New Frontiers For An Artificial Immune System

Julie Greensmith
Digital Media Systems Laboratory
HP Laboratories Bristol
HPL-2003-204
October 7th , 2003*

artificial immune system, document classification, feature vectors, AIRS

AIRS, a resource limited artificial immune classifier system, has performed well on various classification tasks, including data clustering. This thesis proposes the use of this system for the complex task of multi-class document classification. Initially the AIRS system is validated using a standard machine learning dataset, which has not been used previously with this classifier. The use of AIRS for the purpose of document classification was then examined. This includes the pre-processing of HTML documents and the extraction, selection and representation of features, for the purpose of feature vector compilation. AIRS was used to classify various Internet documents, using a variety of datasets. Comparisons were made where the amount of documents, amount of classes and amount of features were varied independently. Additionally, AIRS was compared with another text classification package as a benchmarking exercise. On completion of this we are confident that AIRS is a suitable candidate for increasingly more complex tasks such as hierarchical document classification and multiple taxonomic mappings.

* Internal Accession Date Only
Approved for External Publication

New Frontiers For An Artificial Immune System

Julie Greensmith
School of Computing, University of Leeds
Hewlett-Packard Labs, Filton Road, Bristol, UK
MSc Multidisciplinary Informatics
MSc Thesis

September 5, 2003

New Frontiers For An Artificial Immune System

Julie Greensmith

MSc Multidisciplinary Informatics
(2002/2003)

The candidate confirms that the work submitted is their own and the appropriate credit has been given where reference has been made to the work of others.

I understand that failure to attribute material which is obtained from another source may be considered as plagiarism.

(Signature of student)_____ Julie Greensmith

Abstract

AIRS, a resource limited artificial immune classifier system, has performed well on various classification tasks, including data clustering. This thesis proposes the use of this system for the complex task of multi-class document classification. Initially the AIRS system is validated using a standard machine learning dataset, which has not been used previously with this classifier. The use of AIRS for the purpose of document classification was then examined. This includes the pre-processing of HTML documents and the extraction, selection and representation of features, for the purpose of feature vector compilation. AIRS was used to classify various Internet documents, using a variety of datasets. Comparisons were made where the amount of documents, amount of classes and amount of features were varied independently. Additionally, AIRS was compared with another text classification package as a benchmarking exercise. On completion of this we are confident that AIRS is a suitable candidate for increasingly more complex tasks such as hierarchical document classification and multiple taxonomic mappings.

Acknowledgements

In some ways, this is the most difficult section to write. There are so many people who have provided me with their support for the duration of what has been an awesome 6 months, and it is not easy to find the right words to express my gratitude. However, ‘thanks’ go to my family and pseudo-family for their unquestionable support. I would also like to thank Liz, Jean, and Justin for having the patience to answer all my annoying bash and C++ questions and Jamie for all the advice and inspiration during the early stages of this project. Thanks also go to Marco and the rest of the HP Labs-Bristol students, without whom life would be just one giant classification problem. On a more formal note, I would like to thank Dr’s Dave Cliff, Matt Williamson and Jason Noble for giving me this wonderful opportunity in the first place. Special thanks go to Dr Steve Cayzer who has officially been the most fantastic supervisor, through always being there to answer my incessant questions, despite his obsession with semantic blogging. Last but by no means least, I would like to thank Gillan for his relentless love, support, and encouragement, which will forever mean the world to me.

Contents

1 Introduction **1**

2 Literature Review and Contextual Information **4**

2.1 Machine Learning And Document Classification 4

2.1.1 Information Extraction 6

2.2 Immune Systems, *In Vivo* And *In Silico* 7

2.2.1 The Human Immune System 7

2.2.2 Rhapsody On A Theme Of An Immune System 11

2.2.3 AIRS: A Resource Limited Artificial Immune Inspired Classification System 13

3 Implementation **16**

3.1 Validation Of AIRS Through On A Simple Dataset 16

3.2 Information Preparation 18

3.2.1 Feature Extraction — HTML Reduction 18

3.2.2 Feature Selection —- Information Gain 19

3.2.3 Feature Representation — Feature Vectors For The Classifier . 21

3.2.4 The Datasets 21

3.2.5 A Worthy Competitor: Comparing AIRS With A Naive Bayesian Classification System 22

4 Experimental Analysis - Validation Series **23**

4.1 Optimum Parameter Comparison 27

4.2 Summary 28

5 Experimental Analysis - Document Series **29**

5.1 Document Classification 29

6 Discussion **35**
6.1 Validation 35
6.2 Document Classification 37

7 Future Directions **40**

8 Conclusions **42**

A Scripts and Source Code **48**
A.1 Bash Scripts 48
A.2 Information Gain Calculation Program 52
A.3 Validation Programs 64

B AIRS Configuration File **69**

Chapter 1

Introduction

> *"In the beginning the universe was created. This has made a lot of people very angry and been widely regarded as a bad move."*- Douglas Adams, (1952-2001)

The recent explosion of information that is readily available has brought with it great accessibility and opportunity for the acquisition of knowledge and the advancement of our understanding of a multitude of subject matters. However, we are drowning in our own deep pool of data resources. Perhaps the most notable example of this is the sheer number of pages viewable on the world wide wide web. The latest figures according to *googlefacts* [11] indicate that the google search engine contains over 3 billion catalogued pages. While there in more information available now more that ever before it has become increasingly difficult to navigate in order to find useful information on any desired subject.

From this has emerged the concept of the Semantic web, which is hoped to be an "extension of the current web in which information is given well defined meaning, better enabling computers and people to work in cooperation"[5]. Most pertinently, a current development in semantic web technology uses semantic annotations to provide some form of hierarchical document classification structure. This process involves user annotations to both newly created pages and existing ones. On-line communities provide a collaborative environment in which the user community decide on the classification of documents as a collective. This is based on the specific knowledge of the individuals involved, but again relies on the performance of human annotation. Many web publishers do not have the time, patience or the motivation to perform such a task, which

is viewed as a chore. In order to illustrate this point, the only item in an HTML document that is compulsory to comply with W3C standards is an annotated title tag[23]. Many documents available on the Internet do not even contain this simple tag, (2% of a small sample of cached HTML pages contained this annotation), let alone enriched content tags.

As mentioned, one solution to this problem is to tailor the presentation of Internet documents to the user in a taxonomic structure. This type of infrastructure can be applied to web documents in a post hoc manner, independent of the page author into a standardised format. Such a community, DMOZ [22] (Directory Mozilla), aims to do this. This system relies its users placing the document in to the appropriate classification. This process, while providing an effective and navigable system is labour intensive, and relies on the patience of the users of the system. It is evident that automated systems to perform this task would be a desirable tool, and can be based upon various machine learning principles. Standardised approaches to such classification tasks can include methods such as decision tree based learning, Bayesian classification systems and rule induction based paradigms, all of which rely on pre-defined inductive learning[21].

Biologically-inspired algorithms are becoming more prominent in the field of machine learning and data mining due to the ability of many of these strategies to adapt to unfamiliar data or scenarios. Such solutions have been inspired by a range of biological metaphors; evolution has inspired the development of genetic algorithms and interactions between neurones has inspired artificial neural networks. Another emerging biological metaphor is that of *artificial immune systems*. The concept of providing a computer with an immune system has obvious applications within the field of computer security and network intrusion detection. But, due to the immune system's ability to discriminate between different biological molecules (known as self-nonself discrimination), it also acts as a natural classifier system. Applying some of the principles involved in a functioning immune system to a range of classification tasks is a seemingly logical step.

A number of artificial immune systems have been developed for classification tasks that are unrelated to network intrusion detection. This includes systems developed by Twycross & Cayzer[35] for concept learning, and by Watkins & Boggess[37] for

multi-class data classification. It is a possibility that this system, known as AIRS, has the capacity to perform document classification, possibly in a hierarchical manner, based on the flexibility of the the input method and previous exemplary performance in other data processing tasks.

In alleviating the problem of unstructured information overload, the aim of this project is to extend the use of an artificial immune system to perform document classification. This is to be initially performed on a simple dataset as a validation exercise. Following successful validation, information extraction will be performed on a series of Internet documents, which will then be used to present to the classifier, outputting the classification of the documents.

The objectives of this project are to compare the performance of the AIRS system on a validation set and a document set, with the performance of other tried and tested methods; and to introduce a multi-class artificial immune system classifier to a novel application area, namely document classification.

This thesis is constructed as follows: *Chapter 2* discusses some of the background information that was necessary in order to perform the work. This information is based on concepts derived from machine learning and classifier systems, human immunology and artificial immune systems. The methods and other implementation details are outlined in *Chapter 3* including the validation procedure for the AIRS system. *Chapters 4 and 5* report the experimental findings of the work based on the validation of the system and its use as a document classification system respectively. *Chapter 6* provides a discussion of the results based on the outcome of the classification tasks, and provides a critique of both the experimental practise and of the AIRS system itself. *Chapter 7* provides a brief description of future work that could be performed based on the findings outlined in the previous chapters. *Chapter 8* contains concluding remarks and a summary of the whole thesis. The attached appendices contains the source code for the scripts that were ultimately used, along with some other programs that were implemented for initial testing purposes, plus the code for the classification systems utilised.

Chapter 2

Literature Review and Contextual Information

> *"The realm of the born - all that is nature- and the realm of the made - all that is humanly constructed- are becoming one."* Kevin Kelly, (1952 -)

As this is an interdisciplinary project, there are several research areas that need to be explored to obtain the contextual information necessary to understand possible solutions to this problem. The basic concepts involved in machine learning, with aspects of information theory, will be explored and explained. Additionally, an exposition of existing techniques used in the field of document classification will be presented. The basic principles involved with immune function are summarised, with particular attention paid to the components used as metaphors within the particular artificial immune system (AIS) to be used. Finally, the specific details regarding the immuno-inspired classifier system, AIRS, will be expanded upon.

2.1 Machine Learning And Document Classification

Ever since the dawn of the information age, computer scientists have striven to provide ways of developing systems which can learn, independently of their human counterparts. This involves the self improvement of a system through experience and exposure to unfamiliar scenarios, deriving a level of artificial intelligence. Inductive, rule based system have been developed in addition to more recently introduced evolutionary systems for the purpose of classification. Within the realms of classification, a certain

learning model is used. Classification tasks form a subset of machine learning, and therefore are a specialisation of the techniques that apply to this field in general. There are a number of commonly used classification systems containing similar elements, but again are specialised further, based on their area of application. Fundamentally, classification systems which rely on induction are a multi stage process. They require the use of information in the form of a previously classified dataset, from which a certain proportion is extracted i.e.the *training set*. This is presented to the classifier system, providing initial hypotheses for the classifier, forming the basis of the classification. The remaining data is then used to test the system hence comprising the *test set*. The accuracy of the classifier is measured in terms of the predictive accuracy[1] of the classifier measured. It is important to note that the data comprising the training set does not appear within the test set, as this may provide an unfaithfully high level of classifier accuracy.

One specialised rule based classification system is decision tree learning as demonstrated in [21]. Indeed, a decision tree could easily be applied for document classification on a keyword basis, but practically, the dataset and tree labels would have to be defined in advance, by pre-calculated values.

Bayesian learning techniques, especially *naive Bayesian classification systems* are used for various classification tasks. This is a probabilistic method of classification, and works by calculating the probability of an item belonging to each class within the dataset, and the item is classified as belonging to the class for which this probability is highest. More specifically, if $a = a_1, a_2, \ldots, a_n$ is a series of n features comprising the data item, a_{ij} and $V = \{v_1, v_2, \ldots, v_m\}$ is a set of the classes, then the class $v_{NB} \in V$ which the item is classified as is calculated by the following:

$$v_{NB} = arg \max_{v_j \in V} P(v_j) \prod_{i=1}^{n} P(a_j|v_j), \forall v_j \in V \tag{2.1}$$

This technique is commonly used for various classification tasks and has been demonstrated for use on discrete feature sets by both [35] and [25]. It is naive as it assumes independence between the features comprising the data items. The utilisation of this type of classifier within document classification will provide an excellent comparison for the immune system in question. Naive Bayesian classification may also be used as

[1]The exact details of this are to appear in the implementation chapter, Chapter 3.

a comparison for the system developed by Twycross & Cayzer [35]. The results of this give us a benchmark against which we can compare the AIS classification system.

2.1.1 Information Extraction

In order to use these classification systems, information has to be extracted from the documents into a useful format. This is vitally important, as poor representation of the contents of a document could lead to unnecessarily poor classification accuracy. There are a number of methods that can be used to perform such a task. This will be implemented through abstractions of the documents known as *feature vectors*, which are representations of the information that can be utilised by the classifier system. The use of such feature vectors gives rise to *dimensionality reduction*, which is necessary to avoid representing every single feature of every single document as an independent feature. This is needed for the purpose of efficiency and to focus the classifier toward the more informative features. The transformation from the actual documents into the format of a feature vector is also a multi-stage procedure, consisting of feature *extraction, selection and representation*.

The *extraction* of the text from HTML documents is a relatively trivial task, involving a little text processing. This preprocessing stage involves the removal of items from the HTML documents such as tags and items of java-script. Additionally, stemming algorithms such as the porter stemming algorithm [21] can be used to reduce words of the same family to their root. For example, “excitement”, “excitation” and “excitable” would all be reduced to the single stem of “excit”. However, this method could cause some loss to the richness of the document content, along with loss of meaning in some cases and so should be used with caution.

