

Célia Talma Martins de Pinho Valente Oliveira
Gonçalves

A Tool for Text Mining in Molecular Biology Domains



Universidade do Porto

Faculdade de Engenharia

FEUP

Departamento de Engenharia Informática
Faculdade de Engenharia da Universidade do Porto
Prof. Doutor Eugénio Oliveira
Prof. Doutor Rui Camacho
Abril 2013

Célia Talma Martins de Pinho Valente Oliveira
Gonçalves

A Tool for Text Mining in Molecular Biology Domains



Universidade do Porto

Faculdade de Engenharia

FEUP

*Tese submetida à Faculdade de Engenharia da
Universidade do Porto para obtenção do grau de Doutor
em Engenharia Informática*

Departamento de Engenharia Informática
Faculdade de Engenharia da Universidade do Porto
Prof. Doutor Eugénio Oliveira
Prof. Doutor Rui Camacho
Abril 2013

To my daughter, Edna, truly my most lovely dream.

To my husband, Litos, my everlasting eternal love.

To my parents: Maria Isaura and José de Pinho Valente.

Especially to honor the memory of my dear late Father, the most wonderful Father in the world.

Acknowledgments

At the end of my thesis I would like to thank all the ones who made this thesis possible and an unforgettable experience for me.

It is with deep gratitude that I acknowledge my supervisors: Prof. Eugénio Oliveira and Prof. Rui Camacho, whom I genuinely respect and admire. I would like to thank Prof. Eugénio Oliveira, for the constant support, motivation and patience. I also would like to express my gratitude to Prof. Rui Camacho for the constant guidance, support and invaluable help during the course of this research work.

Many thanks to my friend, Prof. Nuno Fonseca, who gave the embryonic idea of this thesis and was always available.

I wish to thank all the LIACC-NIAD&R colleagues that have accompanied me.

Last but not least, I am infinitely grateful to my parents, brother and sister, for their unconditional love and support. A very special word to my Mother and Father, for being present in all moments of my life, and for the unconditional love and support they gave me all of my life and also for encouraging me to pursue my studies.

A word to my sister, Elsa Marina, who was always a friend, a sister and a mother, whenever I needed her. The day I finished my masters she was the first one to encourage me to pursue a PhD degree. My heartfelt thanks to her for encouraging me. I cannot forget her daughters, my two loving nieces: Joana Aléxia and Diana Zénia whom I love very much, whose love and joy make my life much better.

A word to my brother, Zé Manuel, that looked after Edna together with my mother, whenever it was necessary, since she was born.

A word to the memory of my grandmother Hermínia because it was she who taught me to write, read and make my first calculations in primary school.

A word to my godmother, Maria Ursulina who also taught me how to enjoy studying.

Of course that my most deep word of care goes to the love of my life, my husband Litos, to whom I cannot find words to express my gratitude, for being an amazing husband and so dedicated father. He has accompanied this thesis since it has begun, and has helped me with his deep knowledge in programming. Thank you for the continuous encouragements and for always believing in my skills. I could not have accomplished it without you.

Another most deep word of love goes to my loving daughter, Edna Sofia, that is the most beautiful and loving treasure of my life, my sweet little angel. She was my precious gift at the end of the thesis. Her smile and her joy make my life truly meaningful. Both of you are my daily source of inspiration.

The last word goes to an endless care for the most special Father in the world, who

lives forever in my heart. How I miss you, how I wish you were here! This work has been carried out always remembering the words of encouragement you gave me throughout my life. All this work was meant for you. We will all be together once again one day...

English text reviewed by Ricardo Tavares (ricardoltav@gmail.com)

Abstract

Researchers need to be constantly aware of the work that has been done in their research area. Nowadays, most of the publications are available on the Internet. However there is, usually, an overwhelming amount of information making it impossible for a researcher to be aware of all this available information. Accessing the relevant information through the traditional keyword search engines still results in a huge list of publications to read, that usually have a large number of irrelevant publications. To tackle this problem, research in Text Mining and Information Retrieval has been applied to identify the most relevant publications out of the enormous amount of documents made available in the Internet.

Molecular Biologists have some routine tasks, that we believe may be automatically accomplished through the application of Machine Learning techniques. We have identified one of those tasks. Given a set of genomic or proteomic sequences, return a set of related sequences and a set of papers with information relevant to the study of such sequences. To properly implement this task we have to solve two main problems: to fetch a set of relevant papers; to sort by relevancy the papers resulting from the previous stage.

In this thesis we are proposing a novel method of Information Retrieval, based on Machine Learning techniques, to address the problem of retrieving relevant papers from MEDLINE. We have developed a new Information Retrieval methodology involving the dynamic construction of a classifier in real time for classifying MEDLINE papers. The methodology works as follows. A set of papers, associated with a set of sequences of interest are retrieved from the NCBI database. A data set is constructed using the NCBI retrieved papers, taken as “positive examples” and a set of equal number of papers randomly sampled from MEDLINE, taken as the “negative examples”. The negative examples are constrained to share MeSH terms with the positives ones. This data set is used by a Machine Learning algorithm to induce a classifier. The induced classifier is used to retrieve from MEDLINE a set of relevant papers. Since the retrieved set of papers is usually very large, a ranking of the set is performed (the second step). To address this second problem we are proposing a multi-criteria ranking function. We have used a new methodology to evaluate it automatically. The ranking function is a weighted combination of MeSH terms, number of citations, author’s h-index, author’s number of publications, journal impact factor and journal similarity factor where the original sequences were published.

A web-based search tool was fully implemented integrating all the scientific contributions mentioned.

Resumo

Os investigadores necessitam de estar constantemente informados do trabalho efetuado na sua área de investigação. Atualmente, a maioria das publicações estão disponíveis na Internet. Porém, existe normalmente, uma enorme quantidade de informação que torna impossível para um investigador estar a par de toda a informação disponível. Aceder à informação relevante através dos tradicionais motores de pesquisa baseados em pesquisa de palavras-chave resulta numa enorme lista de publicações para o investigador ler com um elevado número de publicações irrelevantes. No sentido de resolver este problema, a investigação em *Text Mining* e Recuperação de Informação (Information Retrieval) tem sido aplicada para identificar as publicações mais relevantes de entre a gigantesca quantidade de documentos disponíveis na Internet.

Investigadores da Biologia Molecular têm habitualmente um conjunto de tarefas rotineiras, que acreditamos serem susceptíveis de automatização através da aplicação de técnicas de Aprendizagem Computacional (Machine Learning). Identificamos uma destas tarefas. Dado um conjunto de sequências genómicas ou proteómicas, devolver um conjunto de sequências relacionadas e um conjunto de artigos com informação relevante para o estudo dessas mesmas sequências. Para resolvermos esta tarefa tivemos de solucionar dois problemas principais: encontrar e retornar um conjunto de artigos relevantes; ordenar por relevância este mesmo conjunto de artigos.

Nesta tese propomos um novo método de Recuperação de Informação baseado em técnicas de Aprendizagem Computacional, para solucionar o problema de recuperar artigos relevantes da MEDLINE. Desenvolvemos uma nova metodologia de Recuperação de Informação que envolve a construção dinâmica de um classificador em tempo real para classificar os artigos da MEDLINE. A metodologia funciona do seguinte modo. Um conjunto de artigos associado a um conjunto de sequências de interesse é recuperado da base de dados do NCBI. É construído um *data set* usando os artigos recuperados, que constituem os “exemplos positivos” e é gerado um conjunto de artigos escolhidos de forma aleatória da MEDLINE, que constituem os “exemplos negativos”. Os exemplos negativos partilham obrigatoriamente *MeSH terms* com os exemplos positivos. Este *data set* é usado pelos algoritmos de Aprendizagem Computacional para construir um classificador. Este classificador é depois usado para recuperar um conjunto de artigos relevantes da MEDLINE. Como normalmente o conjunto dos artigos recuperados é enorme, é efectuada a ordenação deste conjunto (segundo passo do processo). No sentido de resolver este segundo problema propomos uma função de

ordenação multi-critério. Usamos uma nova metodologia para sintonizar esta função de forma automática. A função de ordenação proposta é uma combinação ponderada dos seguintes factores: *MeSH terms*, número de citações, h-index do autor, número de publicações do autor, o factor de impacto da revista e o factor de similaridade da revista onde as sequências originais foram publicadas.

Foi totalmente implementada, uma ferramenta baseada na Web que integra todas as contribuições científicas mencionadas.

Résumé

Les chercheurs ont besoin d'être renseignés en permanence sur le travail développé dans le domaine de leur recherche. Á l'heure actuelle la majorité des publications sont disponibles sur Internet. Cependant, il existe en général une tellement grande quantité d'information qu'un chercheur ne peut pas en être au courant.

Accéder à l'information révélant par le biais de traditionnels moteurs de recherche basés sur mots-clés donne comme résultat d'énormes listes de publications à lire par le chercheur mais avec un nombre très élevé d'exemplaires sans intérêt ou importance. Pour résoudre ce problème, la recherche en Extraction de Données (*Text Mining*) et Récupération d'Information (Information Retrieval) a été parfois appliquée pour l'identification des publications ordonnées par degré d'importance parmi le gigantesque amoncellement de documents disponibles sur Internet.

Des chercheurs en Biologie Moléculaire ont d'habitude un ensemble de tâches routinières, que nous croyons être susceptibles d'automatisation avec l'application de techniques d'Apprentissage Computationnelle (Machine Learning). Étant donné un ensemble de séquences génomiques ou protéomiques, développer un ensemble de séquences en rapport et un ensemble d'articles pertinents pour l'étude de ces séquences mêmes. Pour résoudre cette tâche nous avons dû trouver la solution pour deux problèmes principaux: trouver et retourner un ensemble d'articles pertinents; ordonner par pertinence cet ensemble d'articles lui-même.

Dans cette thèse nous proposons une nouvelle méthode de Récupération d'Information basé sur des techniques d'Apprentissage Computationnelle, dans le but de résoudre le problème de récupérer des articles pertinents de MEDLINE. Nous avons développé une nouvelle méthodologie de Récupération d'Information qui implique la construction dynamique d'un classificateur en temps réel pour classer les articles de MEDLINE. La méthodologie fonctionne de la façon suivante: un ensemble d'articles associé à un ensemble de séquences d'intérêt est récupéré de la banque de données du NCBI. Un "data set" est construit avec les articles récupérés, lesquels constituent les "exemples positifs" et il est généré aussi un ensemble d'articles choisis de MEDLINE d'une façon aléatoire, lesquels constituent les "exemples négatifs". Les exemples négatifs partagent obligatoirement MeSH terms avec les exemples positifs. Ce "data set" est employé par les algorithmes d'Apprentissage Computationnelle pour construire un classificateur. Ce classificateur est ensuite employé pour récupérer un ensemble d'articles pertinents

de MEDLINE. Comme normalement l'ensemble d'articles récupérés est énorme, la mise en ordre de cet ensemble est effectuée (second pas du procédé). Dans le sens de résoudre ce second problème, nous proposons une fonction de mise en ordre multicritère. Nous employons une nouvelle méthodologie pour accorder cette fonction d'une façon automatique. La fonction de mise en ordre est une combinaison pondérée des facteurs suivants: MeSH terms, nombre de citations, h-index de l'auteur, nombre de publications de l'auteur, le facteur d'impact de la revue et le facteur de similarité des revues où les séquences originelles ont été publiées.

Un outil basé sur la Web a été totalement développé, intégrant toutes les contributions scientifiques mentionnées.

Contents

	i
Abstract	v
Resumo	vii
Résumé	ix
Contents	xiv
Tables Index	xvii
Figures Index	xx
1 Introduction	1
1.1 Context and Problem	1
1.2 BioTextRetriever	3
1.3 Research Questions	5
1.4 Thesis Objectives	5
1.5 Key Contributions	6
1.6 Structure of the thesis	7
2 Information Retrieval and Text Mining	9
2.1 Information Retrieval	9
2.1.1 Boolean Model	12

2.1.2	Vector Space Model	13
2.1.3	Probabilistic Model	15
2.1.4	Performance Evaluation of an Information Retrieval System . . .	17
2.2	Text Mining	18
2.2.1	Pre-Processing	20
2.3	Algorithms for Text Mining	21
2.3.1	Document Indexing	23
2.3.2	Classifier Learning	24
2.3.2.1	Support Vector Machine	24
2.3.2.2	Naïve Bayes Classifier	26
2.3.2.3	K-Nearest Neighbor	27
2.3.2.4	Rocchio Algorithm	28
2.3.2.5	Decision Trees	29
2.3.3	Ensemble Classifiers	29
2.3.4	Inductive Logic Programming	31
2.3.5	Classifier Evaluation	32
2.4	Tools for Information Retrieval and Text Mining	34
2.4.1	General Tools for and Text Mining	34
2.4.2	Tools for Bioinformatics	38
2.5	Ranking Methodologies	48
2.6	Summary	51
3	A tool for Text Mining in Molecular Biology's Domains	53
3.1	BioTextRetriever Overview	53
3.2	The MEDLINE	55
3.3	Local DataBase	56
3.3.1	Extensions to the available Information	58
3.4	Pre-Processing MEDLINE data	59

3.5	Assessing the Pre-Processing Techniques	63
3.6	How to use BioTextRetriever	66
3.7	Summary	70
4	From Sequences to Papers	71
4.1	Constructing a data set	71
4.1.1	Complementing the data sets	72
4.1.1.1	Original sequences characterization	75
4.1.2	Assessing the method of choosing irrelevant papers	76
4.1.3	Classifier Construction Process	79
4.2	Evaluating the Alternatives	82
4.2.1	Alternative 1	82
4.2.1.1	ILP Results	84
4.2.1.2	Comparing propositional and ILP results	85
4.2.2	Alternative 2	86
4.2.3	Alternative 3	87
4.2.4	Alternative 4	88
4.2.5	Alternative 5	89
4.3	Comparing the five alternatives	89
4.4	Summary	91
5	Ranking MEDLINE	93
5.1	The global procedure for the Ranking Function	93
5.2	The Ranking Function	96
5.2.1	Number of MeSH terms	97
5.2.2	Author's Number of Publications	98
5.2.3	Number of citations	99
5.2.4	h-index	100
5.2.5	Journal Impact Factor	102

5.2.6	Journal Similarity Factor	104
5.3	Choosing the Ranking Function Coefficients	105
5.3.1	Experimental Settings	106
5.4	Summary	110
6	Conclusions and Further Research	113
6.1	Thesis Overview	113
6.2	Further Research	115
	References	117
	A Annexes	133
	Index	139

List of Tables

2.1	Confusion Matrix	33
2.2	Machine Learning algorithms used in the study.	35
2.3	Summary of tools for generic text mining applications	38
2.4	Summary of tools for bio text mining applications	46
2.5	Summary of tools for bio text mining applications (cont).	47
3.1	LDB characterization	57
3.2	Machine Learning algorithms used in the study.	64
3.3	Pre-processing combinations and data sets characterization	65
3.4	Results summary table.	66
4.1	Characterization of data sets of the NMV type.	75
4.2	Characterization of data sets of types MRV and RV. In MRV the data sets of Table 4.1 were completed with randomly selected papers. In RV all "negative examples" are randomly selected.	76
4.3	Machine Learning algorithms used in the study.	76
4.4	Accuracy results (%) in NMV data sets. In all data sets the best value isn't different from the second best value in a statistically significant way, according to the t-student test ($\alpha = 0.05$) [SAC07]. The t-student test is a widely used method for comparing the means of two samples.	77
4.5	Accuracy results (%) in MRV data sets. In all data sets the best value isn't different from the second best value in a statistically significant way, according to the t-student test ($\alpha = 0.05$).	77
4.6	Accuracy results (%) in RV data sets. In all data sets the best value isn't different from the second best value in a statistically significant way, according to the t-student test ($\alpha = 0.05$).	78

4.7	Algorithm's Accuracy (%) average for NMVs, RMVs and RVs. There is statistical significance (using t-student test ($\alpha = 0.05$)) between the best and second best methods.	78
4.8	Characterization of data sets used to assess the 5 alternatives.	82
4.9	Accuracy results (%) in data sets. In all data sets the best value isn't different from the second best value in a statistically significant way, according to the t-student test ($\alpha = 0.05$).	83
4.10	Prolog predicates used.	84
4.11	ILP results over the same data sets used with the propositional algorithms.	85
4.12	Comparing ILP and propositional results using accuracy (%).	85
4.13	Ranking of the algorithms in comparison.	86
4.14	Ensemble's Accuracy results in alternative 2. The bold values are statistically different from the second best values according to the t-student test ($\alpha = 0.05$).	87
4.15	Alternative 3 Results	88
4.16	Alternative 4 Results.	88
4.17	Alternative 5 Results.	89
4.18	Best accuracies achieved in each alternative.	89
4.19	Comparing the five alternatives ranking position	90
5.1	The table shows an example of four papers associated with the input sequences.	98
5.2	The α value represents the devaluation coefficient in decreasing order of the paper's scientific age.	100
5.3	Example of h-index computation for h=4 in this case.	101
5.4	Example of four publication journals of four papers associated to the input sequences.	104
5.5	Characterization of data sets regarding the number of attributes and the number of positive and negative examples.	107
5.6	Characterization of data sets used to tune the coefficients of the ranking function. The column number six represents the total number of relevant papers classified as relevant by BioTextRetriever. The last column represents the percentage of relevant papers classified as relevant by BioTextRetriever.	107

5.7	Best combinations results for the fourteen data sets described in Table 5.6. <i>C1</i> represents the coefficient weight for the number of MeSH terms; <i>C2</i> represents the coefficient weight for the number of citations; <i>C3</i> represents the coefficient weight for the author h-index; <i>C4</i> represents the coefficient weight for the impact factor; <i>C5</i> represents the coefficient weight for the number of publications and <i>C6</i> represents the coefficient weight for the Journal Similarity Factor.	110
5.8	Individual combination results for the fourteen data sets described in Table 5.6 for the three combinations presented in Table 5.7.	110
A.1	Accuracy Classification results. Values with an attached '*' were obtained interrupting Weka after 3 months of execution. In each cell is the accuracy and the standard deviation in parenthesis. Bold values are the best results for each data set. The value for cross validation is 10. The last column presents the average of accuracy of the all algorithms for each data set.	135
A.2	True Positive Classification results. Values with an attached '*' were obtained interrupting Weka after 3 months of execution".In each cell is the true positives and the standard deviation in parenthesis. Bold values are the best results for each data set. The value for cross validation is 10. The last column presents the average of true positives of the all algorithms for each data set	136
A.3	F-Measure Classification results. In each cell is the f-measure and the standard deviation in parenthesis. Bold values are the best results for each data set. The value for cross validation is 10. The last column presents the average of f-measure of the all algorithms for each data set.	137

List of Figures

1.1	An example of a DNA structure (U.S. National Library of Medicine) (a) and protein (b).	4
2.1	Sketch of a typical Information Retrieval process	10
2.2	Precision and Recall measures	17
2.3	Text Mining Process	19
2.4	Decision Matrix	21
2.5	The general Text Classification process	22
2.6	Hyperplane that separates two classes	25
3.1	Sequence of steps implemented by BioTextRetriever	54
3.2	Web Page where the user specifies a new request.	67
3.3	Web page where each user can monitor his requests.	68
3.4	Web Page with links for each of the twenty results presented.	69
3.5	Web Page with one of the relevant papers.	70
4.1	Collecting the relevant papers. Irrelevant papers can either be: i) associated with less similar sequences or; ii) randomly collected from MEDLINE papers that have MeSH terms in common with the relevant ones.	72
4.2	Generating a data set with relevant and irrelevant papers.	72
4.3	How “not relevant” (but ”close”) papers are obtained. ϵ is the e-value threshold provided to discriminate the sequences that determine the relevant papers. α and β are parameters for the cut off limiting the “near-misses”.	73

4.4	The accuracy average of the three proposed methods: Near-Miss Values (NMV), MeSH Random Values (MRV) and Random Values (RV). . . .	78
4.5	Alternative 1 consisting in the use of a single classifier applied to the whole data.	80
4.6	Alternative 2 considers the use of an ensemble when basic classifiers are built using the whole data.	80
4.7	Alternative 3 makes a partition on the data set and each sub set of the data is used to train a specific classifier. The set of classifiers are combined in an ensemble.	81
4.8	Alternative 4 makes a partition on the data set and each sub set of the data is trained with all the learning algorithms (T classifiers) and in the end it is made an ensemble of the T*T classifiers.	81
4.9	Alternative 5 makes a partition on the original data set and for each sub set a different ensemble is used with the best basic algorithms obtained in alternative 1. In the end, an ensemble is made of the M ensembles.	82
4.10	Average accuracies of the algorithms in comparison.	86
4.11	Comparing the five alternatives accuracies average.	91
4.12	Comparing the five alternatives position average.	91
5.1	Experimental setting for tuning the ranking function	94
5.2	Construction of the MEDLINE sample.	96
5.3	Extracts the MeSH terms associated with the relevant papers associated with the sequences.	98
5.4	Extracts the journals associated to the relevant papers associated with the sequences.	104
5.5	Information items stored in the LDB to be used in the ranking function. The Journal Impact Factor is the only one not calculated through the local copy of MEDLINE's available information, but is downloaded from the Web of Knowledge website (Thomson Reuters) and is saved in LDB.	105
5.6	Procedure to evaluate the ranking function	109
A.1	Local Database structure.	134

Chapter 1

Introduction

1.1 Context and Problem

It is of capital importance for every researcher to be aware of the work that has been done in his research area. With the advent of the Internet, the amount of information available to everyone is usually overwhelming. One might think that researchers could now get easy access to all related work. There is, however, a very difficult problem that needs to be solved before we reach that situation. Accessing the right information amidst the overwhelming amount of documents available in the Internet is quite difficult, in most cases.

The volume of research publications, in almost all areas of knowledge, has been growing at a phenomenal rate. Nowadays, most publications are available on the Internet, some completely and some partially. The overload of research publications can hinder researchers due to the time spent looking for the real “interesting/relevant” publications. Usually, to find these publications a researcher uses the traditional keyword-based search engines and, as a result, faces a huge list of publications to read with a large number of irrelevant publications. To tackle this problem, research in Text Mining and Knowledge Extraction has been applied to literature mining to help researchers to identify the most relevant publications out of a great amount resulting from simple search strategies.

Although the traditional search engines are quite useful for specific queries, when applied to more complex searches they have strong limitations. For this reason, scientists have focused their attention on Text Mining techniques (information retrieval, information extraction and data mining) as a way to solve this problem. Text Mining helps gathering, maintaining, interpreting and discovering knowledge needed for research in a efficient way. By adding meaning to text, these techniques produce a more structured analysis of textual knowledge than simple word searches [AKT06].

Besides this, information can be found in heterogeneous forms:

- Structured information (databases)
- Unstructured information (text)
- Semi-structured information (XML)

Text Mining can be applied to many areas of research such as literature, journalism, biology, biomedicine, among others.

In this thesis we propose the use of Text Mining to address the information overload problem by automating the process of extracting the relevant parts of the scientific literature. This automation may increase the efficiency of searching for information and allow automated inference of new information.

Another current research topic and subfield of Text Mining is Text Categorization (also known as Text Classification), the task of automatically sorting a set of documents into categories (classes or topics) from a predefined set [Seb05].

The goal of text mining for biologists is to aid researchers in Biology's domains to identify relevant information in a more efficient way by having computers analyse the literature.

Biologists need software that is reliable and can deal with huge amounts of data, as well as interfaces that facilitate human-machine interactions.

Some actual research on text mining and text categorization applications are [BH09], [RHS12], [AZ12]:

- Document Organization;
- Text Filtering;
- Hierarchical Categorization of Web Pages;
- Word Sense Disambiguation;
- Spam Filtering;
- Information Extraction (classify extracted parts of texts);
- Information retrieval: search engines may classify documents to be relevant or non-relevant for a given query, based on user feedback;

We are interested in selecting scientific papers (title and abstracts) that may be in different formats. A rich resource to mine is MEDLINE, a database of citations and abstracts of articles published in major peer reviewed journals since the 1950s, that is available for download. For the scope of the present work we are only considering the paper's abstracts.

One additional problem of Text Mining and Information Retrieval is that the answer not only depends on the query but the answer may also differ according to both the user and the query. In these cases it is not advisable to use Machine Learning

techniques and, in particular, to use classifiers. It is only justifiable to build a classifier if the task is repetitive. The idea is to identify a problem that biologists use frequently and that needs Text Mining and also justifies the construction of a classifier. One of these problems is their regular task of consulting repositories of information to search for similar information.

Another problem is the use of sophisticated and efficient tools that require training and sometimes to be familiar with the techniques used. A biologist is not interested in spending time learning how to use a tool, so the tool must be intuitive, user friendly and efficient in obtaining the desired results.

The major aim of this dissertation is the production of such a tool to help biologists to find relevant information more efficiently. We have developed a Web based search tool to find relevant literature associated with a set of genomic or proteomic sequences, based on Text Mining.

1.2 BioTextRetriever

Within the thesis work we have developed a Web based search tool to find relevant literature in the context of a set of genomic or proteomic sequences, based on Text Mining. With this aim in mind, we have applied Machine Learning techniques to automatically train a classifier that learns, for each set of sequences given by the user, which are the relevant related papers.

BioTextRetriever can handle two types of sequences: DNA (DeoxyriboNucleic Acid) or RNA (Ribonucleic Acid) sequences and Protein sequences. A DNA sequence that determines the genetics of a living organism, is a succession of the letters A, C, T and G that representing the 4 nucleotide bases (A-Adenine, C-Cytosine, G-Guanine, T-Thymine)¹ [SSL10]. The DNA sequences are represented, in the FASTA format, as a sequence of the letters A, C, G and T: “*CATTCCATGGTCCCGCAGCCCCAG...*”. An example of a DNA sequence can be seen in Figure 1.1 (a). A protein sequence is a living chain of aminoacids (21 possible aminoacids) and an example is represented in Figure 1.1 (b).

¹In RNA the letter T is replaced by U-Uracil

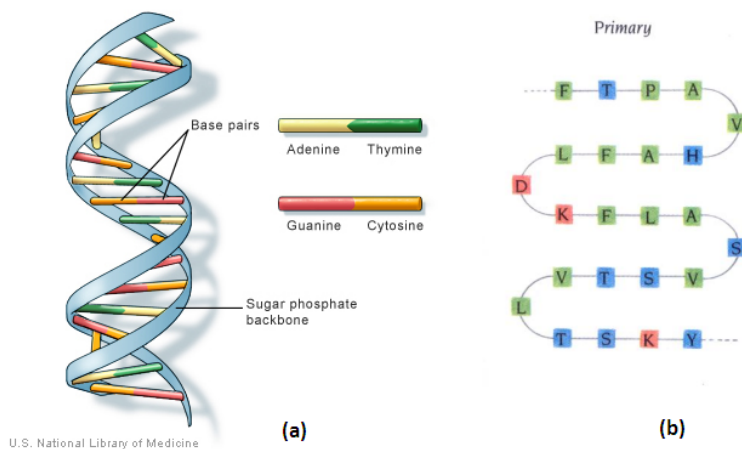


Figure 1.1: An example of a DNA structure (U.S. National Library of Medicine) (a) and protein (b).

To implement our prototype we have started with a case study based on a set of sequences suggested by a Biology expert.

We have started our research by analyzing scientific papers (abstracts). For the aim of this thesis we have only used the title and abstracts of the scientific papers available in MEDLINE 2010.

The most widely used retrieval tool in Molecular Biology is Entrez [JHGJ96], [ent], the PubMed Information Retrieval System provided at the US National Center for Biotechnology (NCBI) [WBB⁺06]. PubMed Central (PMC) is the U.S. National Institutes of Health (NIH) free digital archive of biomedical and life sciences journal literature which is the database we will use. The PubMed database maintained by the National Center for Biotechnology Information (NCBI) is a key resource for biomedical science, and it is our first base of work. The NCBI's PubMed system is a widely used method for accessing MEDLINE. PubMed employs a Boolean search strategy in which users enter search terms and logic operators (AND, OR, NOT) to retrieve documents from MEDLINE. The expressive power of such logic experiences is not enough to avoid an overload of documents in the answer to queries.

The first step is to recover useful information from the PubMed, i.e., to search for papers that include references to similar sequences to those that were introduced by the biologist (in the FASTA format) and return all the references of papers found in the PubMed database that are related to those sequences. With the help of NCBI BLAST we will search in the PubMed database for similar sequences, and gather all the references to papers related to these sequences (here we will use the e-value introduced by the biologist as a cut off value).

Besides PubMed we could also use Ensembl. Ensembl [ens] is a genome browser that is manually curated, its aim is to provide annotation for the biological community that is freely available and of high quality.

Some Information Extraction techniques were applied to the papers' abstracts and title related to the similar sequences found in PubMed, such as: tokenization, stop words removal and stemming among others.

The next step is to apply machine learning techniques to train a classifier that learns to identify papers related to a particular sequence, based on the training set, returned from the extracted papers at PubMed.

One of the main purposes of this thesis is to search, select and rank bibliographic information that is potentially related to a set of sequences S . Afterwards, the purpose is to apply our classifier to all the papers in our local copy of the MEDLINE database. The main idea of this particular Text Mining tool is to automate and accelerate the biologist's routine tasks. At this stage we are able to evaluate the obtained results and present them in our interface to the biologist.

As a final step we have evaluated the impact of our proposed tool.

1.3 Research Questions

The main question that guided this thesis is "Is it possible to construct an automatic web-based tool that given a set of sequences returns an ordered and relevant set of scientific papers? "

To pursue the main question, the following research questions were derived from this main one, through the course of this thesis research:

- What are the best pre-processing techniques to apply to MEDLINE papers?
- Can Machine Learning contribute to the improvement of the Information Retrieval process?
- Which Machine Learning algorithms performed well for the Information Retrieval of MEDLINE papers?
- Is there a good ranking criteria to order by relevance the retrieved MEDLINE papers?

1.4 Thesis Objectives

The following objectives and consequent work developed through the course of this thesis were to answer and prove the above research questions.

The **general and main objective** of the present thesis is:

- The development of a tool that automates the search for relevant documents in Molecular Biology's domains using Text Mining techniques given a set of genomic or proteomic sequences.

The more specific objectives of this thesis, that derive from the main general objective, are:

- To propose and evaluate a methodology for efficient Information Retrieval pre-processing techniques to apply to MEDLINE.
- To include and assess the use of Machine Learning algorithms in the Information Retrieval process.
- To develop and assess a multi-criteria ranking function for the retrieved papers.
- Make the tool user friendly and fully automated.

1.5 Key Contributions

In the quest for those objectives, our developed thesis work led us to the main contributions that can be summarized as follows:

- **Contribution 1** An Information Retrieval methodology involving the dynamic construction of a classifier in real time for categorizing MEDLINE papers.

We have proposed and evaluated a new architecture for Information Retrieval that involves the use of Machine Learning. In this proposal we have derived three ways of producing a data set starting with the papers associated with the sequences. We have also derived and assessed several ways of partitioning the data set and combining the Machine Learning algorithms in order to achieve a good performance in the classification process.

- **Contribution 2** The proposed ranking function was evaluated with a new methodology that enabled us to automate the assessment process. The ranking function is a weighted combination of: MeSH terms, article number of citations, author's h-index, author's number of publications, journal impact factor and journal similarity factor.
- **Contribution 3** We have, experimentally, evaluated the combination of several pre-processing techniques for the MEDLINE set of papers.
- **Contribution 4** A web-based tool for retrieving relevant literature in Molecular Biology and related domains given a set of genomic or proteomic sequences. We implemented a web-based tool (proof-of-concept) that integrates the scientific contributions 1, 2 and 3 mentioned above.

1.6 Structure of the thesis

The remaining of the thesis is organized in five chapters.

Chapter 2 presents the state-of-the-art in the fields highlighted in this thesis: Information Retrieval, Text Mining and Classification methods and tools for Text Mining and Ranking methodologies.

Chapter 3 presents the BioTextRetriever architecture. A Web-based search tool for retrieving relevant literature in Molecular Biology's Domain and provides a detailed description of the pre-processing techniques to be applied to the paper's title and abstracts.

Chapter 4 addresses the construction of a classifier capable of selecting among the MEDLINE papers the relevant ones associated with a set of sequences. This chapter provides the methodology to construct the classifier as well as a set of experiments that support the choices made.

Chapter 5 presents the ranking function developed to rank BioTextRetriever results to present to the end user. The evaluation of the developed function is based on a set of experiments detailed herein.

Finally, Chapter 6 presents the thesis' conclusions and points out some further research.

Chapter 2

Information Retrieval and Text Mining

We will present the State-of-the-Art in the fields of Information Retrieval, Text-Mining, Biology Text Mining, Classification Algorithms and Ranking Methodologies.

2.1 Information Retrieval

Before being analyzed, any document has to be fetched from a repository. When the repository has a large amount of documents, the task of retrieving the most relevant is not trivial. We now survey different techniques that have been suggested for the retrieval of relevant documents from large repositories. Information Retrieval deals with representation, storage, organization of, and access to information items such as documents, Web pages, etc [ByRN99]. The retrieved documents aim at satisfying a user information need, usually expressed in natural language, that is, given a collection of documents, find the useful information corresponding to a user's query [MC00]. The Information Retrieval process can be described as in the Figure 2.1.

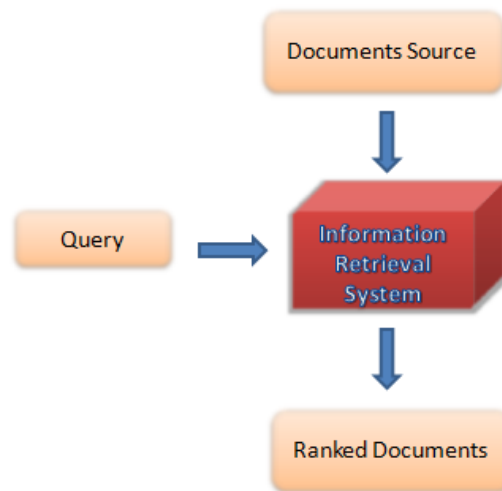


Figure 2.1: Sketch of a typical Information Retrieval process

The wide variety of forms, in which information can be stored and communicated makes Information Retrieval a challenging task. Other several issues make Information Retrieval a challenging task [MC00]:

- the information in the document database is typically unstructured;
- documents are usually written in unconstrained natural language; and
- very often, the documents cover a wide range of subjects.

Some challenges of Information Retrieval can be summarized as:

- Process large document collections efficiently;
- Handle more flexible matching;
- Handle ranked retrieval;
- Handle unformatted data:
 - Textual data: papers, technical reports, web pages, etc;
 - Non-textual data: images, graphics and videos.

An Information Retrieval model specifies how a document and a query are represented and how the relevance of a document to a user query is defined. A retrieval model consists of [Jär07].:

- a representation for documents;

- a representation for queries;
- a retrieval function (a ranking or similarity function which orders the documents with respect to a query, i.e., an ordering of the documents retrieved that reflect the relevance of the documents to the user query).

The classical Information Retrieval models are [ByRN99]:

- Boolean Model;
- Vector Model;
- Probabilistic Model.

This classic Information Retrieval models consider that a document is represented (described) as a set of keywords. These keywords are called index terms. A document is therefore represented as a list of index terms. This approach is also called bag-of-words, where a document is seen as an unordered list of words.

In a recent past, experts on various topics had the task of assigning an appropriate set of keywords to an article. Scientific articles usually have a 3 or 4 keywords representative of the content of the paper. This is called manual indexing. Although manual indexing is more reliable, it's time consuming, and requires a profound knowledge of the classification system. Nowadays, Information Retrieval systems use automatic methods for indexing documents.

However, the different index terms of a document usually have varying relevance when used to describe (summarize) the document contents. To decide the importance of a term for describing (summarizing) the content of a document is a difficult task. This effect is captured through the assignment of weights to each index term of the document.

Some models present ranked results to the user. A ranking is an ordering of the documents retrieved that reflect the relevance of the documents to the user query. A ranking or similarity function or similarity measure can consider several information:

- string comparison;
- probability that documents arise from the same model;
- same terms used;
- same meaning of text.

A similarity measure is a function that computes the degree of similarity between a pair of vectors. Given that both documents and queries can be seen as vectors, a similarity measure represents the similarity between two documents or between a

document and a query.

The literature proposes several similarity measures. The most usual are inner vector product and cosine similarity.

The similarity between a document d_i and a query q can be computed as the inner vector product: $sim(d_i, q) = \sum_{k=1}^t (d_{ik} \cdot q_k)$ where d_{ik} is the weight of term k in document i and q_k is the weight of term k in the query.

The Cosine Similarity measures the cosine of the angle between two vectors. The normalized inner product of the two vectors' length is calculated as:

$$CosSim(d_i, k) = \frac{\sum_{k=1}^t (d_{ik} \cdot q_k)}{\sqrt{\sum_{i=1}^t d_{ik}^2} \cdot \sqrt{\sum_{i=1}^t q_k^2}}$$

We will now detail the classic Information Retrieval Models.

2.1.1 Boolean Model

The first model to appear in the fifties was the Boolean Model. This model is based on Boolean Algebra and uses the operators of George Boole's mathematical logic: AND (the logical product), OR (the logical sum) and NOT (the logical difference).

In this model a document is represented as set of keywords, i.e., a set of terms. A term either is present or not present in a document. So, the index terms' weights are all binary (0,1). A query is a Boolean expression of terms, which may be combined using the Boolean operators: AND, OR and NOT. A document is predicted as relevant to a query if it satisfies the query expression and non-relevant otherwise [ByRN99].

From the literature we can point out some advantages of the Boolean Model [Hie01], [VA10]:

- relatively simple to compute;
- it is easy for the user to understand why a document was retrieved;
- it is easy to understand if the query was too specific (few results) or too broad (many results).

