPDATABLE value="true"/>

<multiClassification2Binary method="one-vs-others"/>
<!-- To test with a thread pool, comment out the line above and
uncomment the
next one.
<multiClassification2Binary method="one-vs-others" thread-pool-
size="1"/>
-->

<!-- Evaluation : how to split the corpus into test and learn? -->
<EVALUATION method="split" runs="2" ratio="0.66"/>

<DISPLAY-NLPFEATURES-LINEARSVM numP="1" numN="-1"/>

 <ENGINE nickname="SVM" implementationName="SVMLibSvmJava"
           options=" -c 0.7 -t 0 -m 100 -tau 0.6"/>


<DATASET>
           <INSTANCE-TYPE>Token</INSTANCE-TYPE>

<WINDOWSIZE windowSizeLeft="5" windowSizeRight="5"/>

           <ATTRIBUTELIST>
                 <NAME>Form</NAME>
                 <SEMTYPE>NOMINAL</SEMTYPE>
                 <TYPE>Token</TYPE>
                 <FEATURE>string</FEATURE>
                 <RANGE from="-5" to="5"/>
           </ATTRIBUTELIST>
```

Continued on the next page.

```
            <ATTRIBUTELIST>
                  <NAME>Ortho</NAME>
                  <SEMTYPE>NOMINAL</SEMTYPE>
                  <TYPE>Token</TYPE>
                  <FEATURE>orth</FEATURE>
                  <RANGE from="-5" to="5"/>
            </ATTRIBUTELIST>
               <ATTRIBUTELIST>
                  <NAME>Tokenkind</NAME>
                  <SEMTYPE>NOMINAL</SEMTYPE>
                  <TYPE>Token</TYPE>
                  <FEATURE>kind</FEATURE>
                  <RANGE from="-5" to="5"/>
            </ATTRIBUTELIST>

            <ATTRIBUTELIST>
                  <NAME>Lemma</NAME>
                  <SEMTYPE>NOMINAL</SEMTYPE>
                  <TYPE>Token</TYPE>
                  <FEATURE>root</FEATURE>
                  <RANGE from="-5" to="5"/>
            </ATTRIBUTELIST>

            <ATTRIBUTELIST>
                  <NAME>Gaz</NAME>
                  <SEMTYPE>NOMINAL</SEMTYPE>
                  <TYPE>Lookup</TYPE>
                  <FEATURE>majorType</FEATURE>
                  <RANGE from="-5" to="5"/>
            </ATTRIBUTELIST>

            <ATTRIBUTELIST>
                  <NAME>EntityType</NAME>
                  <SEMTYPE>NOMINAL</SEMTYPE>
                  <TYPE>Entity</TYPE>
                  <FEATURE>type</FEATURE>
                  <RANGE from="-5" to="5"/>
            </ATTRIBUTELIST>

            <ATTRIBUTE>
                  <NAME>Class</NAME>
                  <SEMTYPE>NOMINAL</SEMTYPE>
                  <TYPE>Resistance</TYPE>
                  <FEATURE>kind</FEATURE>
                  <POSITION>0</POSITION>
            </ATTRIBUTE>

            <ATTRIBUTE>
                  <NAME>Class</NAME>
                  <SEMTYPE>NOMINAL</SEMTYPE>
                  <TYPE>DrugName</TYPE>
                  <FEATURE>rule</FEATURE>
                  <POSITION>0</POSITION>
                  <CLASS/>
            </ATTRIBUTE>
      </DATASET>

