Luís André Brísio Marques dos Santos

Implementation and Evaluation of a Spam Classifier based on the Dynamic Behaviour of Immune Cells

Departamento de Ciência de Computadores
Faculdade de Ciências da Universidade do Porto
Outubro de 2008

Luís André Brísio Marques dos Santos

Implementation and Evaluation of a Spam Classifier based on the Dynamic Behaviour of Immune Cells

U.PORTO
FC FACULDADE DE CIÊNCIAS
UNIVERSIDADE DO PORTO

Tese submetida à Faculdade de Ciências da Universidade do Porto para obtenção do grau de Mestre em Bioinformática

Departamento de Ciência de Computadores
Faculdade de Ciências da Universidade do Porto
Outubro de 2008

Acknowledgments

Este projecto não teria sido possível sem a ajuda e orientação do Professor Doutor Eduardo Correia, que tanto ensinou ao longo deste trabalho. Obrigado também ao Mário Antunes, com quem partilhei o projecto e através de quem obtive muita ajuda e sugestões, ao Professor Doutor Jorge Carneiro, pelas sua contribuições e indicações.

Aos investigadores com quem contactei e de quem me socorri, e que me deram o seu apoio, um muito obrigado e muito boa sorte.

A todos os que me rodearam durante o decorrer deste trabalho: os meus pais, a minha irmã, Rita — obrigado pelo apoio.

This project would not have been possible without the help and guidance of Professor Eduardo Correia, who has shared so much knowledge throughout this work. My many thanks to Mário Antunes, with whom I shared the project and who oftenly helped and provided suggestions, and to Professor Jorge Carneiro as well, for his contribuitions and directions.

A big thank you to all the researchers who I contacted and resorted to, and who have promptly replied back. I wish them all the best of luck.

For all the people that have been around me during the development of this project: my parents, my syster, Rita — thank you all very much for supporting me.

Resumo

Nas rápidas auto-estradas da informação dos dias de hoje, o e-mail revela-se uma ferramenta de valor incalculável, através do qual se trocam mensagens de grande importância — pessoal ou profissional. O spam, que pode ser considerado o "lado negro" do e-mail, tornou-se num constante incómodo que nos afecta a todos em diferentes aspectos e por diferentes vias. Por forma a combater uma situação que só tem vindo a piorar - um pouco por todo o mundo, legisladores elaboram medidas e definem fronteiras legais [26, 25, 73] para tentar conter o spam, enquanto programadores e investigadores desenvolvem novos filtros e métodos de bloqueio [35, 51, 54, 57]. No entanto, e apesar do esforço, os spammers parecem ser sempre capazes de encontrar forma de contornar os obstáculos e chegar até às nossas caixas de correio — e assim o problema subsiste.

Neste trabalho, o Sistema Imune Humano foi analizado como um sistema de processamento de informação [11, 30]. Seguindo os princípios matemáticos do modelo [16] da teoria do Limiar de Activação Ajustável (Tunable Activation Threshold theory - TAT) proposta por Grossman e Paul [37], desenvolvemos e implementámos uma plataforma que esperamos que venha a contribuir na luta contra o spam. Os resultados apresentados revelam a imaturidade do sistema mas também o seu grande potencial quando em comparação com uma implementação clássica de um filtro baseado no algoritmo Naïve-Bayes Multinomial.

Abstract

E-mail has become a priceless tool in the information highway. Maybe more often than we would expect, some of the messages exchanged are at least very important, if not vital. Spam, the dark side of e-mail, is now a constant nuisance and affects us in many different aspects. To counter a worsening situation, legislators around the world have devised legal boundaries [26, 25, 73] trying to contain spam, while programers and researchers develop filtering solutions and blocking methods [35, 51, 54, 57]. However, despite their efforts, spammers always seem to be able to find a way to bypass the systems and target our inboxes — hence the problem persists.

In this work, we look at the Human Immune System as an information processing system [11, 30]. Following a mathematical model [16] of the Tunable Activation Threshold theory proposed by Grossman and Paul [37], we have developed and implemented a framework that we hope will contribute to the spam fighting cause. The results proved its immaturity but show great potential, when compared to an implementation of a spam filter based on the Multinomial Naïve-Bayes version.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

In the last three decades a fast-paced and widely spread technological evolution has been taking place. Since the 70's all aspects in our lives have been suffering remarkable changes. Ranging from transportation to new discoveries and inventions in telecommunications, the whole world changed considerably. Amongst all those novelties and maybe one of the most important ones is the Internet. It came live[1] in October 19th, 1969, under the name ARPANET — *Advanced Research Projects Agency Network* — with a transmission between two machines: one at University of California at Los Angeles and the other at Stanford Research Institute, SRI, in Stanford. The underlying packet switching communication system proved to be a feasible alternative to circuit switching (as in telephone circuits), allowing for time-sharing computers to work together, run programs and retrieve data, as confirmed by Lawrence G. Roberts and Thomas Merril in their 1965 experiment [48]. Vinton Cerf and Robert Kahn developed a transmission protocol (Transmission Control Protocol, TCP) addressing all the transport and forwarding services allowing for multiple applications to make use of the Internet [19].

Its potential was enormous — as it still is nowadays — and soon it became international. By 1985, commercial interest in the network blossomed, leading to the merging of several economical and educational gateways into the structure existent at the time. It took only 30 years for the word *internet* to become common, even though it was referring not only to the original Internet but also to the World Wide Web project [74, 75]. The latter is a system of interlinked hypertext documents, called *web-pages*, and was developed by Sir Tim Berners-Lee in 1990, while at the

[1] Fun fact: the message was 'login' but the system crashed — 'Lo' was the first message ever transmitted on the ARPANET; one hour later a full login was accomplished.

Conseil Européen pour la Recherche Nucléaire (CERN), in Switzerland.

Having information available from and to the whole world became an interesting prospect and soon all kinds of institutions were having a presence in the Web — first came the companies' homepages advertising their products, banks announcing their services, research groups, schools and universities publishing their awards and education opportunities on the web. Web search utilities where in their early years and personal web pages began to sprung. Taking advantage of the network's structure and the TCP/IP protocol design, allowing the easy integration of institutional networks into the Internet, daily life was simplified: checking bank statements is now as easy as logging in into the costumers area on a bank's homepage; paying — or subscribing — the annual fee of a magazine can be done without leaving your home or filling out coupons; and thanks to developments in late 2006 and 2007, rich media streaming allowed for watching on-line a TV show you missed while working late.

1.1 The time is now

Even though its advantages are remarkable and impossible to overlook, the digital world of Internet is also at the forefront of a whole new set of issues and legal battles. On-line services increased and developed to a point where buying a home theater set, a plasma TV, changing auto insurance carrier, or even ordering food, is almost as easy as a double click. In turn, personal details are required for each transaction, being stored in databases accessible on-line and sometimes exchanged between corporations. The ease of access to personal, very sensitive, information is valid both for the rightful user as for a possible intruder with malicious intentions, also known as *hacker*[2].

While it is not easy to trace back to the first hacker, a few names are part of this area of History: The 414's, Kevin Mitnick, Robert T. Morris Jr, amongst others, were responsible for the early threats to both computer networks and systems, raising governments' attention to new kinds of fraud [76, 80]. But if the most mediated events regarding Internet and hackers have always been related to breaking into systems, stealing money from banks or deploying viruses and trojan horses that slow systems down to a near halt, a much different action was started in 1978 and still goes on today, causing a lot of trouble.

[2]not to be confused with *cracker*, which refers to someone who modifies software.

When the ARPANET was established, there were already communication protocols to transfer messages between computer users. The first message sent over a network dates back to 1971 [74] - Ray Tomlinson was responsible for the current addressing format of `user@his.address`, which came to improve single user computer electronic mail (*e-mail*, for short) [70]. In recent years, e-mail has become widely spread, available through any Internet connection and accessible by different devices: computers, mobile phones, PDAs, Blackberries and others. In the busy world of today, information is essential and e-mail is a mean by excellence to communicate — fast, precise and powerful. You can contact your team manager who is travelling to the other side of the world by sending him an e-mail, explaining how your approach can help reduce the project's completion time. Should you wish to be more emphatic on how good your idea is, you can send him a PowerPoint presentation or even an animation. He will get it the next time he checks the mailbox.

From its humble beginning until today e-mail has been empowered by the widespread of public e-mail server access, the in-message use of markup language (HTML), improvements to file attachments and other subtle changes. However, such a fast growing technology with huge massification should have received more attention in what concerns security. Internet and e-mail were designed in a context where transmissions would occur between well known and trustable users, which facilitated communications and growth, with no content impediments, censorship or security concerns. But as more homes got connected and Internet access points popped a bit all over, the ethic rules no longer applied.

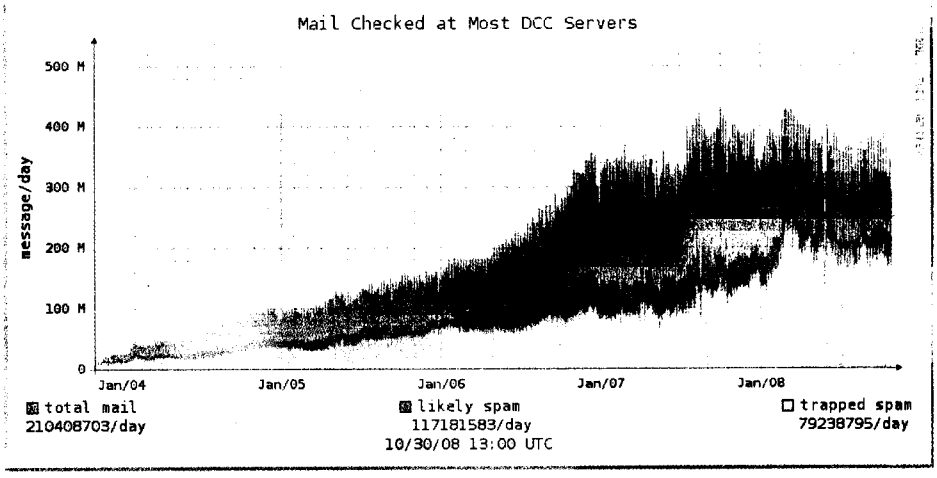
Some unethical people saw the profitable side of the lack of restrictions: sending one e-mail or ten thousand costs, money-wise, nearly the same, and with the appropriate tools, it can be done automatically. The reason why spam exists is simply because it makes people rich — many spammers have become millionaires [50] — thanks to the gullibility of e-mail users. Every time one single person has a positive response to spam by clicking on links, or ordering pharmaceuticals, that person is giving spammers a reason to spam even more. It is not simple to track down every spammer and prosecute them because of the same "no rules" ideology we mentioned in the paragraph above: the lack of security allows for anyone to hide their real identity, thus incapacitating authorities in seizing down these individuals. This particular characteristic, easy anonymity, has been very well portrayed still at the early years of the Internet *boom*, by Peter Steiner, with his *The New Yorker* cartoon (Figure 1.1) [78].

E-mail has been the biggest spam vector — according to 2006 spam statistics [28],

Figure 1.1: Peter Steiner's cartoon

40% of all e-mail was considered spam, totaling an impressive 12.4 billion spam sent in 2006. In February 2007, 90 billion spam were estimated to have travelled the Web, compared to the 55 billion of June 2006 [7][3]— Figure 1.2 depicts the last

Figure 1.2: The last 5 years of e-mails tracked down at the DCC



5 years of total e-mails sent (blue), the ones that would probably be spam but where not detected (red) and spam alone (pink), collected by the Distributed Checksum Clearinghouse[4] . Spam became so annoying and ubiquitous it has to be filtered out in order keep e-mail as a reasonably productive tool — corporations in the developed

[3]Some websites keep track of daily, weekly, monthly and yearly spam numbers. Visit `Spamcop.net` and `Distributed Checksum Clearinghouse Graphs` for some graphs with up-to-date raw statistics

[4]Visit the `Distributed Checksum Clearinghouse` homepage.

countries suffer financial and productivity losses while developing countries, where Internet access is restricted by bandwidth limitations, strive to keep spam out of their network's servers but often end up with a blocked connection to the Internet [69, 32]. Adding up to this scenario, is the growing association between spam and malware distribution, worm spreading and all their associated problems, which have helped infect computers all over the world [49].

Every creational process, be it writing a poem, painting or building something, requires inspiration. A source for ideas can be of any nature, even pre-existing work, but the natural world around us has been providing a huge help for quite a long time. Natural systems have had millions of years to select the best approaches to a variety of situations: evolution, as exemplified by Darwin's revelations about the Galapagos mockingbird's differences, is the process Nature has to allow the most well adapted solution to endure. So, if in need of inspiration, why not turn to the systems that have been — and still are being — perfected throughout the ages? Spider silk provided ideas for bullet proof vests, architecture and engineering look at animals and colonies for structures and mechanics, Newton developed his thoughts on physics by observing the natural occurrences surrounding him. Likewise, computer science has borrowed many ideas from Biology, fetching inspiration in the biological processes and functions — artificial networks and evolutionary algorithms stand as good examples of such interdisciplinary thinking. The relation goes both ways, since engineering and computers have been used to achieve greater knowledge of the biological systems [22].

Natural systems have some key characteristics that make them adequate inspiration sources, often providing good grounds for computational metaphors — where systems are designed in such a way as to incorporate high level representations of a biological system's component or function. The architecture, functionality, mechanism and organization of these naturally occurring systems where deemed important for the development of metaphors [58]. In what concerns the Human immune system, de Castro [22] has pointed out several of its characteristics that are very appealing to computer science: the pattern recognition, self identity, autonomy, anomaly detection, fault tolerance, robustness, self-organization and the integration with other systems are excellent qualities that any developer would like his system to have.

1.2 Motivation and Objectives

With this work we will be merging biological concepts and computer science, by working to implement an immune system-inspired spam filter and assessing its performance.

What attracted us into this project was the appealing idea of a system capable of recognizing what it is meant to protect, and take action towards 'attackers', dynamically adjusting itself as the subject changes throughout time — a bit like a 'lifetime bodyguard'; when the subject is an infant, the bodyguard has to deter specific menaces which are certainly different from those of when the protectee grows up and becomes a man.

The difference between our filter and the commonly used ones is the need to focus on an individual knowledge of the user. The spam filtering solutions commonly used provide results based on a general consensus of what spam is, and are able to return good results but fail the most at the user perception level of spam, since a certain e-mail (such as an invitation) may be seen as unwanted to a large group of people but to some other people it is seen as a very desirable e-mail. As we mentioned before, the idea of a software that 'knows' if a message is spam or not **according to the user's historical behaviour** and can mold that notion as time goes by finds inspiration in the Human immune system, and so we chose to implement some of its properties — recognition of self, non-self and long term adaptability to changes in self.

1.3 Thesis Layout

In this introduction we have explained the present technological context and why aspects such as security, productivity and economics are related with spam. We have also addressed our motivation and main objectives.

Next, we will be looking at the spam problem in more detail in Chapter 2, divided into five main parts: Section 2.1 explains what spam is, how it can be defined, its origins and also, most importantly, describes how spam is a much greater problem than just a few unsolicited e-mails in our inboxes (Subsection 2.1.3). Legal ways to fight spam are described in Section 2.2 along with an explanation on which anti-spam methods exist today and how they work, followed by Section 2.4 were we

enter the function and application of Artificial Immune Systems in the fight against spam. Section 2.3 will introduce and explain some of the main aspects of the Human immune system — essential in our lives and a source of inspiration for computational systems, such as spam filters like the one we have developed. Some of its important features are brought to light in a rather simple, but hopefully clarifying way. To conclude the chapter, the theory that stands behind our project is presented and explained in Section 2.5.

Throughout Chapter 3 we expose the implementation solutions we applied: a benchmarking tool based on one of the filtering techniques available (its description can be found in Section 2.2.2) and our own tool, in Sections 3.2 and 3.3 respectively.

Results from running our program and the benchmarker can be found in Chapter 4. Our final thoughts regarding the results obtained, the performance and possible evolution are available in Chapter 5 and 5.1.

Lastly, in the Appendices, we show two examples of configuration files, one for the optimizer (Appendix A) and another for the TAT simulator (Appendix B).

Chapter 2

State of the Art

Spam is a growing threat to a sound and productive e-mail system worldwide. Defining spam is not an easy task, for "one man's spam is another man's ham" and classifying it correctly is also equally or even more difficult.

2.1 Spam

Opening an inbox, nowadays, it is not uncommon to see e-mails written with capital letters (ROLLEX), numbers instead of letters (v1AGR4) or even a mix of letters and numbers without apparent meaning. More recently, quotes from books, texts, articles or other sources can be found in spam. Those are evolutionary traits that spammers came up with so their messages may succeed in escaping spam filters.

It is a common procedure to start a new endeavour by defining or identifying the subject under study. In our case, since we aim to target a certain type of e-mail, it would be not only appropriate but useful to try to clarify what spam really is. However, while it is easy for each individual to pin-point a spam e-mail in his/her inbox, it is not so easy to obtain a wider definition because one person may actually be interested in another one's spam.

2.1.1 Definition

Spam can be very different, varying from user to user, which is not very helpful when it comes to defining legal terms to fight it. In a rather simplistic way we

may say spam is unwanted e-mail but that would have to include different kinds of content: a breaking up e-mail is (let us assume) not the most of the desired subjects, neither is an e-mail with news that the project on which future work and payment were depending on will not be funded, but still, these two subjects are definitely not spam. So, if the conceptual definition is hard by itself, the legal one is not easier. Some of the well established definitions [55, 17] are:

- **unsolicited commercial e-mail** (UCE) — a typical UCE spam would be an e-mail containing advertisements sent without request or consent of the recipients.
- **unsolicited bulk e-mail** (UBE) — deals with e-mails sent to groups of recipients without their request or consent; UCE is included in UBE.
- and also **junk** e-mail — this is a bit redundant because we then have to ask ourselves how we define 'junk'.