Some disadvantages of the Boolean Model pointed out in the literature are [Hie01], [VA10]:

- difficult from the user's perspective because the query language is complicated, so it is difficult to express some complex user queries;

- very rigid: AND means all; OR means any; the use of binary weights is too limiting;
- all matched documents (that satisfy the query) will be returned, which makes hard to control the number of documents retrieved (too many or too few);
- there is no ranking of the retrieved results;
- all terms are equally important;
- it is slow in big collections;
- terms in a document are considered independent of each other.

2.1.2 Vector Space Model

The Vector Space Model (also known as “bag-of-words“) was introduced by Salton in 1965 [SWY75] and it is the most common modern retrieval model due to its simplicity and effectiveness. The Vector Space Model improves the Boolean Model by removing the limitation of binary weights for index terms. The Vector Space Model is based on geometric algebra. Documents and queries are represented as vectors in a high dimensional space. Queries are a kind of a document, so it can be represented in the same form. A term (word) is a sequence of characters that does not contain any spaces or punctuation.

Each term defines one dimension of the vector. N-terms defines a high-dimensional space. The elements of the vector correspond to term weight, which denotes the importance of term i in the document.

Documents contain a variable number of terms. The idea is to create a vector with the different terms and assign weights to each term.

A distance measure between the query and documents is necessary to rank retrieved documents. Thus we need to calculate the terms’ weights in the document and query representation. Index term weights can be calculated in many different ways. The most used term weights are:

TF: Term frequency

The more frequently a term occurs in a document, the better it describes the document. The idea is that a term is more important, if it occurs more frequently in a document. Term frequency $tf_{t,d}$ indicates how many times a term t occurs in a document d . Document frequency df_i indicates in how many documents a term occurs in document i .

TF-IDF: Term Frequency-Inverse Document Frequency

In Inverse Document Frequency (IDF), for weighting a term it is considered more important if it occurs only in a few documents. Salton defined [Sal89], the Term Frequency Inverse Document Frequency (TF-IDF) as follows:

TF(term, document) = frequency of term in document

$$IDF(term) = \log \frac{\text{number of documents in collection}}{\text{number of documents with term}} + 1$$

$$TF\text{-}IDF(\text{term, document}) = TF(\text{term, document}) * IDF(\text{term})$$

Another way of calculating IDF is:

Inverse document frequency $idf_t = \log \frac{N}{df_t}$, where N is the total number of documents and df_t is the total number of documents where term t occurs.

$$\text{Tf-idf weighting: } tf - idf_{t,d} = tf_{t,d} * idf_t$$

Tf-idf calculates a weight that is directly proportional to the number of occurrences of a term in a document.

Documents are weighted and ranked, in order of similarity to the query based on a measure of distance. Documents that are close together in the vector space talk about the same things.

The distance between vector d_1 and vector d_2 is captured by the cosine of the angle between them. So, relevance is measured by the distance between the query vector and document vector in the vector space. The similarity between a document and a query is measured by calculating the similarity between the two vector representations.

The similarity between the document and the query can be calculated according to the formula [ZSY06]:

$$Sim(\text{document, query}) = \sum_{\text{all query terms}} a * b \quad \text{where}$$

$$a = \text{weight of term in query}$$

$$b = \text{weight of term in document}$$

Some approaches normalize the weight of a document, and the most common approach is cosine normalization [ZSY06] and [W.R03]:

$$\frac{\sum_{\text{all query terms}} \text{weight of term in query} * \text{weight of term in document}}{\sqrt{\sum_{\text{all query terms}} (\text{weight of term in query})^2} * \sqrt{\sum_{\text{all query terms}} (\text{weight of term in document})^2}}$$

There are other variations to the Vector Space Model. Okapi weighting is based on the Poisson distribution [RW94]. Pivot normalization [SBMM96] is a TFIDF document weighting variation.

The advantages of Vector Space Model found in the literature are stated now [VA10]:

- easy to implement;
- widely used;
- many TF-IDF variants;
- provides partial matching which allows retrieval of documents that approximate the query conditions;
- provides ranked results;
- the degree of matching can be used to rank-order (how well a document satisfies a user's information needs) documents;
- improved performance over the Boolean Model because of the weighting scheme

The disadvantages of the Vector Space Model found in the literature are [VA10]:

- assumes that terms are independent (however some terms in a document are related to each other);
- the order of the words in a phrase is not considered;
- each vector is very sparse;
- the semantic of terms is not considered.

The Vector Space Model is used by the web search engines.

2.1.3 Probabilistic Model

The probabilistic model is also known as the Binary Independence Retrieval Model (BIR), because all weights of index terms are binary $\in \{0, 1\}$ and index terms are independent.

In the Statistical Model, a document is usually represented as a “bag of words” (unordered words with frequency).

Probability retrieval models are based on the called Probability Ranking Principle [RW94] where the retrieval system should return retrieved documents in a ranked list in the decreasing order of probability. Probability models can be characterized as methods of estimating the probability of relevance of documents to the user query.

According to [ByRN99]:

Let R be the set of documents known to be relevant. Let \bar{R} be the complement of R (e.g., the non-relevant documents). The similarity between document d_j to the query q , can be defined as $sim(d_j, q) = \frac{P(R|\vec{d}_j)}{P(\bar{R}|\vec{d}_j)}$

Using Baye's Rule, $sim(d_j, q) = \frac{P(\vec{d}_j|R)*P(R)}{P(\vec{d}_j|\bar{R})*P(\bar{R})}$

$P(\vec{d}_j | R)$ Stands for the probability of randomly selecting the document d_j from the set R of relevant documents. Where $P(R)$ and $P(\bar{R})$ are the same for all the documents in the collection

$$sim(d_j, q) = \frac{P(\vec{d}_j|R)}{P(\vec{d}_j|\bar{R})}$$

Assuming the independence of terms $sim(d_j, q) \sim \frac{\prod_{g_i(\vec{d}_j)=1} P(K_i|R)*\prod_{g_i(\vec{d}_j)=0} P(\bar{K}_i|R)}{\prod_{g_i(\vec{d}_j)=1} P(K_i|\bar{R})*\prod_{g_i(\vec{d}_j)=0} P(\bar{K}_i|\bar{R})}$

Taking logarithms the expression becomes:

$$sim(d_j, q) \sim \sum_{i=1}^t W_{i,q} * W_{i,j} * \left(\log \frac{P(K_i|R)}{1-P(K_i|R)} + \log \frac{1-P(K_i|\bar{R})}{P(K_i|\bar{R})} \right)$$

In the beginning there are no retrieved documents so some assumptions can be made:

- $P(K_i | R)$ is constant for all the terms (0.5)
- The distribution of index terms among the non-relevant documents can be approximated to by the distribution of index terms among all the documents in the collection $P(K_i | R) = 0.5$ and $P(K_i | \bar{R}) = \frac{n_i}{N}$

Where n_i is the number of documents which contain the index term k_i and N is the total number of documents in the collection. The advantages of the Probabilistic Model found in the literature are mentioned now [Hie01], [VA10], [SC99].

- documents are ranked in decreasing order of probability of relevance (similarity to the query).

The disadvantages of this model pointed out in the literature are [Hie01], [VA10], [SC99]:

- need to guess initial estimates for $P(K_i | R)$;
- all weights are binary;
- assumes independence of index terms.

2.1.4 Performance Evaluation of an Information Retrieval System

Information Retrieval systems performance are usually evaluated by two measures: Precision and Recall [ByRN99] that can be viewed in Figure 2.2.

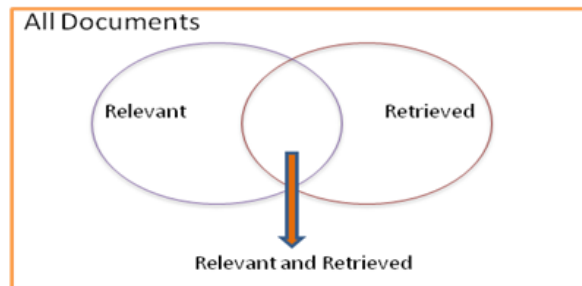


Figure 2.2: Precision and Recall measures

Precision: the percentage of retrieved documents that are in fact relevant to the query (i.e., “correct” responses).

$$Precision = \frac{\{Relevant\} \cap \{Retrieved\}}{\{Retrieved\}}$$

Recall: the percentage of documents that are relevant to the query and were, in fact, retrieved.

$$Recall = \frac{\{Relevant\} \cap \{Retrieved\}}{\{Relevant\}}$$

These two measures: Precision and Recall are inter-dependent measures. Recall increases if we return a higher number of documents. However, precision decreases if the number of retrieved documents increases. A system that returns all documents has 100% Recall.

Precision and Recall are widely used and summarize the behaviour of an Information Retrieval System and evaluate the effectiveness of information retrieval systems [RJB89]. One disadvantage is that it is not always possible to calculate Recall because it requires the knowledge of the total number of relevant items in the collection which usually is not possible to be aware of. The more the collection size grows the more difficult is to calculate recall. In [CR96] the authors state that it is “impossible to assume how many relevant items there are for a particular query in the huge and ever changing Web systems”.

2.2 Text Mining

The aim of text mining is to automatically extract and discover knowledge hidden in the text. Text Mining or Knowledge Discovery from Text (KDT)¹ deals with machine supported analysis of texts.

Citing Hearst, “another way to view text data mining is as a process of exploratory data analysis that leads to heretofore unknown information, or to answers for questions for which the answer is not currently known” [Hea99].

“Text Mining also known as Text Data Mining or Knowledge Discovery from textual databases, refers to the process of extracting interesting and non-trivial patterns or knowledge from text documents ... Text Mining is a multidisciplinary field, involving information retrieval, text analysis, information extraction, clustering, categorization, visualization, database technology, machine learning and data mining.” [hT99].

Text Mining involves the application of techniques from areas such as information retrieval, machine learning, statistics, computational linguistics, and data mining [Hot05].

“Text mining applications integrate a broad spectrum of heterogeneous data resources, providing tools for the analysis, extraction and visualization of information, with the aim of helping biologists to transform available data into usable information and knowledge” [KEV05].

In these methods, a collection of pre-categorized documents is used to train a statistical model of word or phrase and then the statistical model is applied to uncategorized documents.

The Text Mining Process can be divided [EZ02] into the following stages (see also Figure 2.3):

¹Mentioned for the first time by [RI95]

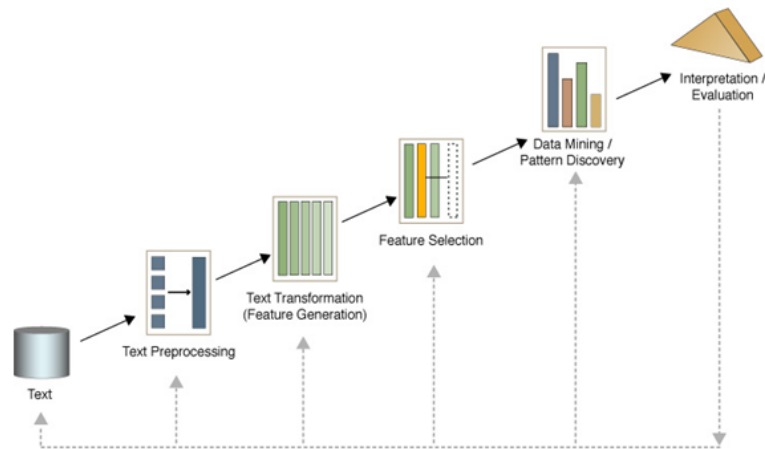


Figure 2.3: Text Mining Process

- Text preprocessing
 - Syntactic/Semantic text analysis
- Features Generation
 - Bag of words
- Features Selection
 - Simple counting
 - Statistics
- Text/Data Mining
 - Classification - Supervised learning
 - Clustering - Unsupervised learning
- Analyzing results

Documents may have many different origins and may be stored also in a wide variety of formats: HyperText Markup Language (HTML), Portable Document Format (PDF), Extensible Markup Language (XML), Microsoft Word (.doc) format, MIME-encoded email messages, plain text and so on.

For mining a huge collection of documents, first it is necessary to pre-process the text documents, and store all the information in a data structure, convenient for further processing.

2.2.1 Pre-Processing

The pre-processing step involves several steps and procedures that we will now summarize as the classic pre-processing steps.

Tokenization and removing unwanted characters

The first step is to remove unwanted characters such as HTML/XML tags and punctuation. Tokenization is the process of splitting a text document into a stream of words by removing punctuation marks and removing HTML tags [AMGG07]. A word must be within white spaces.

Stop Words Removal

The most used filtering method is the stop word filtering, which removes words that are meaningless (such as articles, conjunctions, prepositions, etc – a, the, for, this, etc).

Stemming

Stemming is the process of reducing a word to its root; splits the plural “s” from nouns, the “ing” from verbs, and other affixes [Hot05]. A stem is a natural group of words with equal or similar meaning. The Porter algorithm [Por80] is one of the most used for stemming.

The next step is to represent the document in a convenient way and to store all the information in a convenient data structure for further processing.

The most common way to represent a document is to use the Bag-of-Words approach where a document is represented as a vector of length n , where n is the number of terms in the document. However this approach ignores syntactic and semantic correlations between terms.

After applying the pre-processing only a subset of the entire collection of terms of the document will be used to describe a particular document. There are several methods for keyword selection that are described in Section 2.3.1.

Feature selection is a process that chooses a subset from the original set obeying to some criterions [LLCM03]. This subset has the same meaning as the original one, but it provides a better understanding for the data and the learning process.

According to [DGM⁺08] feature selection is the process of removing irrelevant features from the original data set. Feature selection is very useful because it reduces the dimensionality of the data to be processed by the classifier, reducing execution time and improving predictive accuracy. Feature selection studies how to select a subset of attributes that are used to construct models describing data [DDS07]. Its purpose

includes reducing dimensionality, removing irrelevant and redundant features, and algorithm predictive accuracy due to a lesser amount of data.

Now we can apply the algorithms for text classification.

Classification is a form of data analysis, and it can be used to extract models describing important data classes or make future predictions. The most known Classification methods are described in the next section.

2.3 Algorithms for Text Mining

“Text Categorization (also known as text classification or topic spotting) is the task of automatically sorting a set of documents into categories (or classes, or topics) from a predefined set” [Seb05].

Text Classification attempts to automatically determine whether a document or part of a document has particular characteristics of interest, usually based on whether the document discusses a given topic or contains a certain type of information [CH05].

Text Classification involves two main research areas: Information Retrieval and Machine Learning.

Text Classification is a necessity due to the very large amount of text documents that we have to deal with daily. Text Categorization or Classification, must be efficient, which implies, a short processing time as possible, because of the huge amount of documents to be classified.

Classification is the process of assigning the analyzed document to one or more pre-determined classes. A set of correctly preclassified documents trains the classifier. The Machine Learning approach relies on an initial corpus of Documents $D = \{D_1, D_2, \dots, D_n\}$ that are preclassified under categories $C = \{C_1, C_2, \dots, C_k\}$.

Document classification or categorization may be seen as the task of determining an assignment of a value from $\{0, 1\}$ to each entry of the decision matrix in Figure 2.4, where: [Seb99]

	d_1	d_j	d_n
c_1	a_{11}	a_{1j}	a_{1n}
...
c_i	a_{i1}	a_{ij}	a_{in}
...
c_m	a_{m1}	a_{mj}	a_{mn}

Figure 2.4: Decision Matrix

- $C = \{C_1, C_2, \dots, C_m\}$ is a set of pre-defined categories
- $D = \{D_1, D_2, \dots, D_n\}$ is a set of documents to be categorized
- A value of 1 for a_{ij} is interpreted as a decision to file d_j under category c_i
- A value of 0 for a_{ij} is interpreted as a decision not to file d_j under c_i

A text classifier is automatically generated by a general inductive process. This process infers the characteristics that any document should have to be classified under each category by observing the characteristics of a set of pre-classified documents.

According to [Seb99], in the Machine Learning approach a general inductive process automatically builds a classifier for a category c_i by “observing” the characteristics of a set of documents that have previously been classified manually under c_i by a domain expert; from these characteristics the inductive process gleans the characteristics that a novel document should have in order to be categorized under c_i .

Text Classifier techniques includes probabilistic methods, regression methods, decision trees, neural networks, support vector machines, genetic algorithms, hidden Markov models, among others.

Automated text classification represents the process of assigning labels (the labels for each category are predefined) to new documents based on the knowledge accumulated in the training process [ZIA02]. That is why building a text classifier needs a training set. Both the training and the testing sets must have the documents associated with one or multiple categories. Once the classifier is built, the training set, its effectiveness is determined by comparing the class labels found by the classifier with those already assigned to the testing set.

Resuming, the text classification process needs a set of pre-classified documents for training the classifier and a set of pre-classified documents for testing the effectiveness of the classifier.

Text Classification Process Figure 2.5, can be seen as the following general sequence of steps:

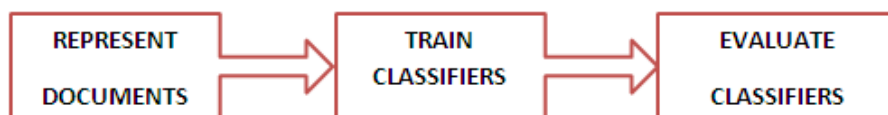


Figure 2.5: The general Text Classification process

According to Fabrizio Sebastiani [Seb05] we can divide the Text Categorization in three phases: document indexing, classifier learning and classifier evaluation.

2.3.1 Document Indexing

Document Indexing deals with how to represent a document.

Although text is stored in machine readable formats such as HTML, XML, PDF, DOC, PostScript, etc., it is not a suitable form of input for most learning algorithms [LTSL07].

We cannot give a text directly to a classifier without a previous pre-processing and indexing procedure. So, document indexing maps a document into a compact representation of its content that can be directly interpreted by a classifier algorithm.

Document Indexing [SW09] describes a document through a set of terms called “index terms” that indicate what the document is about (i.e., summarizes its content).

Text documents must be transformed to match the learning methods input format. The first step in Text Categorization is to transform documents (strings of characters) into a representation suitable for the classifier.

There are different ways to understand what a term is and different approaches to compute term weights.

Most of the learning algorithms use attribute-value representation, which means we have to transform them into a vector space.

Text categorization usually uses a vector model representation of the documents. The vector that represents the document contains the documents terms and also the weights assigned to each term. Support vector machine and K-nearest neighbor are two machine learning approaches where documents are represented as a vector and where each component is associated with a particular word of the document.

The representation of a document d_j is done as a vector of term weights $d_j = W_{i1} \dots W_{|T|j}$ where T is the set of terms (features) that occur at least once in at least one d_j document of T [Seb05]. Each term in a document vector must be associated with a value (weight) which denotes their different importance in the text.

The words of a text are not equally indicative of its meaning. Term weights reflect the (estimated) importance of each term. To each vector component is assigned a value related to the estimated importance (some weight) of the word in the document. Traditionally, this weight is assigned using the Information Retrieval measures TFIDF. The weighting assignment phase, is defined as the assignment of a real number, that lies between 0 and 1, to each keyword and this number indicates the imperativeness of the keyword inside the document. Different methods have been developed and the most widely used model is the tf-idf weighting factor [FB91]. This weight of each keyword is computed by multiplying the term factor (tf) with the inverse document factor (idf) where:

F_{ik} = occurrences of term t_k in document D_i

$Tf_{ik} = \frac{f_{ik}}{\max(f_{ik})}$ normalized term frequency occurred in document

$Df_k = \log \frac{d}{df_k}$ where d is the total number of documents and df_k is number of documents that contains the term t_k

$W_{ik} = tf_{ik} * idf_k$ term weight, the computed W_{ik} is a real number $\in [0, 1]$

The document indexing commonly uses the bag-of-words representation, which is the most common way to represent the content of a document (text). After pre-processing a document or a set of documents, a learning algorithm is used to learn how to classify the documents.

However before applying the classifier, the documents are pre-processed to find a good subset of features. This problem of finding a “good” subset of features is called feature selection. Some feature selection methods that can be applied and constitute some of most common pre-processing techniques:

- Tags removal;
- Stop Words Removal: eliminate non content words (such as “the”, “a”, “for”, etc);
- Stemming: reducing a word to it’s root thus reducing the number of distinct words;
- Pruning infrequent words: words are only considered as features, if they occur at least a few times in the training data.

2.3.2 Classifier Learning

There are several methods (algorithms) for this phase of learning a classifier. Here, will be presented the most common methods and algorithms used in Text Categorization: Support Vector Machine, Naïve Bayes Classifier, K-Nearest Neighbor, Rocchio Algorithm, Decision Trees, Ensemble Classifiers and Inductive Logic Programming.

2.3.2.1 Support Vector Machine

Support Vector Machine (SVM) is one of the machine learning techniques for Text Categorization. This learning model was proposed by [Vap99]. In SVM, documents are represented as points in a vector space, where the dimensions are the selected features. The basic idea of SVM is to find an optimal hyperplane based on the training document vectors, represented in Figure 2.6, to separate two classes with the largest margin from pre-classified data. After this hyperplane is determined, it can be used for classifying data into two classes based on which side they are located. Only a few of the training document vectors define the hyperplane, these are called the “support vectors”.

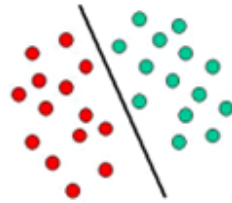


Figure 2.6: Hyperplane that separates two classes

According to [Joa98] SVMs are universal learners. In [Joa98], the author points out some of the reasons why SVMs work well for text categorization:

- High dimensional input space (complexity does not depend on the dimensionality of the feature space);
- Few irrelevant features;
- Document vectors are sparse;
- Most text categorization problems are linearly separable.

When applying SVM to Text Categorization, one basic feature is the words occurring in a training set, i.e., the main idea is that different kinds of documents contain different words and these word occurrences can be viewed as clues for document classification. [KRJ09] says that SVMs exploit statistical learning theory and are capable of overcoming problems associated with high dimensionality (overfitting).

From the literature ([AGOR11], [Tip00], [CWD03]), we can point out the advantages and weaknesses of using SVM. As advantages we can find the following ones:

- Can model real world complex problems
- SVM performs well on data sets that have many attributes, even if there are very few cases with which to train the model.

The weaknesses are:

- It only considers two classes;
- Sensitive to noise, i.e., a relatively small number of mislabeled examples can dramatically decrease the performance

In general, support vector machines accomplish quite high accuracies, if compared to other techniques. There are several applications of SVMs for Text Classification:

[SSKN07] This work describes a framework that allows evaluating SVMs for the categorization problem on a collection, in particular the applicability of SVMs for the Defense Technical Information Center (DTIC). As a result the authors concluded that SVM improves the existing DTIC process.

[DMd⁺03] PreBIND is an information extraction system based on SVM for the detection of protein-protein interactions in the literature. [RNS05] developed a machine learning approach based on SVM to mine protein function predictions from text.

2.3.2.2 Naïve Bayes Classifier

The Naïve Bayes is a simple probabilistic supervised learning classifier based on the the Bayes Theorem and it has strong independence assumptions. Naive Bayes assumes that the existence of a class feature does not depend on the existence of the other features. The Naïve Bayes algorithm requires a set of training documents already classified to create the learning model. After that every set of new documents will be classified based on the probability of belonging or not to a predefined class.

This algorithm calculates the $P(C_k | D)$, the probability that document D belongs to the class C_k . According to Bayes' theorem $P(C_k | D) = \frac{P(C_k) * P(D|C_k)}{P(D)}$

$$P(D | C_k) = \prod_i P(W_i | C_k)$$

From the literature we can point out the advantages and weaknesses of Naïve Bayes [BJ09], [RSTK03], [MAS06]. The advantages are:

- Explicit theoretical foundation (based on the Bayes theorem);
- Relatively effective;
- Very simple and easy to implement;
- Fast in training and classification;
- Robust against noisy data.

The disadvantages referred in the literature are:

- Multinomial model / independence assumption clearly wrong for text;
- Performs worse than other methods in practice;
- On some datasets it really fails badly.

The paper [IM05] presents a module for classifying Polish text, intended for use in automatic processing of job advertisements. Two classifying algorithms were implemented: Naïve Bayes and TFIDF algorithm. The classifier has been tested on a set

of Polish texts: job advertisements and other texts, taken from the Internet. As a conclusion the authors say that the Naïve Bayes classifier outperforms the TFIDF classifier. They also concluded that the use of other words other than nouns, verbs, adjectives and adverbs do not influence the result of the categorization significantly. The Naïve Bayesian classifier is robust and inherently resistant to noise [LS94].

2.3.2.3 K-Nearest Neighbor

The k-nearest neighbor (K-NN) classifier is a supervised learning algorithm. KNN classifies a new example by comparing it to all previous seen examples. The classification of the K most similar previous cases is used for predicting the classification of the current example.

The algorithm for K-NN is [Don03]:

- Given a test document:
 1. Find K's nearest neighbors among training documents;
 2. Calculate and sort score of candidate categories;
 3. Thresholding on these scores.

Decision rule

$$Y(\vec{x}, C_i) = \text{sign} \sum_{d_i \in KNN} \left(\text{sim}(\vec{x}, \vec{d}_i) Y(\vec{d}_i, c_j - b_j) \right)$$

$$Y(d_i, c_j) \in \{0, 1\}$$

$\text{sim}(\vec{x}, \vec{d}_i)$ the similarity between the test document x and the training document d_i
 b_j : category – specific threshold

From the literature we can point out the advantages and the weaknesses of K-Nearest Neighbor [YEH09]. The main advantages are:

- Is simple and intuitive;
- Widely used;
- Is among the top performing text categorization tools;
- Easy to implement (uses standard IR techniques, such as TFIDF).

The disadvantages are:

- Heuristic approach;

- It is difficult to determine the number of neighbors;
- It is computationally heavy for large data sets.

In the work presented in the paper [MSI08] the authors implemented the K-NN and Naïve Bayes algorithm in order to make a practical comparison between these two algorithms applied to the classification of Arabic Text. They conclude that K-NN had better performance than the Naïve Bayes. The corpus consists of 242 documents that belonged to 6 categories.

2.3.2.4 Rocchio Algorithm

The Rocchio Algorithm is an adaptation of the relevance feedback method developed in information retrieval. It uses standard TFIDF weighted vectors to represent documents, and builds a prototype vector for each category by summing up the vectors of the training documents in each category. Test documents are then assigned to the category that has the closest prototype vector, based on a cosine similarity. Rocchios' classifier has a fixed number of categories and known *a priori*. Each document is assigned to exactly one of them.

According to [MR07] the Rocchio classifier is a classifier based on tf-idf, initially proposed as a relevance feedback algorithm but also used in the area of text classification. Both keywords and documents are represented as vectors, and the closer a document \vec{d}_j is to the keyword vector \vec{w}_j the higher is the similarity between the document and the keyword in the vector space.

A document d is represented as $d_j = (d_1, d_2, \dots, d_T)$, where each dimension is the probability of term t_i in the document multiplied by the term's inverse document frequency $IDF(t_i)$, $d_i = P(t_i | D) * IDF(t_i)$.

The inverse document frequency is defined as the logarithm of the inverse of the probability of a term over the entire collection D , $IDF(t_i) = \log \frac{1}{P(t_i|D)}$.

The key idea of Rocchio's algorithm is to construct a so-called optimal query so that the difference between the average score of a relevant document and the average score of a non-relevant document is maximized [CKY⁺08].

The Rocchio technique builds for each category a single prototypical document. The category profile consists of a weighted list of words or terms formed from the word distribution within the category. To decide which categories a new document belongs to, its word distribution is compared to those of the prototypical category documents. When the similarity is high enough, the new document is assigned to the category in question.

The work referred in [SR02] presents refinements to improve the Rocchio algorithm which consist in including negative training examples, taking documents just outside

each category, and using them as negative indicators when computing prototypical document vectors [SR02].

In [FS07] the authors point that this is an algorithm extremely simple to implement and cheap computationally. Its performance, however, is usually mediocre, especially with categories that are unions or disjoints of clusters and in, general, with the categories that are not linearly separable.

2.3.2.5 Decision Trees

Decision tree learning is one of the most successful algorithms due to its various factors: simplicity, comprehensibility, and ability to handle mixed-type data. The main idea is to construct a tree with tests to discriminate between documents of different classes. A classification decision is a sequence of tests terminating in the assignment of a category corresponding to a leaf node of the tree.

Decision trees are constructed by analyzing a set of training examples for which the class labels are known. They are then applied to classify previously unseen examples. If trained on high-quality data, decision trees can make very accurate predictions [CNM04].

“Decision trees are sometimes more interpretable than other classifiers such as neural networks and support vector machines because they combine simple questions about the data in an understandable way. Decision trees naturally support classification problems with more than two classes and can be modified to handle regression problems. Finally, once constructed, they classify new items quickly” [KS08].

From the literature [RM], [PKSR02], we can point out the advantages and the weaknesses of the Decision Tree. The advantages are:

- Relatively easy to interpret by humans;
- Easy to implement;
- Very fast to train and evaluate.

The disadvantages are:

- Analyses one attribute at a time, so there is no way of detecting interaction between variables.

The best known implementation of Decision Trees is the C4.5 of Ross Quilan [Qui93].

2.3.3 Ensemble Classifiers

An ensemble is a collection of models whose predictions are combined by weighted averaging or voting. According to [Die00a] a necessary and sufficient condition for

an ensemble of classifiers to be more accurate than any of its individual members is if the classifiers are accurate and diverse. An ensemble of classifiers is a set of classifiers whose individual decisions are combined in some way (majority or voting) to classify new examples. The main objective of ensemble classifiers is to achieve a better performance than the constituent classifiers. The literature [OM99], [Dv04], [HSA10] refers that:

- Combining predictions of an ensemble is often more accurate than the individual classifiers that make them up
- The classifiers should be accurate and diverse
- An accurate classifier is one that has an error rate of better than random guessing (known as weak learners)
- Two classifiers are diverse if they make different errors on new data points

For performing the experiments we have used the WEKA tool available algorithm implementations: Bagging, AdaBoost and Ensemble Selection. In [SP04] the authors state that bagging and boosting are among the most popular re-sampling ensemble methods that generate and combine a diversity of classifiers using the same learning algorithm for the base-classifiers.

Bagging

Bagging (Bootstrap aggregating) was proposed by [Bre96] and its basic idea is to generate several classifiers from a training set. These classifiers are generated independently. Bagging generates several samples from the original training data set using bootstrap sampling [ET93] and then trains a base classifier from each sample whose predictions are combined by a majority vote among the classifiers.

AdaBoost

In AdaBoost [FS97] the performance of simple (weak) classifiers is boosted by combining them iteratively.

Boosting methods re-weight in an adaptative way the training based on the values of the previous base classifier. The boosting methods run several times on the training data being the data equally weighted and then the classifier iteratively decreases the weight of the corrected classified data and runs again the classifier. Boosting is also a method based on combining several different classifiers. The main differences between Bagging and Boosting are: the way instance samples are generated, and the way final classification is performed. In Bagging, the classifiers are generated independently from each other. The Boosting method uses a more refined way to sample the original training set, where the samples are chosen according to the accuracy of the previously generated classifiers. Each classifier generation takes into account the accuracy of the classifiers generated in the previous steps. According to [SP04] boosting algorithms

are considered stronger than bagging on noise free data. However, there are strong empirical indications that bagging is much more robust than boosting in noisy settings.

Ensemble Selection

Recently, ensemble selection [CNMCK04] was proposed as a technique for building ensembles from large collections of diverse classifiers. Ensemble selection uses more classifiers, allows optimizing to arbitrary performance metrics, and includes refinements to prevent overfitting to the ensemble training data a larger problem when selecting within more classifiers.

2.3.4 Inductive Logic Programming

Inductive Logic Programming (ILP) [Mug90] is a Machine Learning field that uses a subset of First Order Logic (FOL) to represent both data and models. ILP addresses the problem of inducing hypotheses (as predicate definitions) from examples and background knowledge. According to [LF01], an ILP learner requires an initial theory B (background knowledge) and some evidence E (examples), and induces a theory H (hypothesis) that together with B explains some properties of E . We therefore have three main ingredients in an ILP setting: background knowledge (B); examples (E) and hypotheses (H). Another ingredient to ILP is the examples. In traditional ILP examples there are two sorts: positive and negative.

Positive examples ($E+$) are instances of the concept to learn, whereas negative examples are not. Negative examples are used to avoid over generalization. All of these ingredients are represented in a subset of FOL, basically as Prolog programs. The background knowledge is an important feature in ILP over propositional learners since the Background Knowledge is a set of information (predicates/definitions) that the expert considers relevant for the construction of the hypothesis H . Since B is encoded as Prolog programs the expert may provide a wide range of useful information. This information may include grammars, dictionaries, ontologies, numerical algorithms etc. Since H is built taking B 's predicates as "basic blocks" we are able, in an ILP setting, to construct very complex models. The availability of B in ILP supports our view that ILP may perform well in Text Mining applications. We can provide the system with a lot of useful information and algorithms that a text analyzer may find useful. The use of such information is also possible in propositional learners, in most cases, but it is very hard to fit into a attribute-value table.

Due to the use of FOL to encode both data and B we can "easily" handle data with structure. These is another supporting feature for our proposal of using ILP in text analysis. The third ingredient, H , is also encoded in FOL and can therefore represent highly complex models. Traditional ILP systems transform the induction process into a search over a very large space (sometimes infinite) which may cause efficiency problems when dealing with complex problems. To address such a problem the user may constrain the language of H and use a set of parameters for that

purpose. ILP systems like Aleph [Sri04], April [FSC06] or Indlog [Cam00] have a highly powerful expressive language to verify the constraints mentioned above to constrain the hypothesis' language. ILP has relevant applications to problems in complex domains like natural language and molecular computational biology [Mug99]. ILP has successfully been applied to a variety of classification and prediction problems, such as the diagnosis of a patient or a plant disease [LD94].

There are many applications of ILP in medical diagnosis and protein prediction [LD94]. Broadly speaking ILP has applications in the following areas:

- Science: Protein shape prediction, drug structure activity prediction.
- Engineering: Finite element mesh design, satellite fault diagnosis, circuit design, automobile traffic flow analysis, intelligent software agents for Internet
- Language: Part of speech disambiguation from large real world text corpus, learning grammars

One of the potentials of ILP is learning from real-life examples. To use of ILP systems has several advantages such as: it has a powerful representation language; comprehensibility of results; the acceptance of extra-knowledge in a domain for the construction of models; it works with structured data and the models are intelligent; most of the ILP systems are available on the Web; wide applications domain [Cam07]. The advantage of ILP systems in Text Mining is the use of additional information such as dictionaries, statistics analysis, ontologies among others.

There are some applications of ILP on Text Mining, namely:

- Morphological disambiguation [DE00];
- Part-of-Speech Tagging [Cus97], [JdAL99];
- Corpus-based learning of semantic relations [SMNWH00];

2.3.5 Classifier Evaluation

Once a classifier has been built there is the need to measure its effectiveness. Useful measures of a system's quality evaluating results are Precision and Recall, that are used in Information Retrieval.

Precision (P) is the proportion of truly positive examples labeled positive by the system that were truly positive and Recall (R) is the proportion of truly positive examples that were labeled positive by the system [LTSL07]. According to [Seb99] Precision can be viewed as the "degree of soundness" and Recall may be viewed as its "degree of completeness". Neither Precision nor Recall make sense isolated. In fact,

usually higher levels of precision may be obtained at the expense of a low Recall and vice-versa.

$$Precision = \frac{\text{number of retrieved relevant documents}}{\text{total number of relevant documents}}$$

$$Recall = \frac{\text{number of retrieved relevant documents}}{\text{total number of retrieved documents}}$$

F-Measure combines Precision and Recall reflecting the relative importance of Precision versus Recall [ZSY06]:

$$F - Measure = \frac{2 * Recall * Precision}{Recall + Precision}$$

Breakeven point, is where Precision equals Recall.

Micro-Averaging: counts each document equally important.

Macro-Averaging: counts each category equally important.

A Confusion Matrix summarizes all important information, it is a visualization tool typically used in supervised learning. Each column of the matrix represents the classifier output (predicted class), while each row represents the actual class of the instances. One benefit of a confusion matrix is that it clearly shows whether the system is confusing two classes (i.e. mislabeling one as another).

	Classified as Positive	Classified as Negative
Is Positive	True Positive (TP)	False Negatives (FN)
Is Negative	False Positives (FP)	True Negative (TN)

Table 2.1: Confusion Matrix

TP (True Positive) – is the number of instances correctly classified as positive.
 TN (True Negative) – is the number of instances correctly classified as negative.
 FP (False Positive) – is the number of instances incorrectly classified as positive.
 FN (False Negative) – is the number of instances incorrectly classified as negative.

Most evaluation measures can be computed from the confusion matrix: precision, recall, sensitivity, specificity and accuracy [CLCF07]:

Precision (or Positive Predictive Value) = $\frac{TP}{TP+FP}$, is a measure that estimates the probability that a positive prediction is correct.

Recall (or True Positive Rate or Sensitivity) = $\frac{TP}{TP+FN}$, is the proportion of examples belonging to the positive class which were correctly predicted as positive; the proportion of positive classifier results among the relevant documents.

Specificity = $\frac{TN}{FP+TN}$ is the percentage of negative examples correctly predicted as negative; the proportion of negative classifier results among the irrelevant documents.

Classification accuracy is probably the most widely used performance measure amongst

the Machine Learning community [CNM04]. The idea is to determine the success rate of the classifier in classifying unknown instances. Accuracy [PMA⁺07] is the number of correctly classified instances divided by the total number of instances.

Accuracy can be calculated as $\frac{TP+TN}{TP+TN+FP+FN}$, which represents the proportion of correctly classified documents.

2.4 Tools for Information Retrieval and Text Mining

This section presents a survey of tools that have been developed for Text Mining activities. We pay special attention to the ones that have been developed for Text Mining in Molecular Biology applications.