The extraction process is simple in comparison with the feature *selection stage*. The selection of the items within the documents to construct meaningful features vectors representing the text requires a little more thought. An entropy based measure which can be used for this purpose is called *information gain*. Given a dataset information gain can be used to calculate the most informative words within that dataset, therefore providing a method of dimensionality reduction. The presence or absence of words within a document can then be represented in the form of a feature vector, for use within classification tasks. The mathematics behind using information gain is

presented in the implementation section, Chapter 3, of this thesis. As an alternative method to information gain, singular value decomposition [12] a matrix compression algorithm, and principal component analysis[24] which is a statistical measure, can be used.

Once the features have been selected, they need to be represented in a way that can be utilised by the classification systems. There are two main methods that can be explored within this context: *boolean* and *TFIDF* vectors. A boolean vector is a series of ones and zeros indicating the presence or absence of a feature. Alternatively, the TFIDF (term frequency-inverse document frequency) values can be calculated and represented as normalised vectors. As the name suggests, this method takes into account the amount of times a word appears within a document, and how frequently the word appears throughout the document set. This is demonstrated and described in the work of Sebastiani [28]. However, this method may not be the most effective for use on this occasion as there is considerable computational overhead, as the values have to be generated and recalculated every time the data set changes.

2.2 Immune Systems, *In Vivo* And *In Silico*

In this section, basic concepts from immunology will be introduced in addition to the basic principles involved in the design and implementation of artificial immune systems. This enables cross referencing between actual immune systems and their silicon counterparts. A detailed overview of the immune system can be found in Janeway *et al.* [16], but the essential components and concepts which have some relevance in the field of artificial immune systems are highlighted here.

2.2.1 The Human Immune System

The human immune system is understood to be a highly evolved, decentralised and robust system, capable of providing humans with a high degree of protection against various invading organisms (e.g. bacteria, viruses and parasites that are collectively known as pathogens). This is achieved through the orchestration of a symphony of components, some of which are deemed to play a more prominent role than others in providing a high level of protection. The immune system is involved in the recognition,

reaction and subsequent removal of the pathogen from the body. A rough subdivision exists between the components of the system, forming the *innate* and *adaptive* immune systems.

The innate immune system is responsible for triggering an *inflammatory response* on the recognition of a potentially harmful pathogen. This is achieved through the recognition of specific proteins that are present on pathogenic membranes. These proteins cause certain immune cells, namely macrophages, to engulf the pathogen, leading to the subsequent destruction of the pathogen. This event causes a change in the activation state of the macrophage including the secretion of messenger molecules called cytokines. The cascade reaction resulting from the release of the cytokines causes the dilation of local blood vessels enabling the greater concentration of pathogen destroying agents and fluids. This signal also increases the presence of other cells belonging to the other immune sub-system, *the adaptive immune system*.

The adaptive immune system is complex, with an emergent property being the responsive protection of the body from invading pathogens. Some of this complexity arises from the fact that there is no one entity which is primarily responsible for the successful function of the system. Therefore, an overview of the immune system will be presented by examining the individual components, and describing the interactions and dependencies that exist between them. The actions and interactions of the B-Cells will be examined in greater detail, due to their resource limited and adaptive nature which has given rise to many immuno-inspired algorithms.

The most notable components of the adaptive immune system are the white blood cells known as lymphocytes, which can be subdivided into 2 classes: *B-cell and T-cell*. T-cells are further sub-divided into 2 types, *cytotoxic T-cells and helper T-cells*. Molecular proteins known as *major histocompatability complex (MHC) type I and type II* play an integral role in foreign cell recognition. Also, the involvement of *antibodies* within this system is highly important. Now that all the components have been introduced, we shall examine their interactions, and the consequences involved in an immune reaction.

Receptors on the B-cells, which are similar to antibodies, are matched against a protein belonging to a pathogen known as an *antigen*. If there are sufficient B-cells with

an affinity[2] for this antigen, then a level known as the *stimulation threshold* is reached and the B-cell then ingests the pathogen, in a process known as phagocytosis. The pathogenic molecules are then re-expressed as part of the MHC type II molecules of the cell.

Helper T-cells develop in the thymus, where they are presented with MHC type II protein fragments. Here, they are subject to *positive selection* i.e. those T-cells who have a *binding affinity* for the protein, are not destroyed. Successful cells are then subject to the process of *negative selection*, in which they are matched against a range of established self proteins. Any T-cell which matches a self protein is then destroyed. The remaining matured cells have affinity for both MHC and no affinity for self proteins.

The mature T-helper cells act as a co-stimulation for the B-Cells: the binding of the T-helpers to antigen presenting molecules is needed to fully activate the B-cells. Following this activation, B-Cells migrate to the lymph nodes and divide (causing the swelling of glands during infections). The process that occurs in the lymph node is represented in figure 2.1 and is known as somatic hypermutation and clonal selection[8]. As a result of this process, antibodies with a high affinity are produced and attach themselves to the antigens on the surface of the pathogen, acting as a marker for the cell to be destroyed.

Another mechanism of pathogen disposal exists independently from B-cell mediated immunity, and is facilitated by another type of T-cell, namely cytotoxic T-cells. The T-killers are responsible for dealing with *intracellular pathogens*, ones that have invaded the host cells such as viruses. As the pathogen is residing inside the host cell, the antigen is not immediately accessible, therefore the recognition process occurs through the detection of expression of unknown protein fragments bound to a different type of MHC molecule, this time type I. On detection the killer T-cells can destroy the infected cell.

Once an immune response has been exposed to a particular antigen, long living B-cells known as *memory cells* retain a copy of the necessary antibody, therefore if the pathogen is encountered in the future a rapid response sequence can be initiated again

[2]This is the process in which two molecules interact to form a complex at a molecular level

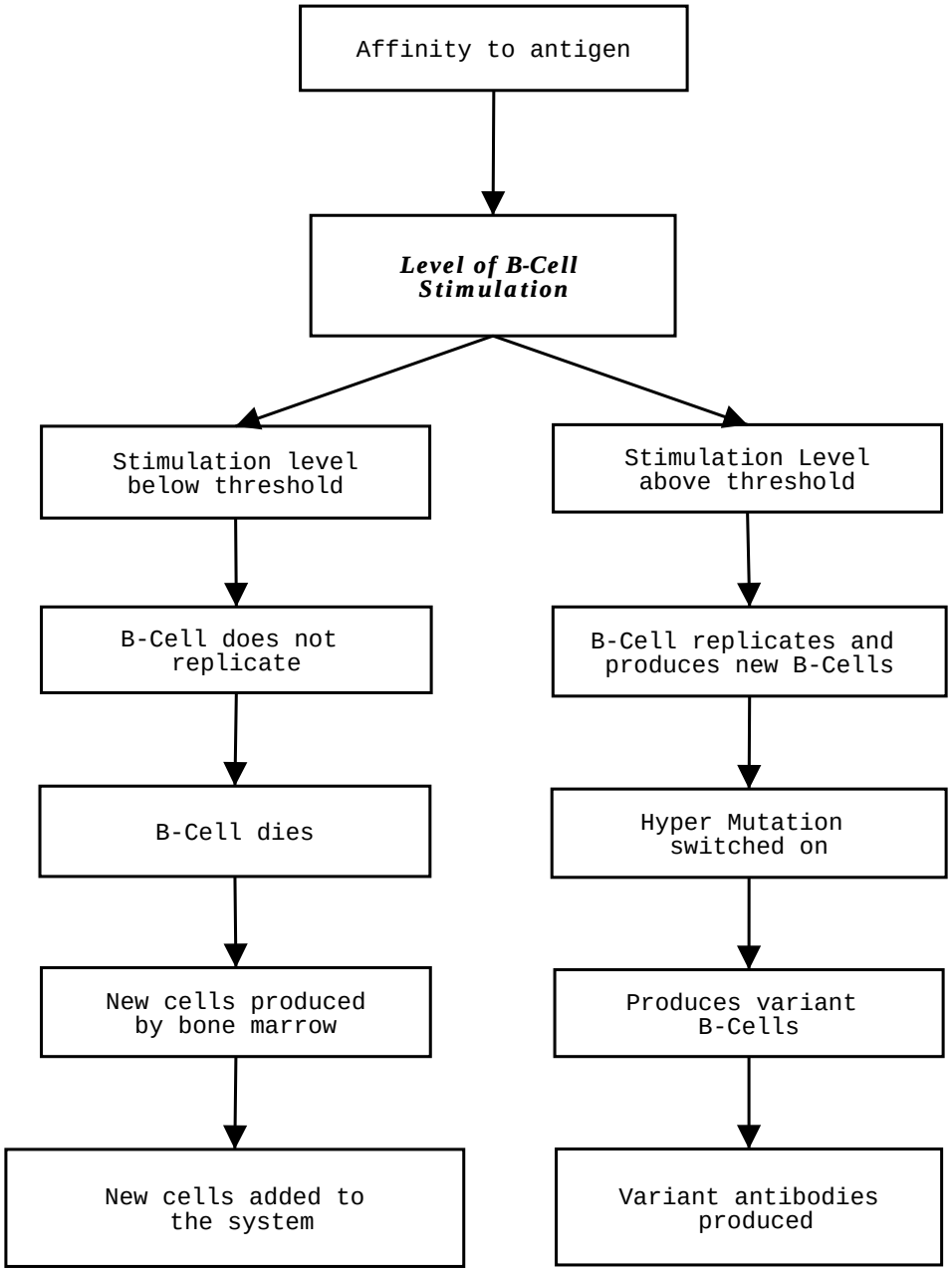

Figure 2.1: The clonal selection processes involved in the development of B-Cells

without having to perform any clonal selection or affinity maturation.

One Minute Muse: Danger Theory

While the self-nonself model described above holds true in a multitude of cases, there are circumstances which suggest that this can not be the only mechanism of pathogen treatment. For example, the definition of self changes over the lifetime of a person, during puberty and pregnancy. Additionally, no immune response occurs to the bacterial flora in the gut, yet autoimmune reactions occur to what is clearly self. One proposed theory to augment the self-nonself model is that another signal is required in order to initiate this immune response. This is known as *danger theory* and hypothesises that the additional signal is based on the mechanism of cell death. This implies that the recognition of a strange protein, needs to be accompanied with higher than normal levels of necrosis (unplanned cell death) in order to facilitate the immune response. Further details of this can be found in [19] and with respect to AIS in [1].

Summary Of The Human Immune System

The human immune system is a multi component, decentralised system which operates successfully through the orchestration of the distinctly different components. The immune system is adaptable and can therefore cope with the presentation of unfamiliar pathogens, therefore suggesting that it could be suitable for a multitude of other classification problems involving unfamiliar data. This forms the basis for the development of AIS and other immunologically inspired algorithms [8].

2.2.2 Rhapsody On A Theme Of An Immune System

The field of artificial immune systems has been growing over recent years. Biological inspiration can successfully be transferred into novel computational paradigms, as shown in the development and successful use of concepts such as artificial neural networks [2], genetic algorithms [7] and artificial ants [9]. The underlying principle of all of these systems is that inspiration has been taken from biology to provide more effective heuristics, but do not attempt to directly or precisely model the function of that actual system [20]. This also applies for the development of artificial immune systems, though it is always good to have an appreciation of the system from which the inspiration has been derived[8]. Successfully implemented AIS draw parallels with

certain aspects of the immune system but are not usually intended as exercises in mathematical or theoretical immunology.

While the field of AIS is relatively new, various attempts have been made to provide an answer to the question 'what qualifies as an immune system, and how can we generate guidelines for building them?'[8]. The selected components of the immune system used for the implementation of an AIS is context specific due to the fact that AIS have applications in a variety of areas. Nevertheless, there are several basic characteristics that frequently appear in AIS including successful representation of immune molecules, with the antibodies being agents of the system and antigens in the form of input data, affinity metrics for the virtual antigen, antibody pattern matching algorithms and meta dynamics i.e. how the system behaves as a whole[8]. The immune system is deemed to be a good model for the following reasons:

- The presence of a content addressable memory;
- It learns by experience;
- It is inherently distributed;
- It is inherently generalist[15]

Successful implementation of AIS have been developed for a multitude of applications. A few examples will be explored in some detail.

Example 1: Lisys

The most obvious and well explored area of the applications of artificial immune systems is within the domain of computer security, in particular the detection of unusual or unauthorised network activity. The system developed by [14] employs various immunological concepts including negative selection, T-cells, B-Cells, MHC molecules and co-stimulatory effects. However, this system is most famous for its implementation of the negative selection principle, as the system is trained on what is known as self or defined as normal network behaviour. This is in the form of bit strings derived from Internet Protocol (IP) data that are subject to the *r-contiguous* bits matching rule. The co-stimulation comes from the detection of anomalies through the matching of the strings and through the validation of the anomaly from an external human moderator. The MHC concept is employed to reduce the amount of false positives in the system.

It was suggested that such a system was not capable of scaling for use on very large networks [17], however, as the system has developed, this has become less of an issue [3]. Current modifications to the system include the removal of the human user as the co-stimulatory signal, and a richer representation of the network behaviour [4].