Spamhaus[1], an international non-profit organization that aims to take down spammers, works with the UBE definition of spam, putting a lot of emphasis in the 'unsolicited **and** bulk' part. To them, from a technical point of view, an e-mail can only be considered spam when it satisfies both the lack of relevance of who the recipient is, since the message is sent to many potential recipients, and at the same time the recipient has not 'verifiably granted deliberate, explicit, and still-revocable permission' for the message to be sent [67].

For our work, we will be considering 'spam' as any unsolicited bulk e-mail but given that our aim is to develop a system capable of adjusting itself according to the content of the messages a certain person A receives and to person's A legitimate e-mail, we do need the more redundant notion of spam as any unsolicited e-mail in person's A inbox that A would filter out as 'spam'.

2.1.2 History

The origins of the term 'spam' are unprecise. SPAM is a type of processed meat by Hormel Foods Corporation, a trademark for Hormel's Spiced Ham (full name being "SPAM Luncheon Meat"). The word 'spam' was repeatedly used in Monty Python's

[1]See: `http://www.spamhaus.org`

sketch *Viking Spam*[2,3], causing noise and disturbing other conversations; flooding IRC chat rooms was very disturbing and was named 'spam' — most likely after the sketch — being the most

Figure 2.1: Hormel Spiced Ham

probable origin for the term [55, 86]. This way we can also claim the term 'ham' to have been associated with legitimate e-mails just like ham is used for *real* meat and SPAM is not.

This is also the probable reason of why the term 'ham' was coined for legitimate e-mail, in a similar relationship as SPAM and real ham.

Today's spam is not much different, in content, from what it was in its origins. The majority of spam received consists of some sort of marketing disguised under different letters or word combinations, which are the consequence of its evolution. The first e-mail to be considered spam was a message someone distributed at Digital Equipment Corporation in 1978. It consisted of an advertisement for the DEC's new systems and had originally more than nine pages of recipients' addresses — however, due to limitations in the program the spammer used, no more than 320 recipients actually received the message [86].

A huge discussion arose on Usenet on how legit such action was and for a reasonable time there was no consensus. As time went by and more and more spam messages where sent, work time started to be consumed with getting mailboxes rid of the spam plague and productivity decreased. Over the years corporations began to lose large amounts of money due to spam, while spammers needed only but a few 'hits' in order to have profits [10].

[2]The sketch in YouTube: `http://www.youtube.com/watch?v=s_-pjudnV3w`

[3]More on the sketch: `http://en.wikipedia.org/wiki/Spam_(Monty_Python)`

2.1.3 How big is this problem?

Let us assume for an instance communications technology had stalled at the analogic copper cable used in telephone lines, so common until some years ago. A direct implication would be the narrower bandwidth available for Internet data transfer, and a much slower access. On such an environment, and at today's spam receiving rate, any user would be spending quite some time downloading nothing but spam. If such scenario may seem a bit old-fashioned, we would like to remind you that there are countries in the World, struggling to grow economically, and whose enterprises also depend on fast access to information, where ISP spam filtering services are below good, good anti-spam software is scarce and expensive, and connection to the Internet is in general only available through one single provider.

This low-tech picture was put into focus by the OECD's Task Force on Spam[4] in a 2006 report [69], where it can be seen how this kind of activity is far more harmful than one may feel at first glance, especially to countries with difficulties in accessing state of the art tech services. Given Internet's weight in the economical balance as to determining companies' and industries' success at their business, and the role such enterprises play in their home country's economic development, it is easy to establish a link between spam-clogged ISP's and loss of vital information, with severe economic damage.

So it is not just a matter of scouting out for legitimate mails in the junk box, or having the inbox full of spam, this problem goes as far as causing whole nation's "mail servers to break down or function at a sub-optimal level" [69].

For a good summary on this problem, A. Gilbert wrote an article on the OECD's document we mentioned, looking into the economic side-effects of spam in developing countries [32].

As a final note, and as it may be read further down in Section 2.2.1, the European Community is also trying to look into the future of spam, hoping to be able to safeguard mobile devices from this kind of threat, something which the law firm mentioned in Fontana's article [29] had a first hands experience with, thanks to their Blackberry devices.

[4]Homepage: `http://www.oecd-antispam.org/sommaire.php3`

2.2 Fighting back

From a global point of view spam is a rather silent menace, clogging up servers and e-mail boxes. Up close and personal, the damage is by far more noticeable, as we have been describing. But ever since spam acquired the 'massive' status, some people have been putting their efforts together to find a way to stop it, or minimize its effects on the Internet and its users. From 1994 to 1997, language analysis-based filters where quite efficient and useful. What happened after 1997 was what made spam and anti-spam 'grow' (as in, evolve) together until what he have today. The methods and tools employed today are covered in Section 2.2.2.

2.2.1 Laws

Anti-spam software certainly does help to control the menace but it will hardly stop it. Anti-spam legal aid appeared in the USA under the form of the 2003 *Controlling the Assault of Non-Solicited Pornography and Marketing* (CAN-SPAM[5]) Act[6] passed by the U.S. Senate on November 23rd, 2003, which states spam is illegal and punishable by law. The document addresses several issues such as the following:

- the requirement for prior consent from the recipient before sending commercial, marketing or sales related e-mail as well as informing that same recipient that his contact details are to be transferred or sold to another party;
- identifies illegal methods through which spam is sent, setting new references as to how an e-mail should be — for example, trying to conceal the true origin of an e-mail is illegal;
- sets legal responsibility on the mail-sending hosts (*i.e.*, the e-mail servers) for its user's actions;
- defines spam filter evasion techniques has illegal.

Anyone caught breaching the terms of CAN-SPAM act applies for a heavy fine or possibly even some time in jail — depending on several aspects, a spammer can be fined up to six million USD (around four million euros) and/or sentenced up

[5]The acronym is somewhat curious, alluding to the idea spam is out on the loose and must be stopped, just like processed meat should remain in its can.

[6]Available at `http://www.spamlaws.com/federal/108s877enrolled.pdf`

to five years in jail; despite the six million fine being the maximum fine applied, if the spam contains false or deceptive headers, then the spammer will be fined $250 USD (€172.4 EUR) multiplied by the total number of violations committed, being "each separately addressed unlawful message received by or addressed to such residents treated as a separate violation" [73] — if we consider it is not uncommon for spammers to send out one or more million spams at a time, that stands for a $25,000,000.00 (25 million) USD fine — a little more than €17.2 million euros.

The European Union also has Directives on protecting personal information, with special regard to preventing unsolicited commercial communications such as direct marketing via electronic messages, snail mail[7], SMS or other forms of publicity. Directive 2002/58/EC is the most recent document regarding this issue and is based on Directive 95/46/EC [26, 25]. Both documents refer to the need of privacy and protection of personal details, emphasising in the fact that even consented marketing must be in the 'opt-out' form — meaning a person must be given the choice to not be sent marketing material whether it is as e-mail, SMS, telefaxes or automated calling machines.

It is up to each Member State of the European Union to determine which sanctions to apply and how. Since there are at the moment 27 member States, it is quite reasonable to say that anti-spam laws are not so easy to understand in the EU rather than in the USA. Still, Governments on both sides of the Atlantic have been making real efforts towards reducing the spam flow and prosecuting individuals found violating the laws — which is not easy, given all the techniques spammers' use to avoid being traced and caught.

2.2.2 Filtering Techniques

Recent statistics show that U.S. corporations spend $8.9 billion USD (€6.14 billion EUR) *per annum* in anti-spam software alone, while individuals end up $255 million lighter - almost €176 million euros [55, 68]. The solutions available are diverse and based on different approaches, some of them are complementary, others simply work alone.

The first anti-spam software consisted of very basic e-mail analysis, looking for known spammers' e-mail addresses, phrases or words thought to exist only in spam

[7] *snail mail* is sometimes used in reference to ordinary, non-electronic mail; e-mail is sent and received within a matter of seconds, minutes at worst, while a letter usually takes at least one day.

and building lists of features which were distributed to users. But its simplicity also allowed for high false positives rates, for one single phrase in an e-mail would easily sentence it to the junk box forcing users to check their spam for legitimate mail - and lose even more time. As spammers developed ways to go around these filters, most became obsolete — as Terry Oda wrote, this is a co-evolutionary problem [55].

Several techniques have been developed and some have had more success than others. Web-based white and black lists 'tell' servers who is allowed to send e-mails and who is not - but be warned, should your e-mail address be wrongfully blacklisted it will not be easy to be unlisted!; more advanced language analysis filtering, server-side software, legal intervention... many are the ways in which the Internet community is working to stop spam.

For spam filters, the way information is extracted from e-mail is a very important step for successful classification. Headers may hold important meta-information such as sender's identification, mailing program, the server through which the e-mail was sent, e-mail encoding and much more. Other methods work by keeping lists of e-mail addresses and checking the incoming e-mail sender's address. Google's Gmail anti-spam filter for instance works not only with Google's own algorithms but also employs machine-learning and OCR (optical character recognition - useful to filter image spam) along with what they call 'community clicks' to filter spam: if the community reports an e-mail as spam, the system learns to recognize similar e-mails as spam and filters them out [34].

2.2.2.1 Language Analysis

One of the first methods to become available in the spam-fighting war were not real filters but instead primitive tools that used a simple language analysis approach to classify spam. E-mails were scanned looking for known senders, various different recipients (cross-posting), or phrases associated with spam (like "Buy Now!" and "Free Trial!"). E-mail clients, such as Outlook, had in their early versions filters based on the message analysis.

For a few years after 1994 these simple filters were all there was needed to fight spam. Their efficiency relied in the fact that spam could be easily identified by cross-checking e-mails with lists of junk mail senders, words or sentences that the world believed to only exist in spam. In fact, success rates as high as 80 percent

were achievable by filtering based on one single word, with a very low chance of missclassification. Almost as a part of the good things about these methods is that they were so simple to implement that any user could alter the lists to their flavor, implying (almost) constant maintenance [68, 86].

Other downsides are pretty much obvious: words like "toll free" would be enough to filter out a otherwise perfectly legitimate message — in the long run these events translated into false positives. And the fact that not everyone thought the same message was spam gave them another reason to check the junk box now and then. As spam grew, the lists with selected 'targets' also grew and became less manageable, and eventually when spammers developed new ways to go around filters, the effort to have them updated was too much and most of these early methods became obsolete.

2.2.2.2 Blacklisting

Blacklisting is a spam controlling system working at the Internet service provider (ISP) level. Its conception is quite simple yet powerful: a *real-time blackhole list* (or RBL) would maintain a list of known spammer networks and by providing that blacklist to the community/ISPs it would be possible to ignore e-mail traffic originating from blacklisted networks. If a network takes care of the problem that got it listed in the first place — *i.e.* gets rid of the spammer — then it may be removed from the listing. That will not always happen, since it is the RBL's administration who decides on such matters (and their mood may influence...).

One interesting form of RBL is the one where a distributed hash database is used. Since a hash is a numerical representation of each message, directly related to its content, it is unique. If a spammer sent a message to 4 servers, when the first two mail servers report the same message's hash to the database, it is flagged as suspected spam. By the time the third mail server is reporting the same hash (since it is the same message), the database tells the server to filter the message — this way any new servers will also filter the message. The *Mail-Abuse Prevention System* RBL[8], deployed in 1997 by Paul Vixie, was the first RBL, available to subscribers. *Spamhaus* also provides RBL's, divided in three different categories, whether it is a database of IP addresses of confirmed spam sources (*Spamhaus Block List* — SBL), a database of IP addresses of violated PCs infected with illegal exploits from other sources (*Exploits*

[8]See: `http://www.mail-abuse.com/`

Block List — XPL) or wether it is a database of IP addresses not meant to deliver non-authenticated SMTP (acronym for Simple Mail Transfer Protocol) e-mails (*Policy Block List* — PBL) [67].

2.2.2.3 Whitelisting

What if instead of a list with untrusted networks, we had a list of trusted networks/senders? Then we could accept mails from anyone in the list knowing it would not be spam. Spamarrest[9] and Habeas[10] offer this kind of service, known as whitelisting, which works at the recipient's mail server level. Quite simply, if person A sends an e-mail to person B, and is on his/her whitelist, then B will get the e-mail. One drawback to this solution is how easy an unknown sender can be blocked - the principle behind whitelisting is that everyone is blacklisted by default except if specifically named in the whitelist. The solution to get an e-mail to a recipient is rather tricky: if A is not on B's whitelist, then A receives an e-mail from B with a challenge - usually, a request to click a link. Should the response be according to the challenge, then A is whitelisted and the e-mail gets through to B. Whitelisting assumes that a human sending an e-mail is legitimate and trustworthy, while spammers do not have time to respond to challenges or do not even see the message, since the reply-to address is often fake.

Problems with this method involve mistakenly replying to every message sent to a mailing list (becoming a nuisance), the time consumed by the party required to answer the challenge and even the inappropriate match between the challenge and the responding individual — an audio challenge for deafs would most certainly be not adequate, 'common knowledge' questions may be to hard for some people, etc.

2.2.2.4 Heuristic Filtering

This technique works by applying a set of common-sense rules designed to match specific spam traits, ranging from content to e-mail construction, typical of spam. The difference between heuristic filtering and primitive language filters is that the former also identify ham, so an e-mail that looks similar to spam may still be correctly delivered if it can match a minimum number of rules. Before SpamAssassin[11]

[9] See: `http://www.spamarrest.com/`

[10] See: `http://www.habeas.com`

[11] See: `http://spamassassin.apache.org/`

even existed, another solution — Brightmail — used heuristic filtering. SpamAssassin's implementation of this method includes many different set of rules and no single characteristic condemns a message.

These filters have significant drawbacks. One is the fact that the rules must fit everyone's e-mail, meaning they can not be too "strict" otherwise very few people will have their e-mails correctly filtered. Secondly, and more serious, if the same set of rules was meant to be used by everyone, spammers could learn how their e-mails were being filtered out and devise a way to bypass the filter. Despite its inconveniences, heuristic filtering is still used today, as part of a set of other methods, as in SpamAssassin [86].

2.2.2.5 Collaborative Filtering

There are many forms of collaborative filtering, like the Gmail's "community clicks" we mentioned earlier in this Section, but it can also be in the form of lexical data containing descriptions of spam characteristics. In general, collaborative filtering takes advantage of a network of intelligence to identify spam; for example, within a group of trusted users, the misfortune of some who receives spam can be used as an "inoculation", building knowledge that will be passed to the others, who in turn will be able to filter against a specific type of spam.

Bill Yerazunis invented the concept of message inoculation, an antigen introduced into the filter which will prevent the user from receiving the original spam that created the antigen, as well as contextually similar spam. Classification groups, another collaborative filtering algorithm, relies on the knowledge of a group of trusted users to assist in a situation when a user's filter is not certain of an e-mail being spam or not. Problems rise due to the fact that spam is not the same for all, initially creating false positives due to some messages [86].

2.2.2.6 Bayesian Filters

Thomas Bayes was a British Presbyterian reverend and mathematician who lived in the 18th century (1702 – 1761) and from whom derives the term *Bayesian probability*, meaning "a notion of probability as something like a partial belief, rather than a frequency" [79]. This notion was first used to classify spam by Patrick Pantel and Dekang Lin in 'SpamCop: A Spam Classification & Organization Program' and also

by Mehran Sahami, Susan Dumais, David Heckerman and Eric Horvitz in their paper 'A Bayesian Approach to Filtering Junk E-mail', both published in 1998.

When a new message arrives in your inbox, you expect its content to be of some sort, depending on the subject or sender of the said message. When you read it, if you see it is spam, the next time you see a similar message you will be more prone not to read it and just toss it into the junk. Should this happen a few more times, your conviction that message is spam gets stronger. However, if for once a message arrives that despite 'looking' like spam it is not, instead of confidently marking it as spam, you would start trusting the message to be ham. Bayesian analysis basically tries to mimic this behavior, classifying messages according to some previously learned information about those messages. Equation 2.1 depicts the general Bayes' theorem.

$$p(c \mid \mathbf{x}) = \frac{p(\mathbf{x} \mid c) \cdot p(c)}{p(\mathbf{x})} \tag{2.1}$$

It works in the following way: $p(c \mid \mathbf{x})$ is what we want to know, the probability of category c given the feature vector x, *i.e.*, we want to know how likely it is for a given category to "produce" the feature vector x; $p(\mathbf{x} \mid c)$ is, similarly, the probability of feature vector x given category c, which is to say, the probability of an e-mail belonging to category c; $p(c)$ is the probability of category c, calculated with basis on the training set; and $p(x)$ is the probability of the occurrence of feature vector x, also in regard to the training set.

The efficiency of Bayesian based filters has become a study matter as seen in Gordon Cormack's and Thomas Lynam paper [21], Paul Graham's 'Better Bayesian Filtering' article [36], Harry Zhang's 'The Optimality of Naive Bayes' [87], Ian Androutsopoulos *et al* [6], just to name a few. Nonetheless, bayesian analysis still gathers some fans and is used together with other methods, providing some of the most accurate filtering results. One of the most relevant works in the bayesian filters field was published by Paul Graham in 2002 and was quickly implemented into most open-source spam filters, like SpamAssassin[12]or DSPAM[13] [68].