2.4.1 General Tools for and Text Mining

We will present here some general and recent tools applied to Text Mining.

The WEKA tool (Waikato Environment for Knowledge Analysis) [WFT⁺99], [HFH⁺09] is a collection of Machine Learning algorithms implemented in Java developed at the University of Waikato, New Zealand. WEKA contains implementations of algorithms for classification, clustering, and association rule mining. Some of the advantages of using WEKA software are [FHH⁺10], [HFH⁺09], [BFH⁺10]:

- it is fully implemented in JAVA and thus is platform-independent;
- provides a wealth of state-of-the-art machine learning algorithms that can be deployed on any given problem;
- its graphical user interface is very easy to use (no data mining knowledge is necessary to use WEKA); however we did not use it, once it was embeded in our code.
- is open source and freely available;
- has widespread acceptance in both academia and business;

One of the WEKA's limitations is that the algorithms need to have all data in the main memory so big data sets are an issue. WEKA has a standard datafile format, called ARFF (Attribute-Relation File Format). It is an ASCII text file that consists of two distinct sections: header information and data. Table 2.2 lists some of the WEKA's algorithms considered in the experiments of this thesis.

We now provide a brief description of these algorithms.

Table 2.2: Machine Learning algorithms used in the study.

Acronym	Algorithm	Type
ZeroR	Majority predictor	Rule learner
smo	Sequential Minimal Optimization	Support Vector Machines
rf	Random Forest	Ensemble
ibk	K-nearest neighbors	Instance-based learner
BayesNet	Bayesian Network	Bayes learner
j48	Decision tree (C4.5)	Decision Tree learner
dtnb	Decision table/Naïve bayes hybrid	Rule learner
AdaBoost	Boosting algorithm	Ensemble learner
Bagging	Bagging algorithm	Ensemble learner
Ensemble Selection	Combines several algorithms	Ensemble learner

- ZeroR Classifier [WFT⁺99] it is a trivial classifier that predicts the majority class or the median (for the numeric values). ZeroR gives a lower bound on the performance of a given data set which should be the baseline performance to compare with other learning classifiers. ZeroR tests how well the class can be predicted without considering other attributes.
- The IBk classifier [AK91] is a k-Nearest Neighbour type classifier used by WEKA. IBK uses a vector space model to determine the distance between an entity pair and the the k closest entity pairs of a given classification.
- The Random Forest Classifier was developed by Breiman [Bre01]. This classifier combines individual decision trees into ensembles. In Random Forest each tree casts a vote for a particular class and the most popular class is selected as the output of the classifier. Each tree has the same weight in voting. This meta learner works very fast especially when large data sets are used but requires more memory than the other WEKA classifiers.
- J48 is a powerfull decision trees classifier. In fact, J48 is the C4.5 [Qui93] implementation for the WEKA tool.
- DTNB is a hybrid classifier that combines a decision table with Naïve Bayes.

The ensemble algorithms (AdaBoost, Bagging and Ensemble Selection) have been described in Section 2.3.3.

The General Architecture of Text Engineering (GATE²) [CMBT02] is an established text mining, open source, framework with an architecture for language processing, information extraction, ontology management and machine learning algorithms. GATE includes an information extraction system called ANNIE (A Nearly New Information

²GATE is freely available at <http://www.gate.ac.uk/>

Extraction System) which is a set of modules comprising a tokenizer (that produces token annotations), a gazetteer, a sentence splitter (that produces sentence annotations), a part of speech tagger, and a name entity recognition (which identifies different types of named entities and creates annotations for each type). GATE provides support not just for standard text mining applications, such as information extraction, but also for tasks such as building and annotating corpora, and evaluating the applications. The GATE architecture is based on components, which are reusable pieces of software that may be deployed in a variety of contexts. The components' structure and reusability are important features of GATE.

A general purpose framework that may be used in Text Mining applications is Rapid Miner [MWK⁺06] (formerly called YALE – Yet Another Learning Environment) which is a framework for Knowledge Discovery and Data Mining (KDD) and Machine Learning. In Data Mining and Knowledge Discovery applications before applying a learning method there is the need to provide the data with specific characteristics for the learning process thus improving the performance of the learning model. To achieve this it is usually necessary to apply pre-processing to the data. YALE allows to easily specify and execute data mining operator chains for pre-processing, especially feature generation and selection, and multistrategy learning. Real world data mining tasks are often solved by a sequence or combination of several data pre-processing and Machine Learning methods. In Rapid Miner, each such method is considered an operator. Rapid Miner is an open source data mining tool. It has flexible operators for data input and output (in different file formats such as Arff, C4.5, csv, excel files, SPSS files and datasets from databases). It includes several Machine Learning algorithms for regression, classification and clustering tasks. It has data preprocessing operators before the learning process. Again the existence of separate pieces for different pre-processing steps may reveal itself to be very useful in Text Mining Applications.

A web text mining process able to discover knowledge in a distributed heterogeneous multi-organization environment is presented by [AMGG07]. The web text mining process is based on a flexible architecture and is implemented in four steps able to examine web content and to extract useful hidden information through mining techniques. These four steps are: crawling, pre-processing, text mining and presentation of results. The first step foresees the recovery of the useful information from the web. This information consists in textual contents present in web pages. The second step is the pre-processing and foresees the creation of a repository of information from the web during the execution of the first step. The third step is the fundamental step on which the whole web mining process is applied. In this step Information Extraction techniques are applied, namely tokenization and lemmatization. The GATE tool (previously mentioned) is used in this step. Subsequent to the Information Extraction, the application of one or more rules on a Document or a Corpus will provide the extraction of “hidden information” that will be made available in a XML format. The fourth step foresees the presentation of the obtained results. The information drawn out during the execution of the third step is stored in a second repository. To fill this repository it is necessary to effect a parsing of the result in XML supplied by the

software during the third phase for the recovery of the necessary information.

Another Web-based tool, called JANE (Journal/Author Name Estimator) [SK08], is a freely available application that, based on a piece of text (e.g. the abstract of a paper), can suggest journals and experts who have published similar articles. JANE helps authors to find appropriate journals where they can publish their work in and editors to find potential reviewers. JANE returns an ordered list of results, with a confidence score for each item (e.g. journal or author). JANE uses the open source search engine Lucene [HG04] to find articles that are similar to the input query. Texts are tokenized using the standard Lucene tokenizer, and are subsequently compared using the Lucene MoreLikeThis algorithm (a very efficient implementation of the traditional TF*IDF vector space model)³. After retrieving the ordered list of most similar records, a weighted k-nearest neighbor approach is used to determine the journal or author list. The final results are ordered by a confidence score.

Lu et al., [ZZJ09] work helps researchers to quickly identify appropriate journals to read and publish in. They have developed a Web application for finding related journals based on the analysis of PubMed log data. A relevant point in their work is the ranking of the, usually journals in the set. Another advantage of the presented tool is that it is web-based and therefore available everywhere. The results are ordered with a measure determined by a journal's past usage.

The Unstructured Information Management Architecture (UIMA) [FL04] is a flexible and extensible architecture for the analysis and processing of unstructured data, especially text. The four main UIMA services are: acquisition, unstructured information analysis, structure information access and component discovery. This architecture facilitates the integration of different analysis tools. UIMA and GATE are similar in design and purpose: both represent documents as text plus annotations and allow users to define pipelines of processes that manipulate the document. The analysis of unstructured content by UIMA applications makes use of a variety of analysis technologies including those from statistical and rule based natural language processing, information retrieval, machine learning, ontologies, automated reasoning, and a diverse of semantic resources (e.g., WordNet, FrameNet, etc).

Apart from Rapid Miner which is a framework, GATE and UIMA, none of the above tools use Machine Learning algorithms able to learn dynamically new needs from the user. Most of them are accessible from the Web which substantially increases their use. Although using different criteria most of them sort the resulting list of papers. This is an important feature since, in most cases, the size of the answer set is huge.

We next present a summary of the presented tools highlighting the most important features for our work. Table 2.3 summarizes the tools presented above and highlights their major features.

³Described earlier in this thesis

	Web-based	Components Architecture	Machine Learning	Use of “background” \ extra Information	GUI
WEKA	No	Yes	ML	No	Yes
GATE	Yes	Yes	ML	Yes (<i>NER</i>)	Yes
RAPID MINER	Yes	Yes	ML	No	Yes
[AMGG07]	Not Available	Yes	ML (GATE)	Yes	No
JANE	Yes	Unique Piece	Yes(K-NN)	No	Yes
Lu et al.	Yes	Unique Piece	No	No	Yes
UIMA	Yes	Yes	ML	Yes	Yes

Table 2.3: Summary of tools for generic text mining applications

2.4.2 Tools for Bioinformatics

“The goal of biomedical text mining is to allow researchers to identify needed information more efficiently, uncover relationships obscured by the sheer volume of available information, and in general shift the burden of information overload from the researcher to the computer by applying algorithmic, statistical and data management methods to the vast amount of biomedical knowledge that exists in the literature as well as the free text fields of biomedical databases ”[CH05]. We now present a set of tools specially designed for Text Mining applied to bioinformatics problems.

MedMiner⁴ [LUL⁺99] is a web-based tool, which filters and organizes large amounts of textual and structured information returned from PubMed or GeneCards (a database of human genes). The aim of MedMiner is to facilitate biologist researchers in their daily literature search, enabling them to collect specific biomedical facts from a large amount of documents, and allowing them to find biological entities in the texts. The MedMiner results page of a query presents summary statistics on the number of abstracts and sentences found that match the search. The keywords are highlighted and a link to the unfiltered abstract is provided. Basically MedMiner identifies sentences from MEDLINE citations where the user’s specified terms and relations are highlighted. Relationship words are from a relatively large lexicon of such terms predefined by the system. The authors identify a weakness of the tool, that is, the required list to identify more general concepts. If this list is not given to the tool, then the tool will miss relevant concepts. Besides MedMiner requires a relevance keyword list so the user must have a prior knowledge of the possible interactions between the two genes. MedMiner uses the traditional keywords search which may be very limiting for some user information needs.

BioMinT⁵ [bio] is an information retrieval and extraction tool for biomedical literature. The goal of the BioMinT project is to develop a generic text mining tool that assists manual annotation by: a) interpreting diverse types of query; b) retrieving relevant

⁴MedMiner is freely available at <http://discover.nci.nih.gov/>

⁵<http://www.biomint.org/>

documents from the biological literature; c) extracting the required information and d) providing the result as database or as a structured report. The BioMinT tool searches the literature and automatically extracts information from abstracts and papers in order to provide two essential research support services: a curator's assistant, accelerating by partially automating the annotation and update of biological databases (database annotation); and a researcher's assistant, generating readable protein family reports in response to queries from biological researchers. BioMinT presents the most relevant articles first.

PubMiner (Publication Text Mining system) [EZ04] is a machine learning based text mining system for mining PubMed abstracts to extract named entities and possible interactions between them. PubMiner consists of three key components: natural language processing, machine learning based inference, and a visualization module. This system allows the visualization of the results in a graph, where the nodes represent the names of genes and of proteins and the arcs represent the possible interactions; the user has also a link between the graph and the documents' treated texts.

TextPresso⁶ [MKS04] is a text-mining system for biomedical scientific literature (PubMed abstracts). TextPresso has two central features: first it searches individual sentences of full text papers, and second, it introduces categories and marks up the instances of the categories in the corpus of literature thus allowing the user to search for instances of these categories in the full text. Textpresso uses an ontology with 33 categories to organize information in a text database. Examples of categories are: biological concepts, relationships between two or more objects and descriptions. If a combination of keywords and categories is found in a sentence, the likelihood that a sentence contains a fact involving the chosen categories and keywords is quite high. If the user chooses co-occurrence within a document, he is more interested in finding a relevant document. The scope of a search can be confined to full text, abstract, title, author, year, or any combination thereof, for document searches as well as sentence searches. TextPresso returns sentences that contain all the query items and categories and thus retrieves facts of interest. TextPresso recognizes terms based on regular expressions. The user may define the context of interest before accessing the results, therefore the results will contain only papers in the selected context. TextPresso is a useful curation tool, as well as search engine for researchers, and can readily be extended to other organism-specific corpora of text.

This is a very interesting and complete tool but it is also based on the traditional keyword search. TextPresso ranks the list of relevant abstracts.

BioRAT [CBLJ04] is a Biological Research Assistant for Text Mining, that accepts a query and autonomously finds a set of papers and highlights the most prominent facts in each paper. BioRAT extracts information from abstracts or full length papers (when available) only in the PDF format. BioRAT combines tools to download papers, to extract information from papers and to design templates to allow this extraction.

⁶TextPresso can be accessed at <http://www.textpresso.org>

When the user enters a query, a list of papers is presented to him (matching titles, date of publication, author, etc) and then the user constructs a template that helps in the extraction of the proper information. The output is presented in the XML format. BioRAT uses GATE (General Architecture for Text Engineering) which is a general purpose text engineering system based on NLP developed at Sheffield University. BioRAT is a web-based friendly tool but it is restricted to documents in PDF format which is a limitation of this system and it also uses the traditional query search.

MedBlast⁷ [TTD04] is a simple web-based application which can gather MEDLINE abstracts related to a given sequence. MEDBLAST uses natural language processing techniques, to retrieve articles related to a given sequence. MedBlast takes a sequence in the FASTA format as input and uses BLAST [SAJ+97] to search and retrieve the corresponding articles of each sequence from PubMed. MedBlast uses BLAST to find abstracts linked to homologous sequences but can also find abstracts with the gene and organism name derived from annotated protein entries. The output is a set of Medline documents which should be read by the user. In our approach we will also mine Biology's scientific literature to retrieve articles related to a given sequence in the FASTA format and we apply the NCBI BLAST tool and machine learning techniques.

GoPubMed⁸ [DS05] is a knowledge-based search engine for biomedical literature from PubMed. GoPubMed submits keywords to the PubMed database obtaining the correspondent biomedical literature to be indexed using the Gene Ontology and the Medical Subject Headings. GoPubMed uses several pre-processing techniques (stemming, tokenization, synonym detection). GoPubMed identifies relevant biomedical concepts associated with the query. The GoPubMed authors highlight the following advantages of GoPubMed in [DS05]: i) the returned abstracts are classified according to the Gene Ontology so that the user can quickly navigate through abstracts by category; ii) the general concepts, that are related to the query, but do not appear in the abstract, are provided automatically; iii) the user can easily verify the classification results since the ontology terms are highlighted in the abstracts and iv) the user can get an overview of the research trends over the time, relevant journals, top authors, and regional research interests (this information can be used to refine the search). GoPubMed does not rank results or provide importance scores for papers.

HubMed⁹ [Eat06] is a simplified interface to the medical literature search engine PubMed, designed for nonexpert searchers, incorporating external web services and providing functions to improve the efficiency of literature search, browsing and retrieval. HubMed shows first the articles that contain the search terms most frequently in the title and/or abstract, i.e., the results are ranked by relatedness. The relatedness is based on a set of initial abstracts saved by the user from previous searches. Special features of HubMed include: date or relevance-ranked search results; Web feeds for

⁷is available at <http://medblast.sibsnet.org>

⁸is available at <http://www.gopubmed.org>

⁹is freely available at <http://www.hubmed.org/>

regular updates of published literature matching any search; clustering and graphical display of related articles; expansion of query terms; direct export of citation metadata in many formats; linking of keywords to external sources of information; manual categorization (tagging) and storage of interesting articles.

DATA CARE (or GetItFull), developed by [NHB⁺06], is a tool that facilitates the downloading of journals (connects to the journal website through its URL) and performs preprocessing, producing as output an XML file for each research article with a specific format (abstract, introduction, methods, results, discussion and figures' legends) and it also identifies journal information such as the journal name, publication year and issue number among others. This is a very limited tool because it only pre-processes articles.

EBIMed¹⁰ [DHM⁺07] is a Web application that combines Information Retrieval and Extraction from MEDLINE. EBIMed's aim is to recognize protein/genes names, GO annotations, drugs and species from PubMed returned abstracts and semantically annotate these abstracts within ontologies (GeneOntology, DrugBank, UniProt, etc). EBIMed also extracts significant co-occurrences between annotated entities. EBIMed retrieves the abstracts from MEDLINE and filters sentences that contain the terms that occur in the same sentence (terms that occur in the same sentence form a pair). All sentences containing pairs are gathered and presented in a table of pairs. For each pair a link is provided to a list of sentences containing the pair. Each sentence is also linked to its original abstract. The concepts that occur frequently are shown in a table and the user can visualize the sentences corresponding to the associations. The main problem with EBIMed is that the user interface is difficult to navigate and it only extracts information from PubMed. Another limitation of this tool is that it only provides quicker results if the number of documents to analyse are limited.

eTBLAST¹¹ [EWHG07] is a text mining application designed to identify similar documents within literature databases such as (but no limited to) MedLine. eTBLAST takes a natural language text as input and then delivers abstracts that are similar to the query with high precision and recall. The user inputs an abstract or paragraph that is submitted to PubMed. The results returned by eTBLAST are ranked by a similarity score. eTBLAST sorts results by relevance, provides the full MedLine abstract and a link to the PubMed page, lets the user iterate the search over several good papers and avoids creating a complicated query. eTBLAST is a text-similarity engine rather than a simple keyword-based search tool, it is claimed that the user does not need to identify and manipulate query keywords and boolean operators, as much as in search engines. eTBLAST aims to help the user to rapidly find references, evaluate novelty, find experts and journals in a given topical area and track the popularity of the topic as defined by the user's query. eTBLAST retrieves the 400 most similar articles using a vector space approach and for these articles, a text-alignment score is calculated and aggregated per journal or author.

¹⁰is available at www.ebi.ac.uk/Rebholz-srv/ebimed

¹¹is freely accessible through the Internet at <http://invention.swmed.edu/etblast/etblast.shtml>

PolySearch¹² [CKY⁺08] is a web based text mining system designed specifically for extracting and analyzing text-derived relationships between human diseases, genes, mutations, drugs and metabolites. PolySearch is designed to mine data from PubMed abstracts, which is similar to EBIMed. Polysearch extracts and analyses not only PubMed data, but also text from multiple databases (DrugBank, SwissProt, HGMD, Entrez SPN, etc). PolySearch can produce a list of concepts which are relevant to the user's query by analyzing the mentioned information sources. It supports multiple types of biomedical text searches from multiples types of databases. PolySearch uses a bag-of-words approach to identify relevant text associations. As frequency by itself is not the best way to rate a paper, in addition to this, PolySearch employs a text ranking scheme to score the most relevant sentences and abstracts associating both the query and the terms with each other. Results are presented to the user using color-coded word highlighting schemes, key sentence display, hyperlinks and database connectivity. One limitation of this tool is that the user has to wait several minutes or hours to receive the results. It only provides quicker responses if the number of documents is limited to a very small number (such as 500 abstracts). PolySearch does not use a Machine Learning approach as our proposed system.

FACTA (Finding Associated Concepts with Text Analysis) [TJS08] is a Text Mining tool that searches for biomedical abstracts that are potentially relevant to a user query. FACTA accepts as queries not only single word concepts, but also arbitrary keywords thus permitting the user to express a concept that cannot be captured by a single keyword. FACTA can discover associations between biomedical concepts contained in MEDLINE articles. The user can navigate these associations and their corresponding articles in a highly interactive manner. The system accepts an arbitrary query term and presents a summary table of co-occurrence concepts based on MEDLINE abstracts. FACTA analyzes the documents retrieved based on a statistical method. Although it is an interesting and recent tool it only searches keywords, unlike the system we are proposing. The advantages of FACTA is that it is easy to use and delivers real-time responses while being able to accept flexible queries. The quick responses are made possible by the pre-indexing of MEDLINE and efficient document/concept retrieval algorithms.

PubFinder [GvdL05] was designed to improve the retrieval rate of scientific abstracts relevant for a specific topic. PubFinder requires as input a set of abstracts representative of the topic the user is searching for. These abstracts are processed to find a list of words indicative of discrimination between abstracts. This list of words is used for scoring all defined PubMed abstracts for their probability of belonging or not to the current topic in descending order of their likelihood score to present to the user. A disadvantage of the PubFinder approach is its high demand of computing time.

PubFocus¹³ [PZC06] performs a statistical analysis of MEDLINE/PubMed search queries by enriching them with bibliometric indicators: the journal impact factor, the number

¹²is freely available at <http://whishort.biology.ua.alberta.ca/polysearch/>

¹³available at www.pubfocus.com

of citations and the authors impact on the field of search. PubFocus provides a list of articles ranked by relevancy.

ReleMed¹⁴ [SSK07] is a search engine, from the University of Virginia's School of Medicine, that searches PubMed for medical literature and presents the results by relevancy based on keywords. ReleMed finds articles that present a close relation among the search terms. ReleMed categorizes each retrieved citation into 8 different levels of relevance, depending on the frequency of occurrence of the search terms within the title, sentences of the abstract and MeSH. ReleMed presents the most relevant results first.

MScanner¹⁵ [PRAS08] is a web-based classifier of MEDLINE citations. MScanner requires as input a corpus of relevant training examples in the form of PubMed IDs and returns results ranked by decreasing probability of relevance. MScanner's main objective is to find relevant documents through classification just like BioTexRetriver. However it is more specific in learning how to distinguish articles relevant to pharmacogenomics versus those that are not by using the MeSH terms.

XplorMed¹⁶ [PIPBA03] is a web-based tool that aims to analyse and extract relations between words of PubMed abstracts. In a certain way XplorMed summarizes results from a MEDLINE search by overcoming the limitations of keyword based search. XplorMed can group abstracts based on the associations between the words they contain. These relations can be filtered and arranged to deduce different subjects in the query and offer a condensed view of the abstract, allowing users to select texts of interest without having to read them all.

MedlineRanker¹⁷ according to their developers [FBSS⁺09] allows a flexible ranking in Medline for a topic of interest without expert knowledge. The MedlineRanker webserver requires as input a list of abstracts relevant to a particular topic and then the tool learns the most discriminative words (common words) in those abstracts. These words are used to score newly published articles. MedlineRanker uses a naïve bayesian classifier for the learning and classification process. The authors claim that if the input contains closely related abstracts MedlineRanker returns relevant abstracts from the recent bibliography with high accuracy and that the tool processes thousands of abstracts from the Medline database in a few seconds, or millions in few minutes. MedlineRanker has a web interface that allows customization of some input parameters. According to the authors the MedlineRanker will produce more accurate results if the user provides a training set with enough abstracts (100-1000) to define the topic of interest.

RefMed¹⁸ [YKO⁺09] is a new system search engine for PubMed with relevance feedback. RefMed ranks the documents based on a machine learning algorithm in a first

¹⁴ <http://www.relemed.com>

¹⁵ is available at <http://mscanner.stanford.edu/>

¹⁶ is available at <http://www.ogic.ca/projects/xplormed/>

¹⁷ <http://cbdm.mdc-berlin.de/~medlineranker/about.html>

¹⁸ is accessible at <http://dm.postech.ac.kr/refmed/>

phase depending on the user query. On a second phase RefMed takes into account the user's feedback judgments on some of the resultant documents while browsing them, and then the system uses a relevance function called RankSVM that ranks the documents based on user's feedback.

SciMiner [HSSF09] is a web-based literature mining and functional analysis tool that identifies genes and proteins using a context specific analysis of MEDLINE abstracts and full texts. SciMiner accepts a free text query (PubMed Entrez search) or a list of PubMed identifiers (PMIDs) as input. SciMiner automatically collects MEDLINE records and available full text documents. Targets (gene and proteins) are extracted and ranked by the number of documents in which they appear. SciMiner searches full text documents; allows users to directly edit the mining results and allows comparisons to be made between search results of multiple queries. SciMiner is implemented in Perl and uses a MySQL database to store compiled dictionaries and identified targets. Although it is an interesting and recent tool to mine biological literature we follow a different approach beginning with the query to search.

A platform for Biomedical Text Mining (BioTM) called @Note [LCC⁺09] promotes a multi-disciplinary research providing support to three different usage roles: biologists, text miners and application developers. @Note is a set of user-friendly tools for biomedical document retrieval, annotation and curation. Its main functional contributions are: the ability to process abstracts and full-texts; an information retrieval module enabling PubMed search and journal crawling; a pre-processing module with PDF-to-text conversion, tokenization and stopword removal; a semantic annotation schema; a lexicon-based annotator; a user friendly annotation view that allows to correct annotations and a Text Mining Module supporting dataset preparation and algorithm evaluation. The basic pre-processing steps are implemented using GATE features. The Text Mining module is also implemented recurring to a low-level plugin to YALE that also includes WEKA. These two open source toolkits, allow the development of different problem oriented text mining experiments, namely feature selection and model evaluation.

The LigerCat¹⁹ (Literature and Genomic Electronic Resource Catalogue) [SSMN09] system was developed to provide a more convenient interface for PubMed searching. The system generates a word cloud for MeSH terms arising in articles reported by an initial user query (gene or drug). The user can then click on the individual terms within the cloud to restrict results in the PubMed search. LigerCat analyses multiple PubMed articles based on their MeSH terms and presents them in a cloud ordered by their frequency.

Genes2WordCloud²⁰ [BJDM11], an open source web application and Java Applet that enables users to create biologically-relevant word-clouds content from several different sources: a single gene or a list of genes, free text, text extracted from the URL of a website, text extracted from abstracts associated with an author, text extracted

¹⁹<http://ligercat.ubio.org/>

²⁰http://www.maayanlab.net/G2W/create_wordcloud.php

from abstracts returned from any PubMed search, and word-clouds created from the abstracts of the most viewed articles on BMC Bioinformatics to examine current trends in the field of Bioinformatics.

A summary of some of the presented tools, adapted from [RKP07], is presented in Tables 2.4 and 2.5.

	Keywords Search	Full Texts or Abstracts	Pre-Processing	Extra Features	Ranking	Implementation
MedMiner	Gene names	PubMed database of gene cards	*	Text filtering using a statistical approach	Yes	Perl
BioMinT	Yes	PubMed literature	*	*	Yes	*
PubMiner	No	PubMed abstracts	Yes (tokenizer)	NLP and ML techniques	No	*
TextPresso	keywords	Titles, abstracts and full texts	Yes (tokenizer sentence delimiter, etc)	No	Yes	*
BioRAT	Yes	Full text	Yes	GATE and BLAST	*	Java
MedBlast	Sequences (FASTA format)	PubMed	*	NLP techniques	*	Perl and BioPerl
GoPubMed	Keywords	PubMed abstracts	*	*	No	*
HubMed	*	PubMed abstracts	*	Clustering	Yes	*
DataCare (GetItFull)	No	PubMed abstracts	Removes HTML tags	No	*	Windows application
EBIMed	Keywords query or PubMed identifiers (PMIDs)	PubMed Abstracts	Tokenization	No	Ranks sentences	*
eTBLAST	Yes	Abstracts	Yes	Statistical approach	Similarity ranked output	*
PolySearch	Yes	Abstracts and Full texts	Yes	Pattern recognition	Text ranking scheme	Perl and HTML
FACTA	Yes	PubMed abstracts	Yes	Statistical approach	statistical approach of concepts	*

* Not Available

Table 2.4: Summary of tools for bio text mining applications

	Keywords Search	Full Texts or Abstracts	Pre-Processing	Extra Features	Ranking	Implementation
PubFinder	Abstracts	set of abstracts	*	*	ranking by relevance to the query	*
PubFocus	Yes	PubMed Abstracts	*	*	ranking by relevance to the query	*
Relemed	Yes	PubMed abstracts	*	*	ranking by relevance to keywords	*
MScanner	PubMed identifiers (PMIDs)	Document corpus	*	ML	Yes	Python
XPlorMed	Yes	PubMed abstracts	*	*	No	*
MedlineRanker	Abstracts	list of abstracts *	*	Machine Learning	Yes	*
RefMed	Yes	PubMed literature	*	ML and IR	ranking functions based on user's judgment	*
SciMiner	Yes	MedLine abstracts and full text	*	Mining rules	statistical approach	Perl and MySQL
@NOTE	Yes	MedLine abstracts / Full texts	Tokenization, stop word removal, etc	ML (Weka)	*	Perl and Java
LigerCat	Yes (Gene/Drug)	Abstracts	*	ML	No	*
Genes2 WordCloud	Text / Genes	PubMed abstracts / Full texts	*	ML	No	JAVA applet

* Not Available

Table 2.5: Summary of tools for bio text mining applications (cont).

The BioCreative [KMS⁺08] challenge (Critical Assessment of Information Extraction in Biology) is a competition with the following objectives: an international evaluation of the state-of-the-art text mining systems in Biology; compare the performance of different methods; produce a gold standard training set; monitor improvements in the field and produce useful evaluation tools/metrics. It is therefore a collaborative effort among researchers from heterogeneous domains (biology, bioinformatics and natural language processing) to provide the evaluation of text mining and information extraction systems applied to the biological domain.

BioCreative is ideally suited to create the conditions necessary for significant scientific advancement in the area of text mining, by providing a framework for testing and evaluating research tools over shared tasks. In particular, three main tasks were defined by the BioCreative organizers: gene mention (GM), gene normalization (GN), and protein-protein interaction (PPI).

An important contribution of BioCreative has been the creation of standard datasets, prepared by domain experts, for the training and testing of text mining applications, which represents an important resource for the continued development and improvement of text-mining applications.

Participants are given a common training corpus, and a period of time to develop systems to carry out the task. At a specified time the participants are then given a test corpus, previously unseen, and a short period of time in which to apply their systems and return the results to the organizers for evaluation. All submissions are then evaluated according to numerical criteria, specified in advance. The results are then returned to the participants and subsequently made public in a workshop and coordinated publication.

As a result, BioCreative has motivated the development of the first text-mining meta-server, which will serve as a framework and platform to improve the accessibility and use of automatically extracted text-derived information by the user community. The server delivers to the user consensus annotations for PubMed abstracts from systems that participate in the BioCreativeII challenge.

BioCreative's main focus is on biologically relevant tasks, which should result in benefits for the biomedical text mining community, the biology and biological database community, as well as the bioinformatics community.

Other complementary evaluations of Text Mining systems in biology have been carried out recently in the KDD Cup ²¹ and the genomics track at the TREC conference ²².

2.5 Ranking Methodologies

Ranking is the process of ordering a set of items in order to show the most relevant first. In fact, Ranking is the core of an Information Retrieval system because we need to know in what order to present the returned documents to the user.

²¹<http://www.sigkdd.org/kddcup/index.php>

²²<http://ir.ohsu.edu/genomics/>

Web Ranking

A ranking is usually obtained by measuring the similarity between a query and a page (content-based features) and a page quality (query independent page quality features) [ABD06].

Algorithms based on link structure have been proposed to rank web pages. PageRank [PBMW99], HITS[Kle99] and SALSA [LM01] are three of these approaches.

PageRank [PBMW99] is widely regarded as the best method for the ranking of Web pages. PageRank is based on the key idea that an interesting page is referenced by other several pages. The more hyperlinks to a page the more relevant that page is. Interesting pages are usually referenced by interesting pages. If a page references several important pages then it might be itself also an important page. PageRank computes the importance of a page (weight) based on the number of pages pointing to that page. PageRank is a graph-based algorithm based on a random walk model to compute the probability distribution of nodes in the graph (Markov Chain). PageRank algorithm has suffered some improvements and there exist some variations of this algorithm, such as FutureRank [SG09].

FutureRank [SG09] ranks scientific articles based on a new measure: which is the expected future PageRank score based on citations that will be obtained in the future.

The HITS (Hyper-link Induced Topic Search) [Kle99] algorithm treats WWW as a directed graph $G(V,E)$, where V is a set of vertices representing pages and E is set of edges corresponding to a link. This algorithm divides the role of a page into a hub or authority. Hub measures the number of links to other pages, and authority measures the importance of a page. HITS relies on the idea that a good authority is pointed to by many good hubs and a good hub points to many good authorities. On the contrary to PageRank, HITS is a query-dependent algorithm. HITS builds a neighborhood graph based on a small set of relevant pages, retrieved from a search engine, and then executes the ranking algorithm on this neighborhood graph. The query is submitted to a search engine.

SALSA [LM01] stands for Stochastic Approach for Link-Structure Analysis, is a probabilistic approach using Markov chains. SALSA is also query-dependent once it is a variation of the HITS algorithm.

The RankNet [BSR⁺05] algorithm learns the patterns of human searches in order to provide more relevant results in the next search. RankNet is widely used in commercial search engines.

Document Ranking

Document ranking tells the user how important a document is to a query. The main idea of ranking scientific papers is to order a set of scientific papers based on relevance, importance or preference. According to the query the ranker orders the corpus of documents presenting the most relevant first, i.e. the results are presented in order of

relevance.

There is some research in document ranking, in fact there is some work in ranking scientific papers.

In [SW] the authors use machine learning to order documents by popularity, or the predicted frequency that an article is viewed by the average PubMed user. The authors claim that the identified method for learning popularity from clickthrough data shows that the topic of an article influences its popularity more than its publication date.

[DLWF12] proposes a unified model, PAV, for ranking heterogeneous objects, such as papers, author, and venues. PAV explores object ranking in bibliographic information where objects are papers, authors and venues. In PAV the bibliographic information network is represented by a weighted directed graph, where a vertex stands for an object, an edge stands for the link between objects, and a weight over an edge stands for the degree of contribution that one object devotes to the importance or reputation of the corresponding object sharing the same edge with the object. The rank (importance or reputation) of an object is the probability that the corresponding vertex is accessed by random walk in the PAV graph. The authors claim PAV is an efficient solution for ranking author, paper, and venues simultaneously. According to their method, the importance or reputation of an author is influenced by his co-authors, his papers, and the venues that published his papers. The importance or reputation of a paper is influenced by its authors, its venue, and the papers that cited it. The importance or reputation of a venue is influenced by the papers that it published and the authors who had papers published by the venue. PAV model transforms the problem of ranking objects into the problem of estimating probability parameters. For estimating probabilities the authors developed an algorithm based on matrix computing. The authors claim their algorithm could be ran efficiently by proving that the underlying computing method is convergent.

The authors in [ZFT⁺11] present an approach that jointly ranks publications, authors and venues. They first constructed a heterogeneous academic network which is composed of publications, authors and venues. A random walk over the network was performed hence yielding a global ranking result of the objects on the network. The mutual reinforcing relationship between user expertise and publication quality was based on users bookmarks. The authors claim that their experimental results with ACM dataset show that their work outperforms all other baseline algorithms, such as Citation Count, PageRank, and PopRank.

In this paper [RBAC⁺07], the authors present three different prestige score (ranking) functions for the context-based environment, namely, citation-based, text-based, and pattern-based score functions. Using biomedical publications as the test case and Gene Ontology as the context hierarchy, the authors have evaluated the proposed ranking functions in terms of their accuracy and separability. They concluded that text-based and pattern-based score functions yield better accuracy and separability than citation-based score functions.

The paper [ZLL11] proposes an iterative algorithm named AP Rank to quantify the scientists' prestige and the quality of their publications via their inter-relationship on an author paper bipartite network. In this method a paper is expected to be of high quality if it was cited by prestigious scientists, while high-quality papers will, in turn, raise their authors' prestige. AP rank weighs the prestige of quoters more than the number of citations. Given that old papers will have more chances to accumulate more citations than recent works the authors proposed a time-dependent AP rank (TAP rank). According to the authors the main advantages of AP rank are that it is parameter-free; it considers the interaction between the prestige of scientists and the quality of their publications and it is effective in distinguishing between prestige and popularity.

The authors in [BHA⁺06] determine whether algorithms developed for the World Wide Web can be applied to the biomedical literature in order to identify articles that are relevant for surgical oncology literature. For this study the authors have made a direct comparison of eight algorithms: simple PubMed queries, clinical queries (sensitive and specific versions), vector cosine comparison, citation count, journal impact factor, PageRank, and machine learning based on polynomial support vector machines. As a result of this study they concluded that the mentioned algorithms can be applied to biomedical information retrieval and that citation-based algorithms were more effective than non citation-based algorithms at identifying important articles. The most effective strategies were simple citation count and PageRank and citation-based algorithms can help identify important articles within large sets of relevant results.

In [LLCL07] the authors propose a ranking function for the MEDLINE citations. This function integrates the Citation Count Per Year and the Journal Impact Factor which are two of the factors that integrate the ranking function we have developed. The goal of this work is to present to the users a reduced set of relevant citations, retrieved and organized from the MEDLINE citations into different topical groups and prioritized important citations in each group.

The referred work uses graphs, existing web-based algorithms, and some propose a more specific ranking function. We may conclude that to choose an existing ranking algorithm or to develop a new ranking function depends on the work to be applied and on what the researchers want to achieve. In our case we decided to develop a multi-criteria ranking function in order to satisfy all the issues we believe to perform better for ranking the MEDLINE papers.

2.6 Summary

This chapter presented the state-of-the-art in the fields of Information Retrieval, Text Mining and Classification algorithms. We have presented the most common models used in Information Retrieval and the classification algorithms used in Text classification, as well as their advantages and weaknesses. We presented a set of general

tools for Text Mining and a set of tools specifically for the Bioinformatics domain. A summary table of these tools is also presented regarding the items that are the focuses of the present study. The chapter ends with a brief state-of-the-art for the ranking methodologies used for web and document ranking that is also one of the important issues of this research.

Chapter 3

A tool for Text Mining in Molecular Biology's Domains

This chapter introduces BioTextRetriever, a Web-based search tool for retrieving relevant literature in Molecular Biology and related domains from repositories such as MEDLINE¹ (MEDical Literature Analysis and Retrieval System Online). Relevant literature means literature related to the scientific paper references associated with the sequences provided by the user.

The present chapter introduces the steps of BioTextRetriever: collect scientific paper references associated with the introduced sequences; pre-process techniques to be applied to the collected scientific paper references; and rank the whole set of collected relevant scientific paper references. A detailed description of the pre-processing techniques to be applied to the scientific paper title and abstract, are presented and finally the chapter ends with an example that presents how to use BioTextRetriever.