Example 2: Co-evolutionary Immune Based Concept Learner

A system used previously in the domain of document classification is the already mentioned work performed in 2002 by Twycross and Cayzer[35]. This was based on the accepted immunological concept of self-nonself discrimination, where a set of discriminatory detectors were created and evolved in a *co-evolutionary manner*, based upon previous work performed by Potter and De Jong[27]. Non-interbreeding species were evolved and combined to form a concentration, which could cover the feature space. Unlike other approaches, this system represented the concepts as the class representations. This system performed well on 2 class problems, on both a test data set and on HTML datasets, and produced results of a competitive standard to naive Bayesian systems and other standard classification methods.

Example 3: AIRS

The AIRS system employs a resource limited B-Cell type metaphor, and is used in the context of multi-category classification. More details about the AIRS algorithm are given in the next section due to the fact that this is the AIS that has been chosen to perform the classification task. In brief, this system uses the maturation, function and hypermutation of B-cells to produce antibodies and memory cells in a one shot training algorithm. This system has been proven to perform well in data mining tasks and other non-linear classification tasks.

2.2.3 AIRS: A Resource Limited Artificial Immune Inspired Classification System

The AIRS systems, developed by Watkins and Timmis, is a successful implementation of an immune algorithm. The algorithm used within the system is a multistage process, consisting of an initial seeding of memory cells within the system, a training period (consisting of adaptive components) and finally a classification stage on test data.

The AIRS system is a particular type of artificial immune system which is inspired by the action of B-cells, in a resource limited manner. This biologically inspired system has already demonstrated good performance on various other classification tasks, in relation to linear and non-linear data clustering[36]. Early indications from such experimentation has suggested that there is more to this system that simple 2 class classification, and that the boundaries can be pushed even further with respect to its applicability. The question is whether AIRS successfully perform the task of multi-class classification in a domain in which it has not been previously used. To enable us to use the system, a basic understanding of how the algorithm functions and how this is applicable to the immune system and within the context of document classification is required.

The One Shot AIRS training Algorithm

1. **Memory Cell Identification And Cloning -** training antigens are presented to the system, and a user defined portion are used in order to create an initial batch of memory cells. A single training feature vector (antigen *ag*) is matched with its closest matching memory cell. This cell is cloned at a certain level, depending upon the strength of the affinity (inverse of euclidean distance), forming a collective known as an *artificial recognition ball* or ARB. These newly created ARBs representing a concentration of cloned cells are added to the ARB pool of other ARBs of the same class.

2. **ARB Affinity Maturation -** an antigen is presented to the ARB pool and are allocated resources depending upon the affinity of the antigen and the the class of the antigen. An ARB with high affinity and the same class as the antigen would be rewarded with a greater proportion than an ARB with lower affinity. Clonal expansion is then allowed to proceed until the average stimulation level of the ARBs is above a user defined stimulation threshold. Once all of the ARB matching has been performed, it is usually the case that the user defined limit for available resources has been exceeded. In this case, the resources in overdraft are removed from each of the ARBs starting with the ARBs of the lowest affinity, until the limit is no longer exceeded. ARBs which have performed particularly badly will be left without any resources, and are removed from the pool. Resources are allocated in a class by class basis.

3. **Selecting the Candidate Memory Cell -** the best ARBs from the same class as the antigen are compared and if this ARB has a higher affinity than the best matching memory cell it is added to become a mature memory cell. If the similarity between the old memory cell and this new candidate cell is sufficiently high, then the old memory cell is replaced in favour of the more accurate memory cell. The process of ARB affinity maturation is repeated for all items of the training data.

Once the system has been trained in this manner, the items of test data are matched with the closest matching memory cell, using a k-nearest neighbour approach[12]. A discussion of how ties are dealt with can be found in [18], based on a selection based on the class proportions of the training items competing for the test antigen. Therefore in the case of a tie, selection will be based on the class with the highest relative antigen classification frequency during the training period.

Now the algorithm has been explained, the time has come to apply the system to the areas outlined in the objectives, namely a validation classification task, and multi class document classification.

Chapter 3

Implementation

> *"C makes it easy to shoot yourself in the foot; C++ makes it harder, but when you do, it blows away your whole leg."*-Bjarne Stroustrup (1950 -)

3.1 Validation Of AIRS Through On A Simple Dataset

The initial part of the experimentation performed was the validation of the AIRS classifier system. This involved the testing of the system on a relatively small, non-linearly separable dataset, in which the categories of the data items cannot be separated by a single hyperplane cutting through the feature space. Linearly separable problems are trivial for machine learning tasks, so the use of a non-linearly separable dataset is imperative. The rationale for performing a task with no real link to document classification was to ensure that the classifier system could perform as well as expected on a dataset explored by [35], and to augment our understanding of the system and how to use it in this context. Additionally, AIRS contains a number of user definable parameters, which can be adjusted to suit the problem. An example of the configuration file used is provided in Appendix B.

The dataset used was taken from the UCI repository[6], which contained voting records from the 1984 US congress. The information of voting details for 16 votes taken within the house were used, as was the information regarding the political affiliation of the senator i.e. use the voting trends to predict if a senator was a democrat or a republican. The representation for this feature vector that had to be presented to the system was in a simple format :

vote	representation
yes	0
no	1
abstain	2

The configuration file used in order to run the system contains a set of user definable parameters, of which 3 were examined. The first parameter investigated was the amount of memory cells used to initially seed the program. This was thought to be of importance as the creation of the initial memory cells within the feature space should lead to a higher predictive accuracy, given a reasonable cover. The stimulation threshold for the clonal selection procedure was also varied. This was deemed important for the training algorithm as the amount of stimulation needed to reproduce will ultimately affect the number of memory cells produced, thus may affect the classification of test items. Finally, the amount of resources that can be allocated was varied, as removal of resources will also have an effect on the clonal selection algorithm.

The predictive accuracy of the classifier was calculated using a 10-fold cross validation procedure. This involved sub-diving the dataset into 10 portions, and using 9 of those portions as the training data, and the remaining one as the test data. The portion used for the test data are rotated until each one has been used. This is performed in the same manner, as in [35] so a direct comparison could be drawn. The results were summarised in a series of tables, box-plots and density plots. Density plots are similar to histograms with large large numbers of bins, which produces a smooth graph of the frequency at each value of predictive accuracy. The output generated contained the predictive accuracy for the training items and the test items with k-nearest neighbour (as described in Chapter 2) values of 1 and 7. The mean value was used to derive optimum settings.

The classification of this particular dataset was previously performed using an AIS and a naive Bayes system, both developed by [35].The results from these previous experiments can be used as a marker regarding the performance of the classifier. For the purposes of the comparison, the configuration for the AIRS system was determined through using the highest mean values obtained in the initial experiment series for the stimulation threshold, number of resources and number of initial training item memory

cells (as presented in Chapter 4).

3.2 Information Preparation

In order to utilise the classifier system specifically for document classification, several preprocessing tasks must be initially be performed. The HTML documents in their raw format are not suitable to give to the classifier for a number of reasons. Firstly, the classifier needs a representation of 'features' to be able to classify a document, which cannot be represented as alphabetical characters. It would be possible to use all of the words contained in all of the documents as features (the presence and absence of each word contained within the dataset) however, it is sensible to perform some process of feature selection to achieve a form of dimensionality reduction, but to a rich enough level so accuracy of classification will not be lost. Finally, the raw HTML files to be classified will include many words which are present for the purpose of HTML and/or java-script / other markup tags, but which do not augment the meaningful content of the document, and therefore must be removed within the component of preprocessing that has to be performed. The transformation from HTML document to boolean feature vector will be explained in detail.

3.2.1 Feature Extraction — HTML Reduction

The files to be used for the testing of the classifier system are taken from a reclassified dataset from [10]. Such documents contain a multitude of tags and small scripts within, all of which is superfluous to the actual meaningful content of the document. The removal of the HTML was performed using a package freely available for Linux known as html2text[34]. This package outputs an ASCII file, containing only the actual words displayed on screen, therefore eliminating the additional tags contained in the document. For the purpose of this exercise, we are only interested in the words contained within the document, so a Perl script was written to extract only the alphabetical characters (excluding numbers and other special characters). Additionally, this script was used to process the file so that every word only appeared once in the dataset, and that each individual word was placed on a separate line, for ease of manipulation for future processing. In preliminary prototype systems, a porter stemming algorithm [26] was also implemented, but this did not appear to enhance the classification accuracy.

3.2.2 Feature Selection —- Information Gain

As stated previously, a good level of classifier performance would not be expected if all of the words within the dataset were used to form the feature vectors. We need to select only useful or informative words. A method to achieve this uses a concept derived from information theory known as **information gain**. This concept is an entropy based method, which is commonly used to distinguish between separate classes of items, and is also used for decision tree methods of learning. The underlying concept of information gain relies on probabilities of a word appearing in a specific class of document. For example, imagine you had three very simple documents each containing 3 words:

Document A	Document B	Document C
cat	dog	keyboard
mouse	mouse	mouse
horse	horse	monitor
Class 1	Class 1	Class 2

The word horse is present in documents A and B, and is not present in document C, and is therefore specific to class 1 and so has a high value of information gain. Conversely the word mouse is present in all documents and so has a very low value of information gain, as no information regarding the class of document can be gained from using this word as a method of distinguishing between the classes. The words 'cat' and 'dog' would have intermediate values. Using the information gained from the appearance and absence of words from different classes, a list of the most informative words can be derived. The exact formula for information gain is represented in the equation below:

$$E(w, S) = I(S) - [P_{(w=pres)} * I(S_{(w=pres)}) + P_{(w=abs)} * I(S_{(w=abs)})] \qquad (3.1)$$

where:

$$I(S) = \sum_{C \in \{C_1,...,C_n\}} -P(S_C) log_2[P(S_C)] \qquad (3.2)$$

Equation 3.1 calculates a value for the expected information gain, E(w, S) of a word(w) belonging to the document set(S). To calculate this we need the probability of a word being present($P_{(w=pres)}$) and absent ($P_{(w=abs)}$), both within the document set as a whole

and in a specific class. The first term in Equation 3.1 takes into account any uneven distribution of the items within classes i.e. class/document distribution skew, and is a measure of how much information can be provided on knowing which class a document would belong to. The second term in Equation 3.1 is a measure of the entropy in sets of documents with and without that word. A low value means that the word has high discriminatory power. The entropy measure in Equation 3.2 measures the spread between classes and is highest for an even distribution of documents.

While the equations and mathematical relationships may appear complicated, they simply describe that the more discriminatory words have a higher value for expected information gain. Words with a high information gain are considered to be more *informative*. The presence or absence of the informative words, (of which we choose the best k) from documents forms the basis of the construction of the feature vectors used in the document representation for the classifier system.

Implementing Information Gain

Following the initial preprocessing of the documents, shell scripts (provided in Appendix A) are used to calculate the statistics regarding the following items of data necessary for calculating the most informative words contained within the dataset.

- total number of documents within each dataset
- number of documents within each class
- number of classes within the dataset
- the presence of each word once in a document, and which class of document in which this word can be found
- the absence of a word, and the classes of the document from which this word is absent
- a list of all the words contained in the entire dataset.

Using this information, a program was written (again, Appendix A), and the k-informative words can be used in the creation of feature vectors. It is also important to note that the probabilities needed could in theory involve a division by zero. In order to overcome this a Laplacian correction was implemented to the relevant parts of code.

3.2.3 Feature Representation — Feature Vectors For The Classifier

Once the information gain of the dataset has been calculated, it is then possible to take the words with the highest information gain, and to represent them as features for the classifier system. Obviously, the number of informative words, k, will be dependent upon the size of the dictionary created in the information gain stage, but on the datasets that we are using, the number of words was varied between 10 and 200 in order to see which will give the most accurate classification of the documents. Once the k most informative words have been outputted, each word within the HTML documents is checked against the list of the k words. The presence of a k-informative word in a document is marked with a 1 and the absence of that word is marked with a 0. This gives rise to the creation of the feature vectors for use within the classifier. For example, our most informative words from the tiny dataset above were horse and monitor. So, the feature vector for the document D containing the words "I rode a horse today" would look like "1 0", thus denoting the presence of the word horse within the document, and the absence of the word monitor. Obviously the datasets used to test the classifier are a great deal larger in size, but the same principle applies.

Once the feature vectors have been created from the k-informative words, and have been labelled, the classifier is all set and ready to go. 5 times 10 fold cross validation was performed in a similar manner to the voting dataset and the results collated. Standard statistical techniques including Wilcoxon Mann-Whitney ranking [33] and Students t-test [30]are used in order to analyse the results. The Wilcoxon ranking was used in the majority of cases as the data was not deemed to be normally distributed and the full datasets were present.

3.2.4 The Datasets

The datasets provided by [10] include classified documents in a similar format, of various different business pages available from 'Yahoo Taxonomies'[32] and from 'The Standard'[31]. There are several different datasets available, but the ones chosen for the purposes of this experimentation contain 2 levels of hierarchy, and approximately 500 documents. Smaller versions of these datasets have been prepared for the purpose of testing the classifier on different size of problems, but using the same standard of

classification. Of course, as with many supervised learning paradigms such as this one, the classification of the training set must be performed. If in this case AIRS performs well, we could hope for a situation in which human intervention would not be needed for the classification of Internet data, saving both time and effort on the behalf of many people. However, this classification originally was performed manually and hence involves some human interaction with the information. As humans are to use the pages, such subjectivity is necessary and even desirable. However, different people may find different subjective classifications more appropriate. As a future application, the use of a personalised structure, which can be user defined would be desirable, as outlined in Greensmith & Cayzer[13].