DSPAM, by Jonathan Zdziarski, is a spam filtering program based on many statistical techniques, aimed mostly at ISP's or large corporations. Also incorporated into the program where a series of research methods like message inoculation standard

[12]See: `http://spamassassin.apache.org/`; please note SpamAssassin employs a series of different methods and not just bayesian analysis.

[13]DSPAM homepage is at `http://dspam.nuclearelephant.com/`

(also included in Yerazunis' CRM114 filter), the bayesian noise reduction algorithm and chained tokens. Message inoculation is one of the algorithms belonging to collaborative filtering, while the bayesian noise reduction is designed to reduce features that may appear as noise and could difficult filtering. Chained tokens is a data processing algorithm that combines adjacent tokens (n-grams at the word level — see `N-grams` in Chapter 3) in order to make new ones and improve the quality of the data fed to filters [85] — or to the DSPAM itself, in this case.

SpamAssassin is also a very well known spam filter, that employs several approaches, including bayesian filtering. Users can check their incoming messages' checksum against special checksum databases — called spam clearinghouses; named Vipul's Razor[14], Pyzor[15] and Distributed Checksum Clearing house (or DCC, for short)[16] — or check the sender's DNS. SpamAssassin also provides blacklists, whitelists and over 700 test rules defined for English spam [64].

2.2.2.7 Other Machine Learning Approaches

Here we would like to refer to Hidden Markov Models (HMM) and k-nearest neighbour, for example. A lot more can be read in Blanzieri's [12], O'Brien and Vogel [54], Tretyakov's [71] and in Diacovo's [24] articles.

The HMM has some differences towards regular Markov models: in these last ones, each state is visible to the observer, making the transition probabilities the only parameters; in a hidden Markov model, the state is not directly visible, so the observer can only take notice of the output — each state has an *emission probability*, which is a probability distribution over the possible outputs. An analysis of a system called *lexicon tree HMM* is described in Diacovo's [24], where a HMM is used to deobfuscate e-mail text (*i.e.*, to change `10@ns` to `loans`, for example). What this method could bring to the battleground is a better pre-processing for e-mails before a filter analysis of any sort — in the same article there is reference to two HMM filtering models but given the fact they work with a dictionary (to obtain the non-obfuscated words), it was verified they produced less than good results — there are many strings in a legitimate e-mail which are not found in dictionaries (smilies, acronym's, URLs, etc.).

If one assumes to have an idea of *distance* between messages, so that two hams

[14] `http://razor.sf.net`

[15] `http://pyzor.sf.net`

[16] `http://rhyolite.com/anti-spam/dcc/`

would be *closer* to each other than one ham and one spam, then one may also use a feature vector from a message to determine how close that e-mail is to the training examples' feature vectors. Should there be more spam *neighbours*, the e-mail is classified as spam. This method is know as the k-nearest neighbour (k-NN). However, it has been reported [71] that this method does not yield good results.

2.2.2.8 Immune System Based Filters

Another different type of spam filters, where our work is included, concerns systems based on the Human immune system. The idea of using biological concepts, theories and strategies in other scientific fields is not new, and a detailed look in computer science reveals genetic algorithms, neural networks, ant algorithms or even virus detectors [44] as examples of biology concepts intertwined with computer science.

Being more popular within the field of Intrusion Detection Systems [46, 30], Artificial Immune Systems have also been applied to spam. In his technical report [13], Brownlee overviews different *computer paradigms* related to the AIS field, listing and separating into 5 categories all the different works: Negative Selection, Clonal Selection, Immune Network, Danger Signal and other approaches.

Abi-Haidar and Rocha [4] have developed a spam detection system based on a model of regulatory T cell dynamics, proposed by Carneiro *et al* [15]. The model is based on the population dynamics of the cells, relying on the interactions between three different types of cells to determine how to classify e-mail messages.

While some effort has been put forward in developing artificial immune based spam filters, the success obtained by non-biologically inspired methods — with a little help from the cultural barriers developed around the AIS subject in general[17] — have made it hard for successful projects to grow outside research and establish themselves as feasible commercial alternatives[18].

[17] In Somayaji's report [65], one may read that "[...]it is generally recognized that biological terms in new papers are signs of poor quality research".

[18] Not exactly in the sense of replacing current software but more as complementing it, as noted in [55, 82].

2.3 Immune System

The human body is a very well adapted system and an attractive one for other organisms to live in: we are able to maintain a constant temperature as well as maintain different substance concentrations and produce compounds that are also the substrate from which many organisms live by. So not so often we see ourselves infected by some virus or bacteria, far too simple organisms that require our cellular and molecular mechanisms to survive, sometimes destroying our cells in the process.

Organisms that cause disease or infection, are called *pathogens*. They are constituted by the same amino-acids present in our body and in other organisms[19] but in unique re-arrangements; amino-acids are proteins' building blocks [52]. Proteins and other molecules that can be identified by the immune system are called *antigens*. Antigens can originate in pathogens or in cells from our body, like cells that reach the end of their lives and die.

Being able to tell which antigens belong to the organism, building up the notion of *self*, from those which do not, and are related to the *non-self*, is paramount for a proper immune function. Such knowledge is acquainted still in the maturation organs, before the cell is released into the blood stream. As organisms, we go through many developmental stages with several molecular configurations — puberty for instance is a stage of extreme changes. Such changes may imply producing new proteins, new hormones or abandoning others, which in turn will be reflected in the immune system's ability to recognize and correctly identify the new self.

2.3.1 Innate and Adaptive Immunity

The immune system is comprised, amongst others, by macrophages, granulocytes (basophils, eosinophils and others), T cells, their sub-populations, and B cells. Some cells originate others and most have functions that are related with both innate and adaptive immune responses. Innate immunity is present at birth time and is constituted by defenses that evolved along with Humans whereas adaptive immunity acquires new defenses throughout the organism's whole life, depending of its experiences — if a young kid catches measles, he will never catch it again and his immune system will be able to recognize the virus; if he is never exposed to it, his immune system will have never produced anti-measles antibodies and the individual

[19] There are only a total of 22 known amino-acids; Zafer lists the standard 20 [83].

would catch the disease.

Specificity is a key difference between innate and adaptive immunity: innate responses can be considered as general, since they target any kind of infectious agent. On the other hand, adaptive responses are only deployed to target a specific microorganism. Despite of their different and almost opposing characteristics, innate and adaptive responses are not mutually exclusive — quite the contrary, many times they complement each other. The innate immunity is the first type of reaction to a pathogen — the front line of host defense — giving our body some time to attempt to defend itself and prepare the adaptive response. Without the innate response, the adaptive response would still step into action, but it would take a few days, leaving us helpless in the meantime [42, 20].

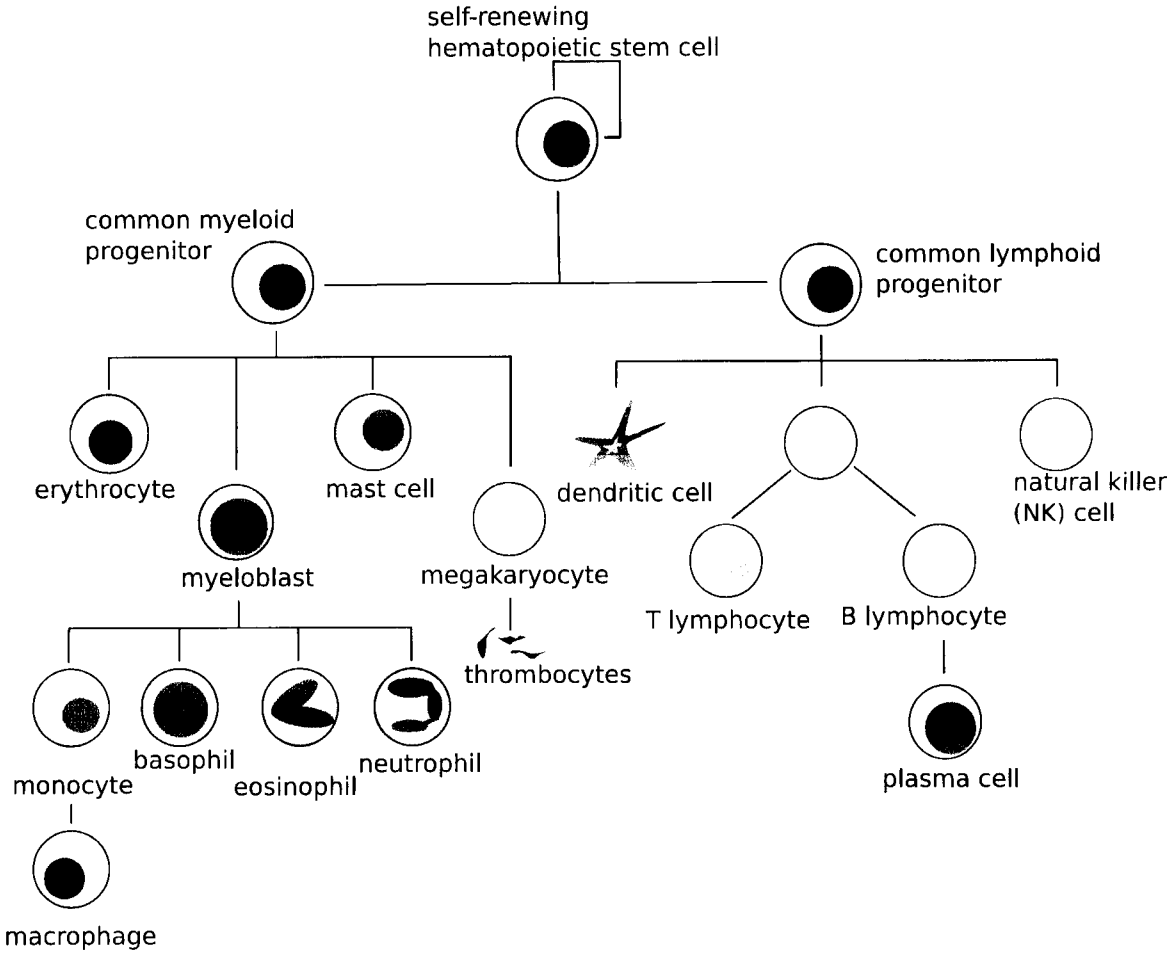
Table 2.1: Comparison between innate and adaptive immunity [33]

	Immunity	
	Innate	**Adaptive**
specificity	limited and fixed	highly diverse, improves during the course of immune response
antigen	not required	required; must be processed and recognized
activated by	antigen's chemical properties	antigen receptor binding
memory	no; response to repeated infection is identical	yes; response to repeated infection is much more rapid
who? (examples)	skin, mucous membranes, temperature, low pH, tissue macrophages, monocytes	T and B lymphocytes, macrophages

Adaptive responses are specific because they rely on the affinity between the antigen and an immune cell receptor to trigger a response. The implication of such mechanism is that the cell must possess the corresponding receptor in order to bind to the antigen and does not bind to any other. Considering the fact our organism also produces antigens, how can the immune system create cells that will bind to 'foreign' antigens?

All blood cells descend from a self-renewing *hematopoietic stem cell* (HSC) located in the bone marrow, in a process called *hematopoiesis*, illustrated in Figure 2.2.

Figure 2.2: A simple hematopoiesis diagram (adapted from Janeway [38])



Different cells end up having different developments: while erythrocytes and mast cells are released into the blood stream to carry oxygen and produce platelets respectively, monocytes will only originate macrophages upon reaching the tissues, which only some will do. B cells mature in the bone marrow and T cells in the thymus — hence their names: **B** for **b**one, **T** for **t**hymus. T cells will be most likely found in lymph nodes, but they also circulate in the blood, alongside with B cells in what is called a *naïve state* - meaning they have matured but not yet encountered their specific antigen [42].

When a monocyte or macrophage finds a pathogen, they will take it up and digest it, destroying it — the whole process is called *phagocytosis*; if an immature dendritic cell phagocytes a bacteria, special enzymes inside the dendritic cell will destroy the

bacteria, 'splitting' it up in antigens that will be transported to the cell's membrane. Dendritic cells then mature into highly effective antigen presenting cell (APC), circulating in the blood vessels with some parts of the digested microorganism in its surface (it is said the cell is 'presenting' the antigen), stimulating other cells — like naïve B and T cells [42].

Immature dendritic cells are classified as being part of the innate immune system, however, upon maturation, they present antigens to cells associated with the adaptive immune system; macrophages and monocytes may also become APCs as well, so it is safe to say that an innate response precedes an adaptive response.

APCs travel from the infection site to the secondary lymphoyd organs, where most of the naïve B and T cells are located. Activated B cells will proliferate and differentiate into antibody-producing plasma cells. The antibodies will bind to the pathogen and help eliminate it using different strategies. T cells have other functions: when activated, they may differentiate into *cytotoxic T cells* (T_C), which kill cells infected with viruses, or they may differentiate into another class of T cells, identified as *helper T cells* (T_H), whose function is to activate other cells like B cells and macrophages [42].

2.3.2 Tolerance and Autoimmunity

In the late 50's, Sir Frank MacFarlane Burnet published a paper [14] explaining his view of the immune system's tolerance mechanism, in an attempt to understand why each individual only produces antibodies in regard to the antigens which he is exposed to (given that antibodies can be induced in response to almost any antigen):

> *How can an immunized animal recognize the difference between an injected material like insulin or serum albumin from another species and its own corresponding substance?*

In the concept he developed, the body was able to generate many cells capable of producing a wide range of antibodies, each with a different specificity. By displaying a membrane-bound form of the antibody to act as antigen-receptor, those cells would be apt to bind to an antigen, upon which the cell would be activated, dividing and producing identical progeny — in their whole known as 'clone'. The clonal cells are capable of secreting antibodies with the same specificity as the surface receptor that was initially triggered.

Besides the **clonal selection theory**, Burnet also suggested it would be possible for the antibody producing cells to have antigen receptors that would bind to self ubiquitous antigens, thus creating a possible reaction to antigens belonging to the organism. To avoid it, those cells would be exposed to self antigens in pre-natal stage; however, the signal received would not be for clonal expansion but instead to enter programmed cell death — *apoptosis*. This is known as **clonal deletion**, and is a form of *negative selection*.

Today, we know that the gene encoding antigen receptors in lymphocytes (T cells' receptor — *TCR* — and B cells' membrane immunoglobulin — *mIg*) suffers many somatic mutations and random segment assortment, hence providing a wide variety of unique antigen receptors. We also know these mutations and random events occur in lymphocyte precursors and also that clonal selection and deletion does occur, but not only in pre-natal stage; in fact, the organism is continuously producing and selecting immune cells, eliminating the auto-reactive ones (capable of reaction to self), thus maintaining **self-tolerance** [43].

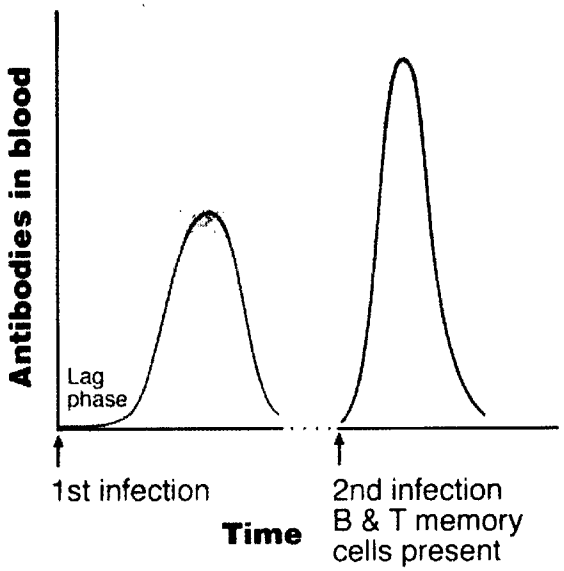
Should an auto-reactive lymphocyte become active, it will perform just like if it had been activated by a pathogenic antigen, *i.e.* it will destroy or promote destruction of the cells to which the antigen belongs. Given that its receptor binds to self antigens, these lymphocytes will be killing cells belonging to the organism, and most likely damaging tissues — the individual under such condition is said to suffer from an *autoimmune disease*, characterized by lack of self-tolerance.

2.3.3 Memory

While immunological memory is not *au par* with negative selection regarding proper functioning of the immune system, it plays a crucial role when it comes to 'repeated offenders'.

Activated B or T cells are capable of proliferating into other similar B or T cells or they may produce one special kind of cell, called *memory cell*. This cell will retain the antigen receptor genes without mutating them, and will last longer that regular B or T cells. Memory cells circulating are more sensitive to the antigen which triggered their maturation from B or T cells, and provide a stronger and faster immune response to that same antigen. This discovery allowed for the development of immunization.

Figure 2.3: Graph of primary and secondary responses vs. time, to two successive infections [72]



When an antigen is first found in the organism, the response is not immediate. Represented as a red line in Figure 2.3, the first immune response suffers a small delay between the presence of the antigen and the rise in antibody concentration. After antigen and pathogen removal from the organism, antibody concentration decreases.

Upon a second infection by the same agent, seen as a blue line in the same Figure, it is noticeable that the presence of memory cells helps to mount a faster and stronger second response: the lag phase is now extremely reduced (or nonexistent), the antibody concentration rapidly increases to noticeable higher values than before but antibodies will also display a higher affinity for the antigen [42].