3.1 BioTextRetriever Overview

BioTextRetriever is a Web-based search tool that accepts a set of genomic or proteomic sequences and returns an ordered set of references to scientific papers reporting work considered relevant for the study of the given sequences. The sequences² are provided by a biologist together with an e-value³ and the type of BLAST⁴. BLAST stands for Basic Local Alignment Search Tool, and finds regions of local similarity between sequences. BLAST returns sequences that are similar to the input query. We have two types of BLAST defined in BioTextRetriever: BLASTP (Protein-protein BLAST) and BLASTX (Nucleotide 6-frame translation-protein). This information constitute the

¹<http://www.nlm.nih.gov/pubs/factsheets/medline.html>

²in FASTA format.

³e-value is a statistic to estimate the significance of a “match“ between 2 sequences [HdVLG06].

⁴<http://blast.ncbi.nlm.nih.gov/Blast.cgi>

input for BioTextRetriever as shown in Figure 3.1. Figure 3.1 also shows the overall processing of the BioTextRetriever tool. We will now describe in detail the structure of the tool as well as how it works. A detailed description of its methodology is provided in the next chapters.

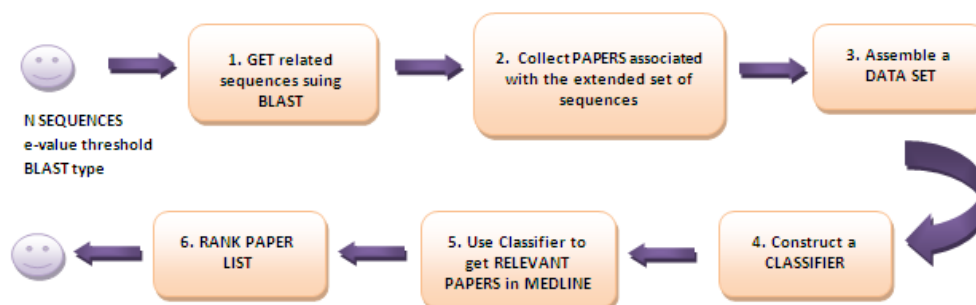


Figure 3.1: Sequence of steps implemented by BioTextRetriever

Initially the expert user specifies a set of N sequences, an e-value and the BLAST type (blastp or blastx). With these items and using the NCBI BLAST tool, a set of similar sequences⁵ is collected from PubMed as a result of Step 1.

The first phase to construct the data sets, only requires the e-value and the type of BLAST to gather the papers associated to the sequences introduced by the user. To accomplish this we have used *ncbi-blast-2.2.26+* that performs a remote blast search at NCBI. *Ncbiblast*⁶ searches for sequences similar to a query sequence introduced by the user according to the type of BLAST that is a request to use *ncbiblast*. The *ncbiblast* tool uses the BLAST (Basic Local Alignment Search Tool) algorithm proposed by Altschul [AGM⁺90] in 1990. This algorithm finds the highest scoring locally optimal alignments between a query sequence and a database. The query and the database searched can be either peptide or nucleic acid in any combination. *Ncbiblast* can search only databases maintained at NCBI. We have embedded this application into the code and automatically have access to the similar sequences which are saved into a text file for further processing. The result of the *ncbiblast* search is a set of sequences similar to the input ones. Along with the new sequences we get their e-value associated. The expectation value or e-value is the number of alignments with the query sequence that would be expected to occur by chance in the database [BO01]. The closer the e-value is to 0, the better the alignment, the more similar are the sequences. Based on the first e-value cutoff introduced by the user we select the similar sequences that have an e-value lower than this. Each of this similar sequences has a set of N papers associated. Using the *ncbiblast* we obtain the pmid's of the papers associated to the similar sequences. With this information we search in a local copy of MEDLINE (LDB - Local DataBase) to obtain each paper's complete information. Step 1 would be the same if we were using Ensembl⁷ [SFS10], instead of NCBI. Details of Step 1 process

⁵Only sequences with e-values smaller than the one specified are collected

⁶ is available for download at <ftp://ftp.ncbi.nlm.nih.gov/blast/executables/release/LATEST/>.

⁷<http://www.ensembl.org/>

will be provided in Section 4.1.

Using the extended set of sequences the tool collects, in Step 2, all the references to the scientific papers related to those sequences. BioTextRetriever then searches their corresponding abstracts in LDB. For this we have previously pre-processed MEDLINE as will be described in Section 3.4.

Step 2 searches and collects the following information in MEDLINE: pmid, journal title, journal ISSN, article title, abstract, list of authors, list of keywords, list of Medical Subject Headings (MeSH) terms and publication date. The MeSH list is a set of controlled vocabulary terms for the indexation of MEDLINE articles (see Section 3.2 for details).

Besides this information, the number of citations of each article and the impact factor of the Journal/Conference where the article was published are the next items of information to be used later in the process of sorting through paper list. Considering the scope of this thesis we are only taking into account paper references that have an abstract available in MEDLINE.

After Step 2 we have an initial version of a data set: a "proto data set". The proto data set is pre-processed and converted into a data set in Step 3 in order to be adequate for the learning process to take place. Step 3 is introduced in Section 3.4.

Step 4 is one of the most important stages of our work and consists in constructing a classifier using Machine Learning techniques as fully explained later in Chapter 4.

The resulting classifier is used to collect an extensive list of articles considered relevant (Step 5).

However, we need to present them to the expert ordered by their relevance. That is the task of Step 6. To achieve that, we developed a ranking function described in Chapter 5 that represents Step 6.

3.2 The MEDLINE

MEDLINE (Medical Literature Analysis and Retrieval Systems Online) is a database of articles published in major peer reviewed journals since the 1950s. Papers are represented by their author information, title, abstract, publication date, keywords list and Medical Subject Headings list. The Medical Subject Headings list consists of single and multi-word terms that are used to index and catalog medical literature [CB01]. The Medical Subject Headings (MeSH) [W.64] is a controlled vocabulary thesaurus maintained by the National Library of Medicine (NLM). MEDLINE's scientific articles are manually annotated using the MeSH ontology [CB01] that is established with a certain degree of consensus in the scientific community. MEDLINE is the premier bibliographic database containing currently more than 22 million references to journal

articles in the Life Sciences with a concentration on Biomedicine. MEDLINE is one of the best known major sources for biomedical literature. PubMed is one, but not the only way to search MEDLINE. PubMed provides additional resources⁸. PubMed⁹ is a web-based literature retrieval service, and is a part of Entrez retrieval system ([JHGJ96], [ent]), a retrieval system that provides access to biomedical literature databases such as MEDLINE and is allocated to the NCBI website. PubMed provides access to a huge amount of biomedical literature databases, and MEDLINE is the largest among those databases.

PubMed comprises more than 22 million citations of biomedical literature from MEDLINE, Life Science journals, and online books. Citations may include links to full-text content from PubMed Central and publisher's web sites¹⁰. PubMed abstracts are manually curated by human experts. The experts read all the documents and assign a set of MeSH terms to each document.

PubMed Central (PMC)¹¹ is the US National Institutes of Health (NIH) free digital archive of Biomedical Life Sciences journal literature. PMC was developed by NLM and is currently managed by NLM's National Centre for Biotechnology Information (NCBI)¹².

3.3 Local DataBase

For a quicker access to MEDLINE registers and to query more rapidly MEDLINE we implemented a local database (LDB). The processing time is also faster in a local database.

Our LDB includes not only the MEDLINE data available but also some extra information not available in MEDLINE's registers. This extra information that includes the Journal Impact Factor, the number of citations of an article, the author's h-index, the author number of publications and the Journal Similarity Factor will be used for the last part of our work, so as to rank the obtained results (Ranking) in order to measure the research impact, e.g., the relevance of the retrieved articles. This extra information is detailed in Section 3.3.1. Now we will detail and characterize the available information in MEDLINE's registers.

We have downloaded the whole XML(eXtensible Markup Language)¹³ files that compose the MEDLINE 2010^{14,15} from the NCBI website. MEDLINE 2010 has nearly 20

⁸<http://www.ncbi.nlm.nih.gov/pubmed/>

⁹http://www.nlm.nih.gov/pubs/factsheets/dif_med_pub.html

¹⁰<http://www.ncbi.nlm.nih.gov/pubmed/>

¹¹<http://www.ncbi.nlm.nih.gov/pmc/>

¹²<http://www.ncbi.nlm.nih.gov/>

¹³The complete structure of a XML file can be seen in Annex A

¹⁴http://www.nlm.nih.gov/bsd/licensee/2010_stats/baseline_doc.html

¹⁵MEDLINE 2010 is distributed as a 80GB set of 617 compressed XML files.

million citations.

Among the recorded features of each paper, we highlight the one's that are most relevant for our work:

- PMID - the PubMed Identifier
- PubDate - the PubMed Date
- Journal Title
- Journal ISSN - has the same value of the ISI Web of Knowledge ISSN
- Article Title
- Abstract (not always available)
- List of Authors
- Mesh Headings List
- Keywords List

For BioTextRetriever to work efficiently it is required to have a local copy of MEDLINE properly pre-processed. We now describe the pre-processing of the local data base. The final schema of LDB is shown in Figure A.1 in the Annexes. The LDB has a central entity named *Citation* where all the other database entities are connected (see Figure A.1).

Table 3.1 characterizes the LDB in terms of the number of registers and size of the most important tables of LDB.

Table 3.1: LDB characterization

Table Name	Number of registers	Size
Citations	9.688.169	21.43 GB
Terms	2.726.179	68.49 MB
MeSH terms	24.971	813.41 KB

3.3.1 Extensions to the available Information

Besides the information contained in MEDLINE, we have added some extra information to our LDB.

Although the set of papers returned by BioTextRetriever are all relevant, we need to point out which is the most important paper to be read by the biologist and the not so important one in the returned list. We believe that a combination of scientific research impact metrics will help to order the returned results in what we expect to be an impartial process. This extra information includes the MEDLINE number of MeSH terms, the article number of citations, the author h-index, the author number of publications, the Journal Impact Factor and the Journal Similarity Factor. All of the items were obtained using the LDB, except the Journal Impact Factor that was obtained from the ISI Web of Knowledge website powered by Thomson Reuters. We will refer the six mentioned items but they are detailed in Chapter 5.

MEDLINE Number of MeSH terms

The MEDLINE Number of MeSH terms refers to the number of MeSH terms in common with the articles associated to the sequences introduced by the user. With this apriori selection we assure that the MEDLINE articles to be classified have a high probability of being somehow related to those sequences.

MEDLINE Author's Number of Publications

The number of publications of an author is usually quite relevant to endorse the author's quality. Usually good authors have a large number of publications. However this may not always be true because an author may have a large number of publications but published in journals or conferences with a low impact factor.

MEDLINE Number of citations

Although not without drawbacks we consider the number of citations an important indicator to evaluate scientific work. The number of citations represents the number of times a scientific paper is cited by other scientific papers excluding self citations. For the present work we calculate the number of citations only inside MEDLINE2010, e.g., scientific papers that cite other MEDLINE scientific papers. We have developed a formula that reflects the fact that old scientific papers have naturally more citations than recent scientific papers.

MEDLINE h-index

The *h-index* [Hir10] is an index that attempts to measure both the productivity and impact of the published work of a researcher. It is based on the set of the scientist's most cited papers and the number of citations that they have in other publications. To calculate the MEDLINE internal h-index we consider only the MEDLINE internal number of citations. Because a scientific paper may have more than one author (which is the most common case) we calculate the h-index for all the authors of an article

and select the highest h-index.

Journal Impact Factor

The Journal Impact Factor (JIF) is a measure of the frequency with which the average article in a journal has been cited in a particular year. Although Journal Impact Factor has advantages and limitations, usually a journal with a high impact factor is considered to have high quality. We have obtained the Journal Impact Factor for each article which was published in the Web of Knowledge website. We have downloaded the complete list of journal's impact factors in October 2010. At this date there were 7347 journals available.

The items considered are not completely independent. Journals with a high impact factor usually have scientific papers highly cited by other published important scientific papers. These highly cited scientific papers are also normally written by important authors and co-authors where important means authors and co-authors with an high h-index.

Journal Similarity Factor

The Journal Similarity Factor, highlights the journals with more papers published that are associated to the original sequences introduced by the user.

3.4 Pre-Processing MEDLINE data

To improve efficiency BioTextRetriever uses a pre-processed local copy of MEDLINE (LDB).

The first step involves processing the MEDLINE XML's encoding files and extract the relevant information to store in LDB. LDB's structure is shown in Appendix A (Figure A.1).

The pre-processing of each article's title and abstract encompasses the following steps:

1. Named Entity Recognition - the task of finding multi-word entities (people, organizations, etc) in the text;
2. Synonyms Handling - handle words' synonyms;
3. Tokenization - break a text into tokens (strings separated by white spaces);
4. Special Characters Removal - remove special characters from a text;
5. Stop-Words Removal - remove a set of stop-words from a text;
6. Word Validation - validate every term of a text;

7. Stemming - reduce inflected words to their root;
8. Pruning - remove too frequent or too infrequent words.

In order to classify MEDLINE documents we need to reduce significantly the data set number of attributes. If we reduce the number of attributes without losing accuracy we get a quicker processing time. The initial number of attributes is usually much larger when compared to the number of examples which may result in an overfitting problem. This information is stored in our LDB.

We will now detail all the pre-processing steps and justify what steps were chosen according to the results of an experimental assessment. We will now describe the pre-processing steps and then describe the experimental assessment.

Named Entity Recognition

Named Entity Recognition (NER) is the task of identifying terms that mention a known entity in the text. Entities typically fall into a pre-defined set of categories such as *person*, *location*, *organization*, etc. For the purpose of our work we are interested in identifying entities from the Life Sciences such as *proteins*, *genes*, etc. For this reason we used the *A Biomedical Named Entity Recognition*¹⁶ (ABNER) tool [Set05], this is a tool that identifies entities in the Life Sciences domain. ABNER is a software tool for molecular biology text analysis. ABNER identifies five types of entities: *proteins*, *RNA*, *DNA*, *cell type* and *cell line*. Entities may be composed by more than one word, so in these cases, after identifying these entities in the title and abstract, the set of words is linked using UNDERSCORE characters. For example the entity “myeloid associated genes” which is identified by ABNER as a protein is replaced by the simple word *_myeloid_associated_genes_*.

Although the referred entity is formed by more than one word, it is stored in LDB and considered for further processing as a single term. It is also added to the table of terms because it is a new term identified as a NER. In the pre-processing procedures the NER identification was the first step applied. If we had applied stemming first the term would not have been identified by ABNER. Although we have implemented this technique in our experiments (see Table 3.3) we have concluded that the identification of NER strongly increased the number of terms which became a serious problem for the papers' classification task. Thus we did not use NER in the pre-processing phase.

Synonyms Handling

Another pre-processing step is **Synonyms Handling**, which we have done using the Word Net¹⁷ (an English lexical database), for regular English (“non technical” words) and Gene Ontology¹⁸ (GO) [ABB⁺00] for technical terms. Handling synonyms allows

¹⁶Available at <http://pages.cs.wisc.edu/bsettles/abner/>

¹⁷<http://wordnet.princeton.edu/>

¹⁸<http://www.geneontology.org/>

us to significantly reduce the number of attributes in the data sets without changing the semantic of words. In this step we have replaced all the synonyms found by one synonym-term thus reducing the number of regular English terms. In the case of biological technical terms, we have used Gene Ontology to identify synonyms. We have chosen to use GO because it is a major bioinformatics initiative to unify the representation of gene and gene-product attributes across all species. A biological term can be, for example, "*mitochondrion inheritance*" that has several synonyms such as: "*mitochondrial inheritance*" and "*mitochondrial genome maintenance*" which are terms with more than one word. We have replaced all the synonyms found by the first Gene Ontology proposed synonym thus reducing the number of biological terms. BioTextRetriever's first step is to handle synonyms because we did not use Named Entity Recognition.

Tokenization

The next step is **Tokenization** that splits the article's title and abstract into tokens, e.g. terms.

Special Characters Removal

This step removes punctuation, digits and some special characters such as (, , , ; , . , ! , ? , ' , " , [,] , *etc*). Characters like + and - are not removed because they might be important in some biology domains and if we remove them, the term may lose sense (for example: "blood-lead").

Stop-Words Removal

Stop-Words Removal removes words that are meaningless such as articles, conjunctions and prepositions (e.g., a, the, at, etc.). We have used a list of 659 stop words to be identified and removed from the article's title and abstract. This step is very useful to discard terms that do not contribute to the evaluation of the document's content. Besides, the removal of these words frees up the space they occupy, and significantly reduces the number of attributes to be given to the classifier.

Word Validation

Another pre-processing step is **Word Validation**. The tool accepts as a valid term those that appear in a dictionary. We have gathered several dictionaries for the common English terms, and for the biological and medical terms.

For the current English terms we have used a set of files available from ISPELL¹⁹ and Word Net. Although ISPELL is a spelling checker, we have only used the dictionary files available to see if a term is a valid term or not. The same was done with Word

¹⁹<http://www.lasr.cs.ucla.edu/geoff/ispell.html>

Net.

For the biological and medical terms, they were validated using The Hosford Medical Terms Dictionary [Hos04], from BioLexicon²⁰ [RSPK⁺08] and Gene Ontology. The Hosford Medical Terms Dictionary which consists of a file that contains a long list of medical terms. BioLexicon is a large-scale terminological resource developed to address text mining requirements in the biomedical domain. The BioLexicon is publicly available both as an XML-formatted term repository and as a relational database (MySQL) and it adheres to the LMF ISO standards [HH10] for lexicon resources. We have also used the Gene Ontology available files that are related to genes, enzymes, chemical resources, species and proteins. We have taken the decision of accepting a term if and only if it appears in one of the mentioned dictionaries. Otherwise the term is discarded.

Stemming

The **Stemming** process reduces the inflected or derived word to its stem/root. If we apply stemming to the words “computer”, “compute”, “computation” and “computing”, the root is “comput” for all of these words. So the learning model can classify the document into a correct category and thus reduce the number of attributes. A disadvantage is that some words may have the same root but may have different meanings thus should be classified in different categories (for example “dessert” and “desert”). Although stemming is not always 100% correct in text classification it is still widely used in text classification because it helps to improve the classifiers [SR02].

Here we have evaluated three different tools with different stemming algorithm implementations: Snowball²¹, Lucene²² and a Porter Stemmer [Por97] implementation. We have compared the number of attributes of these three implementations using 100 titles and abstracts from a MEDLINE 2010 sample and we have concluded that these implementations are very similar because the number of terms returned is almost the same. Snowball had two implementations, one with Porter Stemmer and another with an English Stemmer. Lucene has also a Porter Stemmer implementation. However, in Biology, terms such as “X-RAY” or “N2” may be important terms in the Life Sciences domain, so they should be kept like this. In the tool we are developing these terms are stored like this and not stemmed nor is the punctuation removed. In Lucene the term *n.2.* is transformed by Lucene into *n* which does not have interest in the biology domain. Thus we have decided to use the Porter Stemmer algorithm in our tool.

Pruning

The Zipf's Law [Pow98] is a linguistic measure of frequency of words in a document. According to this law the majority of indexing terms occurs only in a small number

²⁰<http://www.ebi.ac.uk/Rebholz-srv/BioLexicon/biolexicon.html>

²¹<http://snowball.tartarus.org/index.php>

²²<http://lucene.apache.org/core/>

of documents. A word that appears a few times in a document has a low frequency so is usually statistically insignificant. Thus the probability of this word to occur in a new document is also very low.

In text mining, pruning is a useful pre-processing concept because most words in the text corpus are low-frequency words [Hoo11]. In categorization, the pruning often yields the smallest size of the feature space, a smaller classification model and a better performance on testing data set, because of the irrelevance of low frequency words to the text categorization task [MN98, Joa98].

Pruning is the last pre-processing technique to be applied, to discard the terms that appear too frequently and/or too rarely. We have carried out a set of experiments to assess different threshold values for this filtering operation (detailed in Section 3.5). For this set of experiments BioTextRetriever discards the terms that appear rarely in the whole collection.

3.5 Assessing the Pre-Processing Techniques

In order to assess the impact of the pre-processing techniques in the classification of MEDLINE documents, we have carried out a set of experiments that we now describe. Our main objective was to determine the most adequate combination of pre-processing techniques to classify MEDLINE documents.

For this assessment we have created a balanced data set from MEDLINE 2010. Four frequent MeSH terms were selected. The four MeSH terms used were: " *Escherichia coli* "; " *Alzheimer Disease* "; " *Cholesterol* " and " *Lung Diseases* ". These four MeSH terms were chosen, because they were the four most frequent MeSH terms in MEDLINE 2010. These four MeSH terms are from four different domains.

For each MeSH term 20000 papers were randomly selected containing the specified MeSH term and none of the three others. A data set of 80000 papers was assembled in this way.

All of the following pre-processing assessment experiments were based on this balanced dataset. A preliminary work is described in [GGCO10]. The full set of combinations of pre-processing techniques can be seen in Table 3.3.

These include the application or not of: NER; word validation; handling synonyms and pruning of words with very low frequency in the collection of documents. To prune rare words we have used the following thresholds for the filtering out of terms with very low frequency: 5, 10 and 50. When we apply pruning with a threshold of 5 it means that we remove the terms that appear in less than 5 documents of the collection. Pruning of highly frequent terms was not considered because according to our experiments this pruning does not have a considerable impact in the data set. We do not have a significant number of terms that appear at least once in all the collection.

We have tested several thresholds (pruning of 90%, 95%, 85%, 80%). Pruning of 90% means that a term appears more than ninety times, respectively, in the collection, thus we remove these terms from the collection. However we do not have terms that appear 90% in the whole collection, neither for the other thresholds.

Removing stop-words and using stemming was always done in all pre-processing sequences tested.

As we have mentioned in Section 3.4 the sequence of the techniques to apply to construct the data sets in Table 3.3 is not arbitrary. Depending on the options activated we must establish the order by which the techniques are applied.

To compute the number of combinations let us recall that we have 2 possible situations concerning NER (use/ not use). We have four possible situations for pruning (no pruning/ 5/ 10/ 50). We have 2 possible situations for handling synonyms (use/ not use). And finally we have 2 situations for word validation (validate/ not validate). Therefore the total is $2 * 4 * 2 * 2 = 32$ possible combinations shown in Table 3.3.

For this experiments we have used a set of algorithms available in the WEKA [HFH⁺09] tool and listed in Table 4.3. The WEKA tool is described in Chapter 2.

Table 3.2: Machine Learning algorithms used in the study.

Acronym	Algorithm	Type
smo	Sequential Minimal Optimization	Support Vector Machines
ibk	K-nearest neighbors	Instance-based learner
j48	Decision tree (C4.5)	Decision Tree learner

The quality of the classifiers were assessed by their Accuracy, True Positives and F-Measure. Table 3.3 characterizes the 32 data sets used in the experiments in terms of size, number of attributes and the experiments' different pre-processing techniques used.

Table 3.3: Pre-processing combinations and data sets characterization

DataSet ID	NER	Pruning	Handling Synonyms	Word Validation	Number of Attributes	Size (MB)
Dataset1	No	No	No	No	68057	32.6
Dataset2	Yes	Yes-P5	Yes	Yes	10411	43.4
Dataset3	Yes	Yes-P10	Yes	Yes	8264	42.8
Dataset4	Yes	Yes-P50	Yes	Yes	4488	41.5
Dataset5	Yes	No	No	No	120568	65.9
Dataset6	No	Yes-P5	No	No	26311	60.5
Dataset7	No	Yes-P10	No	No	17588	58.7
Dataset8	No	Yes-P50	No	No	7472	53.6
Dataset9	No	No	Yes	No	55691	59.2
Dataset10	No	No	No	Yes	15561	51.1
Dataset11	Yes	Yes-P5	No	No	27922	59.6
Dataset12	Yes	Yes-P10	No	No	18209	57.6
Dataset13	Yes	Yes-P50	No	No	7489	52.4
Dataset14	Yes	No	Yes	No	117848	62.0
Dataset15	No	Yes-P5	Yes	No	26330	62.0
Dataset16	No	Yes-P10	Yes	No	17611	60.2
Dataset17	No	Yes-P50	Yes	No	7484	55.0
Dataset18	No	No	Yes	Yes	12876	45.4
Dataset19	No	Yes-P5	No	Yes	12926	50.1
Dataset20	No	Yes-P10	No	Yes	10211	49.0
Dataset21	No	Yes-P50	No	Yes	5486	47.3
Dataset22	Yes	No	No	Yes	15558	50.0
Dataset23	No	Yes-P5	Yes	Yes	10454	44.4
Dataset24	No	Yes-P10	Yes	Yes	8303	43.8
Dataset25	No	Yes-P50	Yes	Yes	4534	42.5
Dataset26	Yes	Yes-P5	Yes	No	25784	55.5
Dataset27	Yes	Yes-P10	Yes	No	16570	53.6
Dataset28	Yes	Yes-P50	Yes	No	6658	48.9
Dataset29	Yes	No	Yes	Yes	12874	44.4
Dataset30	Yes	Yes-P5	No	Yes	12869	49.1
Dataset31	Yes	Yes-P10	No	Yes	10159	47.9
Dataset32	Yes	Yes-P50	No	Yes	5425	46.2

A complete and detailed list of the results can be found in Appendix A (tables A.1, A.2 and A.3). Table 3.4 summarizes the best results achieved. The 32 data sets built had a very large number of attributes. These results show that the application of pruning, stemming and handling synonyms, reduces significantly the number of attributes without decreasing significantly the accuracy as shown in Tables A.1, A.2

and A.3 in the Annexes. Analyzing the results we can say that the best results are achieved with the J48 algorithm.

Table 3.4: Results summary table.

Algorithms	Accuracy	True Positives	F-Measure
smo	95.19 (0.12) DS14	0.95 (0.001) DS14	0.95 (0.0011) DS14
ibk	71.92 (0.70) DS28	0.72 (0.0050) DS18	0.72 (0.0042) DS18
j48	95.67 (0.33) DS9	0.96 (0.0031) DS10,19,20,21	0.96 (0.0031) DS10,19,20,21,26,27,28

After this exhaustive study we could conclude that the best pre-processing techniques to apply were in the following order:

1. Handle Synonyms
2. Stop-words removal
3. Word validation using a dictionary
4. Stemming
5. Pruning

3.6 How to use BioTextRetriever

BioTextRetriever is a user friendly tool (Figure 3.2) that allows a biologist researcher to obtain a set of interesting scientific papers related to a set of sequences. The user must have an authenticated access to BioTextRetriever.

New Request


Attribute	Value	Help
Sequence	<pre>> h3 GAGAGAACCCACCATGGTGCTGCTCTGCTGCGACAGACCCAGGTCAGGCGCGCCT GGGG ATGTTTCTGTCTTCCACCCAGAGACTACTTCCGCACTTGGAGCTGAGCCAC GGC TCTGCCAGGTTAAGGGCCAGCCAGAGGGTGGCCGAGCCCTGACCCAGCCGG TGGCG CACTGGAGGACATGCCCAGCCGCTGTCCGCCCTGAGGAACTGCAAGGCGACAA GCTT CGGGTGGACCCGGTCACTTCAGCTCTAAGCCACTGCTGCTGGTGAACCTGG CGCC CACTTCCCGGCGSAGTTCACCCCTGCGGTGCAGCCCTCCCTGGACAGTTCTGGC TTCT GTGAGCACCGTCTGACCTCCAAATACGGTTAAGCTGGAGCCCTGGGCCATGC TTCTT GCCCTTGGGCTCCCGCAGCCCTCTCTCCCTTCTGCAACCGTACCCCGTGGT CTT TGAATAAAGTCTGAGTGGGGCGCA</pre>	The sequence must be in the FASTA format.
E-Value	<input type="text" value="0.001"/>	
Blast Type	<input type="text" value="blast"/> 	Blastp or Blastx
Ranking	Select the weights for the Ranking function that will order the final results.	Introduce a value between 0 and 100
Number of MeSH terms in common weight	<input type="text" value="0"/>	
Number of citations weight	<input type="text" value="0"/>	
H-Index weight	<input type="text" value="25"/>	
Impact Factor weight	<input type="text" value="75"/>	
Number of publications weight	<input type="text" value="0"/>	
Journal Factor Weight	<input type="text" value="0"/>	
<input type="button" value="Submit"/>		

Figure 3.2: Web Page where the user specifies a new request.

The sequence(s) introduced by the user must be in the FASTA²³. The e-value is a threshold for BLAST.

The NCBI BLAST tool searches and returns a set of similar sequences. Each similar sequence has a set of papers associated and an e-value. The papers associated to this set of sequences and with an associated e-value between 0.0 and the cutoff, are the most relevant ones. The lower the e-value is the more significant is the alignment. The e-value measures how many alignments with a given score are expected purely by chance [KYB03].

²³<http://blast.ncbi.nlm.nih.gov/blastcgihelp.shtml>

The e-value indicates the statistical significance of a given pairwise alignment, thus the lower the e-value, the more significant is the hit. This means that the lower the e-value the higher the chance that similarity between the e-value, query and database sequences is significant.

Although the user only introduces one cutoff e-value, that is used to determine the most relevant sequences/papers, the tool uses a second cutoff value that is configured in BioTextRetriever to determine the irrelevant sequences/papers resulting from the search (see Section 4.1.1). The user must also specify if it is a protein blast (blastp) or other type of blast covered by blastx. The following percentage parameters (Number of MeSH terms in common, Number of citations, h-index, Impact Factor, Number of publications and Journal Similarity) are detailed and explained in Section 5, although we have a default ranking formula the user may not use our parameters for the ranking formula and may specify himself the parameters he wants to. The parameters should be between 0 and 100, and the sum of all the parameters should be 100%.

After submitting a request the user can monitor it as shown in Figure 3.3.

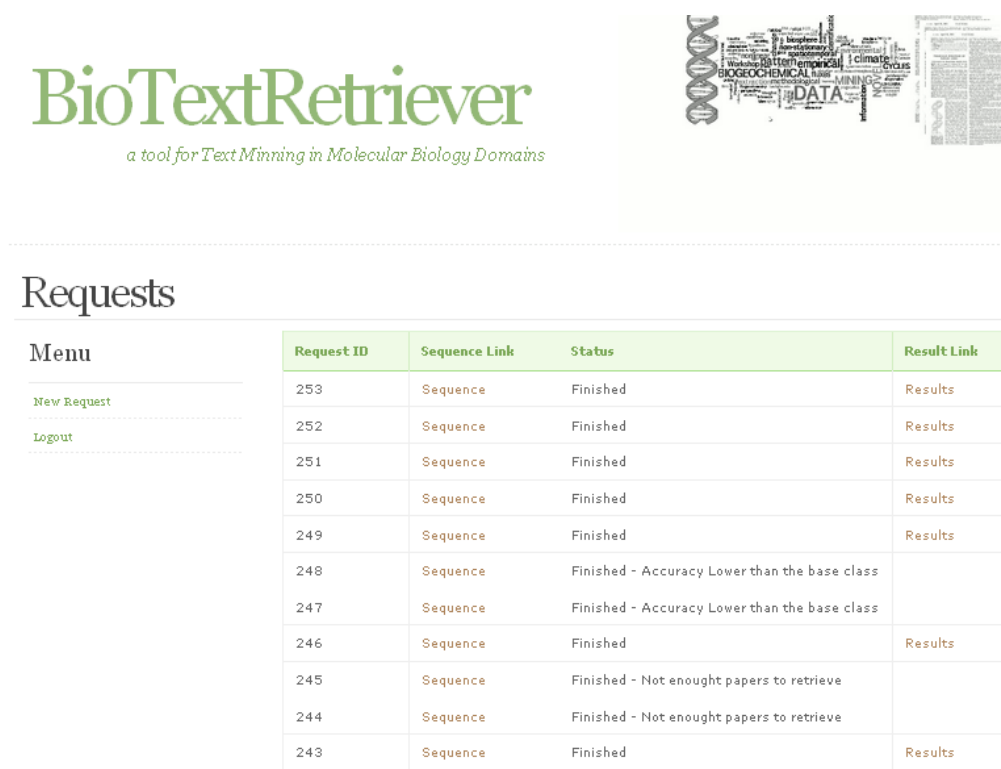


Figure 3.3: Web page where each user can monitor his requests.

When the request is finished the state "Finished" appears and a link for the results is provided. To visualize the results the user should click on the "Results" link and they are presented as in the Figure 3.4.



Request 253

Menu	PMID	Title	Ranking
Sequence	11181995	The sequence of the human genome.	89.7
New Request	11138001	Disruption of a new forkhead/winged helix protein, scurfin, results in the fatal	77.57
Requests	14702039	Complete sequencing and characterization of 21,243 full length human cDNAs.	76.82
Logout	10617205	T cell co stimulation through B7R1 and ICOS.	76.25
	7680768	Primary structure and functional expression of a mouse inward rectifier potassium	76
	7566098	Initial assessment of human gene diversity and expression patterns based upon 83	76
	10508524	The neuronal ceroid lipofuscinoses in human EPMP and mnd mutant mice are associated	75.82
	9916795	Identification of Cd36 (Fat) as an insulin resistance gene causing defective fat	75.82
	9054934	A photoreceptor cell specific ATP binding transporter gene (ABCR) is mutated in	75.82
	11452312	Insights into Wnt binding and signalling from the structures of two Frizzled cysteine	75.75

Figure 3.4: Web Page with links for each of the twenty results presented.

The pmid of each results has a link to that article in particular. A list with the five thousand best ranked papers is presented. For each article it is presented the pmid, the title of the article and the ranking score in descendent order of relevance. The user can consult more information about each article through the pmid that has a link to a web page with more information about that particular article as shown in Figure 3.5.

Each of these results page has the article title, abstract, PubMed id (with a link to the NCBI portal), the PubMed publication date, the Journal Impact Factor, number of citations of the article, and the tool's assigned score (ranking score). The field positive tells the expert if the article belongs or not, to the positive's articles returned by NCBI BLAST. The presented top 5000 results are ordered by a ranking function explained in Chapter 5. The user can also navigate through the 5000 returned papers. A list with 30 papers per page is presented to the user, that may navigate through the other pages returned.

The tool allows several requests being processed at the same time. In this case the user can also follow the state of his own requests in the home page through the request id.