3.2.5 A Worthy Competitor: Comparing AIRS With A Naive Bayesian Classification System

Once results of the classification of the documents in the various different ways has been described and collated the effectiveness of the classifier will hopefully become apparent. However in accordance with good scientific methodology and practise, another standardised method of classification must be used to provide some sort of benchmark to compare the results with. The classifier system that we have chosen to use in this situation is the naive Bayes classifier, due to the fact that this method can often gain high predictive accuracy scores, and has already been used to perform a multitude of classification tasks [35]. For the purpose of this experimentation a comparison will be performed on a medium sized dataset, using a system called Rainbow, which includes naive Bayes classification and *TFIDF* feature vector representation. This system has also performed well on various text classification tasks. A more detailed description of how this system works, and how to work this system is given at the corresponding web address[29]. Details of the configurations used for this system are given in Chapter 5 of this thesis.

Chapter 4

Experimental Analysis - Validation Series

> *"Each problem that I solved became a rule which served afterwards to solve other problems."* - Rene Descartes (1596-1650)

The purpose of this section was to obtain an understanding of the AIRS system, with a view to familiarisation of the mechanisms in which the system works and in order to compare its performance on a standardised classification task.

To use the AIRS system for the purpose of document classification, we initially had to ensure that we could use the system, as it was initially received in the form of uncompiled C++ source code. The code for the classifier was compiled using the g++ compiler version 2.96 for Red Hat Linux 7.3 2.96-113, and was run on one out of 4 of Intel Pentium^{TM} 4 CPU 1.80GHz HP 'e-PC's'. On completion of the compilation process, the iris dataset (provided with the source code) was used to perform preliminary testing on the system. Once it was clear on how to use the various parameter settings, and that classification could be performed, then the system was tested on the voting dataset. These data for the parameter testing are presented as box-plots, with the main square of each plot representing the low and high interquartile range, the horizontal bar down the centre representing the mean value, and the dotted outlying line representing the upper and lower limits of the predictive accuracies of the data. All statistical tests (unless otherwise stated) are Wilcoxon Mann-Whitney ranking tests, performed through the Internet based application available at [33].

This dataset, obtained from [6] was in the form of a 16D vector. Completion of this tested the AIRS system in a novel area. The configuration file for the AIRS system contains many parameters which can be easily altered. Once it was established that classification could be performed on this dataset, then the three investigated parameters, number of resources, stimulation threshold and number of initially created memory cells, all contained within the configuration file, were varied. The amount of resources allowed in the system was varied in the range of 0 - 200, with 200 being the default setting, and the results of this are displayed in figure 4.1. Wilcoxon testing was performed on the data series, and the value for 100 resources compared to all the other values measured. This value was chosen as it had the greatest mean value, as demonstrated in Figure 4.1. The results of the statistical tests are presented in the following table. However, the significance level is a probability of $p < 0.05$, of which none of the results in this experimental series qualify. Thus while on first glance there appears to be an optimum setting (at 100) there is no statistical evidence to support this case.

Number of Resources A	Number of Resources B	Wilcoxon p-value
1	100	0.578
10	100	0.793
200	100	0.882

The second parameter experimented with was the value for the stimulation threshold, which was varied within the range of 0.10 to 0.95, and the corresponding box-plot represented in Figure 4.2. The default setting for this parameter was 0.95, which was deemed by the author to be the optimum setting. A Wilcoxon rank testing was performed comparing the default setting of 0.95 with all of the other tested settings. The results for these tests are shown in the following table. As with the results for the amount of resources in the system, none of these results were statistically significantly different as all had p values of over 0.05, even though there appears to be some increase in the mean as the stimulation threshold increases. It is important to note the the total range of possible values for this parameter was between 0.00 and 1.00

Stimulation Thresh. A	Stimulation Thresh. B	Wilcoxon p-value
0.75	0.95	0.231
0.50	0.95	0.190
0.25	0.95	0.386
0.10	0.95	0.090

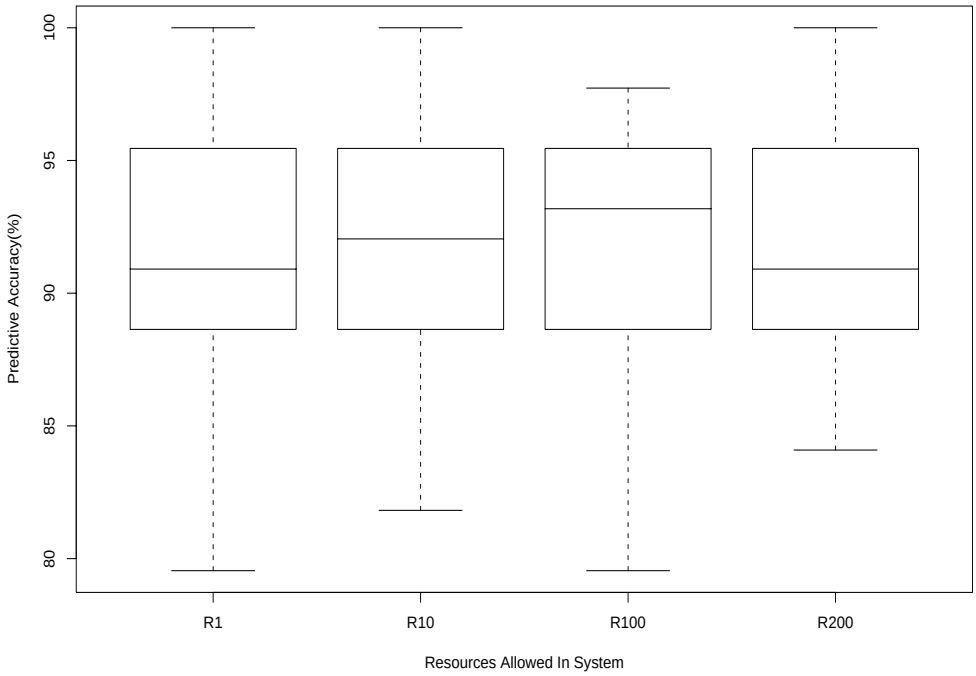

Figure 4.1: Box plot of different numbers of maximum resources allowed in the system

The final parameter to be investigated was the number of initial memory cells (items of training data) that the system created before the main section of the training procedure commenced. As there were 435 data items in the UCI data set, the initial number of memory cells was set between 0 and 400. The representation of this is shown in Figure 4.3.

In reference to Figure 4.3, it appears that there is an increase in classifier performance in between the values of 0 and 200, following by a regression in values between 200 and 400. As for the previous parameters, Wilcoxon ranking was performed and the results of this can be found in the following table.

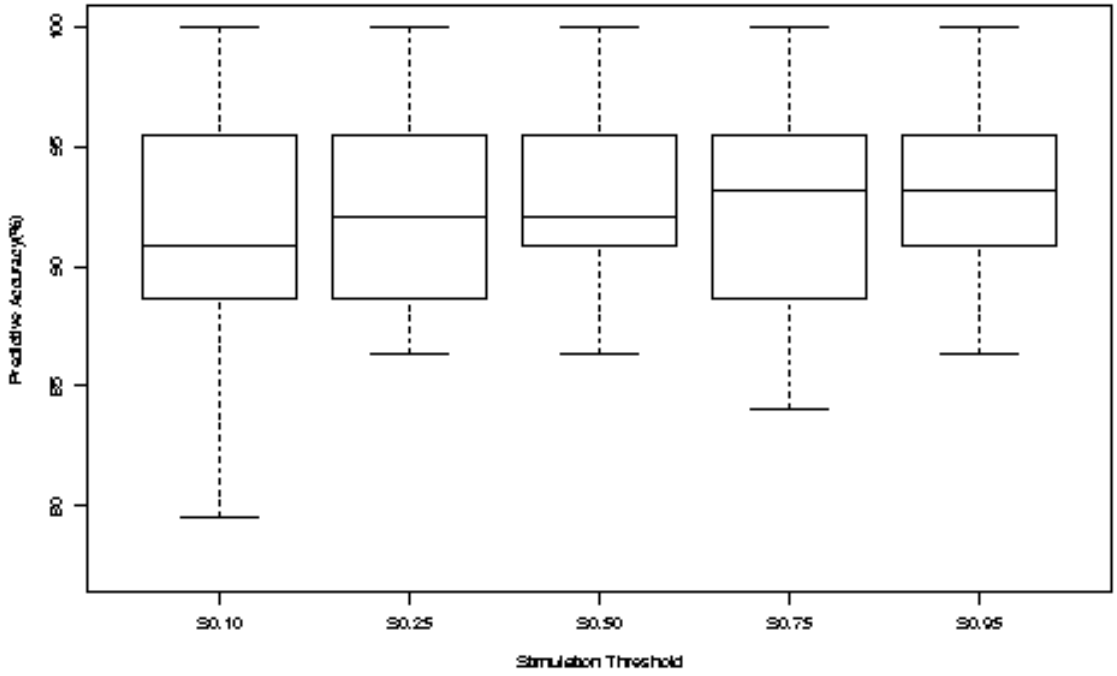

Figure 4.2: Box plot of increasing stimulation thresholds

Number of Memory Cells A	Number of Memory Cells B	Wilcoxon p-value
0	200	**0.0183**
1	200	0.680
10	200	0.217
100	200	0.187
300	200	0.181
400	200	0.217

As highlighted in bold in the relevant table, the Wilcoxon probability value for no initial seeding versus 200 items(selected from 435 training data items) is statistically different. This implies that at least one item of training data is needed in order to achieve a greater level of predicted accuracy. The trend in the means represented in Figure 4.3 also suggests that there could be some over-fitting of data past a certain level of number of seeding memory cells. In order to add clarity to this point, the series of 0 memory cells, 200 memory cells and 400 memory cells are represented in Figure 4.4, where the notion of over-fitting could be suggested if the majority of the

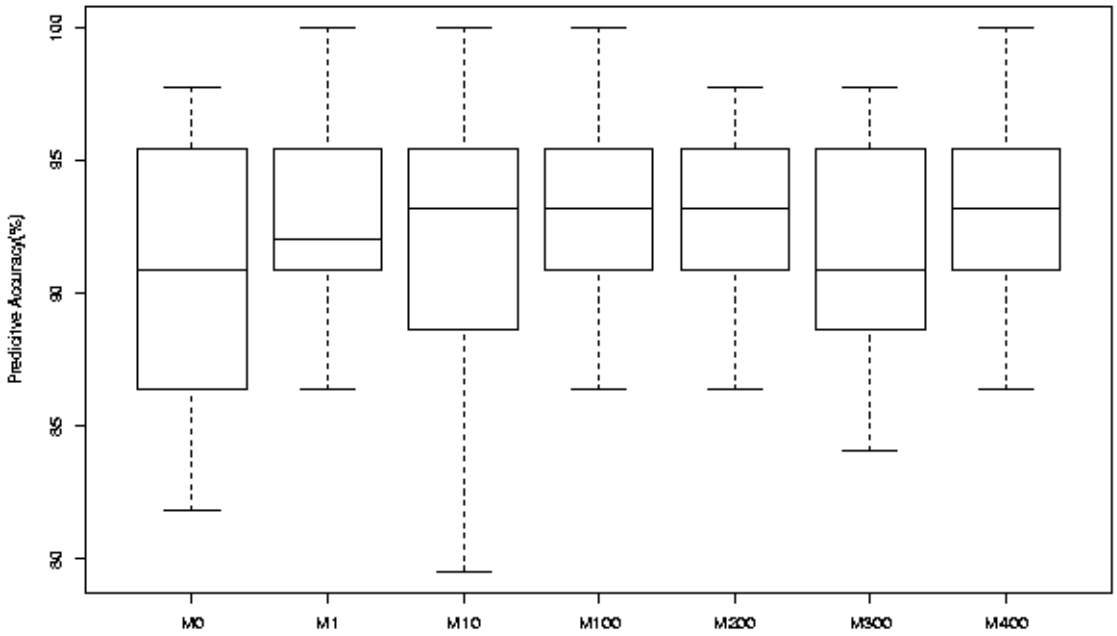

Figure 4.3: Box plot of increasing initially created memory cells

training data was used for seeding purposes.

4.1 Optimum Parameter Comparison

The results presented in Twycross & Cayzer [35] were generated using the same cross-validation method, and the same data set. Therefore the optimum parameter AIRS results can be compared directly, with the selected parameters including a resources level of 200, stimulation threshold of 0.95 and initial seeding value of 200. The performed systems comparison is given in the following table.