2.4 AIS Applications

Artificial Immune Systems (AIS) are, as defined by de Castro [22]:

> *(...) adaptive systems, inspired by theoretical immunology and observed immune functions, principles and models, which are applied to problem solving.*

Some of the "problem solving" performed by AIS could be seen as rather simple pattern recognition, since that is one of the characteristics of the Human immune system: AIS had extended into areas such as host intrusion detection (HID), network intrusion detection (NID), malware and fraud detection (MD and FD respectively), data analysis, scheduling and agent-based systems.

To look at the immune system as an information processing unit [11, 30] is not done intuitively. To understand this we start by looking at T and B cells as elements containing attributes (their antigen receptors) that inspect other elements (the pathogens). This way we can begin to grasp the intricate information flow within the biological intrusion detection process. The difficulty stands not in adding this "new vision" to the Human immune system but instead resides in modelling it appropriately.

Examples of works done in the different fields of the AIS ar the ones by Forrest in HID [30], Kephart with MD [44], Kim and Bentley's in the area of NID [46] and also Mario Antunes' with TAT-based network intrusion detection system (NIDS) [8]. Richard E. Overill's article on anomaly detection [56] details more information on these and other works.

Some AIS algorithms model specific aspects of the immune system, able to be used independently or as different parts integrating a more complex system, and can be divided according to the component and biological theory in which they are based. *Bone marrow models* and *thymus models* constitute the two groups of models after immune system components, whereas the preferred theories for inspiration are *the clonal selection* and *immune networks* [22].

The *bone marrow* model is characterized by a set of attribute strings randomly generated, taking pieces from a pre-existing alphabet (Hamming space-shape type [40]) or by random permutation of as many elements as the string's length (Integer space-shape type) [22]. For the latter, let us assume the model will be generating strings of size 7; they could be created by picking randomly and with repetitions any 7 characters of the working alphabet. Other implementations of bone marrow models may be as complex as using gene libraries as a source for the elements to be created, obtaining rearrangements or even evolutions of the initial forms [22].

Thymus models, like Forrest *et al Negative Selection algorithm* [31], mimic the creation and maturation of T cells. There are agents responsible for target detection and adequate response, which can be seen under two perspectives: positive selection and negative selection. In the former, the detectors are created to match a known

universe of attribute strings, *i.e.*, the cells will look for known patterns; negative selection works with the unknown universe having by basis the knowledge of *self* — the pattern recognition is performed on the complement of the known attributes set [22].

Other AIS are based on clonal selection algorithms. Clonal selection, as better explained in Section 2.3.2, is a process involving cellular proliferation and hypermutation of the antigenic receptors. CLONALG is a clonal selection algorithm [23] used first for pattern recognition but later adapted to optimizing multi-modal functions. Comparatively to evolutionary algorithms, CLONALG takes into account the cell's affinity before defining the proliferation and mutation rates that each member of the population will adjust to [22].

What distinguishes the clonal selection theory from the immune network theory is that the former assummes the immune system is originally at rest and stimulation only occurs when a foreign antigen invades the system while the latter considers a system that has dynamic activity at all times (even when no invading antigen is present). The dynamics of the immune network theory stems from an assumed capability the immune system elements have to be able to recognize and be recognized — the theory suggests certain portions of cells' receptors (idiotopes) allow for such recognition. AIS models belonging to this group have already been successfully employed in solving complex problems like autonomous navigation, automatic control and optimization [22].

2.4.1 A spam filtering model

As mentioned in Section 2.2.2.8, the work of Abi-Haidar and Rocha [4] was developed based on a model of population dynamics of cells of the immune system. In it, three cell types are taken into consideration: effector T cells, regulatory T cells and APCs. It is suggested that T cells proliferate depending on co-localization of effector and regulatory T cells as they bind with an APC.

Three rules determine the population dynamics, as described in their article [4], so that for each antigen presented by an APC:

1. should one or two effector cells bind to an antigen, they proliferate at a fixed rate;

2. the same for regulatory cells, but instead of proliferating, they remain in the

population;

3. if a regulatory cell binds along with an effector one to the same antigen, then the regulatory cell will proliferate at a fixed rate while the effector will remain in population (no proliferation).

The dynamics of the Cross-Regulation model allow for the existence of two stable situations in the same system, and domination flowing from one to the other: a) — effector and regulatory T cells co-exist and are able to identify harmless antigens; or b) — the regulatory cells slowly disappear, and harmful antigens are identified. The results obtained with this method are compared with the results obtained from other methods, including one version of a Naïve-Bayes classifier, and it is shown that the Cross-Regulation Model works just as well or even better than the other methods.

2.5 Tunable Activation Threshold

In Section 2.3.2 we have introduced auto-reactive lymphocytes and the concepts of *self-tolerance* and *autoimmunity*. It was also explained that negative selection helps to eliminate auto-reactive cells from the mature repertoire of immune system cells.

Nonetheless, mature auto-reactive lymphocytes have been identified in normal healthy individuals [60], meaning negative selection does not eliminate all auto-reactive cells. The fact such findings where obtained from **normal healthy** individuals — implying non-corrupted self-tolerance — helps to extrapolate the existence of an activation control of some kind preventing clonal proliferation of the auto-reactive cells. An hypothesis for such a control mechanism was suggested by Zvi Grossman and William Paul, in 1992 [37], in which the repetitive encounters of a cell and its corresponding antigen would eventually make the cell enter a state of *unresponsiveness* — referred to as **anergy** by immunologists, especially when it reflects reduced proliferative responses. In the case of a cell whose receptor matched self antigens (from the organism), such repetitive encounters would happen quite early in the cell's life, leaving the cell anergic. This mechanism was called the Tunable Activation Threshold theory.

Three propositions constitute the working basis for the Tunable Activation Threshold theory [37]:

- **Proposition 1:** *For a stimulus to cause cell activation, the excitation level must*

exceed an activation threshold

- **Proposition 2:** *A lymphocyte is inherently capable of participating in supra- and sub-threshold immune responses*

- **Proposition 3:** *When engaged in persistent sub-threshold interactions, cells are protected against chance activation*

Cells enter activation state only in cases where the excitation level exceeds that of the activation threshold, which is tuned according to exposure to stimulus (antigen binding strength): a cell continuously being stimulated has a very high activation threshold, so high it will eventually be always above the excitation level; on the other hand, cells that rarely bind to their antigens have a lower threshold, more easily surpassed by spikes in excitation levels (rare but strong binding to antigens).

A mathematic model of the TAT theory was developed by Carneiro *et al* in 2005 [16], upon which we have based the development of our work. The difference between the TAT as it was introduced by Grossman and colleagues [37] and the model studied by Carneiro *et al* lies in the fact that the latter restricted adaptation only to the activation-dependent proliferative response, discarding the broader hypothesis by Grossman and colleagues which considered tuning of APCs and also the two-way effect of the relationship between suppression levels and tuning [16].

2.5.1 How TAT Works

In a T cell, two enzymes play a determinant role. Kinase phosphorilates molecules that trigger the cell activation, and phosphatase dephosphorilates them, returning the cell to an inactive state. The TAT theory posits that the relationship between the two enzymes' activity levels is similar to a 'molecular switch', turning the cell 'on' (activation) when the kinase activity overcomes that of the phosphatase, and 'off' if the phosphatase activity is higher.

Two differential equations have been used to represent these dynamics [16]:

$$\frac{dK}{dt} = r_K(K_0(1+\sigma) - K) \tag{2.2}$$

$$\frac{dP}{dt} = r_P(P_0(1+\sigma) - P) \tag{2.3}$$

K and P stand for kinase and phosphatase activity, respectively, r_K and r_P are the turnover rates[20] for kinase and phosphatase, K_0 and P_0 represent the basal steady state kinase and phosphatase activity. σ is the magnitude of the stimulus to both enzymes' production rates. Its value will be σ when the cell is conjugated with an APC and nil if the cell is free.

An additional simplification was made by Carneiro [16]: it is assumed that the turnover rate of the kinase activity is very fast, when compared to the conjugate[21] dissociation rate ($r_K >> d$) and to the phosphatase turnover rate ($r_K >> r_P$).

By establishing that the turnover rate of the kinase is higher than that of the phosphatase and that for any given stimuli, the kinase's basal steady state activity is lower than that of the phosphatases, *i.e.* $P_0 >> K_0$, adaptive properties were verified; moreover, with the provided conditions, the adapter may be momentarily switched on, eventually switching off if the stimulus continued.

Summing up:

- both kinase and phosphatase are exposed to the same stimulus (σ);
- phosphatase's basal steady state is higher than kinase's ($P_0 >> K_0$);
- kinase's turnover rate is higher than phosphatase's ($r_K >> r_P$);
- lymphocyte activation only occurs if kinase activity is higher than phosphatase activity, after dissociation from the APC ($K >> P$);
- if the stimulus persists, phosphatase activity will exceed kinase's and the cell will become inactive;

In such a model, the stimulation history of a T cell is reflected in the kinase and phosphatase activity values at any given time, reason why the frequency of interactions between APCs and individual T-cells (*i.e.* how frequent a T cell is stimulated) modulates the T-cells activation threshold.

[20]The turnover number of an enzyme is equivalent to the number of substrate molecules converted to product in a given unit of time on a single enzyme molecule when it is saturated with substrate [53].

[21]The complex resulting of an enzyme binding to its substrate.

2.5.2 From Immunology to Computer Science and back

Table 2.2: Immunological metaphor

Immunology	TAT-based spam filter
Self	Legit (ham) e-mail
Non-self	Spam e-mail
T cell	Detector that identifies a pattern
T cell receptor (TCR)	Pattern (string) recognized by the detector
Antigen or peptide	Small data (string) obtained from input
Antigen Presenting Cell (APC)	"Container" object for antigens
T cell repertoire	All available detectors
Antigen or peptide concentration	Peptide frequency in the APC
Phagocytosis	Data pre-processing
T cell — APC affinity	Distance between TCR's pattern and the peptide's pattern

Integrating concepts from one field of knowledge into another is never an easy task, it generally requires some sort of adaptation. Examples can be found in Biology of information transmission and storage (DNA for example) that is in a way far more complex than computer's binary language.

In Table 2.2, we present the metaphor generally used to map immunological entities to their computer counterpart (TAT-based spam filter — TBSF). The denomination throughout this thesis and in the actual code was kept as close as possible to Biology, in an effort to facilitate the approximation of both fields as well as to establish a comfortable conceptual framework.

When Computer Science, more specifically, the field of Artificial Immune Systems, takes its inspiration from Biology, it can return results. This is an important contri-

bution for despite of all advances in laboratorial techniques and technology, some aspects of Biology are still difficult to measure and observe. Through what can be seen as a 'symbiotic' relation, results obtained in the AIS field can sometimes be "translated" into the biological realm, helping to understand the information processing aspects of the biochemical processes and cellular mechanics within. Such a relation provides a clear generalization and applicability of biological theories, in this case of the TAT theory.

Chapter 3

Implementation

Our work's implementation consisted on two main parts: one where a Naïve-Bayes algorithm was implemented as to provide benchmarking comparison point based on well proven and known techniques and a second in which the TAT algorithm was coded, tweaked and evaluated.

Both programs where written in Java, with a small exception (see Section 3.1). We will start this chapter by 'dissecting' an e-mail, to find it is composed of two main parts:

- **header** — in the header of an e-mail we may find information such as who the sender is, to whom the message was sent, what time it was sent at, which server or even which e-mail program sent it, the type of encoding and a lot more; this information is called the message's meta-information.
- **body** — here is where the message content lies; the body of a message is where the actual message data travels.

E-mails may also carry files with them, called 'attachments'. These files are sent along with the message, but their details are kept separated from the e-mail's text content.

Spammers will do anything to make their headers look perfectly unsuspicious, thus guaranteeing the message's arrival at the recipient, but will have to keep the body faithful to what they want to be read, *i.e.* their marketing message. For this reason we have opted to restrict our analysis only to the e-mails' body. In spam filtering it actually is a choice whether to include the message as a whole or to select some

features, as noted by Oda [55] and Pantel [57].

Most spam filters used by ISP's and mail server's look for words, known offenders or e-mail reported as spam by the community. Looking at each e-mail we send, there are common traits that allow us to identify its writer. Those traits can be a type of greeting, a set of words we use often or an expression we like. Our e-mail can even be distinguished by its contents' subject, being closely related to our personal interests, work life, etc., meaning the e-mails we send and receive have characteristics of our personality. As a quick example think of how easy it is to identify a piece of writing by someone close to you just by its content or writing style.

This fact supports our choice of analyzing characters instead of whole words; by looking at the characters individually not only we can extract more information not possible when analyzing each word one by one but the system also becomes less susceptible to written errors. By using n-grams, the e-mail's body is split into small tokens, discarding all characters except numbers and letters (the list of ASCII codes and corresponding characters that where extracted for analysis can be found in Table 3.3). Before entering into details about how both the benchmark and our filter were programmed, let us first look at what n-grams are and how they are usefull.

3.0.3 N-grams

In fields like natural language processing, n-grams are a good analysis tool. Systems working with n-gram matching take advantage of the fact n-grams minimize the effect of the errors present in the text, due to the slicing of the strings into smaller parts (the errors would only affect a limited number of these) [18]. An n-gram is an n subsequence of another sequence, usually a sentence, in one of 2 levels: word or character level; a sentence with length L will generate $L - n + 1$ n-sized n-grams [77].

N-grams have been used in language processing for a long time [44] with good results: with them, we can build language dictionaries to identify what language a text is written in or what subject it is about [18]. More interestingly, in Abou-Assaleh's work [5] n-grams where used to successfully identify never before seen malicious code from several code files. Bi-grams, or 2-sized n-grams, at word level are a specific type of tokenization called chained tokens, which can be used to complement individual tokens [86].

Table 3.1: N-grams at word and character level

THE QUICK BROWN FOX JUMPED OVER THE LAZY DOG	

all the possible 5 words n-grams	`The quick brown fox jumped`
	`quick brown fox jumped over`
	`brown fox jumped over the`
	`fox jumped over the lazy`
	`jumped over the lazy dog`

some 9 character n-grams (spaces replaced with '_')	`The_quick_brown fox...`
	`The_quick_brown_fox...`
	`The_quick_brown_fox...`
	`The_quick_brown_fox...`
	`The quick_brown_fox...`

3.0.4 Datasets

Testing a spam filter requires the use of collections of both spam and ham e-mail (called *corpora*). The Internet can provide some *corpora*, but all with different characteristics. Oda [55] has described the ideal corpus of e-mail as being one that:

- is publicly available,
- has spam and ham e-mails,
- is sorted,
- should be as recent as possible and
- should remain unchanged as much as possible.

Preferably, it should have been collected by one or several e-mail addresses that got both spam and ham.

For our tests, we wanted to use a dataset faithful to an individual's writing characteristics, *i.e.*, that kept the typical chain of sent and replied e-mails, the writing style and other writing traits which can easily identify the writer. This constraint had to be imposed in order to maintain the proximity with the immune system: its cells are intrinsically related to the organism where they live, therefore we needed to create

"cell populations" that were intrinsically related to one single e-mail user. To be eligible for testing, any dataset needed to be as close to the original e-mails as possible, complete with headers and no message tokenization/encryption. Preferably, it would have both spam and ham collected by the same real recipient.

Table 3.2: Publicly available corpora

Corpus	Where
Ling-Spam	`http://www.aueb.gr/users/ion/data/lingspam_public.tar.gz`
Enron-Spam Datasets	`http://www.aueb.gr/users/ion/data/enron-spam/`
PU corpora	`http://www.aueb.gr/users/ion/data/PU123ACorpora.tar.gz`
2005/06 Spam Track	`http://trec.nist.gov/data/spam.html`
SpamAssassin	`http://spamassassin.apache.org/publiccorpus/`

Table 3.2 has some of the available public *corpora*, and their locations — an extract of Blanzieri's list in his report [12] with updated download locations. Apart from these datasets, we have also collected our own personal e-mails.

The Ling-Spam and PU *corpora* were not adequate for our tests: e-mail headers where removed and the content is encrypted, and is thus not suitable to our needs [12]. The Spam Track collection constitutes a special case since all the e-mails are in their original form; however, they have been collected from different users and are all mixed together — this is not suitable for our needs. A similar situation applies regarding the SpamAssassin *corpus*.

As far as the Enron dataset goes, we have to mention this is not the original dataset from the Enron investigation, but a sub-dataset arranged by Metsis and colleagues, which they used in their article "Spam filtering with Naïve-Bayes — Which Naïve-Bayes?" [51]. The authors selected the 6 largest folders from the original dataset, complemented them with spam from SpamAssassin, the Honeypot Project and one of the author's personal spam collection, thus producing 6 folders with personal ham plus spam (one folder per user, six different users in total). In addition to this, the e-mails suffered a light pre-processing which included removing all HTML headers and tags, leaving only the message's subject and body. We decided that this pre-processing did not go against our requirement for "untouched" e-mails since all the information we needed for our analysis was still present (subject and body).

Another dataset was obtained from Luis Andre personal e-mails, which we will now refer to as LA. In this case the e-mails where received in his e-mail address and then downloaded for analysis. Both spam and ham have been collected, having taken special attention to ham e-mails containing mostly text, as some messages may contain only attachments or images.

Our options for analyzing e-mails, besides the n-grams and character selection, included discarding header information. Drucker and colleagues [27], have concluded that the best results are achieved when using both e-mail body and subject, without resorting to a *stop list*[1]. Our reasoning for not including header information is that headers are fairly identical between e-mails, thus leaving subject and content as sources of diversity.