Request 253

<p>Menu</p> <hr/> <p>Sequence</p> <hr/> <p>New Request</p> <hr/> <p>Request 253</p> <hr/> <p>Requests</p> <hr/> <p>Logout</p> <hr/>	<table border="1"> <tr> <td>PMID</td> <td>11181995</td> </tr> <tr> <td>Title</td> <td>The sequence of the human genome.</td> </tr> <tr> <td>Abstract</td> <td>A 2.91 billion base pair (bp) consensus sequence of the euchromatic portion of the human genome was generated by the whole genome shotgun sequencing method. The 14.8 billion bp DNA sequence was generated over 9 months from 27,271,853 high quality sequence reads (5.11 fold coverage of the genome) from both ends of plasmid clones made from the DNA of five individuals. Two assembly strategies a whole genome assembly and a regional chromosome assembly were used, each combining sequence data from Celera and the publicly funded genome effort. The public data were shredded into 550 bp segments to create a 2.9 fold coverage of those genome regions that had been sequenced, without including biases inherent in the cloning and assembly procedure used by the publicly funded group. This brought the effective coverage in the assemblies to eightfold, reducing the number and size of gaps in the final assembly over what would be obtained with 5.11 fold coverage. The two assembly strategies yielded very similar results that largely agree with independent mapping data. The assemblies effectively cover the euchromatic regions of the human chromosomes. More than 90% of the genome is in scaffold assemblies of 100,000 bp or more, and 25% of the genome is in scaffolds of 10 million bp or larger. Analysis of the genome sequence revealed 26,588 protein encoding transcripts for which there was strong corroborating evidence and an additional approximately 12,000 computationally derived genes with mouse matches or other weak supporting evidence. Although gene dense clusters are obvious, almost half the genes are dispersed in low G C sequence separated by large tracts of apparently noncoding sequence. Only 1.1% of the genome is spanned by exons, whereas 24% is in introns, with 75% of the genome being intergenic DNA. Duplications of segmental blocks, ranging in size up to chromosomal lengths, are abundant throughout the genome and reveal a complex evolutionary history. Comparative genomic analysis indicates vertebrate expansions of genes associated with neuronal function, with tissue specific developmental regulation, and with the hemostasis and immune systems. DNA sequence comparisons between the consensus sequence and publicly funded genome data provided locations of 2.1 million single nucleotide polymorphisms (SNPs). A random pair of human haploid genomes differed at a rate of 1 bp per 1250 on average, but there was marked heterogeneity in the level of polymorphism across the genome. Less than 1% of all SNPs resulted in variation in proteins, but the task of determining which SNPs have functional consequences remains an open challenge.</td> </tr> <tr> <td>Journal Title</td> <td>Science New York NY</td> </tr> <tr> <td>Date</td> <td>2001</td> </tr> <tr> <td>Impact Factor</td> <td>29.747</td> </tr> <tr> <td>Citations</td> <td>860</td> </tr> <tr> <td>Ranking</td> <td>89.7</td> </tr> </table>	PMID	11181995	Title	The sequence of the human genome.	Abstract	A 2.91 billion base pair (bp) consensus sequence of the euchromatic portion of the human genome was generated by the whole genome shotgun sequencing method. The 14.8 billion bp DNA sequence was generated over 9 months from 27,271,853 high quality sequence reads (5.11 fold coverage of the genome) from both ends of plasmid clones made from the DNA of five individuals. Two assembly strategies a whole genome assembly and a regional chromosome assembly were used, each combining sequence data from Celera and the publicly funded genome effort. The public data were shredded into 550 bp segments to create a 2.9 fold coverage of those genome regions that had been sequenced, without including biases inherent in the cloning and assembly procedure used by the publicly funded group. This brought the effective coverage in the assemblies to eightfold, reducing the number and size of gaps in the final assembly over what would be obtained with 5.11 fold coverage. The two assembly strategies yielded very similar results that largely agree with independent mapping data. The assemblies effectively cover the euchromatic regions of the human chromosomes. More than 90% of the genome is in scaffold assemblies of 100,000 bp or more, and 25% of the genome is in scaffolds of 10 million bp or larger. Analysis of the genome sequence revealed 26,588 protein encoding transcripts for which there was strong corroborating evidence and an additional approximately 12,000 computationally derived genes with mouse matches or other weak supporting evidence. Although gene dense clusters are obvious, almost half the genes are dispersed in low G C sequence separated by large tracts of apparently noncoding sequence. Only 1.1% of the genome is spanned by exons, whereas 24% is in introns, with 75% of the genome being intergenic DNA. Duplications of segmental blocks, ranging in size up to chromosomal lengths, are abundant throughout the genome and reveal a complex evolutionary history. Comparative genomic analysis indicates vertebrate expansions of genes associated with neuronal function, with tissue specific developmental regulation, and with the hemostasis and immune systems. DNA sequence comparisons between the consensus sequence and publicly funded genome data provided locations of 2.1 million single nucleotide polymorphisms (SNPs). A random pair of human haploid genomes differed at a rate of 1 bp per 1250 on average, but there was marked heterogeneity in the level of polymorphism across the genome. Less than 1% of all SNPs resulted in variation in proteins, but the task of determining which SNPs have functional consequences remains an open challenge.	Journal Title	Science New York NY	Date	2001	Impact Factor	29.747	Citations	860	Ranking	89.7
PMID	11181995																
Title	The sequence of the human genome.																
Abstract	A 2.91 billion base pair (bp) consensus sequence of the euchromatic portion of the human genome was generated by the whole genome shotgun sequencing method. The 14.8 billion bp DNA sequence was generated over 9 months from 27,271,853 high quality sequence reads (5.11 fold coverage of the genome) from both ends of plasmid clones made from the DNA of five individuals. Two assembly strategies a whole genome assembly and a regional chromosome assembly were used, each combining sequence data from Celera and the publicly funded genome effort. The public data were shredded into 550 bp segments to create a 2.9 fold coverage of those genome regions that had been sequenced, without including biases inherent in the cloning and assembly procedure used by the publicly funded group. This brought the effective coverage in the assemblies to eightfold, reducing the number and size of gaps in the final assembly over what would be obtained with 5.11 fold coverage. The two assembly strategies yielded very similar results that largely agree with independent mapping data. The assemblies effectively cover the euchromatic regions of the human chromosomes. More than 90% of the genome is in scaffold assemblies of 100,000 bp or more, and 25% of the genome is in scaffolds of 10 million bp or larger. Analysis of the genome sequence revealed 26,588 protein encoding transcripts for which there was strong corroborating evidence and an additional approximately 12,000 computationally derived genes with mouse matches or other weak supporting evidence. Although gene dense clusters are obvious, almost half the genes are dispersed in low G C sequence separated by large tracts of apparently noncoding sequence. Only 1.1% of the genome is spanned by exons, whereas 24% is in introns, with 75% of the genome being intergenic DNA. Duplications of segmental blocks, ranging in size up to chromosomal lengths, are abundant throughout the genome and reveal a complex evolutionary history. Comparative genomic analysis indicates vertebrate expansions of genes associated with neuronal function, with tissue specific developmental regulation, and with the hemostasis and immune systems. DNA sequence comparisons between the consensus sequence and publicly funded genome data provided locations of 2.1 million single nucleotide polymorphisms (SNPs). A random pair of human haploid genomes differed at a rate of 1 bp per 1250 on average, but there was marked heterogeneity in the level of polymorphism across the genome. Less than 1% of all SNPs resulted in variation in proteins, but the task of determining which SNPs have functional consequences remains an open challenge.																
Journal Title	Science New York NY																
Date	2001																
Impact Factor	29.747																
Citations	860																
Ranking	89.7																

Figure 3.5: Web Page with one of the relevant papers.

3.7 Summary

In this chapter we have presented BioTextRetriever, a Web-based search tool for retrieving relevant literature in Molecular Biology and related domains. We have made a general overview of the tool and described its detailed structure. We have also described and evaluated the pre-processing techniques in order to determine the most adequate combination of techniques to apply to BioTextRetriever. We conclude the chapter with an example that shows how the tool works.

Chapter 4

From Sequences to Papers

The core of BioTextRetriever is the construction of a classifier capable of selecting the relevant papers among the whole MEDLINE bibliographic database. “Relevant” papers, in this context, means papers related to the set of sequences that were provided as input to the tool. In this chapter we describe the methodology to construct such a classifier as well as a set of experiments that support the choices we have made. We describe in this chapter the procedure going from step 1 to 4 when using the tool as shown in Figure 3.1 of Chapter 3 until reaching our goal.

Before we address the classifier construction issue, we must address the construction of the data set. For most of the learning algorithms we must provide positive and negative examples, in our case both relevant and irrelevant papers. The way of obtaining the irrelevant papers is an important issue that we describe in this chapter.

4.1 Constructing a data set

We have created a SQL local database called *LDB* using MySQL software and we have also used data warehouse concepts to represent terms and papers in the database. Figure A.1 of the Annexes shows the tables and the structure of such a database.

Step 1 of BioTextRetriever, explained in Chapter 3, elicits the retrieval of the relevant papers associated with the similar sequences, according to the user provided e-value, together with some less similar papers.

Figure 4.1 shows how we obtain the relevant papers associated with all the original and similar sequences.

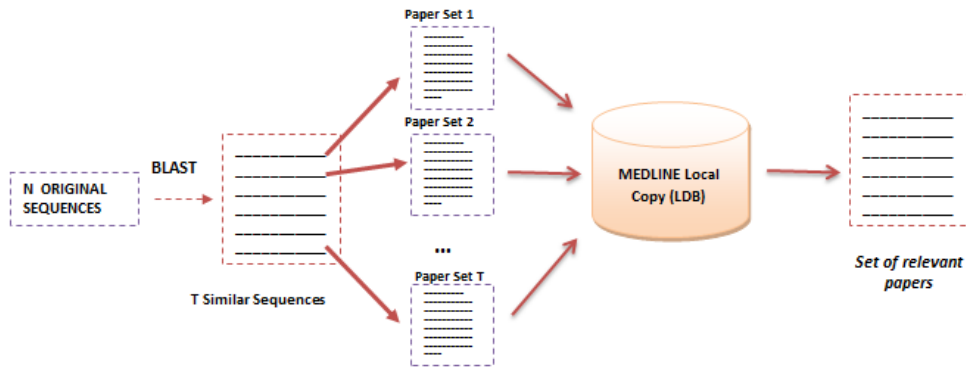


Figure 4.1: Collecting the relevant papers. Irrelevant papers can either be: i) associated with less similar sequences or; ii) randomly collected from MEDLINE papers that have MeSH terms in common with the relevant ones.

The relevant papers are considered to be the set of papers associated with the set of sequences with e-value below to the user provided threshold. This information is obtained directly from the LDB.

We believe that training the classifier with both relevant and irrelevant examples makes it more robust.

4.1.1 Complementing the data sets

Figure 4.2 shows how we obtain the final data set with relevant and irrelevant papers to construct the classifier.

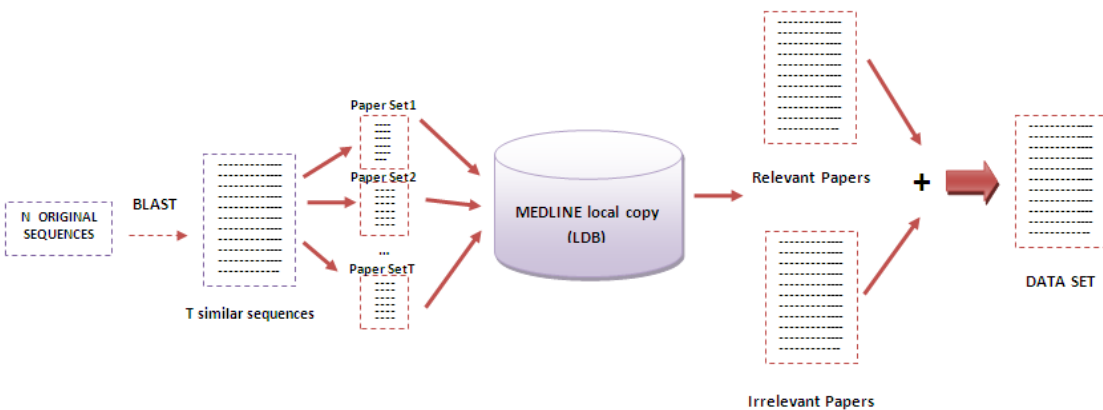


Figure 4.2: Generating a data set with relevant and irrelevant papers.

In this study we have empirically evaluated three different ways of obtaining the irrelevant papers, which we named as: Near-Miss Values (NMV), MeSH Random Values (MRV), and Random Values (RV). We will now explain what are these alternatives and how they are implemented.

To understand the process of selecting irrelevant papers we refer to Figure 4.3. The relevant papers are the ones associated with the sequences with an e-value lower than the e-value cutoff introduced by the user (ev). The left box represents the similar sequences that will be used to identify the relevant papers. To obtain the Near-Miss Values (NMV) we collect the papers associated with the similar sequences that have e-value above and far apart from the first threshold (ev) but close to the second threshold (β). In the experiments we have tried values from the set $\{ 1, 2, 5 \}$ for the second threshold (β). But first, we established a “no man’s land“ zone that is 10% of the number of “not similar“ sequences associated with the introduced sequence, if they exist. This “no man’s land“ zone is represented in Figure 4.3 by the gray box to better discriminate what are relevant and irrelevant papers. The papers associated with the sequences in this gray box are very close to the relevant papers so we discard them. Instead, we gather the ones that are farthest away from the relevant ones. In Figure 4.3 the box on the right represents the “not so near” sequences that provide the “near-misses” papers.

Near-Miss Values (NMV)

The examples in the box on the right of the Figure 4.3 are “near-misses” examples, because they are not considered relevant but have a certain degree of similarity to the sequences.

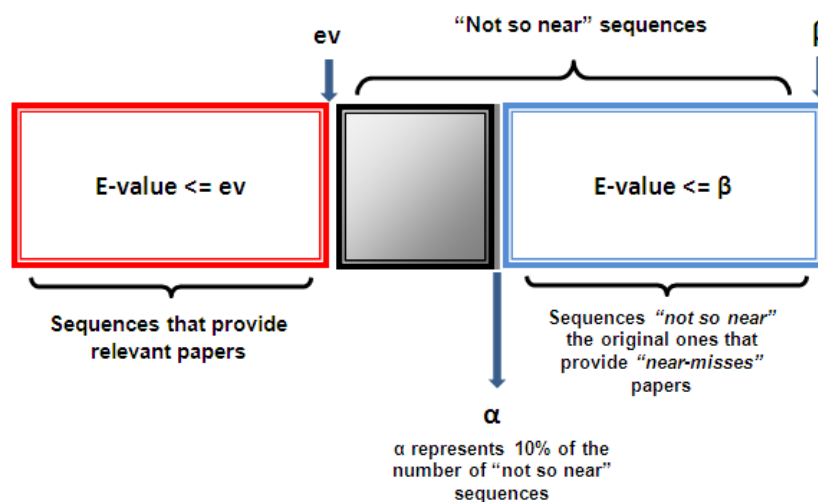


Figure 4.3: How “not relevant” (but “close”) papers are obtained. ev is the e-value threshold provided to discriminate the sequences that determine the relevant papers. α and β are parameters for the cut off limiting the “near-misses”.

In our experiments we have considered different thresholds, for both the values of ev and (β) to obtain the “near-misses“ examples. For the upper limit (β) we have considered values of 1, 2 and 5 in the experiments. The papers associated with the similar sequences with e-value less than ev are considered relevant (left box); the papers associated with the similar sequences that have an e-value greater than ev and less than $ev + 10\%$ ($\alpha = 10\%$) of the “not similar” sequences, which are represented in the gray box, are discarded; the papers associated to the similar sequences with e-values greater than ev plus (α) of the “not similar” sequences and less than (β) are considered “not relevant” papers (“near-misses“).

$$\left\{ \begin{array}{ll} \text{Sequences that provide Relevant Papers,} & \text{sequences with evalue} \leq ev \\ \text{Discarded sequences,} & ev \leq \text{sequences with evalue} \leq ev + 10\% \\ \text{Sequences that provide } near\text{-misses' papers,} & ev + 10\% \leq \text{sequences with evalue} \leq \beta \end{array} \right.$$

In Tables 4.1 and 4.2 the names of the data sets include the different threshold values. For example, the data set named “EC15NMV” means that we have used the sequence EC1, the second digit “5” is the (β) value and NMV was the technique used.

The NMV method for producing a second class of less relevant papers was used on data sets that have a minimum number of negative examples. After a few experiments we have established a minimum number of 10 examples in order for this approach to be usable.

MeSH Random Values (MRV)

Some set of sequences do not have the established minimum number of irrelevant papers associated. To overcome this situation we have the MeSH Random Values (MRV) alternative. This alternative is adopted when we do not have a sufficient number of irrelevant papers for the classifier to learn but still have some irrelevant papers generated by the NMV method. Thus in this case, the irrelevant papers are obtained combining the existing number of “near-miss“ values (if they exist and are few) with some N randomly selected irrelevant papers generated by the LDB. However, these irrelevant papers must have the maximum number of MeSH terms in common with the relevant papers. The idea is to generate papers that although being far apart (as far as “e-value“ criteria is concerned), still have something in common with the relevant papers to improve classifier robustness. We guarantee that in these randomly chosen papers there is none of the relevant ones. The number of irrelevant papers generated in this case plus the existing near-misses is equal to the number of relevant papers.

Random Values (RV)

This RV approach randomly chooses papers from MEDLINE in a number equal to the number of relevant papers. This option is the particular case of MRVs with “near-misses” examples equal to zero. We also guarantee that in this set of irrelevant papers there are no relevant papers (from the ones associated to the original sequences). And these randomly selected papers have also the maximum number of MeSH terms in

common with the positive relevant papers.

4.1.1.1 Original sequences characterization

As standard procedure in Machine Learning we have generated several data sets based on sequences that belong to nine different domains, with the following distribution:

- RNASES: 2 sequences
- Escherichia Coli: 2 sequences
- Cholesterol: 1 sequence
- Hemoglobin: 1 sequence
- Blood Pressure: 2 sequences
- Erythrocytes: 1 sequence
- Hypertension: 3 sequences
- Blood Glucose: 3 sequences
- Lung Disease: 1 sequence

The data sets used are characterised in Tables 4.1 and 4.2. Table 4.1 characterizes data sets for which the negative examples consist only of “*near-miss*” examples. Table 4.2 characterizes the data sets for which there were not enough “*near-miss*” examples and therefore we have used the MRV and RV strategies.

Table 4.1: Characterization of data sets of the NMV type.

Data Sets	Number of Attributes	Positive Examples	Negative Examples	Total Examples
T15	1040	52	29	81
BP15	1270	88	26	114
BP25	1270	80	10	90
EC45	826	37	45	82
EC15	789	28	12	40
C35	611	24	14	38
HYP65	1240	67	19	86

Table 4.2: Characterization of data sets of types MRV and RV. In MRV the data sets of Table 4.1 were completed with randomly selected papers. In RV all "negative examples" are randomly selected.

Data Sets	Number of Attributes	Positive Examples	Negative Examples	Total Examples
T15	1040	52	52	104
BP15	1270	88	88	176
BP25	1270	80	80	160
EC45	826	37	37	74
EC15	789	28	28	56
C35	611	24	24	48
HYP65	1240	67	67	134

4.1.2 Assessing the method of choosing irrelevant papers

We have evaluated the three alternatives proposed to obtain the irrelevant papers by generating several data sets for each of them.

For this experiments we have used a set of algorithms available in the WEKA [HFH⁺09] tool and listed in Table 4.3. The WEKA tool was described in Chapter 2.

Table 4.3: Machine Learning algorithms used in the study.

Acronym	Algorithm	Type
ZeroR	Majority class predictor	Rule learner
smo	Sequential Minimal Optimization	Support Vector Machines
rf	Random Forest	Ensemble
ibk	K-nearest neighbours	Instance-based learner
BayesNet	Bayesian Network	Bayes learner
j48	Decision tree (C4.5)	Decision Tree learner
dtnb	Decision table/Naïve bayes hybrid	Rule learner
AdaBoost	Boosting algorithm	Ensemble learner
Bagging	Bagging algorithm	Ensemble learner
Ensemble Selection	Combines several algorithms	Ensemble learner

All the experiments were carried out on a two quad-core Xeon 2.4 GHz with 16 GB of RAM, running Linux Ubuntu 8.10.

Tables 4.4, 4.5 and 4.6 show the accuracy results obtained using the classifiers of Table 4.3. Accuracy results were estimated performing a 10-fold Cross Validation. Cross-Validation is a widely used statistical method for evaluating and comparing learning algorithms [SSLM11]. The basic idea of cross-validation is to divide the data into two segments: one to learn or train the model and another one to validate the model.

Table 4.7 presents the overall average for the three methods.

Table 4.4: Accuracy results (%) in NMV data sets. In all data sets the best value isn't different from the second best value in a statistically significant way, according to the t-student test ($\alpha = 0.05$) [SAC07]. The t-student test is a widely used method for comparing the means of two samples.

Data Set	ZeroR	smo	rf	ibk	BayesNet	j48	dtnb
T15NMV	64.2	32.1 (15.7)	31.0 (14.9)	45.7 (18.6)	64.0 (11.3)	64.0 (11.3)	60.7 (11.7)
BP15NMV	75.5	59.4 (16.3)	71.9 (12.0)	45.5 (19.0)	75.5 (10.2)	75.5 (10.2)	72.7 (11.9)
BP25NMV	88.9	85.6 (10.5)	86.7 (11.5)	88.9 (10.5)	88.9 (10.5)	88.9 (10.5)	88.9 (10.8)
EC45NMV	54.9	38.3 (20.1)	47.4 (22.5)	53.1 (21.6)	54.3 (21.8)	58.6 (14.0)	52.2 (13.0)
EC15NMV	70.0	55.0 (25.8)	80.0 (25.8)	35.0 (21.1)	62.5 (21.3)	77.5 (29.9)	70.0 (15.8)
C35NMV	63.2	40.0 (19.2)	52.5 (20.1)	60.0 (25.4)	59.2 (26.5)	66.7 (28.9)	61.7 (26.7)
HYP65NMV	77.9	65.3 (10.0)	73.5 (13.1)	78.2 (11.4)	76.0 (13.4)	81.8 (15.5)	80.7 (17.1)
Overall Average		53.7 (16.8)	63.3 (17.1)	58.0 (18.2)	68.6 (16.4)	73.3 (17.2)	69.6 (15.2)

Table 4.5: Accuracy results (%) in MRV data sets. In all data sets the best value isn't different from the second best value in a statistically significant way, according to the t-student test ($\alpha = 0.05$).

Data Set	ZeroR	smo	rf	ibk	BayesNet	j48	dtnb
T15MRV	48.1	44.8 (14.8)	47.8 (14.0)	50.8 (12.1)	57.7 (12.3)	71.4 (16.8)	66.6 (10.9)
BP15MRV	50.0	62.5 (11.4)	66.9 (15.0)	51.3 (18.4)	70.0 (12.8)	74.4 (13.0)	67.5 (11.3)
BP25MRV	50.0	73.8 (9.7)	80.0 (14.4)	53.1 (17.0)	78.8 (15.1)	78.8 (12.9)	79.4 (11.8)
EC45MRV	45.9	35.0 (12.7)	42.1 (24.1)	50.4 (21.7)	37.9 (8.1)	61.4 (22.5)	56.4 (13.1)
EC15MRV	46.4	53.0 (23.5)	74.7 (21.6)	51.3 (17.8)	76.0 (21.5)	72.3 (18.1)	86.0 (16.4)
C35MRV	41.7	52.0 (23.0)	67.0 (30.6)	51.0 (30.6)	31.5 (20.0)	60.5 (23.9)	56.5 (19.4)
HYP65MRV	47.8	53.8 (17.2)	60.2 (16.3)	51.0 (16.8)	66.1 (15.5)	76.9 (10.3)	72.3 (9.7)
Overall Average		53.6 (16.0)	62.7 (19.4)	51.3 (14.4)	59.7 (15.0)	70.8 (16.8)	69.2 (13.3)

Table 4.6: Accuracy results (%) in RV data sets. In all data sets the best value isn't different from the second best value in a statistically significant way, according to the t-student test ($\alpha = 0.05$).

Data Set	ZeroR	smo	rf	ibk	BayesNet	j48	dtnb
T15RV	48.1	86.5 (8.2)	88.5 (8.4)	52.8 (12.6)	85.6 (10.6)	88.5 (12.1)	90.5 (13.1)
BP15RV	50.0	89.4 (8.4)	92.5 (8.2)	58.8 (16.7)	90.0 (9.4)	91.3 (9.4)	87.5 (10.2)
BP25RV	50.0	81.3 (9.8)	92.5 (7.7)	60.6 (14.5)	89.4 (11.0)	90.0 (10.3)	89.4 (8.9)
EC45RV	45.9	84.8 (10.7)	79.3 (14.8)	52.0 (15.2)	78.0 (16.7)	87.9 (8.0)	75.5 (14.4)
EC15RV	46.4	69.3 (15.5)	81.7 (15.6)	51.3 (17.8)	80.0 (17.1)	87.3 (15.1)	85.7 (16.5)
C35RV	41.7	82.0 (17.5)	87.5 (17.2)	57.0 (29.7)	85.5 (16.7)	81.5 (17.7)	78.5 (20.3)
HYP65RV	47.8	73.1 (10.0)	79.1 (9.5)	57.5 (11.7)	81.2 (13.9)	79.8 (6.1)	79.8 (13.3)
Overall Average		80.9 (11.4)	85.9 (11.6)	55.7 (16.9)	84.2 (13.6)	86.6 (11.2)	83.8 (13.8)

Table 4.7: Algorithm's Accuracy (%) average for NMVs, RMVs and RVs. There is statistical significance (using t-student test ($\alpha = 0.05$)) between the best and second best methods.

Algorithms	NMVs	MRVs	RVs
smo	53.7 (16.8)	54.0 (16.0)	80.9 (11.4)
rf	63.3 (17.1)	62.7 (19.4)	85.9 (11.6)
ibk	58.0 (18.2)	51.3 (14.4)	55.7 (16.9)
BayesNet	68.6 (16.4)	59.7 (15.0)	84.2 (13.6)
J48	73.3 (17.2)	70.8 (16.8)	86.6 (11.2)
dtnb	69.6 (15.2)	69.2 (13.3)	83.8 (13.8)
Overall Average	64.4 (16.8)	61.3 (15.8)	79.5 (13.1)

Figure 4.4 shows a graphical representation of the accuracy average of the three proposed methods: Near-Miss Values (NMV), MeSH Random Values (MRV) and Random Values (RV) presented in Table 4.7. The overall average for each method involves the averages obtained for the six proposed basic algorithms.

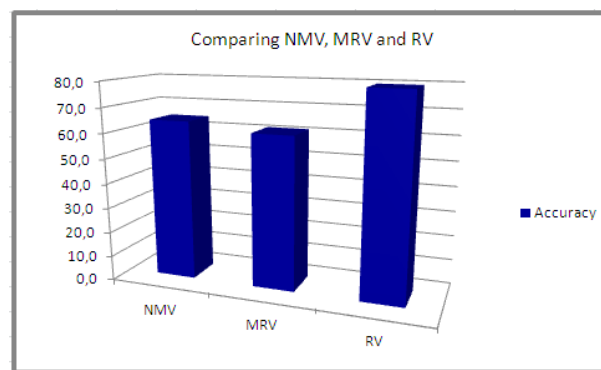


Figure 4.4: The accuracy average of the three proposed methods: Near-Miss Values (NMV), MeSH Random Values (MRV) and Random Values (RV).

From the result tables we may conclude that the best results are generated using the RV method. This because there is statistical significance, according to the t-student test ($\alpha = 0.05$), when comparing the NMV and RV results and the MRV and RV results. However, there is no statistical significance between NMV and MRV. The results suggest that generating randomly the negative examples produces better results.

After these comparative studies and from now on the data sets used are all of the RV type. As the number of examples are few we have chosen another set of sequences with more examples to pursue the following experiments.

4.1.3 Classifier Construction Process

Once we have decided how to collect the relevant and irrelevant papers we can now address the question of how to construct a classifier to filter relevant papers in MEDLINE. To address this problem we have considered to combine different partitions of the original data set with different ways of using the Machine Learning algorithms (either isolated or in an ensemble of classifiers).

We have considered and evaluated five possible alternatives for these combinations. We now describe the different alternatives and present their evaluation in Section 4.2. The different alternatives considered different amounts of data to construct the classifiers and the use of single classifiers or ensembles of classifiers. In all cases a 10-fold cross-validation method was used to evaluate the alternatives. In all experiments the base algorithms are the ones in Table 4.3.

We have developed and used a wrapper to tune each of the Machine Learning algorithms' parameters and also to apply different parameters for each of the stand alone algorithms.

In the following presented five alternatives we have used T for representing the number of basic algorithms, which is 6 in our case. And M represents the number of Ensemble learners which is 3 in our case.

Alternative 1

Figure 4.5 shows our first alternative where a single algorithm is applied to the whole data.

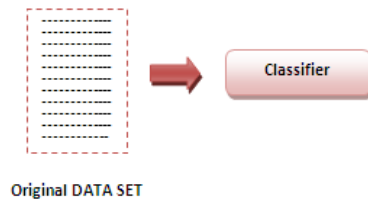


Figure 4.5: Alternative 1 consisting in the use of a single classifier applied to the whole data.

Alternative 2

In Alternative 2 (shown in Figure 4.6) we have used the whole data and the use of an ensemble of T basic classifiers (C_i) when each basic classifier uses the whole data. For the Ensemble we have used three well known algorithms: AdaBoost, Bagging and Ensemble Selection. Different "ensemble parameters" were tested as explained latter on Section 4.2 when the experimental evaluation of this alternative is described in detail.

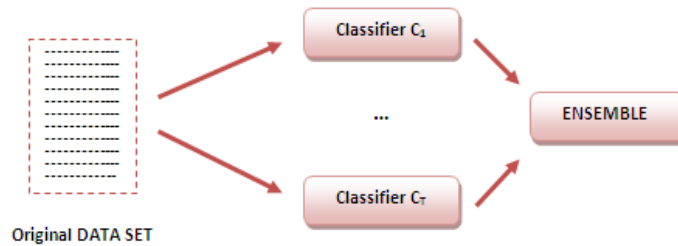


Figure 4.6: Alternative 2 considers the use of an ensemble when basic classifiers are built using the whole data.

Alternative 3

The third alternative, that is shown in Figure 4.7, divides the original data set into T equal parts, where T is the number of basic classifiers. To obtain T balanced data sets in terms of positive and negative examples there has been the implementation of an algorithm that divides the original data set into T balanced data sets. We will execute each sub data set with a different learning algorithm. In the end we ensemble the results of the T classifiers adding a voting scheme algorithm to the wrapper. We have implemented the simple plurality vote scheme [DZ04], where each base classifier assigns a vote for its prediction, and the example is classified in the class with more votes.

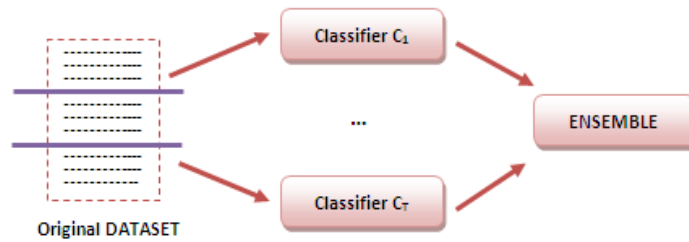


Figure 4.7: Alternative 3 makes a partition on the data set and each sub set of the data is used to train a specific classifier. The set of classifiers are combined in an ensemble.

Alternative 4

Figure 4.8 shows the fourth alternative that also divides the original data set into T equal parts as in alternative 3. But in this alternative each part of the original data will be used to train all the different learning algorithms, and the ensemble is made in the end. Thus we will train T classifiers in each part of the original data set and in the end we will ensemble $T * T$ classifiers. Like alternative 3 we have used the same simple plurality voting scheme for the ensemble algorithm.

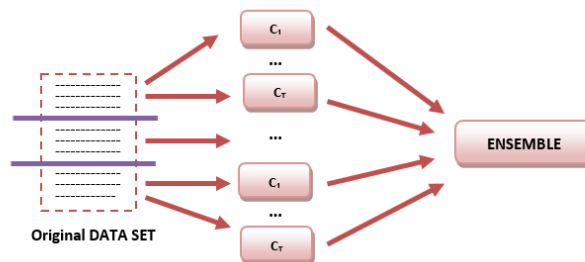


Figure 4.8: Alternative 4 makes a partition on the data set and each sub set of the data is trained with all the learning algorithms (T classifiers) and in the end it is made an ensemble of the $T * T$ classifiers.

Alternative 5

The fifth alternative (shown on Figure 4.9), divides the data set into M equal parts (where M is the number of ensemble algorithms, which is 3 in our case). For each sub set an ensemble is made using the best parameters of the basic algorithms (obtained in alternative 1). In the end, we make an ensemble of these M ensembles using a voting schema.

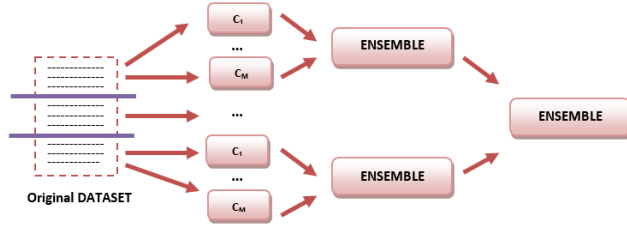


Figure 4.9: Alternative 5 makes a partition on the original data set and for each sub set a different ensemble is used with the best basic algorithms obtained in alternative 1. In the end, an ensemble is made of the M ensembles.

4.2 Evaluating the Alternatives

We have made a set of experiments, using propositional and relational algorithms that are now described. The ILP was only used in alternative 1 because of its results as explained latter on. The experiments cover the 5 alternatives referred in Section 4.1.3.

The sequences used in our experiments are characterized in Table 4.8. In all experiments a 10-fold cross-validation was made. We have used different data sets from the ones used in Section 4.1.2 because of the lower number of examples. The data sets used for the experiments have more than 150 (positive and negative) examples.

Table 4.8: Characterization of data sets used to assess the 5 alternatives.

Data Sets	Number of Attributes	Positive Examples	Negative Examples	Total Examples
S12RV	1602	128	128	256
H11RV	1461	120	120	240
ERYT21RV	1592	118	118	236
HYP11RV	1706	130	130	260
HYP21RV	1944	194	194	388
BG11RV	1546	97	97	194
BG21RV	1631	115	115	230
BG31RV	1859	149	149	298
LUNG21RV	1535	120	120	240

4.2.1 Alternative 1

In the first alternative basic algorithms were used with the whole data and their results compared. We have used the WEKA classifiers: smo, rf, ibk, bayesnet, j48, and dtnb. This first alternative allows the evaluation of stand alone algorithm's performance

with the data. Within this alternative we have also considered the use of the ILP system Aleph.

Propositional Learners Results

Table 4.9 shows the accuracy results obtained using the WEKA classifiers of Table 4.3.

Table 4.9: Accuracy results (%) in data sets. In all data sets the best value isn't different from the second best value in a statistically significant way, according to the t-student test ($\alpha = 0.05$).

Data Set	ZeroR	smo	rf	ibk	BayesNet	j48	dtNB
S12RV	49.6	86.1 (10.3)	92.7 (4.2)	53.3 (23.0)	94.2 (12.5)	94.2 (12.5)	97.5 (7.9)
H11RV	50.0	88.8 (6.5)	94.2 (5.3)	66.3 (17.5)	95.4 (5.0)	96.7 (5.1)	99.2 (1.8)
ERYT21RV	50.0	81.0 (8.2)	91.1 (5.6)	54.3 (9.6)	90.7 (6.3)	95.4 (4.2)	91.6 (4.4)
HYP11RV	50.0	81.9 (5.5)	86.5 (5.2)	55.4 (6.1)	91.2 (4.1)	87.3 (6.6)	84.6 (6.3)
HYP21RV	49.0	91.3 (3.2)	93.8 (4.4)	59.3 (6.4)	93.2 (3.8)	92.5 (4.6)	90.2 (4.3)
BG11RV	48.5	86.6 (9.5)	92.2 (4.5)	54.6 (12.9)	93.3 (3.5)	93.2 (3.6)	93.2 (3.6)
BG21RV	47.8	85.7 (8.5)	92.2 (5.3)	51.3 (10.6)	93.5 (4.7)	94.4 (5.0)	95.7 (3.6)
BG31RV	49.7	84.6 (7.6)	88.3 (3.9)	53.7 (11.0)	91.6 (4.0)	91.0 (5.5)	91.3 (5.0)
LUNG21RV	50.0	83.8 (8.2)	90.8 (6.8)	66.3 (13.0)	90.8 (5.5)	88.3 (6.8)	91.3 (5.0)
Overall Average	49.4	85.5 (7.5)	91.3 (5.0)	57.2 (12.2)	92.7 (5.5)	92.6 (6.0)	92.7 (4.7)

As a global result we can see that all algorithms have, in general, very good performances, well above the majority class predictor (that is around 50%). We can also conclude that *BayesNet*, *dtNB*, *j48* and *rf* are the best algorithms. According to the t-student test ($\alpha = 0.05$) there is however no statistical difference between them. We can also conclude that the *ibk* algorithm performed worse.

ILP Experimental Results

Within this first alternative we have considered the use of an ILP system [FCS⁺03]. An ILP system learns a theory using positive and negative examples. The theory is encoded as a logic program (a set of definitive clauses) and is learned using a set of examples and the background knowledge. The background knowledge represents the domain knowledge and is also represented as a logic program.

The Aleph [Sri] ILP system was applied to the same data sets. All the information was encoded in Prolog. To estimate the predictive quality of the classification models we performed 4 fold cross-validation.

The set of files built to give to the Aleph system were generated from the ARFF files and from the information stored in our local MEDLINE database, which we have used

in our previous experiments using propositional-based algorithms. We have encoded the features used in the propositional learners as background knowledge and have added a set of additional predicates (listed in Table 4.10).

We now describe the extra predicates encoded exclusively for the ILP experiments.

Table 4.10: Prolog predicates used.

Predicate	Description
<i>inText/2</i>	identifies if the word is in text.
<i>contiguous2Words/3</i>	detects if two words are contiguous, in the title or abstract.
<i>contiguous3Words/3</i>	detects if three words are contiguous, in the title or abstract.
<i>bagFrequentWords/4</i>	gets the word frequency in the text (title and abstract).
<i>bagGoodIdfWords/4</i>	gets the inverse document frequency of words in the text (title and abstract).
<i>diffWrds/2</i>	verifies if two words are different.
<i>countWords/2</i>	counts the total number of words in the title or abstract.

The predicate *inText/2* verifies if a word is or not in the text of the document. The predicates *contiguous2Words* and *contiguous3Words* verify if two or three words are contiguous. If two/three words appear several times in the text as contiguous it means that they are somehow closely related, thus these words should be given more importance if they appear contiguous in the text rather than isolated. The predicate *bagFrequentWords/4* calculates the word frequency in the text (title and abstract). The predicate *bagGoodIdfWords/4* gets the inverse document frequency of words in the text (title and abstract) The predicate *diffWrds/2* detects if two words are different. The predicate *countWords/2* counts the total number of words (title and abstract).

4.2.1.1 ILP Results

ILP results for all the data sets used in the experiments are shown in Table 4.11.

Table 4.11: ILP results over the same data sets used with the propositional algorithms.

Data Sets	Accuracy	True Positives	F-Measure
S12RV	91.9 (3.3)	89.8 (4.8)	92.1 (3.3)
H11RV	90.8 (5.2)	92.4 (4.0)	90.8 (5.2)
ERYT21RV	89.5 (6.0)	87.7 (9.4)	89.9 (5.2)
HYP11RV	84.3 (3.2)	86.8 (3.8)	84.6 (2.9)
HYP21RV	91.2 (3.5)	88.0 (3.0)	91.5 (3.5)
BG11RV	90.6 (5.3)	90.7 (7.0)	90.7 (5.3)
BG21RV	91.8 (3.2)	90.0 (2.5)	91.9 (3.1)
BG31RV	86.9 (3.4)	83.8 (2.6)	87.3 (3.6)
LUNG21RV	88.7 (3.2)	86.0 (3.5)	89.1 (3.1)
Overall Average	89.5 (4.0)	88.4 (4.5)	89.8 (3.9)

4.2.1.2 Comparing propositional and ILP results

To compare the propositional and ILP results obtained, we have used the statistical t-student test.

Table 4.12 presents the accuracy (%) results between all the propositional algorithms and ILP results.

Table 4.12: Comparing ILP and propositional results using accuracy (%).