Classifier	Mean	S.D.
AIRS	92.86	3.46
Co-Ev[35]	97.40	2.60
Naive Bayes[35]	90.10	4.90

Wilcoxon ranking could not be performed as I did not have access to the complete dataset for the experiments performed by Twycross & Cayzer[35], therefore a nor-

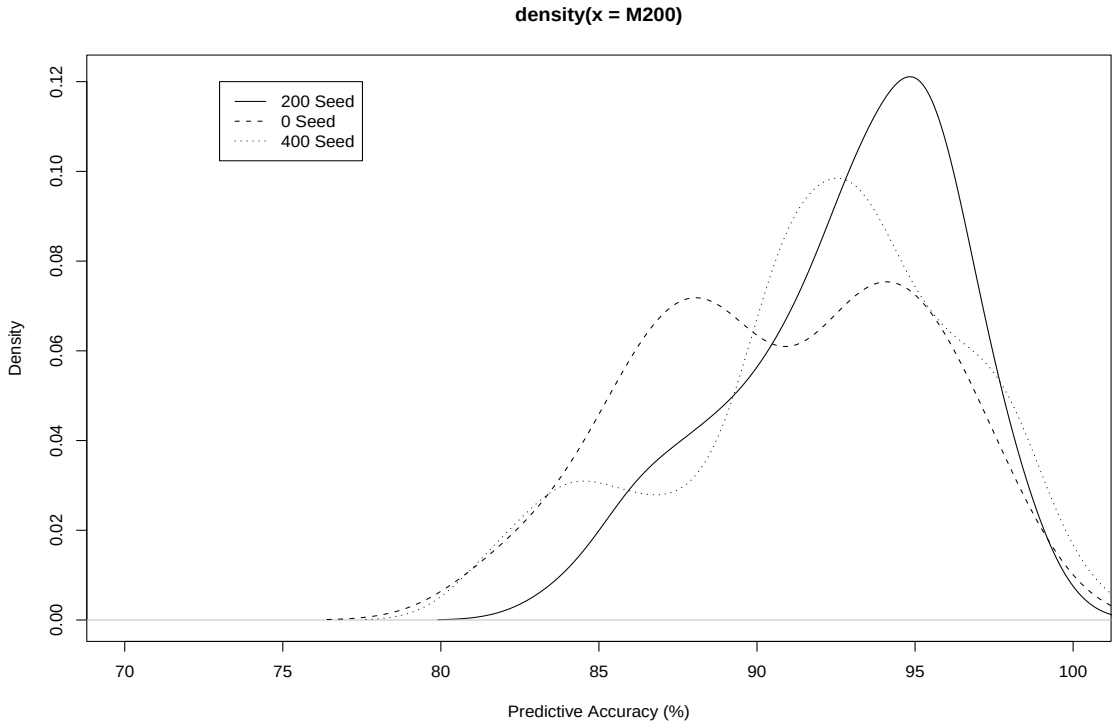

Figure 4.4: Box plot of increasing initially created memory cells

mal distribution had to be assumed. The significance testing was thus performed using a students T-test. The AIRS classifier performs statistically worse that the co-evolutionary AIS (p = 0.0001), but significantly better than the naive Bayes comparison (p= 0.0016). The differences in classification accuracy will be accounted for in the discussion chapter, Chapter 6.

4.2 Summary

The AIRS classification system has been tested on a novel dataset, namely a voting dataset comprising of 16 features and 435 items. The predictive classification accuracy has been measured, and the user definable parameters of initial memory cells created, the total number of resources allowed in the system and the stimulation threshold, have been varied.

Chapter 5

Experimental Analysis - Document Series

> *"In theory, there is no difference between theory and practise. But, in practise, there is."* - Jan L.A. van de Snepscheut (1953-1994)

Once it was clear beyond reasonable doubt that the AIRS classifier was functioning as expected, attention was turned to the successful derivation of feature selection from the document datasets. A collection of datasets were produced and a variety experiments performed on each. Information extraction from the datasets was performed as outlined in Chapter 3. The implementation of the pre-processing and information gain was validated by cross checking the numerical values with manually calculated values, and performance on a small dataset where the most informative words were immediately obvious. Once the testing conditions had been met, the process was repeated using the larger and more complex datasets.

5.1 Document Classification

Initially, we needed a comparison to gain some insight into the relative performance of the classification system, relative to other classifiers. This was achieved through using a small dataset, containing 2 classes and 69 documents. The configuration of the Rainbow system was as suggested as default through its on-line manual pages[29]. In this instance, the version of Rainbow used was bow-970112. The only parameter that was changed was the amount of testing items used in the system, which was changed

from 33 to 10 percent of the total dataset. In order to provide a fair comparison, the data presented to this system was also preprocessed through the html2text process to remove tags and other unwanted markup items.

The results of the 50-times cross validation of both systems are presented in Figure 5.1. The number of initial memory cells used for this was 8, with the stimulation threshold being 0.95 and the amount of allowed resources 200. The number of features used in the AIRS system for this was 100. The values for the mean value and standard deviation of both system are presented in the following table.

	AIRS	Rainbow
mean	85.71	43.23
SD	14.86	16.03

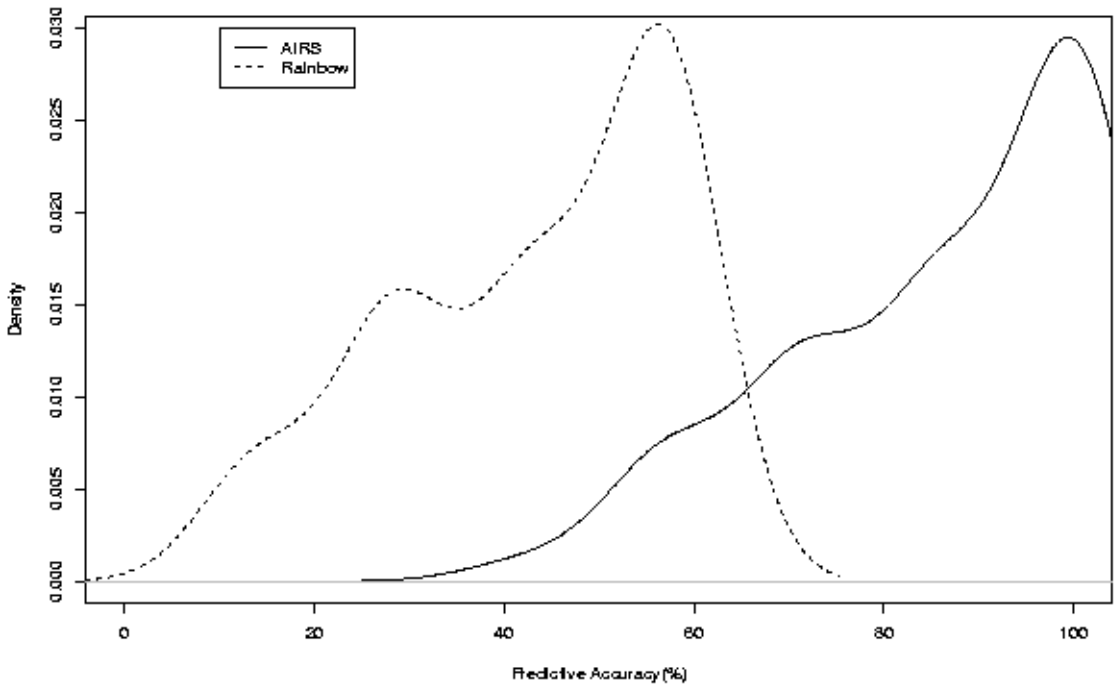

Figure 5.1: A direct comparison between AIRS and the Rainbow Text Classification System

A Wilcoxon ranking was performed on this data, which returned a p value of $p < 0.00001$, which was deemed to be statistically significant. This implies that the AIRS

achieved a statistically significantly higher classification rate than the Rainbow system (in naive Bayes mode). The results were indicative that the system can perform equally well on document classification, and further experimentation regarding different datasets could commence.

The next experimental series to be performed was to investigate the influence of the number of features on the predictive accuracy of the classifier. This small dataset consisted of 35 documents, to be classified into 2 classes. A density plot for the predictive accuracy on 5 times 10-fold cross-validation is shown in Figure 5.2, with the dimensionality of the feature vectors compared. On performance of a Wilcoxon test, it was found that the performance of AIRS with a greater number of features was statistically better than with the lower number of features, with a probability value of $p < 0.0001$. The values for the mean and standard deviation for both data are presented in the following table.

	10 Features	50 Features
mean	37.1	92.7
SD	22.15	17.18

Further complexity was added to the problem through a comparison of the same dataset containing different numbers of documents. This particular dataset was again comprised of 2 class data, with one series containing 64 documents, and the other series containing 457, both consisting of vectors comprising of 200 features. The predictive accuracy on 5 times 10-fold cross-validation is shown in Figure 5.3. On performance of a Wilcoxon test, it was found that predictive accuracy of AIRS on a dataset of 128 documents was statistically significantly higher than the larger dataset of 457 documents (with $p < 0.00001$). The statistical results of this experiment are shown in the following table.

	Small Set	Large Set
mean	94.28	71.48
SD	12.58	6.35

Finally, feature vectors were created from a multi-class dataset. This contained 80 documents which were classified into 4 different categories. This was compared against

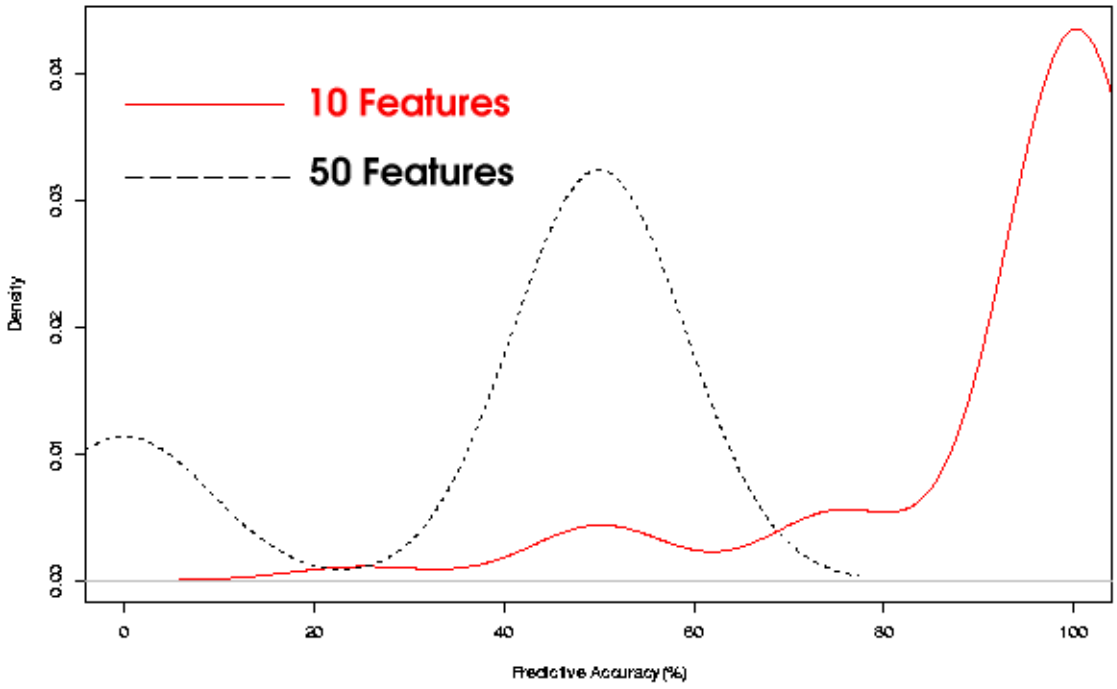

Figure 5.2: Classifier performance for 10 and 50 features, based on a small document dataset of 35 documents

a dataset of 64 documents, and 2 classes. The predictive accuracy on 5 times 10-fold cross-validation is shown in Figure 5.4. On performance of a Wilcoxon test, it was found that predictive accuracy of AIRS on the 2 class dataset was statistically no different than on the 4 class dataset, based on a probability which was greater than 0.05 ($p = 0.09$). The implications of this result are highly significant and will be discussed in detail. The other statistical information regarding this dataset can be found in the following table.

	Two Classes	Four Classes
mean	94.28	87.20
SD	12.58	20.10

As with the previous chapter, a full discussion of the presented results and their implications is given in Chapter 6.