3.1 Pre-Processing

As discussed previously, we have adopted a classification approach requiring e-mail pre-processing. This treatment is common for both the benchmarking Naïve-Bayes and TAT classification. E-mail pre-processing starts differently for the two datasets we have used. The Enron dataset had already been subject to pre-processing, with some HTML removed, while the LA collection was still in its raw form, and it would have to be transformed into the same format as the Enron. For that matter, Aris Kosmopoulos[2] kindly provided a simple program that would pre-process the e-mails into a format similar to the Enron dataset — this small application was written in C++. From this point forward, the pre-processing is the same.

Loading up specific information from a configuration file, our pre-processing program reads the e-mail files, one by one, and compiles them in one single file, in yet another format. This was optimized to fit the TAT simulator and the Naïve-Bayes special input structure, described in Section 3.3: each e-mail file generates a series of tokens that will be stored along with information about the file in one single line, meaning the input file will have as many lines as e-mails processed.

Algorithm 1 contains the pseudo-code explaining how we load up data from an e-mail on file. Certain fields are preserved as a result of the Metsis and colleagues'

[1] A stop list is a list with common words, such as determinants, or prefixes and suffixes.

[2] From the Institute of Informatics and Telecomunication — NCSR Demokritos; the code used in Metsis and colleagues' paper was not available, however one of the authors suggested using Aris Kosmopoulos' code due to its very similar output.

pre-processing method, such as the `To:`, `From:` and even the multipart boundary dashes (- - - - -), which are not of our interest, so we look for the `Subject:` field and any other line that has none of these tags. With this approach, we are able to attain in one single variable (called *concat* in Algorithm 1) the e-mail's subject and content in lower case, but still preserving the writing structure (paragraphs, new lines, etc.).

Following, *concat* will be tokenized by one of two slightly different ways, one for word tokenization and another for n-grams. For the latter, the string is analyzed character by character, and each character is cross-checked against a list of acceptable characters (see Table 3.3) and only afterwards the n-gram will be created and stored. If the tokenizing is to be done by word, then each word that is over 3 characters long will be subject to a Porter stemming method[3] and then stored.

Apart from that slight difference, both routes yield a `<Vector>` type object with a given amount of elements. Following the same strategy as Abi-Haidar [4], we too select the first and last 25 elements from the vector, representing the first and last 25 tokens obtained from the e-mail — since people usually greet and sign off in a consistent fashion throughout their e-mails we hope to be able to catch recurrent patterns that could easily help to identify ham e-mails. The pre-processing ends by writing the the e-mails' meta-information (file name, original classification — ham/spam — and other data) followed by the selected 50 elements into an output file — see the TAT input example in Section 3.3.

Selecting 50 tokens as classification features — be it words or n-grams — is an approach that we hope will allow us to extract the parts with the most individual-related information, *i.e.*, the data on the e-mail that can be more easily associated to the person writing it, which includes the subject, the greeting and signature of the text. N-gram size is configured by setting the desired value in a configuration file and can be varied, although all e-mails end up being pre-processed with the same chosen size.

Naïve-Bayes classifiers learn on sets of ham and spam before they can classify while the TAT-based classifier, similarly to the immune system, learns on one set of ham with up to 20% of spam, as an inoculation simulation. The objective is to create a few cells capable of targeting the spam "antigens" right from the start, thus identifying spam e-mails more easily during the testing phase. As a programming solution, the pre-processing produces, besides the testing file, two output learning files for the

[3]The algorithm is publicly available at `http://tartarus.org/~martin/PorterStemmer`.

Naïve-Bayes classifier (one set of ham and one set of spam) and only one file for the TAT-based classifier.

Algorithm 1: Processing e-mail files

```
      input : email ⟸ File object
1     line ⟸ String (null);
2     concat ⟸ String (empty);
3     while email has lines to read and is not null do
4     |     line ⟸ one line from email;
5     |     if line does not contain "mime —", "content —", "charset :",
      |     "message —", "cc:", "to :", "- - - - -", "sent:", "forwarded:" or
      |     "from :" then
6     |     |     if line contains "subject:" and "subject:" is not the full string
      |     |     then
7     |     |     |     line ⟸ substring of line without "subject:";
8     |     |     end
9     |     |     append line to concat;
10    |     end
11    |     lowercase concat;
12    end
      return: concat
```

```
    tokens ⟸ Vector<String> (empty);
    if Config.ngram == 1 then
        // pre-processing into n-grams
13      processing concat character by character, validating (see Table 3.3
        for a list of valid characters) and building n-grams;
14      loading each n-gram into tokens;
15      if tokens.size() is bigger than 50 then
16          tmpV ⟸ Vector<String> (empty);
17          iterate through the first 25 elements in tokens and load
            them into tmpV;
18          iterate through the last 25 elements in tokens and load
            them into tmpV;
19          clear tokens;
20          load the 50 elements from tmpV into tokens;
21      end
    end
```

```
        else
            // pre-processing into words
22          stk ⇐ tokenizing concat with the standard separator (blank
            space);
23          tmpVV ⇐ Vector<String> (empty);
24          while stk has more tokens do
25              tkn ⇐ String (empty);
26              tkn ⇐ token from stk;
27              if tkn is above 3 characters in length then
28                  tkn ⇐ Porter stemming;
29                  load tkn into tmpVV;
30              end
31          end
32          if tmpVV.size() is bigger than 50 then
33              iterate through the first 25 elements in tmpVV and load
                them into tokens;
34              iterate through the last 25 elements in tmpVV and load
                them into tokens;
35          else
36              iterate through all the elements in tmpVV and load them
                into tokens;
37          end
        end
        print the meta-information and all the elements in tokens into file;
        print a new line character;
```

3.2 Benchmarker

As mentioned before, the benchmarking program classifies e-mails according to the Multinomial Naïve-Bayes algorithm. The Bayesian classification takes *feature vectors* from e-mails and calculates the probability of elements of the vector belonging to a certain *category*: in this case, either **spam** or **ham** [79, 71, 54]. A fairly good example was suggested by Tretyakov [71]: imagine we found a word in an e-mail that we knew it could never occur in ham - then we would move the message to

Table 3.3: ASCII codes and characters accepted for inclusion in pre-processing data

DECIMAL	CHARACTER	DEC.	CHAR.	DEC.	CHAR.
1	SOH (start of heading)	72	H	119	w
4	EOT (end of transmission)	73	I	120	x
5	ENQ (enquiry)	74	J	121	y
6	ACK (acknowledge)	75	K	122	z
7	BEL (bell)	76	L	128	Ç
8	BS (backspace)	77	M	129	ü
14	SO (shift out)	78	N	130	é
15	SI (shift in)	79	O	131	â
16	DLE (data link escape)	80	P	132	ä
17	DC1 (device control 1)	81	Q	133	à
18	DC2 (device control 2)	82	R	134	a
19	DC3 (device control 3)	83	S	135	ç
20	DC4 (device control 4)	84	T	136	ê
21	NAK (negative acknowledge)	85	U	137	ë
22	SYN (synchronous idle)	86	V	138	è
23	ETB (end of trans. block)	87	W	139	ï
24	CAN (cancel)	88	X	140	î
25	EM (end of medium)	89	Y	141	ì
26	SUB (substitute)	90	Z	142	Ä
27	ESC (escape)	97	a	143	A
28	FS (file separator)	98	b	144	É
29	GS (group separator)	99	c	145	æ
30	RS (record separator)	100	d	146	Æ
31	US (unit separator)	101	e	147	ô
48	0 (start of heading)	102	f	148	ö
49	1 (end of transmission)	103	g	149	ò
50	2 (enquiry)	104	h	150	û
51	3 (acknowledge)	105	i	151	ù
52	4 (bell)	106	j	153	Ö
53	5	107	k	154	Ü
54	6	108	l	156	£
55	7	109	m	157	¥
56	8	110	n	159	ƒ
57	9	111	o	160	á
65	A	112	p	161	í
66	B	113	q	162	ó
67	C	114	r	163	ú
68	D	115	s	164	ñ
69	E	116	t	165	Ñ
70	F	117	u		
71	G	118	v		

the spam folder, *i.e.* the probability of a feature vector x (representing the message) belonging to class S (spam) is 1.

From Equation 2.1, we can see a Bayesian classifier will indicate a message as belonging to the category which maximizes $p(\mathbf{x} \mid c) \, . \, p(c)$, *i.e.*, the category which is more likely to have "originated" the message. In our case, we have two categories

— spam and ham — thus Equation 2.1 can be rewritten as

$$\frac{p(c_s) \,.\, p(x \mid c_s)}{p(c_s) \,.\, p(x \mid c_s) \;+\; p(c_h) \,.\, p(x \mid c_h)} > T \tag{3.1}$$

where $T = 0.5$ and c_h and c_s refer to the ham and spam categories [51]. The *a priori* probabilities $p(c)$ for each category are obtained according to Equation 3.2, where the probability of a feature vector x belonging to a given category ($p(x \mid c)$) is calculated differently depending on the Nïve-Bayes version used. We have opted for a Multinomial NB, with term frequencies.

$$P(x \mid c) = \frac{\text{number of training messages of } c}{\text{total number of training messages}} \tag{3.2}$$

The Multinomial NB with term frequencies considers a message *d* as a set of tokens *t*, each occurring as many times as it is present in the message, so it can be represented as a vector of occurrences. Also, a message can be seen as if one independently selected tokens from F with replacement, with probability $p(t_i \mid c)$ for each token (F being the set of tokens resulting from selecting m attributes). This way, $p(x \mid c)$ is a multinomial distribution, and $|d|$, the message created with the selected tokens, does no depend on the category c [51].

Through these conditions, an e-mail will be spam if:

$$\frac{p(c_s) \cdot \prod_{i=1}^{m} p(t_i \mid c_s)^{x_i}}{\sum_{c \in \{c_s, c_h\}} p(c) \cdot \prod_{i=1}^{m} p(t_i \mid c)^{x_i}} > T \tag{3.3}$$

For each m_i, the corresponding $p(t \mid c)^{x_i}$ (where x_i is the frequency of term i from the training set) is calculated by

$$p(t \mid c) = \frac{1 + N_{t,c}}{m + N_c} \tag{3.4}$$

where

$$N_c = \sum_{i=1}^{m} N_{t_i,c} \tag{3.5}$$

In this NB version, $N_{t,c}$ represents the total occurrences of token t in the training messages belonging to category c.

During the training phase, the NB classifier creates the feature set for both spam and ham, which constitute its learning basis. As mentioned in Section 3.1, for each e-mail we have selected a 50 features vector. Each token creates an <Ngram> Java element, with the following variables:

- ngram - its unique string;
- tspamfreq - this token's frequency in all training spam messages;
- thamfreq - this token's frequency in all training ham messages;

<Ngram> objects created in this learning stage will remain in memory as the program reads and pre-processes the e-mails. Upon starting the testing phase, e-mails will also go through pre-processing and feature selection, but the resulting vector will be providing the strings with which to fetch the values from the learning features still in memory and use those values for the calculations required to classify the message — it may happen that for a given message, its feature vector has elements that were not stored during training and for these elements their relevance in the calculations is minimal (as it is defined by Equation 3.3).

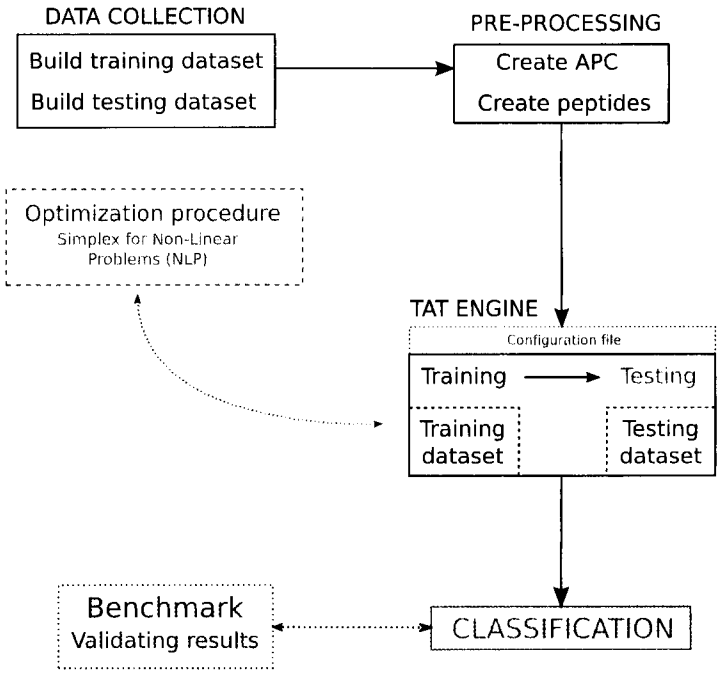
3.3 TAT-based

The idea of using TAT as a model for spam filtering developed from a project where TAT was being deployed for a Network Intrusion Detection System (NIDS) [8]. The e-mail filtering and the intrusion detection algorithm stem from the TAT theory, in a joint development to put TAT into work. Thus, a framework has been developed that suits both spam filtering and NIDS. The general idea of the framework can be seen in Figure 3.1.

Despite the differences between NIDS and spam filtering, the framework works in the same way towards either purpose, providing its input is setup according to a pre-determined structure so it can be understood by the TAT engine. The pre-processing algorithm in section 3.1 complies with such structure and the output files generated have lines according to the following organization:

meta-info1:meta-info2:meta-info3: data1 data2 data3 data4

Figure 3.1: The TAT framework for NIDS and spam filtering



Each line corresponds to a single e-mail, becoming an APC. The **meta-info** fields can comprise anything the user wants — we have used them to place information such as the filename and a tag (ALERTA/NORMAL) for later reference. All meta-information fields must be separated by ":". The other fields are the peptides obtained during pre-processing and must separated by a blank space (see Figure 3.2).

Figure 3.2: Examples of 3 APC formated according to TAT model

```
APC:0208-2000-01-17-farmer-ham-txt:TimeI:0:TimeF:0:NORMAL: glo ons lob ion
oba tio bal ati alr rat lri era ris per isk ope skm top kma nto man ent ana
men nag eme age gem gem age eme nag men ana ent man nto kma top skm ope isk
per ris era lri rat alr ati bal
APC:0210-2000-01-17-farmer-ham-txt:TimeI:0:TimeF:0:NORMAL: 18n xls 8no nxl
nom anx oms jan mst lja sto plj toc hpl oci shp cit lsh ity xls tyg nxl yga
anx gat jan ate lja tet plj etr hpl tro ehp roy leh oya ile yab fil abe dfi
ben edf eno hed noi che oit ach
APC:0006-2003-12-18-GP-spam-txt:TimeI:0:TimeF:0:ALERTA: dob ibe obm rib bme
cri meo scr eos bsc osw ubs swi sub wit nsu ith uns thh eun hhg teu hgh ite
ghm sit hmy bsi mye ebs yen web ene swe ner isw erg his rgy thi gyl tth yle
att lev eat eve rea vel ore
```

The TAT engine simulator implements the TAT model by creating `Tcells` able to recognize and bind to different `peptides`. The `Tcells` also have two variables representing the kinase (`K`) and phosphatase (`P`) enzymes of the real-life T lymphocytes and a proliferation value (`C`) awarded upon creation — assuming 5 in the training phase and 10 in the test phase — that will help in identifying rare `APCs`; how this help is put to work is explained in Section 3.3.2. The TAT implementation was based on the following principles:

1. `Tcells` are created with initial `K` and `P` values, more specifically, K_0 and P_0;

2. upon receiving a certain amount of stimuli (`S`), `K` and `P` may increase up to a *steady-state*, calculated by multiplying the stimuli with a maximum value for each — $S * K_{max}$ and $S * P_{max}$;

3. `K` and `P` increase at different rates, corresponding to the slopes of two straight lines — ϕK and ϕP;

4. ϕK is higher than ϕP, but `K`'s *steady-state* value is lower than `P`'s; this translates into a faster increase in `K`'s values but it will also stabilize sooner than `P`.