Data Sets	ZeroR	smo	rf	ibk	bayesnet	j48	dtmb	ILP
S12RV	49.6	86.1(10.3)	92.7(4.2)	53.3(23.0)	94.2(12.5)	94.2(12.5)	97.5(7.9)	91.9(3.3)
H11RV	50.0	88.8(6.5)	94.2(5.3)	66.3(17.5)	95.4(5.0)	96.7(5.1)	99.2(1.8)	90.8(5.2)
ERYT21RV	50.0	81.0(8.2)	91.1(5.6)	54.3(9.6)	90.7(6.3)	95.4(4.2)	91.6(4.4)	89.5(6.0)
HYP11RV	50.0	81.9(5.5)	86.5(5.2)	55.4(6.1)	91.2(4.1)	87.3(6.6)	84.6(6.3)	84.3(3.2)
HYP21RV	49.0	91.3(3.2)	93.8(4.4)	59.3(6.4)	93.2(3.8)	92.5(4.6)	90.2(4.3)	91.2(3.5)
BG11RV	48.5	86.6(9.5)	92.2(4.5)	54.6(12.9)	93.3(3.5)	93.2(3.6)	93.2(3.6)	90.6(5.3)
BG21RV	47.8	85.7(8.5)	92.2(5.3)	51.3(10.6)	93.5(4.7)	94.4(5.0)	95.7(3.6)	91.8(3.2)
BG31RV	49.7	84.6(7.6)	88.3(3.9)	53.7(11.0)	91.6(4.0)	91.0(5.5)	91.3(5.0)	86.9(3.4)
LUNG21RV	50.0	83.8(8.2)	90.8(6.8)	66.3(13.0)	90.8(5.5)	88.3(6.8)	91.3(5.0)	88.7(3.2)
Overall Average	49.4	85.5(7.5)	91.3(5.0)	57.2(12.2)	92.7(5.5)	92.6(6.0)	92.7(4.7)	89.5(4.0)

The accuracy results obtained in Table 4.12 are very similar. The t-student test ($\alpha = 0.05$) gave no statistical significant difference between *bayesnet*, *dtmb*, *j48* and *rf*. Thus according to these results ILP is the fourth best algorithm.

Figure 4.10 presents a graphical representation of the overall used algorithms. This graphic also highlights that the best algorithms in terms of accuracy are the *bayesnet*, *dtmb*, *j48* and *rf*.

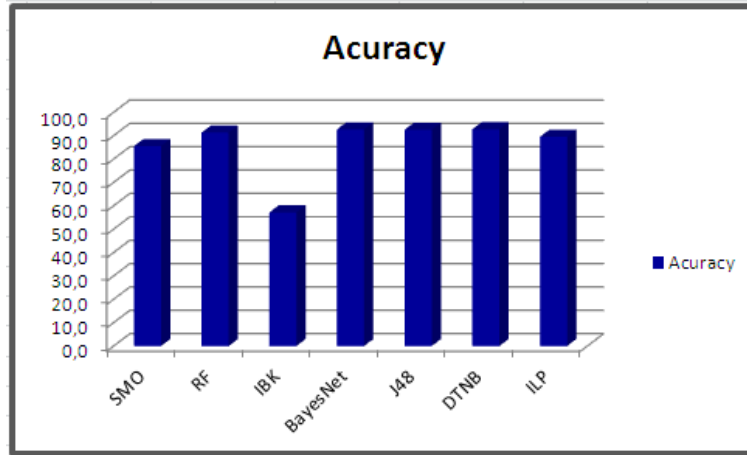


Figure 4.10: Average accuracies of the algorithms in comparison.

Table 4.13 presents, the ranked position of each algorithms' performance. The algorithm with the highest accuracy is assigned the first position, the following is assigned the second, and so on until a maximum of 7 (worst one).

Table 4.13: Ranking of the algorithms in comparison.

Data Sets	smo	rf	ibk	bayesnet	j48	dtnb	ILP
T11RV	6	4	7	2	2	1	5
S12RV	6	4	7	3	2	1	5
H11RV	6	3	7	4	1	2	5
ERYT21RV	6	3	7	1	2	4	5
HYP11RV	4	1	7	2	3	6	5
HYP21RV	6	4	7	1	2	2	5
BG11RV	6	4	7	3	2	1	5
BG21RV	6	4	7	1	3	2	5
LUNG21RV	6	3	7	2	5	1	4
Overall Average	5.8(0.7)	3.3(1.0)	7.0(0.0)	2.1(1.1)	2.4(1.3)	2.2(1.7)	4.9(0.3)
Overall Position	6	4	7	1	3	2	5

From the results presented in Table 4.13 we can conclude that the best algorithms are *bayesnet*, *dtnb* and *J48* because according to the t-student test ($\alpha = 0.05$) there is no statistically significant difference between these two values. There is statistically significant difference for *J48* and *ILP*. The *ILP* is the fifth best algorithm.

Regarding the results we have decided not to use *ILP* in the following alternatives because we did not achieve very good results in this first alternative.

4.2.2 Alternative 2

The second alternative is quite similar to the first one. The difference is that an ensemble of the basic classifiers is used. This alternative uses the best parameter

combinations found in alternative 1 for each basic classifier. In the end the ensemble is made using the WEKA's ensemble classifiers Bagging, AdaBoost and Ensemble Selection.

For the Bagging and AdaBoost classifiers we need to specify a base learner. We have used the experimental work with the base classifiers described previously in alternative 1. For each base classifier and data set we have used the parameter combinations that achieved the best results. We have used and developed a wrapper for the ensembles that automatically tunes ensemble-level parameters. The Ensemble Selection algorithm allows us to specify the set of base learners as well as the best options for each individual learner. Table 4.14 shows the results obtained.

Table 4.14: Ensemble's Accuracy results in alternative 2. The bold values are statistically different from the second best values according to the t-student test ($\alpha = 0.05$).

Data Sets	AdaBoost	Bagging	Ensemble Selection
S12RV	99.2 (1.7)	98.1 (3.3)	97.3 (3.7)
H11RV	99.2 (1.8)	99.2 (1.8)	96.3 (5.0)
ERYT21RV	95.4 (4.2)	95.8 (3.9)	93.3 (4.5)
HYP11RV	91.2 (4.1)	90.0 (6.3)	91.2 (4.1)
HYP21RV	95.1 (1.9)	95.1 (3.3)	89.7 (6.5)
BG11RV	93.8 (3.4)	94.3 (4.6)	94.3 (5.1)
BG21RV	95.7 (3.6)	94.4 (4.6)	93.9 (5.1)
BG31RV	93.9 (3.2)	91.6 (4.8)	92.3 (5.0)
LUNG21RV	93.8 (4.1)	92.5 (4.7)	92.5 (4.3)
Overall Average	95.3 (3.1)	94.6 (4.1)	93.4 (4.8)

As a global result we can see that all algorithms have in general very good performance, well above the majority class predictor (see the ZeroR result in Table 4.12). As expected through the literature [Die00b] ensemble learners have a higher and uniform performance than base learners.

The three ensemble learners used (Bagging, AdaBoost and Ensemble Selection) according to the t-student test ($\alpha = 0.05$) have no statistically significant difference. Thus we conclude that one can use either one of the three proposed ensemble learners.

4.2.3 Alternative 3

The alternative 3 applies the following steps to obtain the presented results. In the first step we apply cross-validation to the original data set. The second step divides each of the data sets into train/test sets. The third divides the data set (the train data sets) into T equal parts, where T is the number of classifiers. To each of the T parts one of the base learning algorithms of alternative 1 was used creating a model. In the end the models were used in the test parts created in the second step. Then we make an ensemble of the T classification results through the wrapper-ensemble developed.

This wrapper-ensemble implements the voting algorithm on the ensemble. We have implemented the simple plurality vote scheme [DZ04], where each base classifier assigns a vote for its prediction, and the example is classified in the class with more votes. Table 4.15 presents the results obtained, and from which we can conclude that the accuracy of this alternative is around 87%.

Table 4.15: Alternative 3 Results

Data Set	Voting
S12RV	89.1 (10.9)
H11RV	91.8 (5.4)
ERYT21RV	89.7 (3.4)
HYP11RV	78.6 (4.8)
HYP21RV	92.0 (1.4)
BG11RV	87.4 (4.3)
BG21RV	84.4 (2.5)
BG31RV	86.3 (1.1)
LUNG21RV	84.4 (6.2)
Overall Average	87.1 (4.4)

4.2.4 Alternative 4

The fourth alternative, like alternative 3, divides the data set into T equal parts where T is the number of classifiers. The difference from alternative 3 is that in alternative 4, T classifiers are constructed for each of the T parts. $T * T$ classifiers are constructed in total. In the end we make an ensemble using the voting algorithm of the wrapper-ensemble. Table 4.16 presents the results obtained and from which we can conclude that the accuracy of this alternative is around 89%.

Table 4.16: Alternative 4 Results.

Data Set	Voting
S12RV	87.8 (11.3)
H11RV	94.6 (1.2)
ERYT21RV	93.1 (3.1)
HYP11RV	80.5 (4.8)
HYP21RV	94.5 (0.8)
BG11RV	89.9 (0.2)
BG21RV	85.1 (2.1)
BG31RV	91.2 (3.4)
LUNG21RV	87.1 (6.6)
Overall Average	89.3 (3.7)

4.2.5 Alternative 5

The fifth, and last alternative, like alternative three and four, also divides the data set into M equal parts where M is the number of ensemble algorithms. Just like alternative fourth applies the WEKA's propositional algorithms to each divided subset, then uses the WEKA's ensembles algorithms for the ensemble of the T parts. In the end uses the wrapper-ensemble to make an ensemble of an ensemble and then applies the voting scheme algorithm. Table 4.17 presents the results obtained and from which we can conclude that the accuracy of this alternative is around 90%.

Table 4.17: Alternative 5 Results.

Data Set	Voting
S12RV	91.2 (5.4)
H11RV	92.0 (3.5)
ERYT21RV	94.0 (4.0)
HYP11RV	84.6 (4.3)
HYP21RV	94.1 (3.2)
BG11RV	91.0 (1.3)
BG21RV	88.2 (3.2)
BG31RV	93.5 (3.2)
LUNG21RV	87.3 (6.4)
Overall Average	90.7 (3.8)

4.3 Comparing the five alternatives

The proposed alternatives include the combination of different partitions of the data together with different types of classifiers.

Table 4.18 shows the accuracies of the best classifier of each of the five alternatives.

Table 4.18: Best accuracies achieved in each alternative.

Data Set	Alt1	Alt2	Alt3	Alt4	Alt5
S12RV	97.5 (7.9)	99.2 (1.7)	89.1 (10.9)	87.8 (11.3)	91.2 (5.4)
H11RV	99.2 (1.8)	99.2 (1.8)	91.8 (5.4)	94.6 (1.2)	92.0 (3.5)
ERYT21RV	95.4 (4.2)	95.8 (3.9)	89.7 (3.4)	93.1 (3.1)	94.0 (4.0)
HYP11RV	91.2 (4.1)	91.2 (4.1)	78.6 (4.8)	80.5 (4.8)	84.6 (4.3)
HYP21RV	93.8 (4.4)	95.1 (1.9)	92.0 (1.4)	94.5 (0.8)	94.1 (3.2)
BG11RV	93.3 (3.5)	94.3 (4.6)	87.4 (4.3)	89.9 (0.2)	91.0 (1.3)
BG21RV	95.7 (3.6)	95.7 (3.6)	84.4 (2.5)	85.1 (2.1)	88.2 (3.2)
BG31RV	91.6 (4.0)	93.9 (3.2)	86.3 (1.1)	91.2 (3.4)	93.5 (3.2)
LUNG21RV	91.3 (5.0)	93.8 (4.1)	84.4 (6.2)	87.1 (6.6)	87.3 (6.4)
Overall Average	94.3 (2.7)	95.4 (2.4)	87.1 (4.4)	89.3 (3.7)	90.7 (3.8)

From the results presented in the accuracy Table 4.18, we can say that ensemble learners performed better, as expected, than base learners. Although alternative 2 has the highest average over the set of data sets, alternative 2 is not statistically significantly different from alternative 1 according to t-student test ($\alpha = 0.05$).

Table 4.19 shows for each data set, the ranking position of each alternative from 1 to 5. In the end we have made the average to conclude about the best ranking position, using the t-student test, and concluded that alternative 2 got the best ranking position, because there is statistically significant difference between this alternative and the second ranked.

Table 4.19: Comparing the five alternatives ranking position

Data Set	Alt1	Alt2	Alt3	Alt4	Alt5
S12RV	2	1	4	5	3
H11RV	1	1	5	3	4
ERYT21RV	2	1	5	4	3
HYP11RV	1	1	5	4	3
HYP21RV	4	1	5	2	3
BG11RV	2	1	5	4	3
BG21RV	2	1	5	4	3
BG31RV	3	1	5	4	2
LUNG21RV	2	1	5	4	3
Overall Average	2.0 (0.9)	1.0 (0.0)	4.9 (0.3)	3.8 (0.8)	3.0 (0.5)
Overall Position	2	1	5	4	3

Figure 4.11 is a graphical representation that compares the five alternatives in terms of accuracy results. Figure 4.12 is a graphical representation that compares the five alternatives in terms of the ranking position. These two graphical representations are a graphical summary of the results presented in Tables 4.18 and 4.19. Both tables and graphics are consistent since the highest accuracy value corresponds to the lowest ranking (1st place).

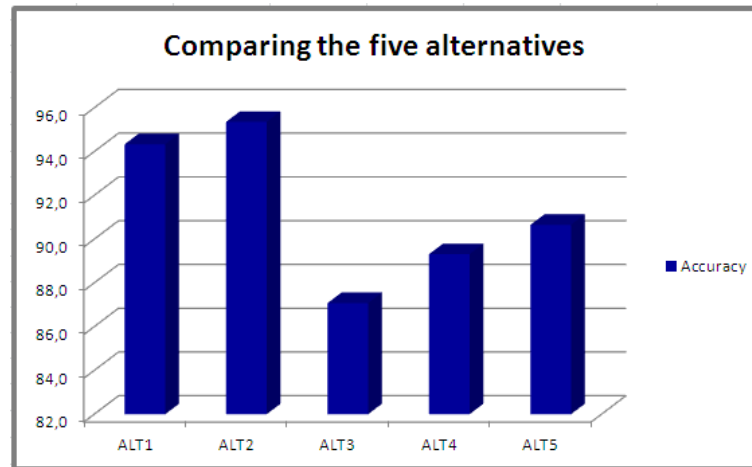


Figure 4.11: Comparing the five alternatives accuracies average.

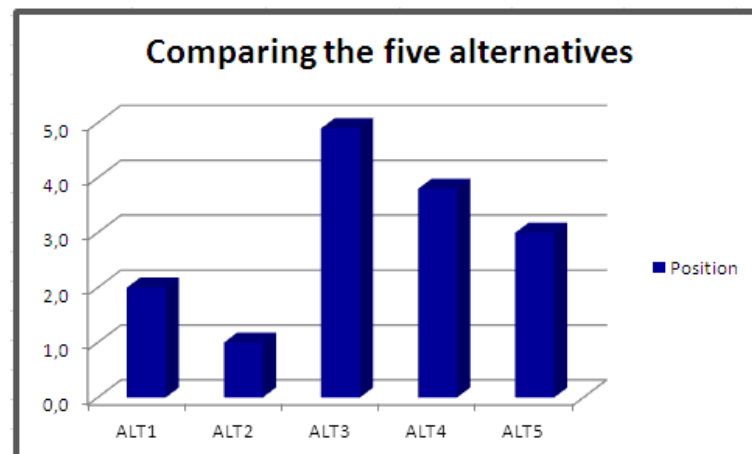


Figure 4.12: Comparing the five alternatives position average.

4.4 Summary

This chapter focused on the construction of the classifier, the main part of step 4 of BioTextRetriever's architecture. The construction of a data set for the classifier was also the object of a deep study considering three possible solutions to extend the set of papers for the negative examples. We could conclude from this study that the best way to generate the negative examples is to use the random values method, since in the preliminary study presented this alternative achieved the highest accuracy for the several data sets presented. The most important part of the study presented in this chapter was the exhaustive study of the five possibilities to address the construction of the classifier figuring out the best alternative of the experimented alternatives. The first alternative shows that the best stand alone classifiers were *bayesnet*, *dtnb*, *j48* and

rf. We can also conclude that the application of ILP performed worst than most of the others. Regarding the best alternative we concluded that using ensemble learners (alternative 2) achieves the best results.

Chapter 5

Ranking MEDLINE

The final step in the BioTextRetriever procedure is to filter the potentially relevant papers, using the classifier constructed in the previous step. The filtered papers are then sorted by relevance before being presented to the user. In the last chapter we have shown how the classifier was constructed. However the set of papers returned by BioTextRetriever is quite large and it is impractical to show all of the papers to the user as they are collected. There is a need to order them by relevance and to present only the most relevant ones. In this chapter we describe and present a ranking procedure that has been developed to order by relevance the final list of papers to be presented.

5.1 The global procedure for the Ranking Function

In order to understand the procedure involved in the last step of the tool we will use Figure 5.1 to provide a context.

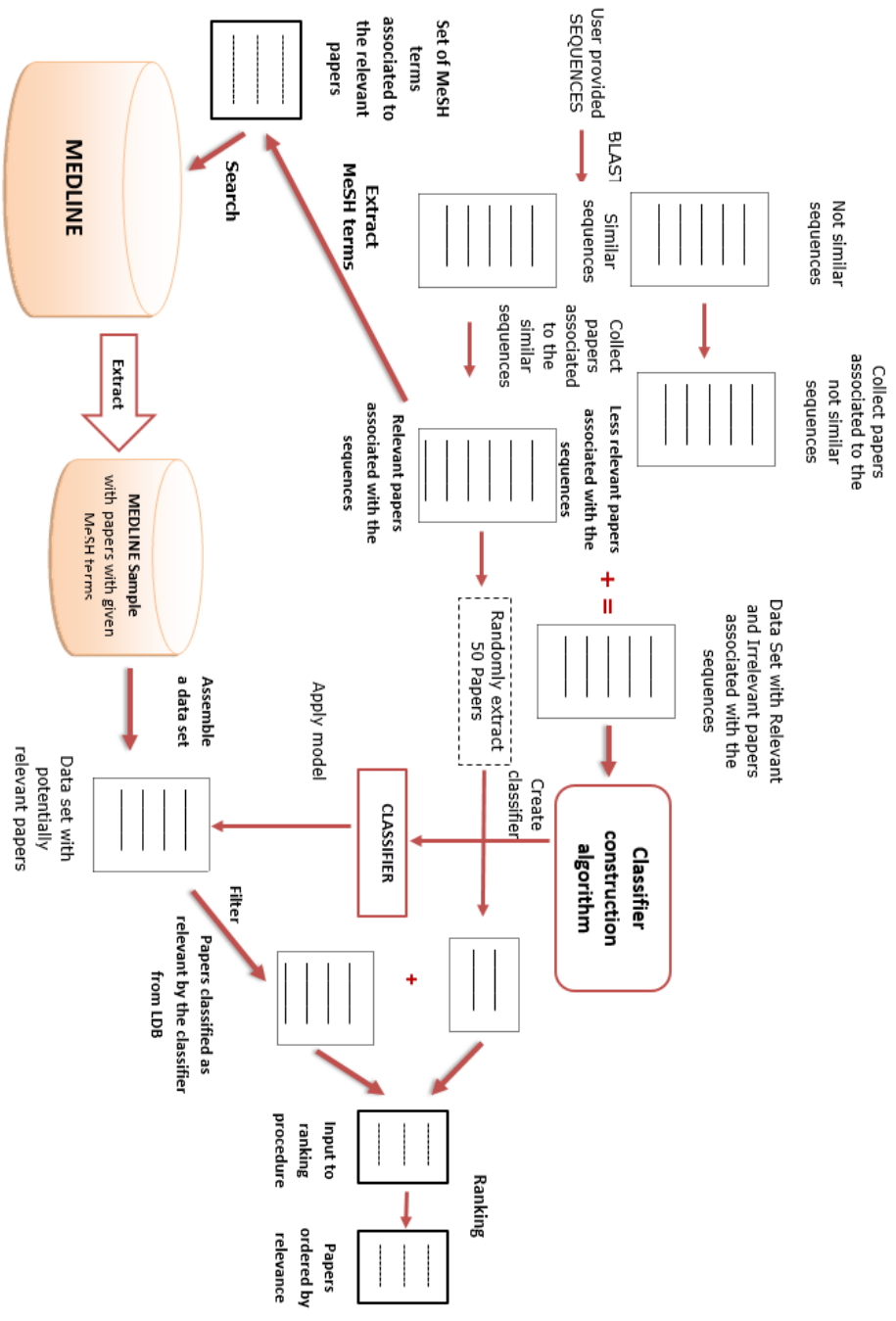


Figure 5.1: Experimental setting for tuning the ranking function

We will first present a summary of the tool's steps up to this point. The user provides a set of sequences, to which the NCBI BLAST tool associates a set of similar sequences. Each of these similar sequences has in turn a set of papers associated. We collect all the papers associated to these similar sequences to build the "relevant papers" part of a data set. The data set is completed by adding a set of "irrelevant papers", through the Random Value method described in the previous chapter. With this data set as input to a machine learning algorithm a classifier is constructed. This classifier will be used, later in the process, to filter the relevant/irrelevant papers from MEDLINE.

From the set of relevant papers associated to the input sequences we extract a set of MeSH terms appearing in these papers. We then search the MEDLINE 2010 database for the papers that have the MeSH terms in common with the set's. The reason for this procedure was already explained in Chapter 4 and has to do with the fact that MEDLINE is a huge database and that the classifiers cannot classify in an acceptable time 20 million documents. Thus we create a MEDLINE sample with papers for classification, that are, somehow related to the introduced sequences because they possess the highest number of MeSH terms in common with the relevant papers we have so far. With this we construct a data set of papers to be given to the classifier for classification. After classification we get a set of papers classified as relevant. To this data set of relevant papers we add a random set of 50 papers, extracted from the set of papers associated to the similar sequences, associated themselves in the first step with the initial sequences entered by the biologist. This procedure was only performed in the experiments intending to decide on the components of the ranking function. The procedure is not performed in the current use of the tool.

To make the tool efficient we have to address some implementation issues. In the 20 million references to scientific papers of MEDLINE 2010 only 9 million have abstracts. To apply our classifier to 9 million abstracts is unfeasible in an acceptable time. Besides we want references to scientific papers that are related to the input sequences. One way to do this is to make an a priori selection of these papers based on specific criteria. The chosen criteria is to select the papers that have the highest number of MeSH terms in common with those extracted from the relevant examples associated to the input sequence. Figure 5.2 shows this procedure. We collect all the papers returned by NCBI BLAST and we make an a priori selection of MEDLINE papers based on the presented criteria. This way we assure that the set of papers that will be classified by BioTextRetriever are somehow related to the input sequences. The number of papers included in the MEDLINE sample was also subject to experimental testing because it affects the efficiency of the BioTextRetriever. In the first experiments we have made a selection of 20000 papers but the SQL instruction to collect the papers was very time consuming. So we reduced this subset to a 10000 subset and then to a 5000 subset. The 5000 subset is created in an acceptable time and we believe that 5000 papers are more than enough to be classified by our model and to be presented to the user in the end.

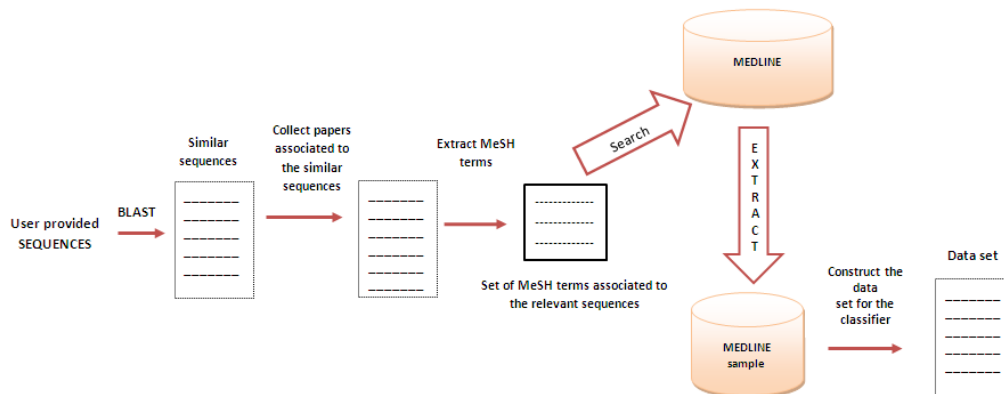


Figure 5.2: Construction of the MEDLINE sample.

5.2 The Ranking Function

Despite the potential relevance of the papers returned by the BioTextRetriever, we need to point out which are the most important papers to present to the user first. A ranking is an ordering of the documents that should reflect their relevance to a user query [ByRN99].

The traditional methods for ranking web pages are not suitable to rank scientific articles, since the traditional ranking algorithms (PageRank, Hits, Salsa and Ranknet to name a few) are based on the number of links to a web page. Besides this reason, the existent algorithms do not take into account the items we believe are the most important to consider in the ranking such as the MeSH terms, the number of publications of the authors, the number of citations of a paper, the h-index of the author of a paper, the journal impact factor and the Journal Similarity Factor. None of the mentioned algorithms involves a function that contemplates the mentioned items. Thus, we have proposed a function that reflects the specific criteria we believe to be the best to use in this case. We propose an integrated ranking of MeSH terms, Pubmed number of citations, author Pubmed h-index, journals impact factor, authors' number of Pubmed publications and the journal similarity factor where the relevant papers were published. The combined use of several indicators that give information on different aspects of scientific output is generally recommended [VVM⁺03].

As explained in the previous section we have used a sample of MEDLINE with papers that have the highest number of common MeSH terms, that we believe to be the best one to use with the relevant papers associated with the introduced sequence(s).

After classification the resulting¹ paper references are ordered by the following ranking function:

¹at most 5000

$$C1 * MeSH + C2 * Citations + C3 * hindex + C4 * IFactor + C5 * Pub + C6 * JSFactor$$

where:

- *MeSH* is a weighted sum of MeSH terms in common with the papers associated with the introduced sequence(s);
- *Citations* is the paper's number of citations in MEDLINE;
- *hindex* is the highest h-index among the authors' h-index;
- *IFactor* is the Journal Impact Factor;
- *Pub* is the number of publications of the author with the highest number among them all;
- *JSFactor* is a weighted sum of the number of papers published in a journal that has papers associated to the introduced sequence(s).

Coefficients C1, C2, C3, C4, C5 and C6 may vary between 0 and 100 and their sum must be 100. The set of experiments to determine the values of these coefficients is described in detail in Section 5.3.

Besides the information contained in MEDLINE, we have added some extra information to the LDB. Besides the Journal Impact Factor all the terms in the ranking function (number of MeSH terms, number of citations, author h-index, author number of publications and the Journal Similarity Factor) are computed using LDB. The Journal Impact Factor was obtained from the ISI Web of Knowledge website powered by Thomson Reuters. We have normalized the coefficient factors between 0 and 100 to obtain a coherent formula.

The paper references are ordered by the ranking function and presented in decreasing order of relevance to the user. Although it is impossible for a human to make an exhaustive reading of all the presented papers, the user can access and see all the papers returned in decreasing order of relevance. The tool presents 30 results per page.

We will now detail each of the terms that integrate the ranking function.

5.2.1 Number of MeSH terms

BioTextRetriever collects all papers related to the relevant sequences. The MeSH terms of those papers are extracted as explained in Figure 5.3. After constructing a classification model (Step 4 Figure 3.1 of Chapter 3), based on this set of papers, the classifier applies the model to a MEDLINE sample. This sample (see Figure 5.2 of Section 5.1) is composed by the papers that have the most common MeSH terms with the MeSH terms of the relevant papers.

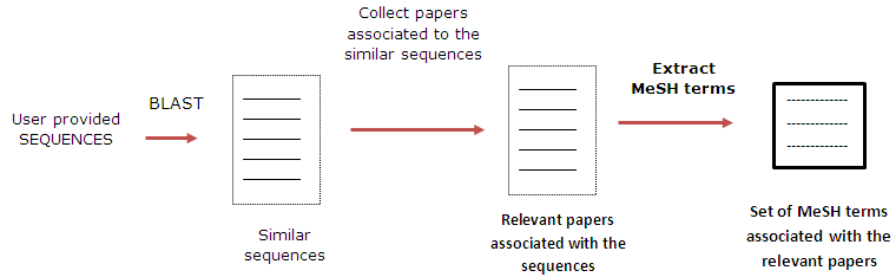


Figure 5.3: Extracts the MeSH terms associated with the relevant papers associated with the sequences.

With this procedure we guarantee that the MEDLINE sample has papers that have a higher number of common MeSH terms with the sequence related papers, and is taken as the first step to a collection of relevant papers.

In order to better highlight even more the number of MeSH terms in common with the ones associated with the sequences, we have introduced a ponderation for the papers that have more of these MeSH terms in common. Table 5.1 shows an example of four papers associated with the input sequences. Suppose that we have a paper in MEDLINE that has MeSH2, MeSH3 and MeSH4. The ponderation factor for this paper should be equal to $3 + 4 + 3$, which equals 10. If we now have a paper in MEDLINE with MeSH1, MeSH7 and MeSH8 its weight would be 2 reflecting the potential weak "conection" with the relevant papers.

This way the papers that have more common MeSH terms associated to the sequences are valued.

Table 5.1: The table shows an example of four papers associated with the input sequences.

Papers associated with the sequences	MeSH1	MeSH2	MeSH3	MeSH4
Paper1	0	1	1	1
Paper2	1	1	1	1
Paper3	0	1	1	0
Paper4	1	0	1	1
Total	2	3	4	3

5.2.2 Author’s Number of Publications

The number of publications of an author has also been considered in the formula. However it may happen that an author may have a large number of publications but with few citations, and these citations are in journals with a small impact factor,

whilst another author may have a smaller number of publications with a high number of citations published in journals with high impact factors, which should be more relevant in the ranking formula. For each author we count the number of publications available. The authors with more than fifty publications get a number of publications of fifty. We believe that fifty is a good number of publications for a good author. However the most common case is paper's with more than one author. In this case, we consider the highest number of publications. As we do not disambiguate the author's name, there is a slight chance that authors with the same name induce an error in the effective number of publications of a particular author.

5.2.3 Number of citations

A citation is a unique reference to an article, a book, a WebPage, a technical report, a thesis or other published item. The number of citations of a paper has become a major indicator to evaluate scientific work. Although it has some drawbacks and it is not unanimously accepted by the scientific community, nevertheless we consider that it is one of the most important measures to estimate the impact of scientific published work in the scientific community. Citation indexes provide a means with which to measure the relative impact of articles in a collection of scientific literature [Bra03]. The concept of citation indexing and searching was invented by Eugene Garfield [Gar64] and anticipated the Science Citation Index. There are several citation indexing systems such as Google Scholar, CiteSeer and Scopus. These systems allow to search for a researcher's number of citations however, we could not use them to obtain the number of citations of around 20 million MEDLINE publications due to PubMed service restriction policies.

We have computed, using LDB, the number of citations of MEDLINE scientific papers. Each sequence has a set of paper references associated, and each of these references has the bibliography associated. Most of the referenced papers are available in MEDLINE so we can obtain the number of citations of a paper cited by other MEDLINE papers inside MEDLINE 2010.

The number of citations for a particular paper is shown to be more relevant and important in comparison with the number of publications. This is because an author may have a higher number of publications but that are not cited, whilst another author may have a smaller number of publications but highly cited. The impact of a piece of research is the degree to which it has been useful to other researchers [SBCH06]. However, the number of citations does not take into account the distribution of the several publications, e.g., a high number of citations but with very few highly cited scientific papers. However, we could not use them to obtain around 20 million citations for the MEDLINE publications due to PubMed service restriction policies.

Papers that have a high number of citations but that are not recent, e.g., should be devalued when compared with recent papers with a high number of citations. In fact,

recent papers have naturally less citations than older papers. We have implemented the following formula for the number of citations.

$$\text{Number of citations} = \frac{\text{Effective Number of citations}}{\alpha * \text{Number of years}}$$

Where α represents the devaluation coefficient used, specified in Table 5.2.

Table 5.2: The α value represents the devaluation coefficient in decreasing order of the paper’s scientific age.

Age of Papers	α
≤ 5 years	1.0
≥ 5 years and ≤ 20 years	0.8
≥ 20 years	0.6

For example, the paper ”The sequence of the human genome”, a well known paper in the Biological and Medical communities, has a score of 100 for the item ”number of citations” in MEDLINE 2010, and in Google it has 10240 citations (searched in 18.December.2012). As we are only considering counting in LDB, on one hand we have found less papers inside MEDLINE 2010 that cite the scientific paper ”The sequence of the human genome” than in Google, besides BioTextRetriever devalues the number of citations by the paper scientific age, thus the number of citations in MEDLINE 2010 is much lower than the one found by Google.

5.2.4 h-index

Hirsch [Hir10] proposed the *h-index*: ”A scientist has an index h if h of his or her N papers have at least h citations each, and the other $(N - h)$ papers have less than or equal to h citations each.” In other words, the h -index bases itself on publications ranked in descending order according to their number of citations. *h-index* is approximately proportional to the square root of the total citation counts [FM10].

The *h-index* is an index that attempts to measure both the productivity and impact of the published work of a researcher. It is based on the set of the scientist’s most cited papers and the number of citations that they have received in other publications. It combines both the number of papers and their quality (impact, or citations to these papers) [Glä06]. The *h-index* is recognized by the ISI Web of Science by Thomson Reuters or Scopus by Elsevier, as an important indicator for assessing research impact [CM06, ADD10, Hir10].

Like the other bibliometric measures, *h-index* has advantages and limitations. Mathematically it is very simple to compute and it is easy to understand [Hir10, Glä06].

Hirsh claims in [Hir10] that the *h-index* performs better than other single-number criteria commonly used to evaluate the scientific output of a researcher (impact factor, total number of documents, total number of citations, citations per paper rate and number of highly cited papers).

For young researchers the *h-index* is not a very promising measure since they have few publications highly cited and thus will probably have a low *h-index*. One might say that the *h-index* favours the researchers that have many cited publications. A scientist with very few highly cited papers or a scientist with many lowly cited papers will have a weak *h-index* [CM06], [Egg06]. To address this issue, Hirsh presented the “*m* parameter” in [Hir10] that divides *h* by the scientific age of a scientist (number of years since the author’s first publication) to attenuate this problem.

Besides this, the *h-index* also depends on the database in use, reason which, alongside problems with common names and different spellings, makes its flaws very visible [HSO12]. Hirsh [Hir10] also refers this technical problem in obtaining the complete list of publications of scientists with very common names. To overcome this problem, the authors in [BD07] recommend that the *h-index* should be calculated with a list of publications authorized by the scientist and found in the Web of Science using a combination of the scientist’s name and address or affiliation.

The *h-index* should not be used to compare scientists from different disciplines [Hir10]. The *h-index* does not take care of self-citations which can increase a scientist’s *h-index* [Van05]. The *h-index* can also be used to measure the scientific output of institutions and research groups [ER08].

As was already mentioned in Section 5.2.3, we have obtained using LDB, the number of citations of each paper’s reference inside MEDLINE 2010. We collect and store for each paper author the number of publications. For each publication we count the MEDLINE internal number of references to that particular publication to obtain the number of citations.

Table 5.3 shows an example of how to calculate the *h-index* of an author inside MEDLINE 2010.

Table 5.3: Example of *h-index* computation for $h=4$ in this case.

Rank of publications	1	2	3	4	5
Number of citations	1988	8	7	6	4

The first line indicates the order of each publication in ascendant order. The second line presents the number of citations in descending order. The author’s first publication has 1988 citations, the second publication has 8 citations, and so on. We know from the literature that the *h-index* of an author is *h* when the number of citations is equal or greater than the number of publications. A researcher has *h-index* *h* if, in the list of articles arranged in decreasing order of the number of citations of these articles, $r=h$ is the highest rank such that the papers on rank 1, 2,..., *h* each have at least *h*

citations [Egg12]. Thus in the presented example the h-index is 4, because the author has four papers with more than four citations each.

As a paper may have more than one author (which is the most common case) we calculate the h-index for all the authors of a paper and select the highest h-index. If an author has a high h-index and the other authors have a smaller h-index, it means that at least one author is recognized by the scientific community as having prestige, and a prestigious author has valuable publications.

5.2.5 Journal Impact Factor

For the ranking we have considered only the journal Impact Factor because MEDLINE only references papers that are published in Journals. The Journal Impact Factor (JIF) [Gar99] is a measure of the frequency with which the average article in a journal has been cited in a particular year or period, thus JIF may change overtime. JIF is based on information obtained from citation indexes. The most widely accepted and used JIF is from the Journal Citation Report (JCR), a product of Thomson Reuters ISI (Institute for Scientific Information) (only considers ISI journals). The Journal Citation Report has been published annually since 1975.

Garfield developed the journal's impact factor metric that is defined by the following formula: $JIF = \frac{C_2}{P_2}$, where C_2 is the number of citations in the current year of any of the items published in a journal in the previous 2 years and P_2 is the number of papers published in the previous 2 years.

For example, the 2012 JIF is calculated by the formula: $\frac{C_2}{P_2}$, where: C_2 is the number of times papers or other items published during 2010-2011 were cited in indexed journals during 2012, and P_2 is the number of items published in 2010 plus 2011.

Thomson Reuters released in 2009, the new 5-year journal Impact Factor in addition to the standard 2-year journal Impact Factor. The 5-year journal Impact Factor is the average number of times articles from a journal published in the past five years have been cited in Journal Citation Report year. And it is calculated by dividing the number of citations in the Journal Citation Report year by the total number of articles published in the five previous years.

The Journal Impact Factor is used to compare different journals only within the same field. The ISI Web of Knowledge indexes more than 11,000 science and social science journals.

A journal with a high impact factor is usually considered a high quality journal and high quality journals usually have high quality papers.

The Journal Impact Factor has some limitations stated by [Seg97], [Lip09] and [Smi07]:

- Journal Impact Factor does not control self-citations;

- Journal Impact Factor varies significantly from field to field;
- The Journal Impact factor depends on the dynamics of the research field;
- Journals databases are not always accessible, i.e., neither all papers are available for free in the Web;
- High citation rates do not always reflect the high quality of a journal/paper;
- The Journal Impact Factor is calculated over a short period of time (the last two or five years);
- A citation in a "low impact" journal is counted equally to a citation in a "high impact" journal, however they should be distinguished, since the second one it is more valuable than the first one;
- A journal score is highly influenced by its total number of citable papers;
- Journal Impact Factor does not assess the quality of individual papers (only a small percentage of Journal papers are highly cited but they have a huge impact in the total number of citations of a Journal).