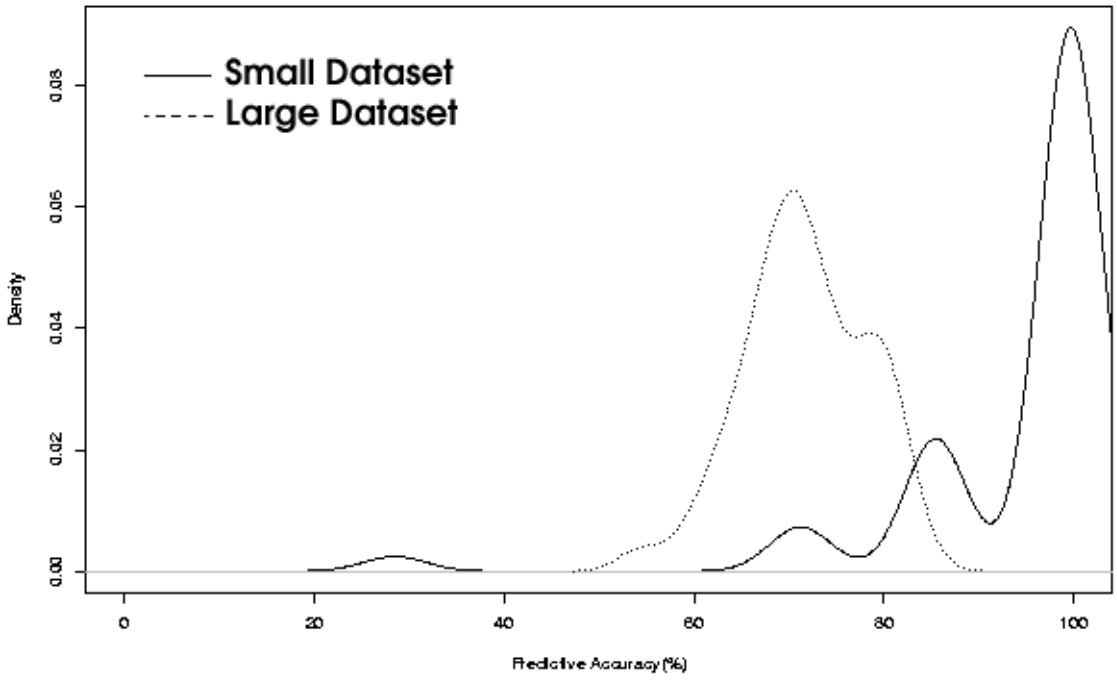

Figure 5.3: Comparison of classifier performance on medium and large document datasets, 64 and 457 documents respectively

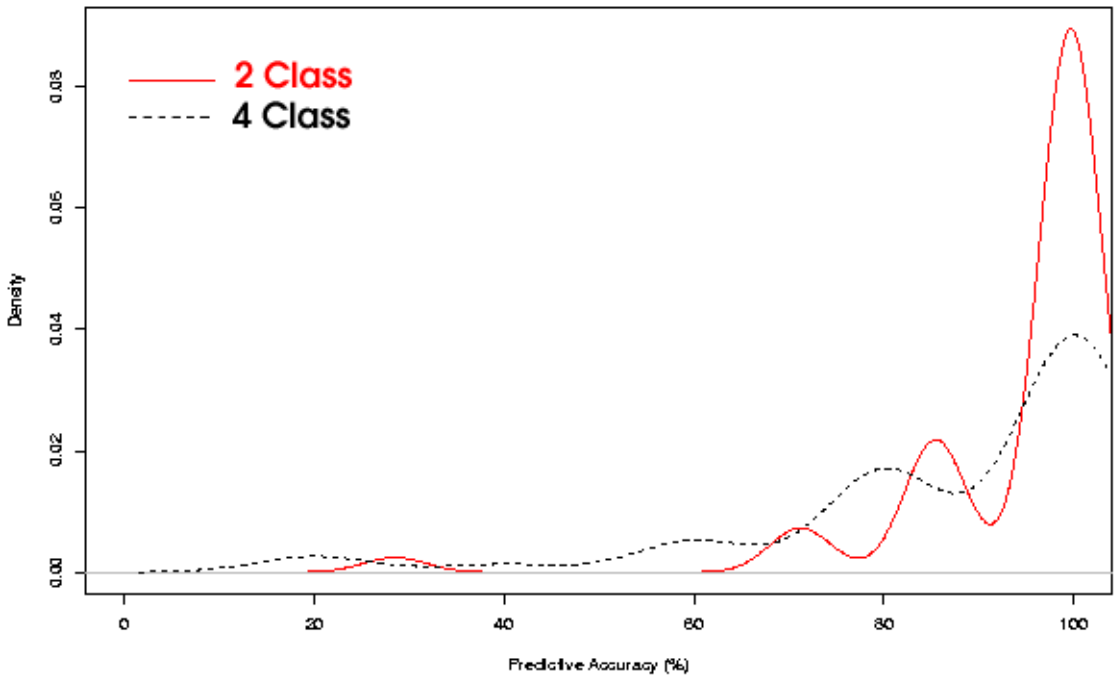

Figure 5.4: Comparison between 2 class and 4 class datasets containing 64 and 80 documents respectively

Chapter 6

Discussion

> *"Once you eliminate the impossible, whatever remains, no matter how improbable, must be the truth."* - Sherlock Holmes (by Sir Arthur Conan Doyle, 1859-1930)

The purpose of this chapter is to review and discuss the results presented in the Experimental Analysis chapters, considering the performance of the AIRS system for the various tasks. As with the analysis, the discussion will occur as two separate sections for the sake of clarity.

6.1 Validation

The validation of the AIRS system through the use of the voting dataset was a novel experimental series, as the system had not been tested on such a dataset before. The other purpose of this exercise was to ensure that the system could classify items of data to a suitable standard, and to familiarise ourselves with the basic parameter settings.

The three parameters examined were chosen because they appeared to be both instrumental to the functioning of the algorithm, and they were very easy to change (involving minor modifications to a configuration file). Firstly, the stimulation threshold parameter was examined. An decrease in the stimulation threshold should in theory provide an overall lower predictive accuracy, as it could result in the excessive proliferation of antibodies with an affinity lower that really necessary for effective classification. Even though there is variation present between the different settings for this

parameter, in comparison with the default setting of 0.95 no other value was significantly different. This could be expected as the creator of the system does remark that AIRS is robust to changes in the default settings of the configuration[37].

Secondly, the amount of resources available in the system was varied. Again, when compared with the default setting of 200, the resulting Wilcoxon test values prove that no significant difference was apparent in the predictive accuracies of the various number of resources. One explanation for this could be as the allocation of the resources allowed in the system shapes the positioning of the memory cells and limits the growth of the matching ARBS. A lower level of resources would imply that only ARBS with a high antigen affinity would be allowed to proliferate, thus providing under-fitting of the data. However, this is not statistically shown from the results of this experiment, but could be due to the quite restricted nature of the dataset (it contains a mere 16 features per vector).

The final parameter tested was the initial proportion of the training antigens used to create initial memory cells before the training procedure commenced. On comparison of using slightly under half of the total training items (i.e. 200 out of 435) to seed the system against a system that had not been prepared with initially created memory cells, the seeded system performed significantly better. Other comparisons with the rest of the values for this setting were not deemed statistically significant. It may be the case that values over 200 are subject to over-fitting of the data as seen from the decrease in the mean value. This could be due to the creation of too many memory cells initially so the antibody 'cover' of the feature space is not generalised enough to cope with the unseen data. However, this is merely suggested by the results presented in Figure 4.3. This could be problematic on a more sparse dataset, containing fewer training items or a very uneven training data class distributions.

In short, AIRS can be used successfully for the classification of this standardised dataset, giving a mean predictive accuracy of 92.86% (SD=3.46), using a stimulation threshold of 0.95, 200 resources and an initial creation of memory cells with 200 out of 435 training antigens. These positive results provide a new addition to the growing body of datasets on which AIRS has proven to be an effective classification system. On comparision with the AIS developed by Twycross & Cayzer[35] and the naive

Bayesian system, AIRS performed better than the Bayesian system, but not as effectively as the co-evolutionary AIS. The AIRS system, unlike the naive Bayesian system, does not assume independence between the different items comprising a feature vector, as the information is processed through a distance based measure. The poorer performance on this dataset in comparison to the Twycross & Cayzeer can be accounted for through their system containing 'wildcard' values which involve a generality bias i.e. uneven weighting of features (this is described in detail in the evaluation section of [35]). The potential incorporation of this concept to the AIRS system could be beneficial and will be discussed in Chapter 7.

6.2 Document Classification

On performance of various experiments, there is evidence to suggest that the AIRS system is suitable for use in the field of document classification. The experiments described in Chapter 5 have provided evidence that AIRS can be used not only for use in two-class classification problems, but can also be used in multi-class document classification, on a variety of different size problems.

Initially, a small document dataset was used in order to compare AIRS with another text classification system, namely Rainbow. The results of this were that the performance of the AIRS system was significantly better than Rainbow for this task, as is evident from Figure 5.1 and from the statistically significant result given by the Wilcoxon test. As with the result for the voting dataset, the reason for the good performance of the AIRS system in this case is likely due to the test data matching being performed using a distance based measure in feature space, which takes into account any potential relationships due which may be interrelated. This comparison is fair, as exactly the same preprocessing was performed on the input files, and both systems used information gain as the method of feature selection. Therefore the observed differences in the classifier performance are likely to be due to the classifiers themselves. The implications for these results are far reaching; AIRS can perform document classification with a higher level of predictive accuracy than the naive Bayesian based system, and therefore further experiments using datasets with varying characteristics can be performed using AIRS.

The next test performed using the system was to see if any difference occurred as the number of features investigated increased. It was found that this made a significant difference in the classification accuracy between using 10 features and 50 features on a small dataset. This implies that a rich feature representation is needed in order to perform such a classification task. If the number of informative words is below a certain proportion of words in the dictionary, then some of the words important in class discrimination may be omitted, accounting for the poor performance of the classifier under that particular set of conditions. However, I feel that this experiment does not provide conclusive evidence of this as only 2 datasets were used to demonstrate this point. A further investigation into this would include the use of greater numbers of features on datasets of different sizes, to determine more precisely how many features should be used. Nevertheless, the results of this have highlighted the importance of good feature selection and representation in order to achieve successful classification.

In line with the previous test, different size datasets were compared in order to test the ability of the AIRS system to scale with increasing problem sizes. While the system performed significantly better with a smaller dataset, the results of using a large dataset give a higher value for the mean than those derived from using the Rainbow system. Again, further experiments with a multitude of datasets of increasing size would be preferable to fully investigate the effect of document set size on the performance of the classifier set. It is not impossible at this point to suggest that document sets of a greater size may require a richer form of feature representation, or a more representative number of k-informative words in order to achieve greater levels of classification accuracy.

The results presented in the final section of Chapter 5 are perhaps the most significant. It was established that on a relatively small sized dataset, there was no statistically significant difference between the performance of AIRS on a 2 class problem or a 4 class problem. This infers that AIRS definitely a potential candidate for use as a multi-class document classification system. This hypothesis is based on the fact that the system has already been used for other forms of multi-class classification [36], [18], but the results of these test suggest that the system can be used in the realm of multi-class document classification. This has implications that the system will be suitable for future performance in the areas of hierarchical document classification and Semantic web re-

lated applications. As with the other experiments, more evidence is needed, through the use of of increasingly more complex datasets, before the exact usefulness of this system as a multi-class classifier will be determined. Nevertheless, the foundations for such works have been firmly laid and will be discussed in Chapter 7.

In summary, the AIRS system has out-performed another text classification system and has provided high levels of predictive accuracy in various document classification tasks. It has also been established that there are several different factors which may have an effect on the classification accuracy of the system, including the relative amount of features used and the size of the document dataset. However, most significantly, these early indications show that the AIRS system is a suitable tool for use in multi-class document classification. Further experimentation in this area will be useful in demonstrating exactly how effective AIRS is at performing document classification.

Chapter 7

Future Directions

> *"The best thing about the future is that it comes only one day at a time."*
> -Abraham Lincoln (1809-1865)

The applications and implications of the ability of such system to perform on a difficult classification task are far reaching, but there are still a number of different avenues relating to this project which still have not as yet been explored. There are several areas which this project can be expanded, namely the improvement of the information extraction techniques, modifications to the AIRS system itself, and more novel application areas for the system.

Firstly we could use a richer method of feature representation based on the word frequency and information gain values and compare the existing classifier performance, perhaps incorporating a method similar to a *tfidf* metric. One other possibility would be to use the values calculated for the word frequency in combination with the actual values calculated for the information gain of a dataset. These methods still rely on knowing the exact content of the dataset before the classifier is used. A method which could perform the information extraction in a more dynamical manner would be beneficial in improving the applicability of such a system to an actual Internet environment.

Modifications could be made to AIRS in order to improve its performance as a multi class document classification system. The affinity calculations at present are a distance based measure, which is perhaps not the most effective way of performing this, based on the fact that the co-evolutionary AIS produced a higher classification accuracy on the validation exercise. The addition of wild-cards or the utilisation of another

non-Euclidean distance metric could perhaps improve the classification accuracy when presented with increasingly difficult classification tasks. Additionally, a major modification to the system would include changing the algorithm from a 'one-shot training system' to one which also learns from the classification of the test items, as indeed the actual immune system does. This would apply a reinforcement learning approach on the system. Alternatively, the incorporation of other co-stimulation factors by human intervention [3], or even as a result of 'danger signals'[1].

As mentioned in the discussion section, this thesis has achieved an initial insight into the possibility of using AIRS in the domain of document classification. However, there is still a vast number of areas relating to this which can be explored. Immediately following on from this work, investigations into increasing document set sizes and more complex multi-class problems can be explored. The extension of the system into the realm of taxonomic or hierarchical classification seems a distinct possibility, as preliminary testing in this area has achieved potentially positive results. The use of AIRS as a hierarchical classifier would have strong implications for its use within a real world environment. The application of AIRS within the context of the Semantic Web would be the ultimate aim of this exercise. This would use the placing of the document into a taxonomy as a *semantic feature*, in addition to the information extracted out of the document. In accordance with the personalisation of such web tools, the mapping between the taxonomies into individual user defined hierarchies would assist in the navigation of the Internet, and provide automatically generated links between documents, without the intervention of human annotation. Further details of the proposed research into this area are highlighted in Greensmith and Cayzer [13].

But why stop at document classification? There are many other text based data sets (including the content of e-mails) from which such a classification would be preferable. The versatile nature of the AIRS system indicates that it may be able to perform well on a multitude of other classification tasks, providing that a suitably representative method of feature extraction can be performed. It will be interesting to see just how applicable this system can be to future classification tasks.