Principles 2 and 3 comprise the core dynamics of the TAT model; this central piece

can be re-written in pseudo-code in the following manner:

Algorithm 2: TAT dynamic engine pseudo-code

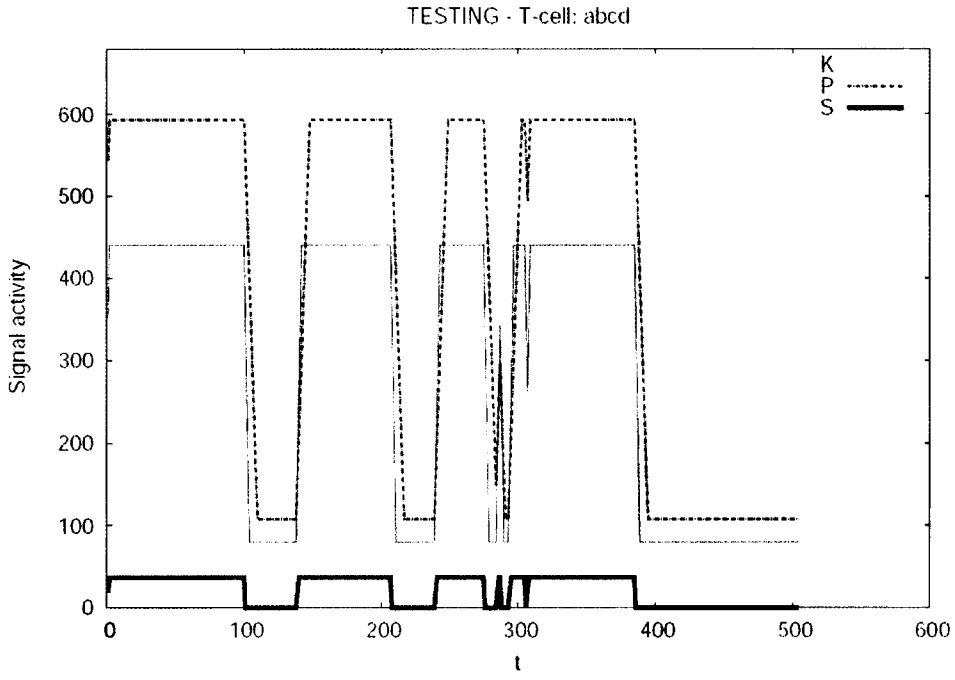
foreach *cell* **do**
  if $K_{max}\cdot$ (S + basal stimuli) is bigger than current K value **then**
    current $K \Longleftarrow$ minimum(($K_{max}\cdot$ (S + basal stimuli)), (current K + $\phi K \cdot APC\_time$));
  end
  else
    current $K \Longleftarrow$ maximum(($K_{max}\cdot$ (S + basal stimuli)), (current K + $\phi K \cdot APC\_time$));
  end
  if $P_{max}\cdot$ (S + basal stimuli) is bigger than current P value **then**
    current $P \Longleftarrow$ minimum(($P_{max}\cdot$ (S + basal stimuli)), (current P + $\phi P \cdot APC\_time$));
  end
  else
    current $P \Longleftarrow$ maximum(($P_{max}\cdot$ (S + basal stimuli)), (current P + $\phi P \cdot APC\_time$));
  end
end

Figure 3.3 is a graphical example of the variation of a `Tcell`'s `K` and `P` values with the stimulus (`S`) received. We can clearly see that `K` grows faster than `P` but also reaches a plateau (corresponding to K_{max}) faster.

3.3.1 Training

The TAT simulator requires training, which is comprised by two different steps. We have split the training dataset into two sub-datasets, each for one of the training phases: *treino_p* and *teste_p*. The former is used for initial simulator training, having only ham `APC`'s — `Tcells` have their `K` and `P` values updated according to the TAT dynamics expressed in Algorithm 2; in this step, the created `Tcells` repertoire is stored for later use. The other subset has ham and spam `APC`'s (but no more spam than 20% of the total `APCs`) and functions as a "dry run" of the simulator, following the same dynamics as before but this time also creating `Tcells` that will target *non-self*.

Figure 3.3: K and P values change according to the stimulus (S) received



The *teste_p* phase not only updates K and P values of the Tcells but also changes the proliferation value — the starting value remains 5, just like in *treino_p* and when an APC is classified, the simulator will check if that classification is correct (if a ham is ham and a spam, spam). If it is, the proliferation value of all the Tcells whose TCR's matched the APC's peptides and have $K > P$ will be doubled ($C \cdot 2$). If, on the other hand, the classification awarded is not correct (*i.e.*, the APC is ham and was classified as spam, or vice-versa) then the proliferation value of the Tcells matching the APCs peptides which also have $K > P$ will be reduced to half ($\frac{C}{2}$).

Teste_p works as a kind of vaccine, where weakened antigens are introduced into the system in order to produce cells ready to deploy the necessary responses when the real, full-active pathogens invade the organism (in the training subset context, "weakened" means a small proportion — hence the 20% limit) .

A typical TAT parameter set comprises K and P, their slopes (ϕK and ϕP), K_{max}, P_{max}, and basal stimuli s_0. However, TAT relies on the affinity strength between the peptide and the Tcell and other parameters. These and their role in the algorithm are explained in table 3.4.

To find a combination of TAT parameters that simultaneously produces low false

Table 3.4: TAT runtime parameters and their meaning (from [8])

Parameter	Application
rare APC threshold	multiplies with the number of `Tcells` that had a high enough match to yield the minimum number of activated `Tcells` required to identify a given `APC` as **rare**
rare peptides threshold	multiplies with the number of `Tcells` that had a high enough match to yield the minimum number of activated `Tcells` necessary to determine if the peptide is rare
affinity threshold	threshold for the binding affinity between the `Tcell`'s pattern (`TCR`) and the `peptides` — defines whether a `Tcell` binds with a `peptide` or not

positives (FP's) and a good filtering (high rate of true positives - TP), we employ a non-linear meta-heuristic simplex optimizer [59] which will find a good set of runtime parameters for the TAT simulator such as the ones in Table 3.4; this is done in the *Teste_p* phase. A feedback system between the optimizer and the simulator is in place, so that the optimizer may suggest combinations of parameters more adequate for a good result. Given the equation in Function 3.6 to minimize, the optimizer will propose a configuration to the simulator, which will run the training phases with it. Function 3.6 tells the optimizer to keep searching for a configuration that allows to achieve the lowest levels of false negatives, *i.e.*, we are telling the optimizer we do not mind a few spams in our inbox if that prevents hams from being wrongly classified. The *bias* is a value the user provides in a configuration file for the optimizer and is meant to introduce some bias to the calculus, depending on whether it is preferable to penalize obtaining false negatives or false positives. Values for false positives and false negatives are are fed back into the optimizer and the starting function will be recalculated with the new values. The process repeats

itself until one of the stopping conditions is reached:

- if the proposed parameters are repeated, in successive or intervaled iterations;
- if there have been more iterations than the ones allowed by the user (via configuration file parameter);
- if there is a large enough number of valid iterations (configurations with non-valid parameters are not taken into account — see the optimizer constraints Table 3.5);
- and finnaly if the total of valid iterations is larger than the configuration value.

If any of these checkpoints is true, the optimizer feeds the configuration which obtained the best results thus far to the simulator, and then the testing phase begins.

$$\texttt{Fo} = \text{false positives} + (bias \cdot \text{false negatives}) \tag{3.6}$$

Table 3.5: Optimizer value constraints and derived parameters [8]

Constraints	Derived Parameters
$0 < Affinity < 1$	$K_0 = S_0 \cdot K_{max}$
$0 < \frac{\phi K}{\phi P} < 1$	$Pmax = (\frac{P_{max}}{K_{max}}) \cdot K_{max}$
$0 < \frac{P_{max}}{K_{max}} < 1$	$P_0 = S_0 \cdot P_{max}$

Some constraints are imposed to the optimizer regarding the values of the different thresholds. These limits are used so that the optimizer will look for values that still make sense. In an attempt to reduce the number of input parameters, we were able to derive some from others, according to table 3.5.

3.3.2 Classifying

Two different classification algorithms were developed, one relating to a user-defined threshold for classification, called TAT version A, and another making use of the number of *active* `Tcells` in the `Tcells` population — TAT version B. The pseudo-code of the version A is described in Algorithm 3, where α is the minimum amount

of affinity between TCR's necessary to be considered the existence of a "match" (the *affinity threshold*); τ is the *rare peptide threshold* and ψ the *rare APC threshold*.

Algorithm 3: TAT classification — using threshold (adapted from [8])

input : $APC \Longleftarrow$ `<Vector>` with the `peptides` from the test dataset
$V_{tcells} \Longleftarrow$ `<Vector>` with the `Tcells` created during training

1 $C_{tcells} \Longleftarrow$ `<Vector>` (null) ;
2 $C_{activeTs} \Longleftarrow$ `<Vector>` (null);
3 $C_{rareAPC} \Longleftarrow$ `<Vector>` (null);
4 **forall** *pep* in APC **do**
5 $S \Longleftarrow 0$;
6 **forall** *tcell* in V_{tcells} **do**
7 **if** affinity between $APC[pep]$ and $V_{tcells}[tcell]$ is bigger than α **then**
8 $S \Longleftarrow C * Aff$;
// C is the number of occurrences
// of each peptide $APC[pep]$
9 add $(V_{tcells}[tcell])$ to C_{Tcells};
10 **end**
11 update ($V_{tcell}[tcell]$, S);
12 **end**
13 **forall** *ctcell* in C_{tcells} **do**
14 **if** $K > P$ **then**
15 $C_{activeTs} \Longleftarrow$ add $(C_{tcells}[ctcell])$;
16 **end**
17 **end**
18 $prolifcalc \Longleftarrow \frac{\sum \text{active Tcells' proliferation}}{\text{number of active Tcells}}$;
19 **if** $C_{activeTs}$ is bigger than or equal to $C_{tcells} \cdot \tau$ and $prolifcalc$ is bigger than or equal to 5 **then**
20 $C_{rareAPC} \Longleftarrow$ add $(C_{tcells}[ctcell])$;
21 **end**
22 **end**
23 **if** $C_{activeTs}$ is bigger than $C_{tcells} \cdot \psi$ **then**
24 trigger alert for this APC;
25 **end**

The simulator has to iterate through every `peptide` *pep* of a vector of `peptides`

(which is the APC) and see if there is any Tcell *tcell* in the Tcells vector that matches the peptide in more than α. This match, or affinity, between the peptide and the Tcell TCR, is measured according to the Hamming's distance — which calculates the minimum number of character substitutions needed to change one string into the other [41].

Should the affinity be greater than the threshold, then the matching Tcells will be stimulated by a value which is calculated by multiplying the affinity value by the number of occurrences of the peptide in the APC. The Tcells that have had a match will be part of a *matched Tcells set* — C_{Tcells}.

All the Tcells in the C_{Tcells} set which satisfy the $K > P$ condition are placed in an active Tcells vector. This vector, in turn, will account for the identification of rare peptides by following the conditions:

- the vector must not be empty;
- its size must be bigger than or equal to the size of the C_{Tcells} set times τ (rare peptide threshold);
- the sum of the proliferation of all the active Tcells divided by the number of active Tcells is bigger than or equal to 5.

It was determined that the value of the sum of the proliferation of all the active Tcells divided by the number of active Tcells needed to be bigger or equal to 5 because 5 is the proliferation value assigned to a Tcell when created in the training phase; we are guaranteeing that should there be only one Tcell in the active Tcells vector, even if that Tcell was created in the training phase, it will be used in the classification instead of discarded.

When all three conditions are met, the peptides that matched and activated the corresponding Tcells are marked as rare. These peptides will influence the classification an APC receives, for if the total number of active Tcells is bigger than or equal to the number of Tcells that have had a match times the ψ (the rare APC threshold), then the APC will be classified as spam. If the total number of Tcells with $K > P$ happens to be lower, then the APC will be tagged as being ham.

The B version of the TAT classification algorithm was developed in an effort to escape the thresholds — and introduce a bit of the populational dynamics of the immune system [15, 4] — the APC classification of *rare* (spam) is only granted if, for a given

APC, the absolute value of the total number of active Tcells minus the total number of rare `peptides` divided by the total number of active Tcells is equal or less than 0.5. In other words, from line 22 until the end, Algorithm 3 would become:

Algorithm 4: TAT classification — second approach

$val \Longleftarrow |\frac{ActiveTCells - RarePeptides}{ActiveTCells}|$;
if val is equal or less than 0.5 **then**
    trigger alert for this APC;
end

With this slight change we take into account the rare `peptides` that did not activate `Tcells` as well as the ones which did, compensating a possible situation where rare `peptides` could exist in the APC that did not activated `Tcells` and where not being taken into account.

The case not depicted in Algorithm 3 is the one where no `Tcell` matches a `peptide`. In such case, a new `Tcell` is created, with a TCR identical to the `peptide` that was at miss. The TAT dynamics also foresees situations in which a `Tcell` has not been stimulated for quite some time: its K and P values decrease until they eventually reach K_0 and P_0, point at which the `Tcell` is removed from the repertoire — similar to cellular death.

Chapter 4

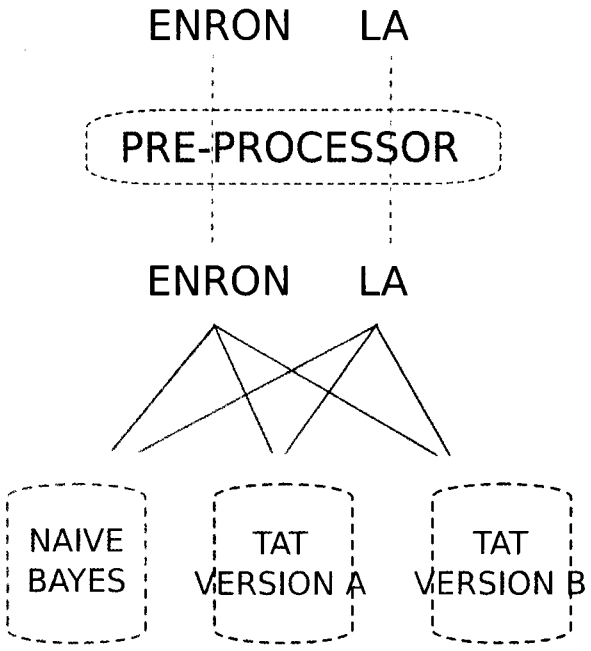
Results

We have used the enron dataset and LA's collection in several runs on both the NB and TAT-based filters, using different training and testing set configurations. A 10-fold cross validation [63, 47] was used with the enron dataset to compare our pre-processing with that of previously published results of a similar bio-inspired approach ([4, 51]). To compare the Naïve-Bayes versus the TAT-based versions A and B and also both versions between each other, using the enron and LA datasets, we prepared two datasets of e-mails per enron folder, totalling 12 datasets, plus the LA e-mails.

The datasets were analyzed three times, one by the Multinomial Naïve-Bayes with Term Frequencies and twice by TAT, for as described in Section 3.3.2, we have implemented two versions of TAT for comparison. As to follow the guidelines on how to deploy statistical tests, both datasets were pre-processed once and then three identical copies of the resulting sets where used in the tests, without any change — Figure 4.1 visually illustrates this idea.

Two-tail paired sample *t*-tests were applied [84] to the F-score and Accuracy [61, 81] values calculated from the results obtained from running the filters with the datasets mentioned above. We have been able to test whether the results' variation was the same between the results of our Multinomial Naïve-Bayes with Term Frequencies implementation and that of Abi-Haidar [4] (Multinomial Naïve-Bayes with Boolean Attributes) (Table 4.7), between our Naïve-Bayes implementation and each TAT version and also between the two TAT versions (Tables 4.8, 4.9 and 4.10). Paired-sample hypothesys testing is done when data from one sample is somehow related to the data in the other sample, thus allowing to test whether the mean difference

Figure 4.1: Enron and LA dataset handling from raw until being analyzed by the two programs.



between samples is equal or not[84].

4.1 Benchmark

Benchmarking is used to compare how well a system performs. In our case, it will tell us how accurate our filter is and how our decisions regarding retrieving information affect classification.

Table 4.1 details the configuration of the training and testing setup for the 10-fold cross validation, per each of the six enron folders. After selecting 1500 ham and 1500 spam in each of the enron dataset folders, we created 10 subsets, each with 150 ham and 150 spam and ran the program 10 times, each time with a different testing subset. Folder "enron2" only had 1496 spam, thus one of the ten subsets only had 146 spam — meaning that when that subset was the testing one, the test was done in 146 spam versus the 150 from the other folders; when any other subset was the testing one, then the learning set would have 1346, against the 1350 of the other folders. This is the reason behind the difference in values in Table 4.1.

Table 4.2 presents the Accuracy and F-score computed from the results of the 10-

Table 4.1: Enron Dataset 10-fold cross validation setup

	Training		Testing	
Dataset	Ham	Spam	Ham	Spam
enron1	1350	1350	150	150
enron2	1350	1346	150	196
enron3	1350	1350	150	150
enron4	1350	1350	150	150
enron5	1350	1350	150	150
enron6	1350	1350	150	150

fold runs on each folder separately, with e-mails pre-processed into character level n-grams sized 3, 4 and 5 and another approach using e-mail words. It is also shown the total Accuracy and F-score separated by each pre-processing type (the different n-gram sizes plus the words). The formulas used are show in Equations 4.1, 4.2, 4.3 and 4.4. Calculations for the totals were performed averaging the individual F-score and Accuracy and computing the standard deviation of all results, per n-gram/word. TP, TN, FP and FN respectively denote the true positive, true negative, false positive and false negative rates.

$$Accuracy = \frac{(TP + TN)}{(TP + TN + FP + FN)} \tag{4.1}$$

$$F = \frac{2 \cdot Precision \cdot Recall}{Precision + Recall} \tag{4.2}$$

$$Precision = \frac{TP}{(TP + FP)} \tag{4.3}$$

$$Recall = \frac{TP}{(TP + FN)} \tag{4.4}$$

Because we used the same datasets and the same Naïve-Bayes version as other authors, the results from the 10-fold testing setup will be compared against results already published in the literature, directly comparing different methods of pre-processing and feature selection approaches.

Another test setup was arranged, which used the previously selected enron subsets and also the LA dataset, fed to both the Naïve-Bayes and TAT filters, to compare their classification efficiency. For each enron folder, the 10 subsets were divided in two groups of 5; the training was performed on the first subset while the testing was done on the remaining four — and then repeated, has there were 2 groups of datasets per enron folder. This setup, designated as "5-by-2", was meant to provide enough testing e-mails as to introduce the changes in subjects within each enron folder. However, to abide to the TAT-based filter learning mechanism, the training sets only had 20 spam. Again, the folder enron2 has a smaller number of spam. Table 4.3 discriminates the amount of ham and spam in the training and testing sets used. The LA set was also divided into training and testing sets, as can be seen in the same Table.