[Lip09] and [Smi07] also enumerate the determining factors associated with journals with high impact factors:

- Indexing in most known databases: PubMed/MEDLINE, Scopus and Google Scholar;
- Papers written in the English language;
- Availability of the full-text paper, preferently for free;
- Availability of the paper abstract;
- Submissions from authors with an higher reputation;
- Publications of an higher number of review papers, because review papers are often more cited;
- To cite papers previously published in the same journal;
- Focus on dynamic "excellence" research fields that generate more citations.

Although the Journal Impact Factor has the above mentioned limitations we included it in the ranking formula.

We have obtained the 2-year Journal Impact Factor for the papers that have been published in the Web of Knowledge website. We have downloaded the complete list

of Journals Impact Factors available in October 2010 ². For each paper references BioTextRetriever retrieves, we gather the Journal Impact Factor of the publication (Thomson Reuters).

5.2.6 Journal Similarity Factor

The Journal Similarity Factor, highlights the journals with more papers published that are associated to the sequences introduced by the user. A paper can be published in one and only one Journal. The key idea is that the papers that are associated to the sequences introduced by the user should have a higher impact in the formula. Figure 5.4 illustrates this procedure.

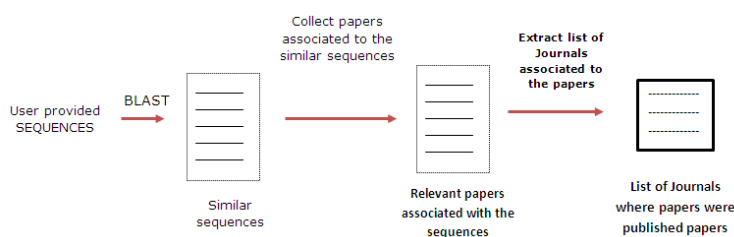


Figure 5.4: Extracts the journals associated to the relevant papers associated with the sequences.

Table 5.4 shows an example of four papers from the set of papers associated with the input sequences. Suppose that a paper we collect from MEDLINE, was published in Journal1. As this particular journal has published three papers associated with the sequences, the formula should emphasize this fact by assigning the weight 2 to this factor in detriment of a journal that has, for example zero papers published.

Table 5.4: Example of four publication journals of four papers associated to the input sequences.

Papers associated to the sequences	Journal1	Journal2	Journal3	Journal4
Paper1	1	0	0	0
Paper2	0	1	0	0
Paper3	0	0	1	0
Paper4	1	0	0	0
Total	2	1	1	0

Some of the items in the ranking formula, namely the number of MeSH terms in common with the papers associated with the sequences, the author's h-index, the number of publications, the number of citations and the Journal Similarity Factor are

²At this date there were 7347 journal classifications available

calculated and stored jointly to the authors. The number of citations and the number of publications are independent of the other items. However the h-index relates the number of publications and the number of citations.

Figure 5.5 summarizes the six issues mentioned that are part of the ranking function developed. The Journal Impact Factor is an item that is not calculated through the LDB but is obtained through an external source (Thomson Reuters).

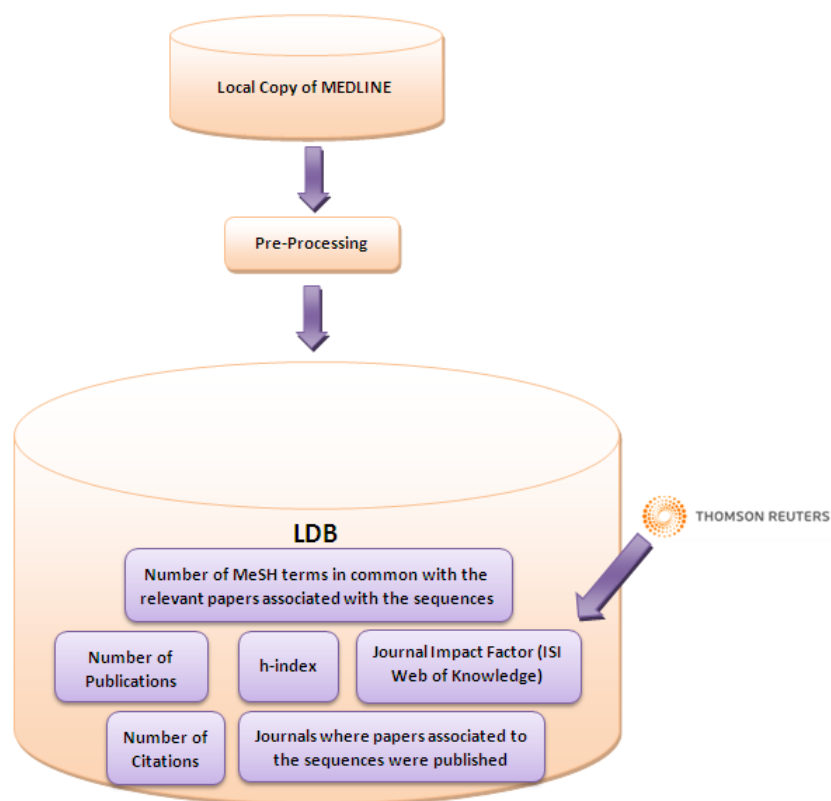


Figure 5.5: Information items stored in the LDB to be used in the ranking function. The Journal Impact Factor is the only one not calculated through the local copy of MEDLINE's available information, but is downloaded from the Web of Knowledge website (Thomson Reuters) and is saved in LDB.

5.3 Choosing the Ranking Function Coefficients

As described in Section 5.2, the ranking function combines the following six components:

1. The number of MeSH terms associated with the papers connected to the sequences introduced by the user;

2. Number of PubMed publications;
3. Number of citations;
4. Author h-index;
5. Journal Impact Factor;
6. Journal Similarity Factor.

In order to assure the usefulness of these coefficients to the relevance of the retrieved papers and also to propose default values for the formula coefficients, we undertook a set of experiments that are next described.

5.3.1 Experimental Settings

Data Description

We have used 14 data sets, each one composed by more than 90 relevant papers. These data sets resulted from using sequences from 7 different domains with the following distribution:

- Rnases: 1 sequence
- Alzheimer: 1 sequence
- Blood Pressure: 1 sequence
- Erythrocytes: 2 sequences
- Hypertension: 2 sequences
- Blood Glucose: 4 sequences
- Lung Disease: 3 sequences

The data sets used are characterized in Table 5.6 concerning the number of attributes, and the number of relevant papers returned and classified by BioTextRetriever as relevant. Some of these data sets were already characterized in Table 4.8 in Chapter 4.

Table 5.5: Characterization of data sets regarding the number of attributes and the number of positive and negative examples.

Data Sets	Number of Attributes	Positive Examples	Negative Examples	Total Examples
S12	1602	128	128	156
BP25	441	63	31	94
ALZ31	1485	114	114	228
ERYT11	1505	99	99	198
ERYT21	1592	118	118	236
HYP11	1706	130	130	260
HYP21	1944	194	194	388
BG11	1546	97	97	194
BG21	1631	115	115	230
BG31	1859	149	149	298
BG41	1812	161	161	322
LUNG11	1553	124	124	248
LUNG21	1535	120	120	240
LUNG31	1054	74	74	148

Table 5.6: Characterization of data sets used to tune the coefficients of the ranking function. The column number six represents the total number of relevant papers classified as relevant by BioTextRetriever. The last column represents the percentage of relevant papers classified as relevant by BioTextRetriever.

Data Sets	Total Relevant Papers classified by BioTextRetriever	Percentage of Relevant Papers classified by BioTextRetriever
S12	3947	78.9%
BP25	2071	41.4%
ALZ31	1751	35.0%
ERYT11	2498	50.0%
ERYT21	4397	87.9%
HYP11	4235	84.7%
HYP21	4638	92.8%
BG11	4423	88.5%
BG21	5	0.1%
BG31	2301	46.0%
BG41	3288	65.8%
LUNG11	4103	82.1%
LUNG21	4182	83.6%
LUNG31	1644	32.9%

The study carried out in the previous chapter showed that the best alternative was to use the Ensemble algorithms (Alternative 2) for the classification problem. Consequently we have used the results provided from this alternative in the experiments.

Experimental Procedure

Since we do not have access to an expert to evaluate the results of the application of the ranking function, and also because the sorted set of paper is still very large we have adopted the following procedure. For each data set we performed the following actions:

1. Run step 1 through step 5 of the tool to get a set of potentially relevant papers;
2. Add to the extracted set of papers classified as relevant in the previous step of the tool, 50 papers extracted randomly from the relevant papers associated to the input sequences. Since these papers are guaranteed to be relevant (by the ownership of the original sequences) we use them to alternate the fact of not having access to an expert.
3. Count how many of the guaranteed relevant papers (obtained in 2.) will appear in high positions of the ranked set.
4. For each data set of the Table 5.6:
 - (a) Create 10 new sub data sets, each of them with 50 randomly examples added from the relevant ones.
5. The average of the combinations for each of the 10 sub data sets is obtained and represents the value achieved for each data set.

In these experiments we have tested the six coefficients with values from the set $\{ 0, 25, 50, 75, 100 \}$ with the restriction that the sum of all coefficients must be 100%. The combination of all these values for the five coefficients gives a total of one hundred and twenty six possible combinations.

The ranking function is evaluated by analysing the first 20 papers that are presented to the user in descendent order of relevance and counting the number of papers from the 50 relevant ones inserted in the data set that appear in this 20 first.

The combination that returns the higher number of relevant papers associated with the references constitute the best coefficient combinations for the proposed ranking function. Figure 5.6 summarizes the procedure.

The combination that has more hits in average for all the data sets is considered the best combination for the default ranking formula.

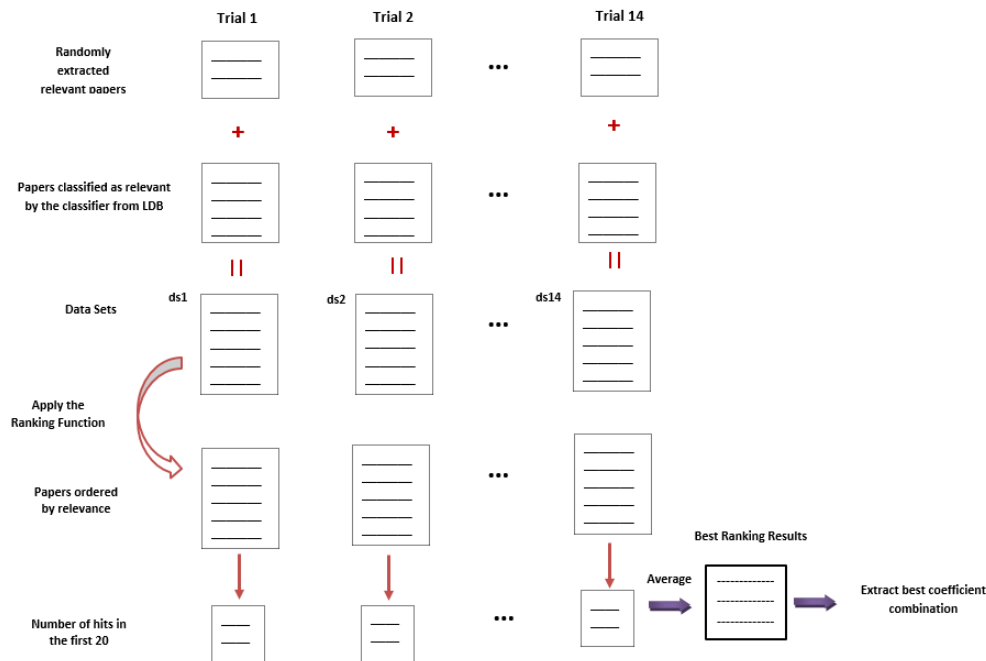


Figure 5.6: Procedure to evaluate the ranking function

Evaluation and Discussion

The columns C1 to C6 of Table 5.7 represent the best ranking coefficient combinations for the presented methodology.

The columns C1 to C6 of Table 5.7 represent the values of the six items coefficients: where $C1$ is the coefficient weight for the number of MeSH terms; $C2$ is the coefficient weight for the number of citations; $C3$ is the coefficient weight for the author h-index, $C4$ is the coefficient weight for the impact factor, $C5$ is the coefficient weight for the number of publications and $C6$ is the coefficient weight for the Journal Similarity Factor. The last column represents the hits average for each combination for the fourteen data sets used. Table 5.8 represents the individual combination results for the fourteen data sets described in Table 5.7 for each line of this Table.

The best results highlight the number of citations and the h-index factors. We have applied the t-test to analyze these three best results. The t-test ($\alpha = 0.05$) gave no statistical significance between the three best results presented.

Table 5.7: Best combinations results for the fourteen data sets described in Table 5.6. $C1$ represents the coefficient weight for the number of MeSH terms; $C2$ represents the coefficient weight for the number of citations; $C3$ represents the coefficient weight for the author h-index; $C4$ represents the coefficient weight for the impact factor; $C5$ represents the coefficient weight for the number of publications and $C6$ represents the coefficient weight for the Journal Similarity Factor.

C1	C2	C3	C4	C5	C6	Average
0	75	25	0	0	0	3.8 (3.8)
0	100	0	0	0	0	3.7 (4.1)
0	50	50	0	0	0	3.6 (4.1)

Table 5.8: Individual combination results for the fourteen data sets described in Table 5.6 for the three combinations presented in Table 5.7.

S12	BP25	ALZ31	ERYT11	ERYT21	HYP11	HYP21	BG11	BG21	BG31	BG41	LUNG11	LUNG21	LUNG31
1.9 (1.2)	3.4 (1.2)	3.1 (1.8)	1.0 (1.2)	4.1 (1.8)	0.9 (0.9)	1.9 (1.6)	3.1 (1.3)	17.6 (0.7)	2.8 (1.7)	2.0 (1.3)	3.8 (2.2)	2.7 (1.6)	4.4 (1.2)
1.9 (1.2)	3.4 (1.2)	3.3 (1.7)	1.3 (1.2)	3.8 (1.8)	0.9 (0.9)	1.7 (1.2)	3.2 (1.7)	16.4 (0.7)	2.8 (1.7)	2.1 (1.5)	3.6 (1.9)	2.8 (1.7)	4.1 (0.9)
1.6 (1.3)	3.4 (1.2)	2.8 (1.7)	0.9 (0.9)	3.7 (1.9)	0.7 (0.8)	1.8 (1.5)	3.6 (1.4)	17.5 (0.9)	2.6 (1.6)	2.0 (1.3)	3.7 (2.2)	2.1 (1.1)	3.7 (0.9)

From the presented best combinations, BioTextRetriever was configured with the combination presented in the first line of Table 5.7. Although BioTextRetriever was configured with the aforementioned weights, the user may introduce the weights.

5.4 Summary

This chapter presented our proposal for a ranking function that integrates several issues in one function, namely MeSH terms, number of citations, author h-index, journal impact factor, authors' number of publications and the Journals associated to the sequences. All of these issues were referred and justified in their importance in the ranking of MEDLINE papers. The experimental procedure is presented and illustrated through practical examples. This set of experiments are described to achieve a suitable combination of the ponderation items to propose a default formula for the ranking function. This study showed that the three best combinations involve considering a high score for the number of citations and the author h-index, according to the results obtained in our experiments. The number of publications is already used in the calculation of either the h-index as well as the number of citations. The Journal Similar Factor was not contemplated, possibly because if the papers associated to the original sequences are all published in different journals then our formula does not

take into account this particular fact. The MEDLINE sample was generated with the papers having the highest number of MeSH terms in common with the original sequences, thus highlighting this issue again in the ranking function possible may not have the desired impact since these papers already have the highest number of MeSH terms. We have configured BioTextRetriever with the best combination (by default) achieved that only considers the number of citations and the author h-index.

Chapter 6

Conclusions and Further Research

This chapter summarizes the main conclusions of the research work that has been done. The main contributions of this thesis are highlighted, as well as its limitations, and future research directions are proposed.

6.1 Thesis Overview

Undoubtedly researchers need to be aware of all of the relevant scientific research in their area of knowledge. However the volume of scientific and technical publications in almost all areas of knowledge is growing at a phenomenal rate. Most of these publications are available on the Internet. Thus, accessing the right and relevant information amidst this overwhelming amount of information available in the Web is indeed of great importance, albeit difficult in most cases [CH00].

When trying to find relevant publications, researchers turn to the well known traditional keyword-based search engines, which return as a result a huge list of publications, that usually includes a large number of irrelevant ones [APTK10].

To tackle this problem, research in Text Mining and Information Retrieval has been applied to literature mining in order to help researchers to identify the most relevant publications [CH00], [LGG01].

We have developed an approach that automates some routines identified in the biologist's frequent tasks. One such task is the search for those papers that make explicit references to a particular genomic or proteomic set of sequences.

We have thus elected the search for relevant scientific papers that reference a set of genomic or proteomic sequences as the main problem to be addressed in this thesis. The application of Machine Learning techniques, Information Retrieval and Text Mining was proposed and implemented as a solution to this problem.

The main research question that guided this thesis was "Is it possible to construct an automatic web-based tool that, given a set of genomic or proteomic sequences, provided by the user, returns an relevant and ordered set of papers?" "

To answer this main research question we had to solve two main problems. The first is how to collect a set of relevant papers and the second is how to sort by relevance the papers resulting from the solution of the first problem. Each of these two problems raised new research questions, that we believe were answered throughout this thesis.

For the first problem: to retrieve the relevant papers of MEDLINE, we have proposed a novel method of Information Retrieval, based on Machine Learning techniques. This method involved in the first stage the study and evaluation of the most adequate combination of the text pre-processing techniques. We have made a set of experiments (described in Chapter 3), involving 32 data sets, out of four different domains. The experiments showed that the most adequate techniques were the following and in the presented order: 1. Handle Synonyms; 2. Stop-words removal; 3. Word validation using a dictionary; 4. Stemming and 5. Pruning. The application of these techniques, reduces significantly the original number of attributes without decreasing significantly the accuracy. This enables a considerable speed-up to the classifier construction process, which is important for a Web-based service like BioTextRetriever. This evaluation of the combination of several pre-processing techniques for the MEDLINE set of papers, constitutes the first contribution of this thesis.

To solve the first problem, a second and crucial phase was carried out, i.e., to develop an Information Retrieval methodology that involves the dynamic construction of a classifier in real time to classify MEDLINE papers. This second phase (detailed in Chapter 4) required first the construction of a data set. The data set is required to dynamically construct a classifier that will act as a filter for the main source of papers (MEDLINE). We have empirically proposed and evaluated three different ways of producing a data set starting with the papers associated to the sequences, that we have called Near-Miss Values (NMV), MeSH Random Values (MRV) and Random Values (RV). The experimental results showed that the average accuracy of the Machine Learning algorithms on the several data sets used were higher for the RV method.

To construct a classifier in real time for classifying MEDLINE papers we have developed a new methodology based on Machine Learning techniques. We have devised and assessed several ways of partitioning the data and combining the Machine Learning algorithms in order to achieve a good performance in the classification process. From this study we were able to conclude that the best Machine Learning algorithms to achieve a good performance are the Ensemble of Classifiers (a method that combines the individual decisions of a set of classifiers through majority or voting). We were also able to conclude that the best stand alone classifiers were "*decision table*", *j48* (an implementaion of the C4.5 algorithm) and the "*random forest*". In terms of the accuracies of the results, the Ensemble of algorithms achieved an accuracy of 95.3% and the stand alone classifiers achieved an accuracy of 92.7%. The results show that the use of Machine Learning is extremely valuable to automate the Information Retrieval

process with good performance results.

For the second problem we have proposed a new methodology that enables the automation of the assessment process of a multi-criteria ranking function (detailed in Chapter 5).

BioTextRetriever's last procedure is to organize the papers selected as relevant by the classifier. In fact, this set of papers classified as relevant is quite large and it is not advisable to present such a huge number of papers to the user. We proposed an integrated ranking function that combines MeSH terms, Pubmed number of citations, author Pubmed h-index, journals impact factor, authors number of Pubmed publications and journal similarity factor¹.

Since we do not have access to an expert to evaluate the results of the ranking function, we have adopted a procedure where the relevant papers associated to the original sequences are the ones that maximize the presented ranking function if they appear in the first 20 results. Since these papers are guaranteed to be relevant (because they are associated to the original sequences) we use them as an alternative to the fact that we do not have access to an expert. The ranking function is evaluated by analysing the first 20 papers that are presented to the user in descendent order of relevance by the ranking function, and counting the number of papers from the relevant papers associated with the introduced sequences that maximize the ranking function. The best combinations maximize the number of citations and the h-index. BioTextRetriever was configured, by default, with this coefficients combination, however the user can introduce other weights for each factor.

As a final contribution, we have implemented a web-based software tool for retrieving relevant literature in Molecular Biology and related domains given a set of genomic or proteomic sequences. This tool integrates all the technical scientific contributions mentioned and is itself a proof-of-concept of this thesis' developed work. The tool is available only for registered users at <http://nilson.fe.up.pt/>.

6.2 Further Research

For the thesis experiments we have only used the MEDLINE 2010 database but we could also have used the Ensembl database to collect other related papers associated to the original sequences that are not on MEDLINE and increase this way the relevant papers associated to those original sequences.

We have based our search on MeSH terms, but we could also allow the user to introduce keywords and gather only the papers including those referred keywords. This procedure would probably reduce, even more, the number of returned documents. We

¹The journal similarity factor highlights the journals with more papers published that are associated to the original sequences.

could gather the keywords associated to the positive documents (returned by BLAST with e-value less than the one specified by the user) and create the MEDLINE sample alongside the MESH terms and also these keywords, and compare with the results obtained.

Another open issue, and a complex one, is the desambiguation of the authors' names. When we calculate the number of publications of an author we do not desambiguate the cases where there are more than one author with the same name, which indeed can exist.

For the aim of this thesis we have only used the papers' abstracts, however in a future work we will extend the abstracts to full texts and then compare the results achieved.

Another aspect concerning the results obtained by using our proposed tool is that after making the requests the user must re-visit the page to see the state of the request until it reaches the finished state. One improvement could be to send a notification, through email, to the users when the process reaches the "finished" state. This will save time to the user if the requests are still being processed.

References

- [ABB⁺00] M. Ashburner, C. A. Ball, J. A. Blake, D. Botstein, H. Butler, J. M. Cherry, A. P. Davis, K. Dolinski, S. S. Dwight, J. T. Eppig, M. A. Harris, D. P. Hill, L. Issel-Tarver, A. Kasarskis, S. Lewis, J. C. Matese, J. E. Richardson, M. Ringwald, G. M. Rubin, and G. Sherlock. Gene ontology: tool for the unification of biology. The Gene Ontology Consortium. *Nature genetics*, 25(1):25–29, May 2000.
- [ABD06] Eugene Agichtein, Eric Brill, and Susan T. Dumais. Improving web search ranking by incorporating user behavior information. In Efthimis N. Efthimiadis, Susan T. Dumais, David Hawking, and Kalervo Järvelin, editors, *SIGIR*, pages 19–26. ACM, 2006.
- [ADD10] Giovanni Abramo, Ciriaco Andrea DâAngelo, and Flavia Di Costa. Citations versus journal impact factor as proxy of quality: could the latter ever be preferable? *Scientometrics*, 84(3):821–833, 2010.
- [AGM⁺90] S. F. Altschul, W. Gish, W. Miller, E. W. Myers, and D. J. Lipman. Basic local alignment search tool. *Journal of molecular biology*, 215(3):403–410, October 1990.
- [AGOR11] Davide Anguita, Alessandro Ghio, Luca Oneto, and Sandro Ridella. In-sample model selection for support vector machines. In *IJCNN*, pages 1154–1161. IEEE, 2011.
- [AK91] D. Aha and D. Kibler. Instance-based learning algorithms. *Machine Learning*, 6:37–66, 1991.
- [AKT06] Sophia Ananiadou, Douglas B B. Kell, and Jun-Ichi I. Tsujii. Text mining and its potential applications in systems biology. *Trends Biotechnol*, 2006.
- [AMGG07] Aprile A., Castellano M., Mastronardi G., and Tarricone G. A web text mining flexible architecture. *International Journal of Computer Science and Engineering.*, 2007.
- [APTK10] S. Ananiadou, S. Pyysalo, J. Tsujii, and D. B. Kell. Event extraction for systems biology by text mining the literature. *Trends in Biotechnology*, 28(7):381–390, 2010.

- [AZ12] Charu C. Aggarwal and ChengXiang Zhai. A survey of text classification algorithms. In Charu C. Aggarwal and ChengXiang Zhai, editors, *Mining Text Data*, pages 163–222. Springer, 2012.
- [BD07] Lutz Bornmann and Hans-Dieter Daniel. What do we know about the *h* index? *JASIST*, 58(9):1381–1385, 2007.
- [BFH⁺10] Remco R. Bouckaert, Eibe Frank, Mark A. Hall, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H. Witten. Weka - experiences with a java open-source project. *Journal of Machine Learning Research*, 11:2533–2541, 2010.
- [BH09] Iyad Batal and Milos Hauskrecht. Boosting knn text classification accuracy by using supervised term weighting schemes. In *Proceedings of the 18th ACM conference on Information and knowledge management, CIKM '09*, pages 2041–2044, New York, NY, USA, 2009. ACM.
- [BHA⁺06] Elmer V. Bernstam, Jorge R. Herskovic, Yindalon Aphinyanaphongs, Constantin F. Aliferis, Madurai G. Sriram, and William R. Hersh. Research paper: Using citation data to improve retrieval from medline. *JAMIA*, 13(1):96–105, 2006.
- [bio] Biomint <http://biomint.pharmadm.com/>.
- [BJ09] P. Bhargavi and S. Jyothi. Applying naive bayes data mining technique for classification of agricultural land soils. *International Journal of Computer Science and Network Security.*, 9(8):117–122, 2009.
- [BJDM11] Caroline Baroukh, Sherry Jenkins, Ruth Dannenfelser, and Avi Ma’ayan. Genes2WordCloud: a quick way to identify biological themes from gene lists and free text. *Source code for biology and medicine*, 6:15+, 2011.
- [BO01] Andreas D. Baxevanis and Francis B. F. Ouellette. *Bioinformatics: A Practical Guide to the Analysis of Genes and Proteins, Second Edition*. Wiley-Interscience, April 2001.
- [Bra03] Shannon Bradshaw. Reference directed indexing: Redeeming relevance for subject search in citation indexes. In *In Proc. of the 7th ECDL*, pages 499–510, 2003.
- [Bre96] Leo Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, 1996.
- [Bre01] Leo Breiman. Random forests. *Mach. Learn.*, 45(1):5–32, October 2001.
- [BSR⁺05] Chris Burges, Tal Shaked, Erin Renshaw, Ari Lazier, Matt Deeds, Nicole Hamilton, and Greg Hullender. Learning to rank using gradient

- descent. In *ICML '05: Proceedings of the 22nd international conference on Machine learning*, pages 89–96, New York, NY, USA, 2005. ACM Press.
- [ByRN99] Ricardo Baeza-yates and Berthier Ribeiro-Neto. Modern information retrieval, 1999.
- [Cam00] Rui Camacho. Inducing models of human control skills. In *in Lecture Notes in Computer Science*, pages 107–118. Springer-Verlag, 2000.
- [Cam07] Rui Camacho. *Extracção de conhecimento.*, 2007.
- [CB01] Margaret H. Coletti and Howard L. Bleich. Technical milestone: Medical subject headings used to search the biomedical literature. *JAMIA*, 8(4):317–323, 2001.
- [CBLJ04] David P. A. Corney, Bernard F. Buxton, William B. Langdon, and David T. Jones. Biorat: extracting biological information from full-length papers. *Bioinformatics*, 20(17):3206–3213, November 2004.
- [CH00] Aaron M. Cohen and William R. Hersh. A survey of current work in biomedical text mining. *Briefings in bioinformatics*, 6(1):57–71, March 200.
- [CH05] Aaron M. Cohen and William R. Hersh. A survey of current work in biomedical text mining. *Brief Bioinform*, 6(1):57–71, January 2005.
- [CKY⁺08] Dean Cheng, Craig Knox, Nelson Young, Paul Stothard, Sambasivarao Damaraju, and David S. Wishart. Polysearch: a web-based text mining system for extracting relationships between human diseases, genes, mutations, drugs and metabolites. *Nucleic Acids Research*, 36(Web-Server-Issue):399–405, 2008.
- [CLCF07] E.P. Costa, A.C. Lorena, A.C.P.L.F. Carvalho, and A.A. Freitas. A review of performance evaluation measures for hierarchical classifiers. In C. Drummond, W. Elazmeh, N. Japkowicz, and S.A. Macskassy, editors, *Evaluation Methods for Machine Learning II: papers from the AAAI-2007 Workshop, AAAI Technical Report WS-07-05*, pages 1–6. AAAI Press, July 2007.
- [CM06] Blaise Cronin and Lokman Meho. Using the h-index to rank influential information scientists: Brief communication. *J. Am. Soc. Inf. Sci. Technol.*, 57(9):1275–1278, July 2006.
- [CMBT02] H. Cunningham, D. Maynard, K. Bontcheva, and V. Tablan. Gate: A framework and graphical development environment for robust nlp tools and applications. In *Proceedings of the 40th Annual Meeting of the ACL*, 2002.

- [CNM04] Rich Caruana and Alexandru Niculescu-Mizil. Data mining in metric space: an empirical analysis of supervised learning performance criteria. In *KDD '04: Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 69–78, New York, NY, USA, 2004. ACM.
- [CNMCK04] Rich Caruana, Alexandru Niculescu-Mizil, Geoff Crew, and Alex Ksikes. Ensemble selection from libraries of models. In Carla E. Brodley, editor, *ICML*, volume 69 of *ACM International Conference Proceeding Series*. ACM, 2004.
- [CR96] H. Chu and M. Rosenthal. Search engines for the world wide web: a comparative study and evaluation methodology. pages 127–135, 1996.
- [Cus97] James Cussens. Part-of-speech tagging using progol. In Nada Lavrac and Saso Dzeroski, editors, *ILP*, volume 1297 of *Lecture Notes in Computer Science*, pages 93–108. Springer, 1997.
- [CWD03] Yisong Chen, Guoping Wang, and Shihai Dong. Learning with progressive transductive support vector machine. *Pattern Recognition Letters*, 24(12):1845–1855, 2003.
- [DDS07] M. Indra Devi, Rajaram D.R., and K. Selvakuberan. Page categorization without the web page. In *Proceedings of the International Conference on Computational Intelligence and Multimedia Applications*, volume 2, pages 116–120, 2007.
- [DE00] Saso Dzeroski and Tomaz Erjavec. Learning to lemmatise slovene words. In *Cussens and S. DzĚeroski, Learning Language in Logic, Number 1925 in Lecture notes in artificial intelligence*, pages 69–88. Springer-Verlag, 2000.
- [DGM⁺08] Shyamala Doraisamy, Shahram Golzari, Noris Mohd.Norowi, Md.Nasir B Sulaiman, and Nur Izura Udzir. A study on feature selection and classification techniques for automatic genre classification of traditional malay music. In *9th International Conference on Music Information Retrieval (ISMIR2008)*, 2008.
- [DHM⁺07] Rebholz-Schuhmann D., Kirsch H., Arregui M., Gaudan S., Riethoven M., and Stoehr P. Ebimed - text crunching to gather facts for proteins form medline. *Bioinformatics*, 23(2):e237–e244, 2007.
- [Die00a] Thomas G. Dietterich. Ensemble methods in machine learning. In Josef Kittler and Fabio Roli, editors, *Multiple Classifier Systems*, volume 1857 of *Lecture Notes in Computer Science*, pages 1–15. Springer, 2000.
- [Die00b] Thomas G. Dietterich. Ensemble methods in machine learning. In *Proceedings of the First International Workshop on Multiple Classifier Systems, MCS '00*, pages 1–15, London, UK, UK, 2000. Springer-Verlag.

- [DLWF12] Zhi-Hong Deng, Bo-Yan Lai, Zhonghui Wang, and Guo-Dong Fang. Pav: A novel model for ranking heterogeneous objects in bibliographic information networks. *Expert Syst. Appl.*, 39(10):9788–9796, 2012.
- [DMd⁺03] Ian Donaldson, Joel Martin, Berry de Bruijn, Cheryl Wolting, Vicki Lay, Brigitte Tuekam, Shudong Zhang, Berivan Baskin, Gary D. Bader, Katerina Michalickova, Tony Pawson, and Christopher W. Hogue. Pre-bind and textomy—mining the biomedical literature for protein-protein interactions using a support vector machine. *BMC bioinformatics*, 4, March 2003.
- [Don03] Kim Dongho. A survey on text classification, 2003.
- [DS05] Andreas Doms and Michael Schroeder. Gopubmed: Exploring pubmed with the geneontology. *Nucleic Acid Research*, 33:W783–W786, 2005.
- [Dv04] Saso Džeroski and Bernard Ženko. Is combining classifiers with stacking better than selecting the best one? *Mach. Learn.*, 54:255–273, March 2004.
- [DZ04] Saso Dzeroski and Bernard Zenko. Is combining classifiers with stacking better than selecting the best one? *Machine Learning*, 54(3):255–273, 2004.
- [Eat06] A. D. Eaton. Hubmed: a web-based biomedical literature search interface. *Nucleic acids research*, 34:W745–747, July 2006.
- [Egg06] L Egghe. An improvement of the h-index: The g-index. *ISSI Newsletter*, 2(1):1–4, 2006.
- [Egg12] Leo Egghe. Averages of ratios compared to ratios of averages: Mathematical results. *J. Informetrics*, 6(2):307–317, 2012.
- [ens] Ensembl <http://www.ensembl.org/>.
- [ent] Entrez pubmed <http://www.ncbi.nlm.nih.gov/entrez/>.
- [ER08] L. Egghe and I. K. Ravichandra Rao. Study of different h-indices for groups of authors. *J. Am. Soc. Inf. Sci. Technol.*, 59(8):1276–1281, June 2008.
- [ET93] B. Efron and R. J. Tibshirani. *An Introduction to the Bootstrap*. Chapman & Hall, New York, 1993.
- [EWHG07] Mounir Errami, Jonathan D. Wren, Justin M. Hicks, and Harold R. Garner. etblast: a web server to identify expert reviewers, appropriate journals and similar publications. *Nucleic Acids Research*, 35(Web-Server-Issue):12–15, 2007.

- [EZ02] Y. Even-Zohar. Introduction to text mining., 2002.
- [EZ04] Jae-Hong Eom and Byoung-Tak Zhang. Pubminer: Machine learning-based text mining system for biomedical information mining. In *AIMSA*, pages 216–225, 2004.
- [FB91] Norbert Fuhr and Chris Buckley. A probabilistic learning approach for document indexing. *ACM Trans. Inf. Syst.*, 9(3):223–248, 1991.
- [FBSS⁺09] Jean-Fred Fontaine, Adriano Barbosa-Silva, Martin Schaefer, Matthew R. Huska, Enrique M. Muro, and Miguel A. Andrade-Navarro. MedlineRanker: flexible ranking of biomedical literature. *Nucl. Acids Res.*, 37(suppl.2):W141–146, July 2009.
- [FCS⁺03] Nuno Fonseca, Vitor Santos Costa, O Silva, Nuno Fonseca, Vitor Santos Costa, O Silva, and Rui Camacho. On the implementation of an ilp system with prolog, 2003.
- [FHH⁺10] Eibe Frank, Mark Hall, Geoffrey Holmes, Richard Kirkby, Bernhard Pfahringer, Ian H. Witten, and Len Trigg. Weka-a machine learning workbench for data mining. In Oded Maimon and Lior Rokach, editors, *Data Mining and Knowledge Discovery Handbook*, pages 1269–1277. Springer, 2010.
- [FL04] David Ferrucci and Adam Lally. Uima: an architectural approach to unstructured information processing in the corporate research environment. *Nat. Lang. Eng.*, 10(3-4):327–348, 2004.
- [FM10] Fiorenzo Franceschini and Domenico A. Maisano. Analysis of the hirsch index’s operational properties. *European Journal of Operational Research*, 203(2):494–504, 2010.
- [FS97] Yoav Freund and Robert E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *J. Comput. Syst. Sci.*, 55(1):119–139, 1997.
- [FS07] Ronen Feldman and James Sanger. *The Text Mining Handbook: Advanced Approaches in Analyzing Unstructured Data*. Cambridge University Press, 2007.
- [FSC06] Nuno A. Fonseca, Fernando M. A. Silva, and Rui Camacho. April - an inductive logic programming system. In *JELIA*, pages 481–484, 2006.
- [Gar64] Eugene Garfield. ”Science Citation Index”—A New Dimension in Indexing. *Science*, 144(3619):649–654, May 1964.
- [Gar99] E Garfield. Journal impact factor: a brief review. *CMAJ Canadian Medical Association Journal*, 161(8):979–980, 1999.