Chapter 8

Conclusions

> *"I may not have gone where I intended to go, but I think I have ended up where I intended to be."*- Douglas Adams (1952-2001)

As we are nearing the end of the thesis, it is time to briefly review the content of the previous chapters. Chapter 1 outlined the problem area and the general themes and and concepts involved in the work to be performed. It highlighted the extent of the problem: we are experiencing a data overload, and there are various methods available to help us achieve this, including the use of biologically inspired systems. The concept of using an artificial immune system, AIRS to perform the task of document classification is proposed. Chapter 2 provided the reader with the necessary information in order to understand the processes involved in the classification procedure and immune systems, both natural and artificial are explained with reference to this context. Chapter 3 contained the implementation details of how classification could be achieved through using the AIRS system. Chapters 4 and 5 contained the results of such experiments, based a standardised machine learning dataset and for document classification, respectively. In these chapters we found that AIRS was a comparably good classification system both on the validation series of experiments and for small problem document classification. The system performed significantly better than another established classification system in both cases. Additionally, AIRS functioned as a classifier system within the domain of multiple-class document classification. The implications of these results were presented in Chapter 6. Further work and future applications of the AIRS system were presented in Chapter 7.

As a final review of the system, it is important to re-state the aim of the project and to

evaluate the outcome based on this original information. As stated in Chapter 1, the objectives of this project are to compare the performance of the AIRS system on a validation set and a document set, with the performance of other methods, including one other artificial immune system; and to introduce a multi-class artificial immune system classifier to a novel application area, namely document classification. As outlined in the opening paragraph of this chapter, the AIRS system was able to consistently perform classification tasks to a level which was comparable with, if not better than other well established methods. The outcome of this thesis is highly significant, because not only have we shown the place for biologically inspired approaches to document classification, but we have pushed forward the use of the AIRS system into this novel application area.

Bibliography

[1] Uwe Aickelin and Steve Cayzer. The danger theory and its application to artificial immune systems. In Jonathan Timmis and Peter J. Bentley, editors, *Proceedings of the 1st International Conference on Artificial Immune Systems (ICARIS)*, pages 141–148, University of Kent at Canterbury, September 2002. University of Kent at Canterbury Printing Unit.

[2] J. Ambühl, D. Cattani, and P. Eckert. Classification of meteorological patterns. In *Proc. ICANN'97, 7th International Conference on Artificial Neural Networks*, volume 1327 of *Lecture Notes in Computer Science*, pages 1119–1124. Springer, Berlin, 1997.

[3] Justin Balthrop, Fernando Esponda, Stephanie Forrest, and Matthew Glickman. Coverage and generalization in an artificial immune system. In W. B. Langdon, E. Cantú-Paz, K. Mathias, R. Roy, D. Davis, R. Poli, K. Balakrishnan, V. Honavar, G. Rudolph, J. Wegener, L. Bull, M. A. Potter, A. C. Schultz, J. F. Miller, E. Burke, and N. Jonoska, editors, *GECCO 2002: Proceedings of the Genetic and Evolutionary Computation Conference*, pages 3–10, New York, 9-13 July 2002. Morgan Kaufmann Publishers.

[4] Justin Balthrop, Stephanie Forrest, and Matthew R. Glickman. Revisting LISYS: Parameters and normal behavior. In David B. Fogel, Mohamed A. El-Sharkawi, Xin Yao, Garry Greenwood, Hitoshi Iba, Paul Marrow, and Mark Shackleton, editors, *Proceedings of the 2002 Congress on Evolutionary Computation CEC2002*, pages 1045–1050. IEEE Press, 2002.

[5] Tim Berners-Lee, James Hendler, and Ora Lassila. The Semantic Web. *Scientific American*, 284(5):34–43, May 2001.

[6] C.L. Blake and C.J. Merz. UCI repository of machine learning databases, 1998. http://www.ics.uci.edu/∼mlearn/MLRepository.html.

[7] Dave Cliff. Biologically-inspired computing approaches to cognitive systems : a partial tour of the literature. Technical Report HPL-2003-11, Hewlett Packard Laboratories, January 16 2003. http://www.hpl.hp.com/techreports/2003/HPL-2003-11.html; http://www.hpl.hp.com/techreports/2003/HPL-2003-11.pdf.

[8] L.N. de Castro and J. Timmis. *Artificial Immune Systems: A New Computational Approach*. Springer-Verlag, London. UK., September 2002.

[9] Elizabeth K Dicke. Designing storage area networks with biologically-inspired approaches. Master's thesis, University of Sussex, September 2003.

[10] AnHai Doan, Pedro Domingos, and Alon Y. Halevy. Reconciling schemas of disparate data sources: A machine-learning approach. In *SIGMOD Conference*, 2001.

[11] Google Press Office: Google Fun Facts. http://www.google.com/press /fun-facts.html.

[12] AA Freitas. *Data Mining and Knowledge Discovery with Evolutionary Algorithms*. Spinger-Verlag, Berlin, 2002. Page 34.

[13] Julie Greensmith and Steve Cayzer. An artificial immune approach to semantic document classification. In Jonathan Timmis, Peter J. Bentley, and Emma Hart, editors, *Proceedings of the 2nd International Conference on Artificial Immune Systems (ICARIS)*, pages 141–148. LNCS, 2003.

[14] Steven A. Hofmeyr and Stephanie Forrest. Architecture for an artificial immune system. *Evolutionary Computation*, 8(4):443–473, 2000.

[15] J. E. Hunt, D. E. Cooke, and H. Holstein. Case memory and retrieval based on the immune system. *Lecture Notes in Computer Science*, 1010:205–216, 1995.

[16] Charles A. Janeway, Paul Travers, Mark Walport, and Mark Shlomchik. *Immunobiology The Immune System in Health and Disease*. New York : Garland ; Edinburgh : Churchill Livingstone., 2001.

[17] Jungwon Kim and Peter J. Bentley. An evaluation of negative selection in an artificial immune system for network intrusion detection. In Lee Spector, Erik D. Goodman, Annie Wu, W. B. Langdon, Hans-Michael Voigt, Mitsuo Gen, Sandip Sen, Marco Dorigo, Shahram Pezeshk, Max H. Garzon, and Edmund Burke, editors, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2001)*, pages 1330–1337, San Francisco, California, USA, 7-11 July 2001. Morgan Kaufmann.

[18] Gaurav Marwah and Lois Boggess. Artificial immune systems for classification: Some issues. In Jonathan Timmis and Peter J. Bentley, editors, *Proceedings of the 1st International Conference on Artificial Immune Systems (ICARIS)*, pages 149–153, University of Kent at Canterbury, September 2002. University of Kent at Canterbury Printing Unit.

[19] Polly Matzinger. The danger model: A renewed sense of self. *Science*, 296(1):301–305, 2002.

[20] Melanie Mitchell. *An Introduction to Genetic Algorithms*. Complex Adaptive Systems. MIT-Press, Cambridge, 1996.

[21] Tom M. Mitchell. *Machine Learning*. McGraw-Hill, New York, 1997.

[22] Directory Mozilla (DMOZ) open directory project. http://dmoz.org/.

[23] W3C Standards pages. http://www.w3.org/tr/html4/struct/global.html.

[24] M. G. Partridge and R. A. Calvo. Fast dimensionality reduction and simple PCA. *Intelligent Data*, 2(3), 1998.

[25] M. J. Pazzani. Searching for dependencies in bayesian classifiers. In *Proceedings of the Fifth International Workshop on Artificial Intelligence and Statistics*, pages 239–248. Springer-Verlag, 1996.

[26] M.F Porter. An algorithm for suffix stripping. *Program*, 14(3):130–137, 1980.

[27] Mitchell A. Potter and Kenneth A. De Jong. The coevolution of antibodies for concept learning. *Lecture Notes in Computer Science*, 1498:530–535, 1998.

[28] Fabrizio Sebastiani. Machine learning in automated text categorization. *ACM Computing Surveys (CSUR)*, 34(1):1–47, 2002.

[29] Rainbow Text Classification System. http://www-2.cs.cmu.edu/afs/cs/project/theo-11/www/naive-bayes.html.

[30] Students T-Test. http://home.clara.net/sisa/t-test.htm.

[31] The Standard Taxonomies. http://www.thestandard.com.

[32] Yahoo Taxonomies. http://www.yahoo.com.

[33] Wilcoxon Mann-Whitney Test. http://www.fon.hum.uva.nl/service/statistics/wilcoxontest.html.

[34] HTML to ASCII Converter for Linux. http://rpmfind.net/linux/rpm2html/search.php?query=html2text.

[35] Jamie Twycross and Steve Cayzer. An immune-based approach to document classification. Hewlett-Packard Technical Report Number HPL-2002-288, http://www.hpl.hp.com/techreports/2002/HPL-2002-288.html.

[36] Andrew Watkins and Jonathan Timmis. Artificial immune recognition system (AIRS): Revisions and refinements. In Jonathan Timmis and Peter J. Bentley, editors, *Proceedings of the 1st International Conference on Artificial Immune Systems (ICARIS)*, pages 173–181, University of Kent at Canterbury, September 2002. University of Kent at Canterbury Printing Unit.

[37] Andrew B. Watkins and Lois C. Boggess. A Resource Limited Artificial Immune Classifier. In *Proceedings of Congress on Evolutionary Computation*, pages 926–931. IEEE, May 2002. Part of the 2002 IEEE World Congress on Computational Intelligence held in Honolulu, HI, USA, May 12-17, 2002.

Appendix A

Scripts and Source Code

This section contains scripts and programs used in combination with the AIRS classification system. Scripts `parse`, `preprocess`, `masterUniverse`, `word2class` and `wordlist` were used to derive the necessary information for the information gain calculation. The program `readIn.cc` was used to perform the information gain calculation, and the features were processed using the `features` script. The programs `val.cc` and `stat.cc` were used to automate the validation procedure. It is important to note that additional scripts were run on the command line in order to complete the process, but are not included in this appendix. I would like to further stress that these programs would not have been written without the support of Liz Dicke, Justin Balthrop, Steve Cayzer and Gillan Cash.

A.1 Bash Scripts

```
#!/bin/bash

for i in *.html; do
html2text -nobs $i | wordlist > ${i%.html}.out;
done
```

```
#!/usr/bin/perl

print "<classified>\n";
print "Water\n";
```

```
while (<>) {
    @list = split /[^a-zA-Z]/;
    foreach $word (@list) {
       print lc $word, "\n" if $word;
    }
}
```

```
#! /bin/bash

# check usage
if [ -z "$1" ] ; then
    echo "Usage: $0 <category>"
    exit 1
fi

for i in $(ls *.out); do
    cat $i | grep -v "<classified>" | grep -v $1 | sort
 | uniq > $i.tmp
    mv $i.tmp $i
done
```

```
#! /bin/bash

# check usage
if [ -z "$1" ] ; then
    echo "Usage: $0 <category>"
    exit 1
fi

# concatenate all files to make dictionary
rm words_$1
numfiles=0;
for i in $(ls *.out); do
```

```
    cat $i >> words_$1
    let numfiles=$numfiles+1
done

# sort out dictionary
cat words_$1 | awk '{FS=" "; print $1}' | sort | uniq > tmp
mv tmp words_$1

# process all words in dictionary
rm count_$1_*
for i in $(cat words_$1); do
    echo "processing word $i"
    count=0;
    for filename in $(ls *.out); do
        if ( grep "^$i$" $filename 1>/dev/null )
        then
            let count=$count+1
        fi
    done

    echo $count >> count_$1_present
    let absent=$numfiles-$count
    echo $absent >> count_$1_absent
    echo $numfiles >> count_$1_numfiles
done
paste words_$1 count_$1_present count_$1_absent
count_$1_numfiles> matrix_$1
#! /bin/bash

#take all files and compile a master dictionary

numcats=0;
totDocs=0;
rm all_words
```

```
for i in $(ls words_*); do
    cat $i >> all_words
    let numcats=$numcats+1
done

#sort the dictionary

sort all_words | uniq > tmp
mv tmp all_words

# output present and absent totals for all words
#in each categories
for j in $(ls matrix_*); do
    cat=${j:7}
    echo " *** Processing category $cat"

    rm augmented.$j
    for i in $(cat all_words); do
        echo -n "$i "
        line=$(grep "^$i\>" $j )
if [ -n "$line" ] ; then
            stats=${line/$cat/}
            stats=${stats//$i/}
            stats=${stats/[ ]+/ }
            echo "... FOUND  - $stats "
    echo $stats >> augmented.$j
else
            # numdocs=$(ls $cat/*.out | wc | cut -f6 -d' ')
            numdocs=$(ls $cat/instances/*.out | wc
| awk '{FS=" "; print $1}')
    echo "... NOT FOUND - 0      $numdocs $numdocs"
    echo 0 $numdocs $numdocs >> augmented.$j
    ndcs=$numdocs
fi
```

```
    done
    let totdocs=$totdocs+$ndcs
    echo $totdocs $numcats > info


done

# finally stitch all the categories together
paste all_words $(ls augmented*) > dictionary
echo "****** THE FINAL DICTIONARY ******"
cat dictionary
```

```
#!/bin/bash

for file in *.out; do
    out=''
    for word in `cut -d' ' -f1 $1`; do
if (grep $word $file > /dev/null); then
    out="$out 1"
else
    out="$out 0"
fi
    done
    echo ${file%.out} $out `pwd`
done
```