Table 4.2: Benchmark results from the 10-fold cross validation runs

		enron1	enron2	enron3	enron4	enron5	enron6	**Totals**
3	Accuracy	0.72 ± 0.05	0.74 ± 0.02	0.74 ± 0.05	0.74 ± 0.04	0.74 ± 0.19	0.78 ± 0.02	$\mathbf{0.74 \pm 0.09}$
	F-score	0.77 ± 0.03	0.78 ± 0.02	0.75 ± 0.05	0.73 ± 0.04	0.81 ± 0.14	0.81 ± 0.01	$\mathbf{0.77 \pm 0.07}$
4	Accuracy	0.59 ± 0.05	0.59 ± 0.02	0.60 ± 0.04	0.63 ± 0.04	0.53 ± 0.04	0.66 ± 0.03	$\mathbf{0.60 \pm 0.05}$
	F-score	0.66 ± 0.04	0.66 ± 0.02	0.63 ± 0.05	0.65 ± 0.04	0.62 ± 0.03	0.70 ± 0.02	$\mathbf{0.65 \pm 0.04}$
5	Accuracy	0.54 ± 0.04	0.56 ± 0.02	0.52 ± 0.04	0.62 ± 0.04	0.48 ± 0.09	0.62 ± 0.02	$\mathbf{0.56 \pm 0.07}$
	F-score	0.63 ± 0.03	0.62 ± 0.02	0.57 ± 0.04	0.65 ± 0.04	0.58 ± 0.05	0.66 ± 0.02	$\mathbf{0.62 \pm 0.05}$
words	Accuracy	0.87 ± 0.04	0.91 ± 0.02	0.94 ± 0.02	0.91 ± 0.02	0.87 ± 0.08	0.90 ± 0.01	$\mathbf{0.90 \pm 0.05}$
	F-score	0.88 ± 0.04	0.91 ± 0.03	0.93 ± 0.02	0.90 ± 0.02	0.87 ± 0.07	0.90 ± 0.01	$\mathbf{0.90 \pm 0.04}$

Table 4.3: Enron and LA "5-by-2" dataset setup

	Training		Testing	
Dataset	Ham	Spam	Ham	Spam
LA	136	12	437	115
enron1 first group	150	20	600	600
enron1 second group	150	20	600	600
enron2 first group	150	20	600	600
enron2 second group	150	20	600	596
enron3 first group	150	20	600	600
enron3 second group	150	20	600	600
enron4 first group	150	20	600	600
enron4 second group	150	20	600	600
enron5 first group	150	20	600	600
enron5 second group	150	20	600	600
enron6 first group	150	20	600	600
enron6 second group	150	20	600	600

Similarly to the 10-fold dataset setup, Table 4.4 contains the results of the "5-by-2" setup computed as F-score and Accuracy values as well, but calculated as totals per n-gram size, *i.e.*, the values of Accuracy and F-score shown correspond to the average of all groups' Accuracy and F-score values in all folders and their respective standard deviation, per n-gram size. Due to its reduced sized, there was only one run with the LA dataset, hence no standard deviation is present. The color coding is used for future reference.

In Section 4.3 we will address these results and compare them side-by-side with previously published values [4], for the first testing run, and with the TAT-based results, in what regards the second testing run.

4.2 TAT-based

Section 3.3 explains how two different versions to classify an APC as rare were developed and their differences. In this Section the threshold approach described in

Algorithm 3 will be designated as **version A** and **version B** will refer to the second approach (Algorithm 4).

Using the datasets prepared to run with the Naïve-Bayes filter, more specifically the ones referred to in Table 4.3, and running them in the TAT-based filter once for each method, we obtained the results shown in Table 4.5. As before, the LA dataset only had one single run; the enron results are once more the averages of the Accuracy and F-score of both groups of all the six enron folders.

Finding a set of configuration parameters was done by repeatedly calling the parameter optimizer on the TAT simulator. Due to the fact that a single TAT configuration file takes on many numeric parameters, the number of their possible combinations is enormous and varies with the dataset, reason why each dataset needed its own configuration (the configuration optimizing process was run every time a new dataset was used).

There are no results for pre-processing with words since our aim was to use n-grams. Tokenizing by words was taken into account in the Naïve-Bayes tests as it was usefull to help analyze the pre-processing solutions applied to the datasets — the results of Abi-Haidar Naïve-Bayes implementation were obtained by pre-processing e-mails into words and like so we could directly compare results and assess differences in pre-processing.

Presented in Table 4.6 is the merging of two tables: the results from the (second) Naïve-Bayes (as seen in Table 4.4) and TAT-based runs (Table 4.5). The color code used previously is meant to easily identify related columns. The Totals column refers to F-score and Accuracy for all enron **and** LA results, per n-gram size. In colored bold one can see the best Accuracy result while the black bold highlights the F-score best result, per group (enron, LA or Totals) in a given n-gram size. Here, we can easily identify which filter has the best performance comparatively with either the enron or LA dataset, or even both combined.

Table 4.4: Benchmark results from the "5-by-2" setup run

		enron	LA
3	Accuracy	0.47 ± 0.06	0.69
	F-score	0.53 ± 0.03	0.21
4	Accuracy	0.47 ± 0.04	0.51
	F-score	0.14 ± 0.05	0.16
5	Accuracy	0.51 ± 0.02	0.78
	F-score	0.12 ± 0.05	0.17

Table 4.5: TAT-based results

		version A		version B	
		enron	LA	enron	LA
3	Accuracy	0.52 ± 0.07	0.74	0.53 ± 0.05	0.62
	F-score	0.55 ± 0.09	0.20	0.62 ± 0.08	0.34
4	Accuracy	0.51 ± 0.07	0.6	0.52 ± 0.06	0.26
	F-score	0.46 ± 0.17	0.35	0.62 ± 0.11	0.36
5	Accuracy	0.50 ± 0.03	0.41	0.48 ± 0.06	0.78
	F-score	0.57 ± 0.11	0.40	0.49 ± 0.14	0.21

Table 4.6: Naïve-Bayes vs TAT-based

		Naïve-Bayes			version A			version B		
		enron	LA	Totals	enron	LA	Totals	enron	LA	Totals
3	Accuracy	0.47 ± 0.06	0.69	0.49 ± 0.08	0.52 ± 0.07	**0.74**	$\mathbf{0.54 \pm 0.09}$	$\mathbf{0.53 \pm 0.05}$	0.62	0.53 ± 0.06
	F-score	0.53 ± 0.03	0.21	0.50 ± 0.09	0.55 ± 0.09	0.20	0.52 ± 0.13	$\mathbf{0.62 \pm 0.08}$	**0.34**	$\mathbf{0.60 \pm 0.11}$
4	Accuracy	0.47 ± 0.04	0.51	0.47 ± 0.04	0.51 ± 0.07	**0.60**	$\mathbf{0.51 \pm 0.07}$	$\mathbf{0.52 \pm 0.06}$	0.26	0.50 ± 0.09
	F-score	0.14 ± 0.05	0.16	0.14 ± 0.05	0.46 ± 0.17	0.35	0.45 ± 0.16	$\mathbf{0.62 \pm 0.11}$	**0.36**	$\mathbf{0.60 \pm 0.13}$
5	Accuracy	$\mathbf{0.51 \pm 0.02}$	**0.78**	$\mathbf{0.53 \pm 0.08}$	0.50 ± 0.03	0.41	0.49 ± 0.04	0.48 ± 0.06	**0.78**	0.50 ± 0.10
	F-score	0.12 ± 0.05	0.17	0.12 ± 0.05	$\mathbf{0.57 \pm 0.11}$	**0.40**	$\mathbf{0.56 \pm 0.12}$	0.49 ± 0.14	0.21	0.47 ± 0.15

4.3 Statistical Analysis

Assessing if a result is similar or different from another is best done by means of statistical tests. Such tests help to bring forward evidences that may be misleading or even not clearly understandable — from Table 4.6, we can see there are some differences between TAT-based filtering with version A and version B, and just by looking at the results in that same table, one would feel inclined to state that version B is more efficient in correctly identifying spam. However, the difference between the results each method yields may not be statistically significant and if that is the case, then one can not conclude there is a better method.

Earlier in the Chapter we identified the hypotheses to be tested as well as the statistical test to be used. The two-tail paired t-test is appropriate for our study since the datasets used to test the filters are the same in all experiments — becoming our *dependent variable*. The Student's t-test formula we applied is [84]:

$$t = \frac{\overline{d}}{S_{\overline{d}}} \tag{4.5}$$

In our case, for two series of results, $\overline{d}$ and $S_{\overline{d}}$ are, respectively, the *mean* and the *standard error* of the difference between each pair of results.

Let us state once more what the two main questions we are interested in:

a) does our pre-processing method have any influence in the classification (for better, or worse)?

b) is our TAT-based filter as good as a Multinomial Naïve-Bayes filter with Term Frequencies?

These two questions can be re-formulated into finding if the results variation is equal between filters and pre-processing methods.

Answering question **a)** was done by comparing the results from the 10-fold cross validation tests of the enron dataset on the Multinomial Naïve-Bayes filter with Abi-Haidar's Multinomial Naïve-Bayes results [4] (the author kindly provided the F-score and Accuracy values from each of the 60 runs) to see if there was no difference in the results' variation — $H_0 : \mu_1 - \mu_2 = 0$, $H_A : \mu_1 - \mu_2 \neq 0$, which, if defining a population difference $\mu_d = \mu_1 - \mu_2$, can be re-written as $H_0 : \mu_d = 0$, $H_A : \mu_d \neq 0$ [84]. The test outcome can be seen in Table 4.7, where Abi-Haidar's results [4]

(designated as NB-A) were paired with our own (MultiNB-LA, for reference), in the following combinations, for both F-score and Accuracy: NB-A vs MultiNB-LA n-gram size 3; NB-A vs MultiNB-LA n-gram size 4; NB-A vs MultiNB-LA n-gram size 5 and NB-A vs MultiNB-LA words. The results are shown in Table 4.7[1], and the critical t value was found to be $t = 2.001$, for a 0.05 confidence interval at 59 degrees of freedom.

Question **b)** leads us to the results in Table 4.6, where it seems that the TAT-based filter performs a bit better than our Naïve-Bayes implementation, either thanks to one method or another. We have performed another set of two-tailed paired t-tests to see if the results from both filters and within the two TAT-based methods had identical variation; once more, our null hypothesis were $H_0 : \mu_d = 0$, $H_A : \mu_d \neq 0$ [84].

[1]The $p - value$ values where calculated on-line, thanks to Daniel S. Soper Statistical Calculator — p–value for Student t-test [66].

Table 4.7: Paired t test results for the two Naïve-Bayes versions

	NB-A vs MultiNB-LA3		NB-A vs MultiNB-LA4		NB-A vs MultiNB-LA5		NB-A vs MultiNB-LAwords	
	F-score	Accuracy	F-score	Accuracy	F-score	Accuracy	F-score	Accuracy
t	1.4386	0.9800	2.8762	1.9086	2.8565	1.8652	0.3117	0.0874
p-value	0.1555	0.3310	0.0057	0.0611	0.0029	0.0671	0.7563	0.9306
H_0	not rejected	not rejected	rejected	not rejected	rejected	not rejected	not rejected	not rejected

Table 4.8: Paired t test results for Multinomial Naïve-Bayes versus TAT-based filter - n-gram size 3

	MultiNB-LA3 vs TAT v. A		MultiNB-LA3 vs TAT v. B		TAT v. A vs TAT v. B	
	F-score	Accuracy	F-score	Accuracy	F-score	Accuracy
t	1.599	0.528	4.325	0.955	1.448	0.742
p(t)	$0.20 > p(t) > 0.10$	p(t)> 0.5	$0.001 >$ p(t)	$0.20 >$ p(t)> 0.10	$0.50 >$ p(t)> 0.20	$0.20 >$ p(t)> 0.10
H_0	not rejected	not rejected	rejected	not rejected	not rejected	not rejected

Table 4.9: Paired t test results for Multinomial Naïve-Bayes versus TAT-based filter - n-gram size 4

	MultiNB-LA4 vs TAT v. A		MultiNB-LA4 vs v. B		TAT v. A vs TAT v. B	
	F-score	Accuracy	F-score	Accuracy	F-score	Accuracy
t	6.116	15.412	12.580	12.833	2.337	0.465
p(t)	$0.001 > p(t)$	$0.001 > p(t)$	$0.001 > p(t)$	$0.001 > p(t)$	$0.05 > p(t) > 0.02$	$p(t) > 0.5$
H_0	rejected	rejected	rejected	rejected	rejected	not rejected

Table 4.10: Paired t test results for Multinomial Naïve-Bayes versus TAT-based filter - n-gram size 5

	MultiNB-LA5 vs TAT v. A		MultiNB-LA5 vs TAT v. B		TAT v. A vs TAT v. B	
	F-score	Accuracy	F-score	Accuracy	F-score	Accuracy
t	10.525	19.893	6.733	13.193	1.546	0.145
p(t)	$0.001 > p(t)$	$0.001 > p(t)$	$0.001 > p(t)$	$0.001 > p(t)$	$0.20 > p(t) > 0.10$	$p(t) > 0.5$
H_0	rejected	rejected	rejected	rejected	not rejected	not rejected

The tests resumed in Table 4.7 lead us to conclude that the pre-processing introduces changes in the classification results, namely in what concerns n-grams sizes 4 and 5. N-grams size 3 seem to allow for a similar performance between both Naïve-Bayes versions — we implemented a Multinomial Naïve-Bayes with Term Frequencies; the available results were from a Multinomial Naïve-Bayes with Boolean Attributes.

Although very similar, the Multinomial Naïve-Bayes with Term Frequencies treats each message as a set of boolean attributes of selected features, which is to say, the feature vector is constituted by a variable that says whether a feature was present in the message or not. Accuracy in this case was shown to be identical between both versions and pre-processing methods, for all n-gram sizes as well as for messages pre-processed into words (Table 4.7) — the H_0 was not rejected (P-value for our tests was bigger than 0.05). When working with words, the variation of the F-scores of both Naïve-Bayes versions was also tested to be similar (accepted H_0 with a P-value of 0.76 for the F-score and 0.93 for the Accuracy). The t test for the F-score, however, tells us that when performing classification with n-grams size 4 and 5 the results obtained were different (P-values of 0.0057 for n-gram size 4 and 0.0029 for size 5).

With this slight difference in mind, we now look into the results of the TAT-based filter compared against the ones obtained by our Multinomial Naïve-Bayes implementation. All the results can be seen in Tables 4.6, 4.8, 4.9, 4.10 — the first one with the averaged F-scores and Accuracy values, the remaining ones with the different t test outcomes. Just by looking at the decisions regarding the null hypothesis that resulted from the statistical testing, it is clear that the TAT-based model behaves differently. In fact, by crossing all the mentioned tables, we are able to verify the following:

- for n-grams sized 3, TAT's version B has the best performance, which is stated in the rejecting of the null hypothesis in Table 4.8;
- all the hypotheses regarding the tests with 4 sized n-grams were rejected, exception made to TAT version A versus TAT version B accuracy: Table 4.6 shows very close results.
- regarding n-grams sized 5, once more the t test corroborate what Table 4.6 shows: that although the Naïve-Bayes is more accurate, TAT version A performs better; yet, there is no statistical difference between TAT version A and B.

From Tables 4.8, 4.9 and 4.10 we are able to verify that whether with TAT version A

or B, for all n-gram sizes, we can not state they have equal F-score results, even in Table 4.8, the null hypothesis regarding the test between the F-scores of the Multinomial Naïve-Bayes and TAT's version A was rejected — albeit the only one. In terms of Accuracy, we can see that with exception to n-grams size 3, there is no reason for us to state both methods are similarly accurate.

As far as comparing the two TAT versions implemented (see Subsection 3.3.2), we could verify there is only difference in the results when considering pre-processing the e-mails into n-grams sized 4 which according to Table 4.9 tells us their F-scores are different — Table 4.6 shows a clearly higher value of F-score for the version A. Other than this case, both methods seem to have identical performance and Accuracy.

The fact that the TAT-based filter achieved overall better results than the Naïve-Bayes meets our expectations, although it could be argued that the Naïve-Bayes performed below what has been described in the literature [4, 51] — this can be explained by the previously mentioned results deriving from the pre-processing method applied, which was the same for all datasets submitted to analysis.

However, TAT's accuracy does not seem to be good enough to over pass the Naïve-Bayes, at least not for n-grams sized 5. This could be explained by the rather large-size `Tcells` repertoire created during the training phase: all tokens (antigens) which do not match with a `Tcell` already in the repertoire will give birth to a new `Tcell`, which means all single antigens possible from all training messages will create a `Tcell`. Such solution creates a very big diversity of cells, increasing the probability of having false positives.

Chapter 5

Conclusions

In the beginning our main objective was the study of the feasibility of developing a spam filter inspired by the TAT theory. This was part of a team-work effort to bring to life a framework capable of classifying not just spam, but other data that may be interpreted as a self/non-self dichotomy — for example, another area where the TAT simulator is proving helpful is in the area of Network Intrusion Detection Systems [8, 9].

After successfully completing the implementation of an algorithm capable of replicating the dynamic characteristics of the TAT model [16], its performance was assessed by putting it face-to-face with a well known and proven (as well as controversial [54]) spam filtering Naïve-Bayes algorithm. Chapter 4 described and discussed the results achieved throughout different tests and provided a comprehensive statistical analysis of the results.