</ML-CONFIG>
```

Figure 17: XML Based Configuration File Example

Bibliography

Peter D. Stenson et al. (2003). Human Gene Mutation Database (HGMD): 2003 update. *Human Mutationn* , 577-581.

A. Statnikov, C. A. (2005). A comprehensive evaluation of multicategory classification methods for microarray gene expression cancer diagnosis. *Bioinformatics* , 631-643.

Andersson, D. I. (2003). Persistance of antibioitc resistant bacteria. *Current Opinion in Microbiology* , 452-456.

Andersson, D. I. (2006). The biological cost of mutational antibioitc resistance: any practical conclusions? *Current Opinions in Microboilogy* , 461-465.

Baral, C. G. (2007). CBioC: beyond a prototype for collaborative annotation of molecular interactions from the literature. *Computational Systems Bioionformatics Conference.* San Diego, CA: Life Sciences Society.

Burges, C. J. (1998). A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery* , 121-167.

Cohen, L. H. (2006). Biomedical Language Processing: What's Beyond PubMed. *Molcular Cell , 21* (5), 589-594.

Conly, J. (2002). Antimicrobial resistance in Canada. *CMAJ , 167* (8).

Corney, D. P. (2004). BioRAT: Extracting Biological Information from Full-length Papers. *Bioinformatics* , 3206-3213.

H. Cunningham, D. M. (2002). GATE: A Framework and Graphical Development Environment for Robust NLP Tools and Applications. . *Proceedings of the 40th Anniversary Meeting of the Association for Computational Linguistics (ACL'02).* Philadelphia.

Hans-Peter Kriegel, et. al. (2007). Future trends in data mining. *Data Mining and Knowledge Discovery , 15* (1), 87-97.

Hepple, M. (2000). Independence and Commitment: Assumptions for Rapid Training and Execution of Rule-based Part-of-Speech Taggers. *Proceedings of the 38th Annual Meeting of the Association for Computational Linguistics (ACL-2000).* Hong Kong.

Humphreys., R. G. (1996). *GATE User Guide.* Retrieved March 2009, from GATE-A General Architechture for Text Engineering: http://gate.ac.uk/sale/tao/split.html

J. L. Martinez et al. (2009). A gobal view of antibiotic resistance. *FEMS Microbial Review , 33*, 44-65.

Jose Luis Martinez, et al. (2009). A global view of antibiotic resistance. *FEMS Microbiological Review* , 44-65.

K. Alsabti, S. R. (1998). An efficient k-means clustering algorithm. *Proceedings of the First Workshop on High Performance Data Mining.* Orlando, FL.

Ken-ichiro Fukuda et al. (1998). Toward Information extraction: identifying protein names from biological papers. *Proceedings of the Pacific Symposium on Biocomputing*, (pp. 705-716). Maui, HI.

Kim Kyungduk, S. Y. (2005). POSBIOTM/W: a development workbench for machine learning oriented biomedical text mining system. *Proceedings of HLT/EMNLP on Interactive Demonstrations* (pp. 36 - 37). Vancouver, British Columbia, Canada: Association for Computational Linguistics.

Kohonen, T. (1998). The self-organizing map . *Neurocomputing* , 1-6.

Levin, D. I. (1999). The biological cost of antibiotic resistance. *Current Opinion in Microbiology* , 489-493.

Levy, M. N. (2007). Molecular Mechanisms of Antibacterial Multidrug Resistance. *Cell* , 1037-1050.

Lynette Hirschman, J. C. (2002). Accomplishments and challenges in literature data mining for biology. *Bioiformatics Review , 18*, 1553-1561.

M. Kanehisa, e. a. (2008). KEGG for linking genomes to life and the environment. *Nucleic Acids Research* , 480-484.

Martinez, J. L. (2008). Antibiotics and Antibiotic Resistance Genes in Natural Environment. *Science* , 365-367.

McKusick-Nathans Institute of Genetic Medicine, John Hopkins University, and National Center for Biotechnology Information. (2009). *Online Mendelian Inheritance in Man, OMIM*. Retrieved May 2009, from OMIM: http://www.ncbi.nlm.nih.gov/omim/

Muller HM, K. E. (2004). Textpresso: an ontology-based information retrieval and extraction system for biological literature . *PLoS Biology* .

N. J. Belkin, W. B. (1992, December). Information filtering and information retrieval: two sides of the same coin? *Communications of the ACM* , pp. 29-38.

P. Yue, E. M. (2006). SNPs3D: candidate gene and SNP selection for association studies. *BMC Bioinformatics , 7*.

P.S. Bradley, U. M. (1999). Data Mining: Overview and optimization opportunities. *INFORMS Journal on Computing* , 217–238.

Pedro Larranaga, e. a. (2005, July 29). Machine learning in bioinformatics. *Briefings in Bioinformatics* , pp. 86-112.

Pop, B. L. (2008). ARDB - Antibiotic Resistance Genes Databases. *Nucleic Acids Research* , 1-5.

Quinlan, J. R. (1996). Bagging, boosting, and C4.5. *Proceedings of the Thirteenth National Conference on Artificial Intelligenece* .

Settles, B. (2005). ABNER: An Open Source Tool for Automatically Tagging Genes, Proteins, and Other Entity Names in Text. *Bioinformatics* , 3191-3192.

Syed Toufeeq Ahmed, D. C. (2005). IntEx: A Syntactic Role Driven Protein-Protein Interaction Extractor for Bio-Medical Text. *Proc. of ISMB BioLINK SIG: Linking Literature* (pp. 54-61). Detroit, Michigan: Association for Computational Linguistics.

Tan, A.-H. (1999). Text Mining: The state of the art and the challeneges. *Proceedings of the PAKDD Workshop*, (pp. 65-70).

U. Fayyad, P. S.-S. (1996). Knowledge Discovery and Data Mining: Towards a Unifying Framework. *Proceedings of the second international conference on Knowledge Discovery and Data Mining.* Portland, Oregon: AAAI.

U.S. National Institute of Health (NIH) . (2009, April 7). *PubMed Central*. Retrieved May 2009, from PubMed Central: http://www.pubmedcentral.nih.gov/index.html

Vapnik, V. (1999). An overview of statistical learning theory. *IEEE Transactions on Neural Networks , 10* (5), 988-999.

Wright, C. T. (2005, February 9). Introduction: Antibiotic Ressitance. *Chemical Reviews* .

Y. Li, K. B. (2005). SVM Based Learning System For Information Extraction. . *Deterministic and Statistical Methods in Machine Learning, LNAI* , pp. 319-339.

Ying Zhao, G. K. (2005). Hierarchical Clustering Algorithms for Document Datasets. *Data Mining and Knowledge Discovery , 10* (2), 141-168.

Yoshimasa Tsuruoka, J. T. (2008). FACTA: a text search engine for finding associated biomedical concepts. *Bioinformatics* , 2559-2560.

Z. Lacroix, L. R. (2004). Techniques for Optimization of Queries on Integrated Biological Resources. *Journal of Bioinformatics and Computational Biology , 2* (2), 375.