- [GGCO10] Carlos Adriano Gonçalves, Célia Talma Gonçalves, Rui Camacho, and Eugénio C. Oliveira. The impact of pre-processing on the classification of medline documents. In Ana L. N. Fred, editor, *Pattern Recognition in Information Systems, Proceedings of the 10th International Workshop on Pattern Recognition in Information Systems, PRIS 2010, In conjunction with ICEIS 2010, Funchal, Madeira, Portugal, June 2010*, pages 53–61, 2010.
- [Glä06] Wolfgang Glänzel. On the opportunities and limitations of the h-index. *Science Focus*, 1(1):10–11, 2006.
- [GvdL05] Thomas Goetz and Claus-Wilhelm von der Lieth. PubFinder: a tool for improving retrieval rate of relevant PubMed abstracts. *Nucleic acids research*, 33(Web Server issue), July 2005.
- [HdVLG06] Tim Hulsen, Jacob de Vlieg, Jack A. M. Leunissen, and Peter M. A. Groenen. Testing statistical significance scores of sequence comparison methods with structure similarity. *BMC Bioinformatics*, 7:444, 2006.
- [Hea99] Marti A. Hearst. Untangling text data mining. In *Proceedings of the 37th annual meeting of the Association for Computational Linguistics on Computational Linguistics, ACL '99*, pages 3–10, Stroudsburg, PA, USA, 1999. Association for Computational Linguistics.
- [HFH⁺09] Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H. Witten. The weka data mining software: an update. *SIGKDD Explor. Newsl.*, 11(1):10–18, November 2009.
- [HG04] Erik Hatcher and Otis Gospodnetic. *Lucene in Action (In Action series)*. Manning Publications Co., Greenwich, CT, USA, 2004.
- [HH10] Verena Henrich and Erhard Hinrichs. Standardizing wordnets in the iso standard lmf: Wordnet-lmf for germanet. In *Proceedings of the 23rd International Conference on Computational Linguistics, COLING '10*, pages 456–464, Stroudsburg, PA, USA, 2010. Association for Computational Linguistics.
- [Hie01] Djoerd Hiemstra. *Using language models for information retrieval*. Univ. Twente, 2001.
- [Hir10] J. E. Hirsch. An index to quantify an individual’s scientific research output that takes into account the effect of multiple coauthorship. *Scientometrics*, 85(3):741–754, December 2010.
- [Hoo11] Apirak Hoonlor. Sequential patterns and temporal patterns for text mining, 2011.
- [Hos04] The hosford medical terms dictionary v3.0, 2004.

- [Hot05] Andreas Hotho. A brief survey of text mining. *Journal for Computational Linguistics and Technology*, 2005.
- [HSA10] Homayouni H., Hashemi S., and Hamzeh A. A lazy ensemble learning method to classification. *International Journal of Computer Science Issues*, 7, 2010.
- [HSO12] Dr. Syed Akif Hasan, Dr. Muhammad Imtiaz Subhani, and Ms. Amber Osman. H-index: The key to research output assessment. Technical report, 2012.
- [HSSF09] Junguk Hur, Adam D. Schuyler, David J. States, and Eva L. Feldman. Sciminer: web-based literature mining tool for target identification and functional enrichment analysis. *Bioinformatics*, 25(6):838–840, 2009.
- [hT99] Ah hwee Tan. Text mining: The state of the art and the challenges. In *In Proceedings of the PAKDD 1999 Workshop on Knowledge Discovery from Advanced Databases*, pages 65–70, 1999.
- [IM05] Zak I. and Ciura M. Automatic text categorization, 2005.
- [Jär07] Kalervo Järvelin. An analysis of two approaches in information retrieval: From frameworks to study designs. *J. Am. Soc. Inf. Sci. Technol.*, 58(7):971–986, 2007.
- [JdAL99] Alípio Jorge and Alneu de Andrade Lopes. Iterative part-of-speech tagging. In *Learning Language in Logic'99*, pages 170–183, 1999.
- [JHGJ96] Epstein J., Ohkawa H., Schuler G., and Hans J. Entrez: Moldecular biology database and retrieval system, 1996.
- [Joa98] T. Joachims. Text categorization with support vector machines: Learning with many relevant features. In *European Conference on Machine Learning (ECML)*, pages 137–142, Berlin, 1998. Springer.
- [KEV05] Martin Krallinger, Ramon Alonso-Allende A. Erhardt, and Alfonso Valencia. Text-mining approaches in molecular biology and biomedicine. *Drug discovery today*, 10(6):439–445, March 2005.
- [Kle99] Jon M. Kleinberg. Authoritative sources in a hyperlinked environment. *J. ACM*, 46(5):604–632, 1999.
- [KMS⁺08] Martin Krallinger, Alexander Morgan, Larry Smith, Florian Leitner, Lorraine Tanabe, John Wilbur, Lynette Hirschman, and Alfonso Valencia. Evaluation of text-mining systems for biology: overview of the second biocreative community challenge. *Genome Biology*, 9(Suppl 2), 2008.

- [KRJ09] Antonis Koussounadis, Oliver Redfern, and David T. Jones. Improving classification in protein structure databases using text mining. *BMC Bioinformatics*, 10, 2009.
- [KS08] Carl Kingsford and Steven L. Salzberg. What are decision trees? *Nature biotechnology*, 26(9):1011–1013, September 2008.
- [KYB03] Ian Korf, Mark Yandell, and Joseph A. Bedell. *BLAST - an essential guide to the basic local alignment search tool*. O’Reilly, 2003.
- [LCC⁺09] Anália Lourenço, Rafael Carreira, Sónia Carneiro, Paulo Maia, Daniel Glez-Peña, Florentino Fdez-Riverola, Eugénio C. Ferreira, Isabel Rocha, and Miguel Rocha. @note: A workbench for biomedical text mining. *Journal of Biomedical Informatics*, 42(4):710–720, 2009.
- [LD94] N. Lavrač and S. Džeroski. *Inductive Logic Programming: Techniques and Applications*. 1994.
- [LF01] Nada Lavrač and Peter A. Flach. An extended transformation approach to inductive logic programming. *ACM Trans. Comput. Logic*, 2(4):458–494, 2001.
- [LGG01] N. M. Luscombe, D. Greenbaum, and M. Gerstein. What is bioinformatics? An introduction and overview. Technical report, Department of Molecular Biophysics and Biochemistry Yale University New Haven, USA, 2001.
- [Lip09] Giuseppe Lippi. The impact factor for evaluating scientists: the good, the bad and the ugly. *Clin Chem Lab Med*, 47(12):1585–6, 2009.
- [LLCL07] Yongjing Lin, Wenyuan Li, Keke Chen, and Ying Liu. Model formulation: A document clustering and ranking system for exploring medline citations. *JAMIA*, 14(5):651–661, 2007.
- [LLCM03] Tao Liu, Shengping Liu, Zheng Chen, and Wei-Ying Ma. An evaluation on feature selection for text clustering. In *Machine Learning, Proceedings of the Twentieth International Conference (ICML 2003), August 21-24, 2003, Washington, DC, USA*, pages 488–495, 2003.
- [LM01] R. Lempel and S. Moran. Salsa: the stochastic approach for link-structure analysis. *ACM Trans. Inf. Syst.*, 19(2):131–160, April 2001.
- [LS94] Pat Langley and Stephanie Sage. Induction of selective bayesian classifiers. In Ramon López de Mántaras and David Poole, editors, *UAI*, pages 399–406. Morgan Kaufmann, 1994.
- [LTSL07] Man Lan, Chew Lim Tan, Jian Su, and Hwee-Boon Low. Text representations for text categorization: A case study in biomedical

- domain. In *Proceedings of the International Joint Conference on Neural Networks, IJCNN 2007, Celebrating 20 years of neural networks, Orlando, Florida, USA, August 12-17, 2007*, pages 2557–2562, 2007.
- [LUL⁺99] Tanable L., Scher U., Smith L.H., Lee J.K., Hunter L., and Weinstein J.N. An internet text mining tool for biomedical information, with application to gene expression profiling. *BioTechniques*, 27:1210–1217, 1999.
- [MAS06] Miriam Martinez-Arroyo and Luis Enrique Sucar. Learning an optimal naive bayes classifier. In *ICPR (4)*, page 958. IEEE Computer Society, 2006.
- [MC00] Mandar Mitra and B. B. Chaudhuri. Information retrieval from documents: A survey. *Inf. Retr.*, 2(2/3):141–163, 2000.
- [MKS04] Hans-Michael M. Müller, Eimear E. Kenny, and Paul W. Sternberg. Textpresso: an ontology-based information retrieval and extraction system for biological literature. *PLoS biology*, 2(11), November 2004.
- [MN98] Andrew McCallum and Kamal Nigam. A comparison of event models for naive bayes text classification, 1998.
- [MR07] Joao Magalhaes and Stefan Rueger. High-dimensional visual vocabularies for image retrieval. In *SIGIR '07: Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 815–816, New York, NY, USA, 2007. ACM.
- [MSI08] Bawaneh M.J., Alkoffash M. S., and Al Rabea A. I. Arabic text classification using k-nn and naïve bayes. *Journal of Computer Science*, 4(7):600–605, 2008.
- [Mug90] Stephen Muggleton. Inductive logic programming. In *Proceedings of the 1st Conference on Algorithmic Learning Theory*, pages 43–62, 1990.
- [Mug99] Stephen Muggleton. Inductive logic programming: Issues, results and the challenge of learning language in logic. *Artificial Intelligence*, 114(1-2):283–296, 1999.
- [MWK⁺06] Ingo Mierswa, Michael Wurst, Ralf Klinkenberg, Martin Scholz, and Timm Euler. Yale: rapid prototyping for complex data mining tasks. In *KDD '06: Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 935–940, New York, NY, USA, 2006. ACM.
- [NHB⁺06] Jeyakumar Natarajan, Cliff Haines, Brian Berglund, Catherine Desesa, Catherine Hack, Werner Dubitzky, and Eric Bremer. Getitfull a tool for downloading and pre-processing full-text journal articles. *Knowledge Discovery in Life Science Literature*, pages 139–145, 2006.

- [OM99] David Opitz and Richard Maclin. Popular ensemble methods: An empirical study. *Journal of Artificial Intelligence Research*, 11:169–198, 1999.
- [PBMW99] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The pagerank citation ranking: Bringing order to the web. Technical Report 1999-66, Stanford InfoLab, November 1999. Previous number = SIDL-WP-1999-0120.
- [PIPBA03] Carolina Perez-Iratxeta, Antonio Jesús Pérez, Peer Bork, and Miguel A. Andrade. Update on xplormed: a web server for exploring scientific literature. *Nucleic Acids Research*, 31(13):3866–3868, 2003.
- [PKSR02] Vili Podgorelec, Peter Kokol, Bruno Stiglic, and Ivan Rozman. Decision trees: an overview and their use in medicine. *Journal of Medical Systems*, 26:445–463, 2002.
- [PMA⁺07] Smialowski Pawel, Galiano Antonio J. Martin, Mikolajka Aleksandra, Girschick Tobias, Holak Tad A., and Frishman Dimitrij. Protein solubility: sequence based prediction and experimental verification, 2007.
- [Por80] M. F. Porter. An algorithm for suffix stripping. *Program*, 14(3):130–137, 1980.
- [Por97] M. F. Porter. An algorithm for suffix stripping. pages 313–316, 1997.
- [Pow98] David M.W. Powers. Applications and explanations of zipf’s law, 1998.
- [PRAS08] Graham L. Poulter, Daniel L. Rubin, Russ B. Altman, and Cathal Seoighe. Mscanner: a classifier for retrieving medline citations. *BMC Bioinformatics*, 9, 2008.
- [PZC06] Maksim Plikus, Zina Zhang, and Cheng M. Chuong. PubFocus: semantic MEDLINE/PubMed citations analytics through integration of controlled biomedical dictionaries and ranking algorithm. *BMC Bioinformatics*, 7(1):424+, October 2006.
- [Qui93] J. Ross Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, 1993.
- [RBAC⁺07] Nattakarn Ratprasartporn, Sulieman Bani-Ahmad, Ali Cakmak, Jonathan Po, and Gultekin Özsoyoglu. Evaluating different ranking functions for context-based literature search. In *ICDE Workshops*, pages 261–268. IEEE Computer Society, 2007.
- [RHS12] Muhammad Rafi, Sundus Hassan, and Mohammad Shahid Shaikh. Content-based text categorization using wikitology. *CoRR*, abs/1208.3623, 2012.

- [RI95] Feldman R. and Dagan I. Kdt-knowledge discovery in texts. In *In Proceedings of the First International Conference on Knowledge Discovery (KDD-95)*., 1995.
- [RJB89] Vijay V. Raghavan, Gwang S. Jung, and Peter Bollmann. A critical investigation of recall and precision as measures of retrieval system performance. *ACM Trans. Inf. Syst.*, 7(3):205–229, 1989.
- [RKP07] Dieng-Kuntz R., Khelif K., and Barbry P. Mining biomedical texts to generate semantic annotations, 2007.
- [RM] Lior Rokach and Oded Maimon. Chapter 9 decision trees.
- [RNS05] Simon B. Rice, Goran Nenadic, and Benjamin J. Stapley. Mining protein function from text using term-based support vector machines. *BMC Bioinformatics*, 6(S-1), 2005.
- [RSPK⁺08] D. Rebholz-Schuhmann, P. Pezik, V. Lee J-J Kim, R. del Gratta, Y. Sasaki, J. McNaught, S. Montemagni, M. Monachini, N. Calzolari, and S. Ananiadou. Biolexicon: Towards a reference terminological resource in the biomedical domain. In *Proceedings of the of the 16th Annual International Conference on Intelligent Systems for Molecular Biology (ISMB-2008)*, 2008.
- [RSTK03] Jason D. M. Rennie, Lawrence Shih, Jaime Teevan, and David R. Karger. Tackling the poor assumptions of naive bayes text classifiers. In *In Proceedings of the Twentieth International Conference on Machine Learning*, pages 616–623, 2003.
- [RW94] S. E. Robertson and S. Walker. Some simple effective approximations to the 2-poisson model for probabilistic weighted retrieval. In *In Proceedings of SIGIR 94*, pages 232–241. Springer-Verlag, 1994.
- [SAC07] Mark D. Smucker, James Allan, and Ben Carterette. A comparison of statistical significance tests for information retrieval evaluation. In *Proceedings of the sixteenth ACM conference on Conference on information and knowledge management, CIKM '07*, pages 623–632, New York, NY, USA, 2007. ACM.
- [SAJ⁺97] Altschul S.F., Schaffer A.A., Zhang J., Zhang Z., Miller W., and Lipman D.J. Grapped blast and psi-blast:a new generation of protein database search programs. 25:3389–3402, 1997.
- [Sal89] Gerard Salton. *Automatic Text Processing: The Transformation, Analysis and Retrieval of Information by Computer*. Addison-Wesley, 1989.
- [SBCH06] Nigel Shadbolt, Tim Brody, Les Carr, and Stevan Harnad. The open research web: A preview of the optimal and the inevitable, 2006.

- [SBMM96] Amit Singhal, Chris Buckley, Mandar Mitra, and Ar Mitra. Pivoted document length normalization. pages 21–29. ACM Press, 1996.
- [SC99] Fei Song and W. Bruce Croft. A general language model for information retrieval. In *In Proceedings of the 1999 ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 279–280, 1999.
- [Seb99] Fabrizio Sebastiani. A tutorial on automated text categorisation. In *Proceedings of ASAI-99, 1st Argentinian Symposium on Artificial Intelligence, Buenos Aires*, pages 7–35. 1999.
- [Seb05] Fabrizio Sebastiani. Text categorization. pages 109–129, 2005.
- [Seg97] P. O. Seglen. Why the impact factor of journals should not be used for evaluating research. *BMJ (Clinical research ed.)*, 314(7079):498–502, February 1997.
- [Set05] Burr Settles. Abner: an open source tool for automatically tagging genes, proteins and other entity names in text. *Bioinformatics*, 21:3191–3192, July 2005.
- [SFS10] Giulietta M. Spudich and Xosé M. Fernández-Suárez. Touring Ensembl: a practical guide to genome browsing. *BMC genomics*, 11(1):295+, May 2010.
- [SG09] Hassan Sayyadi and Lise Getoor. Futurerank: Ranking scientific articles by predicting their future pagerank. In *SDM*, pages 533–544. SIAM, 2009.
- [SK08] Martijn J. Schuemie and Jan A. Kors. Jane: suggesting journals, finding experts. *Bioinformatics*, 24(5):727–728, March 2008.
- [Smi07] Derek R Smith. Historical development of the journal impact factor and its relevance for occupational health. *Ind Health*, 45(6):730–42, 2007.
- [SMNWH00] Steffen Staab, Alexander Maedche, Claire Nedellec, and Peter M. Wiemer-Hastings, editors. *ECAI’2000 Workshop on Ontology Learning, Proceedings of the First Workshop on Ontology Learning OL’2000, Berlin, Germany, August 25, 2000. Held in conjunction with the 14th European Conference on Artificial Intelligence ECAI’2000, Berlin, Germany*, volume 31 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2000.
- [SP04] Kotsiantis S. and Pintelas P. Combining bagging and boosting. *International Journal of Computational Intelligence*, 4:324–333, 2004.

- [SR02] Fabrizio Sebastiani and Consiglio Nazionale Delle Ricerche. Machine learning in automated text categorization. *ACM Computing Surveys*, 34:1–47, 2002.
- [Sri] A. Srinivasan. The aleph manual, 2003. available from <http://web.comlab.ox.ac.uk/oucl/research/areas/machlearn/aleph>.
- [Sri04] Ashwin Srinivasan. The aleph manual, 2004.
- [SSK07] Mir Siadaty, Jianfen Shu, and William Knaus. Relemed: sentence-level search engine with relevance score for the MEDLINE database of biomedical articles. *BMC Medical Informatics and Decision Making*, 7(1):1+, January 2007.
- [SSKN07] Zeil S., Zubair S., Maly K., and Ratkal N. A machine learning approach for automatic text categorization: A case study for dtic collection. In *Proceedings of International Conference on Soft Computing, Intelligent System and Information Technology (ICSIT 2007)*, pages 128–132, 2007.
- [SSL10] S.Rajesh, S.Prathima, and L.S.S.Reddy. Article:unusual pattern detection in dna database using kmp algorithm. *International Journal of Computer Applications*, 1(22):1–5, February 2010. Published By Foundation of Computer Science.
- [SSLM11] Richard M. Simon, Jyothi Subramanian, Ming-Chung Li, and Supriya Menezes. Using cross-validation to evaluate predictive accuracy of survival risk classifiers based on high-dimensional data. *Briefings in Bioinformatics*, 12(3):203–214, 2011.
- [SSMN09] I.N. Sarkar, R. Schenk, H. Miller, and C.N. Norton. Ligercat: using ”mesh clouds” from journal, article, or gene citations to facilitate the identification of relevant biomedical literature. *AMIA Annu Symp Proc*, 2009, 2009.
- [SW] L. Smith and W.J. Wilbur. The popularity of articles in pubmed. *The Open Information Systems Journal, National Center for Biotechnology Information, Bethesda, Maryland, USA*.
- [SW09] M. Song and Y.F. Wu. *Handbook of Research on Text and Web Mining Technologies*. Number vol. 1 in Handbook of Research on Text and Web Mining Technologies. Igi Global, 2009.
- [SWY75] G. Salton, A. Wong, and C.S. Yang. A vector space model for automatic indexing. *Communications of the ACM*, 18(11):613–620, 1975.
- [Tip00] Michael E. Tipping. The relevance vector machine, 2000.

- [TJS08] J Tsuruoka, Tsujii J., and Ananiadou S. Facta: A text search engine for finding biomedical concepts. 2008.
- [TTD04] Qiang Tu, Haixu Tang, and Dafu Ding. Medblast: searching articles related to a biological sequence. pages 75–77, 2004.
- [VA10] K. A Vidhya and G Aghila. Text mining process, techniques and tools: an overview. *International Journal of Information Technology and Knowledge Management*, 2(2):613–622, 2010.
- [Van05] Anthony F J Van Raan. Comparison of the hirsch-index with standard bibliometric indicators and with peer judgment for 147 chemistry research groups. *Scientometrics*, 67(3):12, 2005.
- [Vap99] Vladimir N. Vapnik. *The Nature of Statistical Learning Theory (Information Science and Statistics)*. Springer, November 1999.
- [VVM⁺03] T N Van Leeuwen, M S Visser, H F Moed, T J Nederhof, and Afj Van Raan. The holy grail of science policy: Exploring and combining bibliometric tools in search of scientific excellence. *Scientometrics*, 57(2):257–280, 2003.
- [W.64] Sewell W. Medical subject headings in medlars, 1964.
- [WBB⁺06] D. L. Wheeler, T. Barrett, D. A. Benson, S. H. Bryant, K. Canese, V. Chetvernin, D. M. Church, M. Dicuccio, R. Edgar, S. Federhen, L. Y. Geer, W. Helmberg, Y. Kapustin, D. L. Kenton, O. Khovayko, D. J. Lipman, T. L. Madden, D. R. Maglott, J. Ostell, K. D. Pruitt, G. D. Schuler, L. M. Schriml, E. Sequeira, S. T. Sherry, K. Sirotkin, A. Souvorov, G. Starchenko, T. O. Suzek, R. Tatusov, T. A. Tatusova, L. Wagner, and E. Yaschenko. Database resources of the national center for biotechnology information. *Nucleic Acids Res*, 34(Database issue), January 2006.
- [WFT⁺99] Ian H. Witten, Eibe Frank, Len Trigg, Mark Hall, Geoffrey Holmes, and Sally Jo Cunningham. Weka: Practical machine learning tools and techniques with java implementations, 1999.
- [W.R03] Hears W.R. Information retrieval: a health and biomedical perspective. 2003.
- [YEH09] Ashkan Yazdani, Touradj Ebrahimi, and Ulrich Hoffmann. Classification of eeg signals using dempster shafer theory and a k-nearest neighbor classifier, 2009.
- [YKO⁺09] Hwanjo Yu, Taehoon Kim, Jinoh Oh, Ilhwan Ko, and SungChul Kim. Refmed: relevance feedback retrieval system fo pubmed. In David Wai-Lok Cheung, Il-Yeol Song, Wesley W. Chu, Xiaohua Hu, and

- Jimmy J. Lin, editors, *Proceedings of the 18th ACM Conference on Information and Knowledge Management, CIKM 2009, Hong Kong, China, November 2-6, 2009*, pages 2099–2100. ACM, 2009.
- [ZFT⁺11] Ming Zhang, Sheng Feng, Jian Tang, Bolanle Adefowoke Ojokoh, and Guojun Liu. Co-ranking multiple entities in a heterogeneous network: Integrating temporal factor and users' bookmarks. In Chunxiao Xing, Fabio Crestani, and Andreas Rauber, editors, *ICADL*, volume 7008 of *Lecture Notes in Computer Science*, pages 202–211. Springer, 2011.
- [ZlA02] Osmar R. Zaiane and Maria luiza Antonie. Classifying text documents by associating terms with text categories. In *In Proc. of the Thirteenth Australasian Database Conference (ADC-02)*, pages 215–222, 2002.
- [ZLL11] Yan-Bo Zhou, Linyuan Lü, and Menghui Li. Quantifying the influence of scientists and their publications: Distinguish prestige from popularity. *CoRR*, abs/1109.1186, 2011.
- [ZSY06] Wei Zhou, Neil R. Smalheiser, and Clement Yu. A tutorial on information retrieval: basic terms and concepts. *Journal of Biomedical Discovery and Collaboration*, 1:2+, March 2006.
- [ZZJ09] Lu Z., Xien Z., and Wilbur W. J. Identifying related journals through log analysis. 25, 2009.

Appendix A

Annexes

Table A.1: Accuracy Classification results. Values with an attached '*' were obtained interrupting Weka after 3 months of execution. In each cell is the accuracy and the standard deviation in parenthesis. Bold values are the best results for each data set. The value for cross validation is 10. The last column presents the average of accuracy of the all algorithms for each data set.

Data Set ID	smo	ibk	J48	Average
Dataset1	94.95 (0.18)	63.99 (0.41)	95.64 (0.34)*	84.86 (0.31)
Dataset2	93.74 (0.29)	70.05 (0.55)	93.62 (0.48)	85.80 (0.44)
Dataset3	93.42 (0.33)	69.60 (0.55)	93.49 (0.57)	85.50 (0.48)
Dataset4	93.84 (0.24)	71.04 (0.56)	93.49 (0.57)	86.12 (0.46)
Dataset5	95.18 (0.10)	59.20 (1.2)	95.31 (0.27)*	83.23 (0.52)
Dataset6	94.76 (0.15)	65.81 (0.44)	95.63 (0.31)	85.40 (0.30)
Dataset7	94.66 (0.19)	66.19 (0.52)	95.63 (0.31)	85.49 (0.34)
Dataset8	94.60 (0.22)	66.36 (0.92)	95.63 (0.31)	85.53 (0.48)
Dataset9	94.95 (0.22)	67.43 (0.87)	95.67 (0.33)*	86.02 (0.47)
Dataset10	94.37 (0.21)	68.41 (1.07)	95.58 (0.30)	86.12 (0.53)
Dataset11	94.73 (0.12)	63.83 (0.81)	95.55 (0.35)	84.70 (0.43)
Dataset12	94.59 (0.20)	64.69 (0.47)	95.55 (0.35)	84.94 (0.34)
Dataset13	94.46 (0.25)	65.04 (0.91)	95.55 (0.35)	85.02 (0.50)
Dataset14	95.19 (0.12)	61.05 (1.41)	95.35 (0.34)*	83.86 (0.62)
Dataset15	94.91 (0.14)	66.19 (1.24)	95.66 (0.37)	85.59 (0.58)
Dataset16	94.87 (0.14)	67.11 (1.07)	95.66 (0.37)	85.88 (0.53)
Dataset17	94.76 (0.24)	67.04 (0.72)	95.66 (0.37)	85.82 (0.44)
Dataset18	93.50 (0.24)	71.54 (0.49)	93.23 (0.54)	86.09 (0.42)
Dataset19	94.35 (0.23)	68.63 (0.88)	95.58 (0.30)	86.19 (0.47)
Dataset20	94.39 (0.17)	68.84 (0.80)	95.58 (0.30)	86.27 (0.42)
Dataset21	94.60 (0.39)	68.97 (0.59)	95.58 (0.30)	86.38 (0.39)
Dataset22	94.23 (0.25)	68.70 (0.58)	95.47 (0.34)	86.13 (0.39)
Dataset23	93.51 (0.24)	70.67 (0.55)	93.23 (0.54)	85.80 (0.44)
Dataset24	93.56 (0.24)	70.23 (0.59)	93.23 (0.54)	85.67 (0.46)
Dataset25	94.00 (0.24)	71.27 (0.58)	93.23 (0.54)	86.17 (0.45)
Dataset26	94.83 (0.25)	66.70 (1.04)	95.58 (0.30)*	85.70 (0.53)
Dataset27	94.76 (0.14)	68.41 (0.60)	95.58 (0.30)	86.25 (0.35)
Dataset28	94.69 (0.22)	71.92 (0.70)	95.58 (0.30)	87.40 (0.41)
Dataset29	93.31 (0.22)	70.38 (0.63)	93.49 (0.57)	85.73 (0.47)
Dataset30	94.23 (0.26)	68.94 (0.50)	95.47 (0.34)	86.21 (0.37)
Dataset31	94.24 (0.20)	69.00 (0.66)	95.47 (0.34)	86.24 (0.40)
Dataset32	94.47 (0.33)	68.88 (0.60)	95.47 (0.34)	86.27 (0.42)
Number of wins	7	0	25	—

Table A.2: True Positive Classification results. Values with an attached ’*’ were obtained interrupting Weka after 3 months of execution”.In each cell is the true positives and the standard deviation in parenthesis. Bold values are the best results for each data set. The value for cross validation is 10. The last column presents the average of true positives of the all algorithms for each data set

Data Set ID	smo	ibk	J48	Average
Dataset1	0.95 (0.0017)	0.64 (0.0043)	0.96 (0.0034)*	0.85 (0.0031)
Dataset2	0.94 (0.0029)	0.70 (0.0054)	0.94 (0.0050)	0.86 (0.0044)
Dataset3	0.93 (0.0033)	0.70 (0.0056)	0.93 (0.0052)	0.85 (0.0047)
Dataset4	0.94 (0.0023)	0.71 (0.0056)	0.93 (0.0052)	0.86 (0.0044)
Dataset5	0.95 (0.0011)	0.59 (0.0100)	0.95 (0.0027)*	0.83 (0.0046)
Dataset6	0.95 (0.0015)	0.66 (0.0043)	0.96 (0.0032)	0.86 (0.0030)
Dataset7	0.95 (0.0020)	0.66 (0.0050)	0.96 (0.0032)	0.86 (0.0034)
Dataset8	0.95 (0.0022)	0.66 (0.0093)	0.96 (0.0032)	0.86 (0.0049)
Dataset9	0.95 (0.0022)	0.67 (0.0086)	0.96 (0.0034)*	0.86 (0.0047)
Dataset10	0.94 (0.0019)	0.68 (0.0106)	0.96 (0.0031)	0.86 (0.0052)
Dataset11	0.95 (0.0013)	0.64 (0.0080)	0.96 (0.0035)	0.85 (0.0043)
Dataset12	0.95 (0.0021)	0.65 (0.0047)	0.96 (0.0035)	0.85 (0.0034)
Dataset13	0.94 (0.0025)	0.65 (0.0091)	0.96 (0.0035)	0.85 (0.0050)
Dataset14	0.95 (0.0011)	0.61 (0.0096)	0.95 (0.0033)*	0.84 (0.0047)
Dataset15	0.95 (0.0012)	0.66 (0.0109)	0.96 (0.0037)	0.86 (0.0023)
Dataset16	0.95 (0.0014)	0.67 (0.0107)	0.96 (0.0037)	0.86 (0.0053)
Dataset17	0.95 (0.0025)	0.67 (0.0073)	0.96 (0.0037)	0.86 (0.0045)
Dataset18	0.94 (0.0024)	0.72 (0.0050)	0.93 (0.0041)	0.86 (0.0038)
Dataset19	0.94 (0.0024)	0.69 (0.0088)	0.96 (0.0031)	0.86 (0.0048)
Dataset20	0.94 (0.0017)	0.69 (0.0078)	0.96 (0.0031)	0.86 (0.0042)
Dataset21	0.95 (0.0038)	0.69 (0.0058)	0.96 (0.0031)	0.87 (0.0042)
Dataset22	0.94 (0.0025)	0.69 (0.0056)	0.95 (0.0028)	0.86 (0.0036)
Dataset23	0.94 (0.0025)	0.71 (0.0055)	0.93 (0.0041)	0.86 (0.0040)
Dataset24	0.94 (0.0025)	0.70 (0.0061)	0.93 (0.0041)	0.86 (0.0042)
Dataset25	0.94 (0.0025)	0.71 (0.0060)	0.93 (0.0041)	0.86 (0.0042)
Dataset26	0.95 (0.0027)	0.67 (0.010)	0.96 (0.0032)*	0.86 (0.0023)
Dataset27	0.95 (0.0013)	0.68 (0.0061)	0.96 (0.0032)	0.86 (0.0035)
Dataset28	0.95 (0.0022)	0.72 (0.0076)	0.96 (0.0032)	0.88 (0.0043)
Dataset29	0.93 (0.0022)	0.70 (0.0062)	0.93 (0.0052)	0.85 (0.0045)
Dataset30	0.94 (0.0025)	0.69 (0.0055)	0.95 (0.0028)	0.86 (0.0036)
Dataset31	0.94 (0.0021)	0.69 (0.0065)	0.95 (0.0028)	0.86 (0.0038)
Dataset32	0.94 (0.0032)	0.69 (0.0060)	0.95 (0.0028)	0.86 (0.0040)
Number of wins	11	0	21	—

Table A.3: F-Measure Classification results. In each cell is the f-measure and the standard deviation in parenthesis. Bold values are the best results for each data set. The value for cross validation is 10. The last column presents the average of f-measure of the all algorithms for each data set.

Data Set ID	smo	ibk	J48	Average
Dataset1	0.95 (0.0017)	0.64 (0.0033)	0.96 (0.0034)*	0.85 (0.0028)
Dataset2	0.94 (0.0029)	0.70 (0.0048)	0.94 (0.0047)	0.86 (0.0041)
Dataset3	0.93 (0.0033)	0.70 (0.0050)	0.93 (0.0047)	0.85 (0.0043)
Dataset4	0.94 (0.0023)	0.71 (0.0051)	0.93 (0.0047)	0.86 (0.0040)
Dataset5	0.95 (0.0011)	0.59 (0.0098)	0.95 (0.0027)*	0.83 (0.0045)
Dataset6	0.95 (0.0015)	0.66 (0.0042)	0.96 (0.0032)	0.86 (0.0030)
Dataset7	0.95 (0.0020)	0.66 (0.0054)	0.96 (0.0032)	0.86 (0.0035)
Dataset8	0.95 (0.0022)	0.67 (0.0091)	0.96 (0.0032)	0.86 (0.0048)
Dataset9	0.95 (0.0022)	0.68 (0.0091)	0.96 (0.0034)*	0.86 (0.0049)
Dataset10	0.94 (0.0019)	0.68 (0.0118)	0.96 (0.0031)	0.86 (0.0056)
Dataset11	0.95 (0.0013)	0.64 (0.0058)	0.96 (0.0035)	0.85 (0.0035)
Dataset12	0.95 (0.0021)	0.65 (0.0042)	0.96 (0.0035)	0.85 (0.0033)
Dataset13	0.94 (0.0025)	0.65 (0.0097)	0.96 (0.0035)	0.85 (0.0052)
Dataset14	0.95 (0.0011)	0.61 (0.010)	0.95 (0.0033)*	0.84 (0.0048)
Dataset15	0.95 (0.0012)	0.67 (0.0114)	0.96 (0.0037)	0.86 (0.0054)
Dataset16	0.95 (0.0014)	0.67 (0.0089)	0.96 (0.0037)	0.86 (0.0047)
Dataset17	0.95 (0.0025)	0.67 (0.0081)	0.96 (0.0037)	0.86 (0.0048)
Dataset18	0.94 (0.0024)	0.72 (0.0042)	0.93 (0.0040)	0.86 (0.0035)
Dataset19	0.94 (0.0024)	0.68 (0.0098)	0.96 (0.0031)	0.86 (0.0051)
Dataset20	0.94 (0.0017)	0.69 (0.0089)	0.96 (0.0031)	0.86 (0.0046)
Dataset21	0.95 (0.0038)	0.69 (0.0062)	0.96 (0.0031)	0.87 (0.0044)
Dataset22	0.94 (0.0025)	0.68 (0.0066)	0.95 (0.0028)	0.86 (0.0040)
Dataset23	0.93 (0.0025)	0.71 (0.0051)	0.93 (0.0040)	0.86 (0.0039)
Dataset24	0.94 (0.0025)	0.70 (0.0053)	0.93 (0.0040)	0.86 (0.0039)
Dataset25	0.94 (0.0024)	0.72 (0.0054)	0.93 (0.0040)	0.86 (0.0039)
Dataset26	0.95 (0.0027)	0.67 (0.0105)	0.96 (0.0031)*	0.86 (0.0054)
Dataset27	0.95 (0.0013)	0.69 (0.0059)	0.96 (0.0031)	0.87 (0.0034)
Dataset28	0.95 (0.0022)	0.72 (0.0072)	0.96 (0.0031)	0.88 (0.0042)
Dataset29	0.93 (0.0022)	0.70 (0.0060)	0.93 (0.0047)	0.85 (0.0043)
Dataset30	0.94 (0.0025)	0.69 (0.0061)	0.95 (0.0028)	0.86 (0.0038)
Dataset31	0.94 (0.0021)	0.69 (0.0070)	0.95 (0.0028)	0.86 (0.0040)
Dataset32	0.94 (0.0032)	0.69 (0.0059)	0.95 (0.0028)	0.86 (0.0040)
Number of wins	10	0	22	—

Index

- Evaluation and Discussion, 109
- A tool for Text Mining in Molecular Biology's Domains, 53
- Algorithms for Text Mining, 20
- Alternative 2, 86
- Alternative 3, 87
- Alternative 4, 88
- Alternative 5, 89
- Assessing the method of choosing irrelevant papers, 75
- Assessing the Pre-Processing Techniques, 63
- Author's Number of Publications, 98
- BioTextRetriever, 3
- BioTextRetriever Overview, 53
- Boolean Model, 12
- chapter2Summary, 50
- chapter3Summary, 70
- chapter4Summary, 91
- chapter5Summary, 110
- Choosing the Ranking Function Coefficients, 105
- Classifier Construction Process, 79
- Classifier Evaluation, 32
- Classifier Learning, 24
- Comparing propositional and ILP results, 85
- Comparing the five alternatives, 89
- Complementing the data sets, 72
- Conclusions and Further Research, 113
- Constructing a data set, 71
- Context and Problem, 1
- Data Description, 106
- Decision Trees, 28
- Document Indexing, 22
- Ensemble Classifiers, 29
- Evaluating the Alternatives, 82
- Experimental Procedure, 107
- Experimental Settings, 106
- Extensions to the available Information, 57
- From Sequences to Papers, 71
- Further Research, 115
- General Tools for and Text Mining, 33
- h-index, 100
- How to use BioTextRetriever, 66
- ILP Experimental Results, 83
- ILP Results, 84
- Inductive Logic Programming, 30
- Information Retrieval, 9
- Information Retrieval and Text Mining, 9
- Introduction, 1
- Journal Impact Factor, 102
- Journal Similarity Factor, 104
- K-Nearest Neighbor, 26
- Key Contributions, 6
- Local DataBase, 56
- Naive Bayes Classifier, 25
- Number of citations, 99
- Number of MeSH terms, 97
- Original sequences characterization, 75
- Performance Evaluation of an Information Retrieval System, 16
- Pre-Processing, 19

Pre-Processing MEDLINE data, 59
Probabilistic Model, 15
Propositional Learners Results, 83

Ranking MEDLINE, 93
Ranking Methodologies, 47
Research Questions, 5
Results from Alternative 1, 82
Rocchio Algorithm, 27

Structure of the thesis, 7
Support Vector Machine, 24

Text Mining, 17
The global procedure for the Ranking Function, 93
The MEDLINE, 55
The Ranking Function, 96
Thesis Objectives, 5
Thesis Overview, 113
Tools for Bioinformatics, 37
Tools for Information Retrieval and Text Mining, 33

Vector Space Model, 13