A.2 Information Gain Calculation Program

```
#include <iostream>
#include <string>
#include <vector>
#include <fstream>
#include <sstream>
#include <algorithm>
```

```
using namespace std;
///a structure to store the values read in for use in the Info
//gain
class WordsVals{

public:
string label;
vector <int> values;
double iGain;
};
////function prototypes
double logTwo(double);
double CalcInfoGain(WordsVals,int, int, double);
double ISFunc(double[], unsigned int);
bool checkOk(double *, int);
void sort(vector<WordsVals>&);
void createFeatures(int, vector<WordsVals>, string);
void doInfoGain(WordsVals&, int, int);

int main(int argc, char * argv[]){

if(argc !=5){
cerr << "Correct usage is <./exe> <dictionary>
<info>  <int k> <InfoGain.out>";
exit(1);
}

//input from the dictionary in the form
//word present absent numInClass p a nC etc etc

ifstream fin;

fin.open(argv[1]);
```

```
if(fin.fail()){
cerr << "Input file could not open.";
exit(1);
}

//information file containing number of docs overall
// and num classes

ifstream infoin;

infoin.open(argv[2]);

if(infoin.fail()){

    cerr << "Info file did not open. " ;
    exit(1);
}

int tempArr[2];
for (int i=0; i < 2; i++)
    infoin >> tempArr[i];
int numClasses = tempArr[1];
int totNumDocs = tempArr[0];
cout << "We have "<<numClasses <<
"classes and "<< totNumDocs << " documents" << endl;
//okay, so lets read in the items in the file into
// an object we can use

vector <string> items; //tempory reading in vector
vector <WordsVals> WVvec; //nicely assigned vector containing
int cVal;  //the word and the value of the word

while(!fin.eof()){
```

```
string s;
getline(fin, s);
if(s == ""){
cout << "Hello" << endl;
break;
}

items.push_back(s);
}

for (vector<string>::iterator iter = items.begin();
 iter!=items.end(); ++iter)
{

// cout << "Stuff is " << *iter << endl;
string temp;
WordsVals tempWV;

temp = *iter;
cout << "Temp is " << temp << endl;

istringstream ss(temp);

ss >> tempWV.label;

while (ss)
   {
ss >> cVal;

tempWV.values.push_back(cVal);


   }
```

```
WVvec.push_back(tempWV);

}

double ISval = 0.0;

double pClass[numClasses+1];

int pCount =0;
   WordsVals tempWV;
tempWV = WVvec[0];
for (int i = 2; i < tempWV.values.size(); i+=3){

cout << "wordVal stuff" << tempWV.values[i] << endl;

pClass[pCount] =
(double)tempWV.values[i]/(double)totNumDocs;
// pClass[pCount] =
wordVals.values[i];
cout << "PClass is" << pClass[pCount]<< endl;

// pCount++;
// cout << "Here!!!";
// }
}
  ISval = ISFunc(pClass, numClasses);
cout << "ISval is "<< ISval << endl;



// for (vector<WordsVals>::iterator WViter = WVvec.begin();
 WViter!=WVvec.end(); ++WViter){
for (int i = 0; i < WVvec.size(); i++){
```

```
WordsVals tempWV;
tempWV = WVvec[i];

cout << "go! " << tempWV.label << " " << endl;

CalcInfoGain(tempWV, numClasses, totNumDocs, ISval);
WVvec[i] = tempWV;
}

sort(WVvec);

int k = atoi(argv[3]);

ofstream fout;

fout.open(argv[4]);

if(fout.fail()){

cerr<<"Could not open output file.....now exiting!";
exit(1);
}

for(int i =0; i < k; i ++){

fout << WVvec[i].label << " " << WVvec[i].iGain << endl;

}

fin.close();
fout.close();

return 0;
}
```

```
/*void doInfoGain(WordsVals &wordVals, int totNumDocs,
 int totNumClasses){

//calculate the I(s) part
//double ISval = 0.0;
double InfoGain = 0.0;
//probability of a document belonging to a class
double pClass[totNumClasses+1];

int pCount =0;

for (int i = 2; i < wordVals.values.size(); i+=3){

cout << "wordVal stuff" << wordVals.values[i] << endl;

pClass[pCount] = (double)wordVals.values[i]/(double)totNumDocs;
// pClass[pCount] = wordVals.values[i];
cout << "PClass is" << pClass[pCount]<< endl;

// pCount++;
// cout << "Here!!!";
// }
}
  ISval = ISFunc(pClass, totNumClasses);
cout << "ISval is "<< ISval << endl;
//InfoGain = CalcInfoGain(wordVals, totNumDocs, totNumClasses, ISval);

//wordVals.iGain = InfoGain;

return;

}*/

double logTwo(double number)
```

```
{
   double answer = 0.0;
   if (number != 0.0)
// must check in case something else does logTwo(0)
      answer = log(number)/log(2.0);
cout << "ans is in L2 " << answer << endl;
   return answer;
}
double ISFunc(double pClass[], unsigned int totNumClasses)
{
   double ISval = 0.0;
   double val = 0.0;
   double temp = 0.0;
   double theLog = 0.0;


   for(int a=0; a < totNumClasses; a++)
   {
temp = pClass[a];
theLog = logTwo(pClass[a]);
cout << "Temp is " << temp << endl;
val = temp * theLog;
cout << "Val is " << val << "and a is "<< a << endl;
cout << "ISval is " << ISval << endl;
ISval = val + ISval;
cout << "Check" << endl;
   }
cout << "Check" << endl;

   ISval *= -1.0;
cout << "Check";
   return ISval;
}
double CalcInfoGain(WordsVals wordVals,
```

```
 int numClasses, int totNumDocs, double ISval)
{
    double InfoGain = 0.0;
// set up probabilistic variables
    double Ppres, Pabs, pISC, npISC;

    // initialise probabilities
    Ppres = Pabs = pISC = npISC = 0.0;

    int presentInDocsOverall = 0;
int *presentInCat;
presentInCat = new int[numClasses+1];



    //probability of being present is the amount of
//times the word is present
    //overall, divided by the total number of documents
   for (int i = 0; i < numClasses; i++){

cout  << "wordVals.values[i] is " << wordVals.values[i*3]
<< "with i being" << i << endl;
presentInDocsOverall += wordVals.values[i*3];
presentInCat[i] = wordVals.values[i*3];



    }
cout << "Check! Pin DOcs = "<< presentInDocsOverall
 << " and tot num docs is " << totNumDocs<< endl;
     double p1 = (double)presentInDocsOverall;
double p2 = (double)totNumDocs;
Ppres = (double)presentInDocsOverall / (double)totNumDocs;

    // e.g if Ppres = 0.6, Pabs will = 0.4
```

```
    Pabs = 1.0 - Ppres;


cout << "Ppres=" << Ppres << " and Pabs=" << Pabs << endl;

double *pCat; //prob of it being present in a category

pCat = new double[numClasses+1];


for (int j = 0; j < numClasses; j++)
{
       // a word might be present 4 times in Cat[0]
out of a total appearance of 13 times

cout << "Present in cat[" << j << "] is "
<< presentInCat[j] << endl;
double pc1 = (double)presentInCat[j];
double pc2 = (double)presentInDocsOverall;

cout << pc1 << " is pc1 and pc2 is " << pc2 << endl;

pCat[j] = (pc1 + 1) / (pc2 + 2);
cout << "PCat[" << j << "] is " << pCat[j] << endl;



}

      if (!checkOk(pCat, numClasses)) {
 cout << "Error: pCat values do not add up to 1." << endl;
      }
pISC = ISFunc(pCat, numClasses);

cout << "pISC is  RAAAAAAAAAAAA" << pISC;
```

```
double *npCat;

npCat = new double[numClasses+1];

for (vector<int>::iterator iter = wordVals.values.begin()+2;
iter!=wordVals.values.end(); iter+=3)
{
  double csize = (double)*iter;
//how many documents in that particular category
  double p1 = ((double)*(iter-1));
//how many times the word is absent
  double p2 = (double)totNumDocs - (double)presentInDocsOverall;
  double tot = (double)totNumDocs;
  double wpres = (double)presentInDocsOverall;

  npCat[*iter] = (p1+1) / (p2+2);  //zero division correction

}


npISC = ISFunc(npCat, numClasses);
// no values in npCat can be < 0

//do formula
cout << "\t[" << ISval << " -
((" << Ppres << '*' << pISC << ") +
 (" << Pabs << '*' << npISC << "))] \t";
InfoGain = (ISval - ((Ppres * pISC) + (Pabs * npISC)));


delete [] presentInCat;
delete [] pCat;
delete [] npCat;
```

```
return InfoGain;
}
bool checkOk(double *arr, int size)
{
   double total = 0.0;
   for (int i = 0; i < size; i++) {
      total += arr[i];
   }
   if (total != 1.0) {
      cout << '{';
      for (int i = 0; i < size; i++) {
 cout << arr[i] << ' ';
      }
      cout << "} ";
      return false;
   }
   else
      return true;
}

bool WVLess(WordsVals wv1, WordsVals wv2) {
return wv1.iGain < wv2.iGain;
}

void sort(vector<WordsVals> &WVec){

sort(WVec.begin(), WVec.end(), WVLess);

}

/*void createFeatures(int, vector<WordsVecs> &, string){
```

```
}*/
```

A.3 Validation Programs

```
#include <iostream>
#include <string>
#include <fstream>
#include <vector>
#include <string>
#include <stdlib.h>
#include <ctime>
#include <cmath>

using namespace std;

int main(){


char votes[10000];
char* temp;
int count =0;
vector <char*> votingVec;
vector <char*>::iterator iter;

srand(time(NULL));  //seeding the random value


ifstream fin; //to use file input and output
ofstream fout;
ofstream fout2;

fin.open("houseMaster.data");
// this file contains the dataset
fout.open("house.tst");
```

```
// output test data
fout2.open("house.trn");
// output training data

if (fin.fail()){ //some error checking
cerr << "File did not open...aborting" << endl;
exit(1);
}

if (fout.fail()){
cerr << "Test data file did not open...aborting" << endl;
exit(1);
}

if (fout2.fail()){
cerr << "Training data file did not open...aborting" << endl;
exit(1);
}

for (int g =0; g < 435; g++){

fin.getline(votes, (10000));
temp = new char[(strlen(votes)) + 1];
strcpy(temp, votes);

votingVec.push_back(temp);

}
cout <<"size before = "<< votingVec.size() << endl;

for (int k =0; k < 43; k++){

iter = votingVec.begin() + rand() % votingVec.size();
```

```
fout << *iter << endl;

votingVec.erase(iter);
}

cout << "size after = " << votingVec.size() << endl;

for (iter = votingVec.begin(); iter != votingVec.end(); iter++) {

fout2 << *iter << endl;

}

fin.close();
fout.close();
fout2.close();

delete [] temp;

return 0;

}
```

```
#include <iostream.h>
#include <string>
#include <fstream>
#include <vector>
#include <string>
#include <stdlib.h>
#include <ctime>
#include <cmath>
#include <stdio.h>

using namespace std;
```

```
int main(){


char lines[10000];
char* temp;
int count =0;
vector <char*> lineVec;
vector <char*>::iterator iter;
vector <char*>::iterator iter2;

srand(time(NULL));
//seeding the random value


ifstream fin; //to use file input and output
ofstream fout("Seed_300.sts", ios::in | ios::app);
ofstream fout2;

fin.open("house.sts");   // this file contains the dataset
     // output test data
//fout2.open("house.trn");      // output training data

if (fin.fail()){ //some error checking
cerr << "File did not open...aborting" << endl;
exit(1);
}

if (fout.fail()){
cerr << "Test data file did not open" << endl;
exit(1);
}

/*if (fout2.fail()){
cerr << "Training data file did not open" << endl;
```

```
exit(1);
}*/

for (int g =0; g < 30; g++){

fin.getline(lines, (10000));
temp = new char[(strlen(lines)) + 1];
strcpy(temp, lines);

lineVec.push_back(temp);

}
cout <<"size before = "<< lineVec.size() << endl;

// for (int k =0; k < 43; k++){

iter = lineVec.begin() + 29;
iter2 = lineVec.begin() + 23;
fout << *iter <<" " <<  *iter2 <<endl;

// lineVec.erase(iter);
// }

cout << "size after = " << lineVec.size() << endl;


fin.close();
fout.close();

delete [] temp;

return 0;
}
```

Appendix B

AIRS Configuration File

```
# where to output the trained system
NetOut house.net

# where to output the predictions for each vector in the test
# and training set
Pred house.prd

# the number of features in each vector in the data set
NGenes 16

# the number of classes in the data set
Classes 2

# is there a label for each vector?
# if so, this label must be in the 1st column of the data
labels no


#various learning parameters

# number of passes through the training data
# has only been experimented, really, with value of 1
Epochs 1

# the clonal rate
```

```
CRate 10

# the mutation rate
MRate 0.001

# number of training data items to seed the network
InitSize 400

# the Affinity Threshold Scalar (ATS)
Nat 0.2

# the stimulation threshold
StimT 0.95

# the number of resources allowed in the system
NumRes 200

# the K value used for the systems response to data items
BcellK 7

# this is the hyper mutation rate for Memory Cell offspring
# if it is not set, then the default value is 2
HRate 10.0

# if NumRes is not set, then RRate is used as a scalar for
# determining the number of resources in the system
# numres = RRate * number of training items
#RRate 3.0

# this is the rate at which resources are added to an ARB
# if it is not specified then RAddRate = Clonal Rate
#RAddRate 5

# if InitSize is not set, then TrainP is used to determine
# the number of data items to use to seed the network
# initsize = TrainP * number of training items
#TrainP 0.2
```