While the TAT-based model may have performed reasonably well in our tests — or at least, well enough to make us long for further developments — it can not escape our analysis that this model is, at the moment, working with configurations suggested by means of an optimizer, as described in Section 3.3.1. The possible combinations for the configuration parameters are endless and at some point, for this study, we had to choose the one that had provided the best results. Still, despite that feature, the TAT-based model proved our convictions that it could be a useful tool, in this case applied to spam filtering: from the results we obtained, and as explained in Chapter 4, using the same dataset, pre-processed in the same way, there is little difference between the TAT-based spam filter and our Naïve-Bayes implementation.

From this work, we have also learned the importance of the e-mail pre-processing

and how it can affect filtering e-mails — from opting between using n-grams or words, to a simpler choice of n-gram size, or which e-mail parts to analyze, all aspects contribute, positively or negatively, to the final result. We have also strengthened our conviction that human biology provided us with the conceptual tools for this work and although their implementation may not be as fine tuned as in human beings, we were still able to prove such tools work when applied to a different area of knowledge — albeit lacking the millions of years of human evolution to perfect our model. Like so, it is our belief that Biology might provide solutions to other problems in various areas.

The implementation of the TAT theory was successful and the TAT-based spam filter did meet our initial hopes and objectives, of becoming a filter at least as good as a Naïve-Bayes. We feel our work has contributed with another tool to the spam fighting cause as well as to help shed some light into one of the Immune System's mysteries.

5.1 Future Work

The TAT-based model presented in this work is one of the earliest versions. The results we have shown here are hopefully the first batch of many to be released, for there are many ideas to be implemented in our quest for excellent results both in spam filtering as in intrusion detection.

Some of the tests we look forward to performing put directly side-by-side the TAT-based spam filter with other well known methods and may feature other optimizations. We plan on addressing questions that this work revealed, such as the pre-processing differences between different approaches and which would be most suited for the TAT-based model, but also other issues, such as improving the `Tcell`'s proliferation mechanism, optimizing the decision process and refining the conditions that surround `Tcell`'s death.

Resembling a life-long history of infections and diseases, we are also planning on implementing a user supervised feedback mechanism, using false positives and allowing the user the possibility to correct the system's classification. False negatives would be treated as "vaccines", making the system more accurate.

For the future there are also plans to integrate the TAT spam filter with different systems — one idea is to develop a plug-in for SpamAssassin.

By means of working through these points and untying some "knots" we believe we will be able to achieve our goal of successfully implementing an immune system-based framework that may serve different purposes but also provide immunologists with information on how the TAT theory plays along in the Human Immune System.

Appendices

Appendix A

Optimizer Configuration Example

Here we present an example of a configuration file for the optimizer. Since the TAT simulator configuration will be generated having this one as a basis, many of the parameters are identical, specially the directory paths for files.

Worthy of note are some of the parameters related to Network Intrusion Detection System - the TAT simulator works as a framework for both NIDS and spam filtering and changing between the two types of analysis only requires some minor changes in the optimizer's configuration

```
[GERAL]
BASE_DIR:              /home/labrisio/ais
BASE_CONF:             /home/labrisio/ais/conf
BASE_DATASET:          /home/labrisio/ais/datasets
BASE_CONF_FILE:        /home/labrisio/ais/conf/aisim.conf
BASE_SCRIPTS:          /home/labrisio/ais/scripts
BASE_CLASS:            /home/labrisio/ais/build/classes
BASE_LOGS:             /home/labrisio/ais/Logs
BASE_RULES:            /home/labrisio/ais/rules
BASE_GRAPH:            /home/labrisio/ais/GraphicalTCells
BASE_HISTORY:          /home/labrisio/ais/HISTORY/
exec_opt:              /home/labrisio/ais/scripts/OPT/opt
LogResults:            /home/labrisio/ais/Logs/ReportOPT.log
backup:                true
rep_TCell:             TCell_Repertoire
```

```
[PRE_PROCESSING]
p_treino:              75
packet_size:           200

[ANOMALY GENERATOR]
number_ps:             1
network_ps:            192.168.2.0/24
n_attacks_treino:      600
n_attacks_teste:       20
n_unseen_teste:        100
rules_file:            all.rules
ficheiro_ATAQUE:       time_attack
ftester_conf:          ftest.conf
victim:                192.168.2.101
attacker:              192.168.2.106

[CATALOG]
catalog_dir:           /home/labrisio/ais/datasets/CATALOG
src_ip:                192.168.2.106
dst_ip:                192.168.2.101
src_mac:
dst_mac:
src_ip_new:            192.168.5.75
dst_ip_new:            192.168.5.92
src_mac_new:
dst_mac_new:
#src_ip:
#dst_ip:
#src_mac:
#dst_mac:
#src_ip_new:
#dst_ip_new:
#src_mac_new:
#dst_mac_new:

[COLLECT TRAFFIC]
interface:             eth0
```

```
filter:
treino_duration:        40
teste_duration:         120

[SIMULADOR GERAL]
lifeSpanAPC:            2
graphicTCellPath:       false
aislog:                 ais.log
alertLogFile:           LogXPTO3
reportMode:             ReportoryTeste
base64:                 false
kill:                   true
hammingAffinity:        true

[SIMULADOR TAT]
apc_duration:           1
peptide_len:            5
fo_x:                   30
fo_y:                   10
s0:                     5
kmax:                   10
kfi:                    18
rate_0:                 2
min_rate_fi:            0
max_rate_fi:            1
min_rate_max:           1
max_rate_max:           2
min_aff:                0
graphicTCellPath:       false
aislog:                 ais.log
alertLogFile:           LogXPTO3
reportMode:             ReportoryTeste
base64:                 false
kill:                   true
hammingAffinity:        true

[SIMULADOR TAT]
```

```
apc_duration:          1
peptide_len:           5
fo_x:                  30
fo_y:                  10
s0:                    5
kmax:                  10
kfi:                   18
rate_0:                2
min_rate_fi:           0
max_rate_fi:           1
min_rate_max:          1
max_rate_max:          2
min_aff:               0
max_aff:               1
min_apcT:              0
max_apcT:              1
min_kpT:               0
max_kpT:               1
min_T:                 0
max_T:                 0
init_best_obj:         999999
max_itera_obj:         200
penalty:               5000
```

Appendix B

TAT Configuration Example

This is an example of a configuration file for the TAT simulator. All the numerical values are suggested by the optimizer, with respect to the constraints imposed, as mentioned in Subsection 3.3.1.

```
;
executeTrainingMode;false
executeTestingMode;true
trainingInputData;false
testingInputData;/home/labrisio/ais/datasets/teste
trainingInputTCellFile;false
trainingOutputTCellFile;false
testingInputTCellFile;/home/labrisio/ais/datasets/TCell_Repertoire
testingOutputTCellFile;false
kMax;10
pMax;10.9264
k0;50
p0;54.632
s0;5
kFi;18
pFi;3.29319
token;
affinityThreshold;0.706112
lifeSpanAPC;2
timeAPC;1
```

```
graphicTCellPath;/home/labrisio/ais/GraphicalTCells/grafics.txt
alertLogFile;/home/labrisio/ais/Logs/LogXPTO3
aPCThreshold;0.168622
comiteKPThreshold;0.978295
ReportPath;/home/labrisio/ais/Logs/ReportoryTeste
base64;false
kill;true
hammingAffinity;true
KPTolerance;0
```

References

[1] 2006 TREC Public Spam Corpora. http://plg.uwaterloo.ca/~gvcormac/treccorpus0(Access date December 2007.

[2] Enron Email Dataset. http://www.cs.cmu.edu/~enron/. Access date March 2008.

[3] The SpamAssassin Project - Public Corpus. http://spamassassin.apache.org/publiccorpus/. Access date December 2007.

[4] Rocha LM. Abi-Haidar A. Adaptive Spam Detection Inspired by a Cross-Regulation Model of Immune Dynamics: A Study of Concept Drif. *Proceedings of 7th International Conferenc on Artificial Immune Systems (ICARIS 2008) - in press*, 2008.

[5] T. Abou-Assaleh, N. Cercone, V. Keselj, and R. Sweidan. N-gram-based detection of new malicious code. *Computer Software and Applications Conference, 2004. COMPSAC 2004. Proceedings of the 28th Annual International*, 2, 2004.

[6] I. Androutsopoulos, J. Koutsias, KV Chandrinos, G. Paliouras, and CD Spyropoulos. An evaluation of Naive Bayesian anti-spam filtering. *Arxiv preprint cs.CL/0006013*, 2000.

[7] AntiSpam - SpamUnit. Spam Statistics. http://www.spamunit.com/spam-statistics/.

[8] M.J. Antunes and M.E. Correia. TAT-NIDS: a novel immunological bio-inspired network intrusion detection system. *2nd International Workshop on Practical Applications of Computational Biology and Bioinformatics (IWPACBB 2008)*, 2008.

[9] M.J. Antunes, M.E. Correia, and J. Carneiro. Towards an Immune-inspired temporal anomaly detection algorithm based on tunable activation thresholds. *BioSignals 2009 - International Conference on Bio-inspired Systems and Signal Processing*.

[10] S. Atkins. Size and cost of the problem. *Proceedings of the Fifty-sixth Internet Engineering Task Force (IETF) Meeting*, pages 16–21, 2003. `http://www.ietf.org/proceedings/03mar/index.html`.

[11] H. Bersini. Immune System Modeling: The OO Way. *LECTURE NOTES IN COMPUTER SCIENCE*, 4163:150, 2006.

[12] E. Blanzieri and A. Bryl. A survey of anti-spam techniques. Technical report, Technical report# DIT-06-056. 2006.

[13] J. Brownlee. A Coarse Taxonomy of Artificial Immune Systems.

[14] F.M. Burnet. Immunological recognition of self. `http://www.nobel.se/medicine/laureates/1960/burnet-lecture.pdf`.

[15] J. Carneiro, K. Leon, I. Caramalho, C. van den Dool, R. Gardner, V. Oliveira, M.L. Bergman, N. Sepulveda, T. Paixao, J. Faro, et al. When three is not a crowd: a Crossregulation Model of the dynamics and repertoire selection of regulatory CD4+ T cells. *Immunological Reviews*, 216(1):48, 2007.

[16] J. Carneiro, T. Paixão, D. Milutinovic, J. Sousa, K. Leon, R. Gardner, and J. Faro. Immunological self-tolerance: Lessons from mathematical modeling. *Journal of Computational and Applied Mathematics*, 184(1):77–100, 2005.

[17] CAUCE - Coalition Against Unsolicited Commercial Emails. How do you define "spam"? `http://www.cauce.org/archives/37-How-do-you-define-spam.html`. Access date January 2008.

[18] WB Cavnar and J.M. Trenkle. N-Gram-Based Text Categorization. *Ann Arbor MI*, 48113:4001.

[19] V.G. CERF and R.E. KAHN. A Protocol for Packet Network Intercommunication.

[20] William R. Clark. *In Defense of Self: How the Immune System Really Works in Managing Health and Disease*. Oxford University Press, 2007.

[21] G. Cormack and T. Lynam. A study of supervised spam detection applied to eight months of personal email. *unpublished, July*, 1, 2004.

[22] L.N. De Castro and J. Timmis. *Artificial Immune Systems: A New Computational Intelligence Approach*. Springer, 2002.

[23] L.N. De Castro and F.J. Von Zuben. The clonal selection algorithm with engineering applications. In *Proceedings of GECCO'00, Workshop on Artificial Immune Systems and Their Applications*, volume 37. London, UK: Springer, 2000.

[24] R. Diacovo. Filtros de Spam. *Universidade Federal do Rio de Janeiro UFRJ COPPE Instituto Alberto Luiz Coimbra de Ps-Graduao e Pesquisa de Engenharia Programa de Engenharia de Sistemas e Computao Tpicos Especiais em Inteligłncia Artificial II*.

[25] EC Directive. 2002/58/EC of the European Parliament and of the Council of 12 July 2002 concerning the processing of personal data and the protection of privacy in the electronic communications sector (Directive on privacy and electronic communications). *Official Journal of the European Communities (31 July 2002 No L. 201, 2002)*.

[26] EC Directive. 95/46/EC of the European Parliament and of the Council of 24 October 1995 on the protection of individuals with regard to the processing of personal data and on the free movement of such data. *Official Journal of the European Communities (23 November 1995 No L. 281, 1995)*.

[27] H. Drucker, D. Wu, and V.N. Vapnik. Support Vector Machines for Spam Categorization. *IEEE TRANSACTIONS ON NEURAL NETWORKS*, 10(5), 1999.

[28] D. Evett. Spam Statistics. `http://www.spam-filter-review.toptenreviews.com/spam-` Access date January 2008.

[29] J. Fontana. Tallying the true cost of spam. *NetworkWorld.com*, 2003. `http://www.networkworld.com/news/2003/1117spamcost.html?page=1`.

[30] S. Forrest and S.A. Hofmeyr. Immunology as Information Processing. *Design Principles for the Immune System and Other Distributed Autonomous Systems*, 2000.

[31] S. Forrest, AS Perelson, L. Allen, and R. Cherukuri. Self-nonself discrimination in a computer. In *Research in Security and Privacy, 1994. Proceedings., 1994 IEEE Computer Society Symposium on*, pages 202–212, 1994.

[32] A. Gilbert. Developing nations losing spam battle, report says. *ZDNet News*, 2005. http://news.zdnet.com/2100-1009_22-5723435.html.

[33] RA Goldsby, TJ Thomas, BA Osborne, and J. Kuby. *Immunology fifth edition*. WH Freeman and Company, 2003.

[34] Google. Gmail - So much time, so little spam. http://mail.google.com/mail/help/fightspam/spamexplained.html. Access date January 2008.

[35] P. Graham. A plan for spam, 2002. http://www.paulgraham.com/spam.html.

[36] P. Graham. Better Bayesian Filtering, 2003. http://www.paulgraham.com/better.html.

[37] Z. Grossman and WE Paul. Adaptive cellular interactions in the immune system: the tunable activation threshold and the significance of subthreshold responses. *Proceedings of the National Academy of Sciences of the United States of America*, 89(21):10365, 1992.

[38] M. Häggström. Hematopoiesis. http://en.wikipedia.org/wiki/Hematopoiesis. page visited in March 2008.

[39] A.J. Hall, L.J. Yee, and S.L. Thomas. Life course epidemiology and infectious diseases. *International Journal of Epidemiology*, 31(2):300–301, 2002.

[40] R.W. Hamming. Error detecting and error correcting codes.

[41] R.W. Hamming. *Coding and information theory*. Prentice-Hall, Inc. Upper Saddle River, NJ, USA, 1986.

[42] CA Janeway, P. Travers, et al. *Immunobiology: The Immune System in Health and Disease*. Garland Science Publishing, 2005.

[43] CA Janeway, P. Travers, et al. *Immunobiology: The Immune System in Health and Disease*. Garland Science Publishing, 2005. 242-243.

[44] J.O. Kephart. A Biologically Inspired Immune System for Computers. *Artificial Life IV: proceedings of the fourth international workshop on the synthesis and simulation of living systems*, 1994.

[45] J. Kim and PJ Bentley. Towards an artificial immune system for network intrusiondetection: an investigation of clonal selection with a negative selection

operator. *Evolutionary Computation, 2001. Proceedings of the 2001 Congress on*, 2, 2001.

[46] J. Kim, P.J. Bentley, U. Aickelin, J. Greensmith, G. Tedesco, and J. Twycross. Immune system approaches to intrusion detection–a review. *Natural Computing*, 6(4):413–466, 2007.

[47] R. Kohavi. A Study of Cross-Validation and Bootstrap for Accuracy Estimation and Model Selection. In *INTERNATIONAL JOINT CONFERENCE ON ARTIFICIAL INTELLIGENCE*, volume 14, pages 1137–1145. LAWRENCE ERLBAUM ASSOCIATES LTD, 1995.

[48] B.M. Leiner, V.G. Cerf, D.D. Clark, R.E. Kahn, L. Kleinrock, D.C. Lynch, J. Postel, L.G. Roberts, and S.S. Wolff. The Past and Future History of the Internet. *COMMUNICATIONS OF THE ACM*, 40(2):103, 1997.

[49] A. Lev. THE MARRIAGE OF SPAM AND MALWARE: IMPLICATIONS FOR SMTP MALWARE DEFENCE.

[50] B. McWilliams. *Spam kings*. O'Reilly Sebastopol, Calif, 2005.

[51] Androutsopoulos I. Metsis V. and Paliouras G. Spam filtering with naive bayes–which naive bayes. *Third Conference on Email and Anti-Spam (CEAS)*, 2006.

[52] D. L. Nelson and M. M. Cox. *Lehninger Principles of Biochemestry*. Worth Publishers, 3rd edition, 2000. page 70.

[53] D. L. Nelson and M. M. Cox. *Lehninger Principles of Biochemestry*. Worth Publishers, 3rd edition, 2000. page 263.

[54] C. O'Brien and C. Vogel. Spam filters: bayes vs. chi-squared; letters vs. words. *Proceedings of the 1st international symposium on Information and communication technologies*, pages 291–296, 2003.

[55] T. Oda. *A Spam-Detecting Artificial Immune System*. PhD thesis, Carleton University, 2005.

[56] R.E. Overill. Computational immunology and anomaly detection. *Information Security Technical Report*, 12(4):188–191, 2007.

[57] P. Pantel, D. Lin, and M. Winnipeg. SpamCop: A Spam Classification & Organization Program.

[58] RC Paton. Towards a metaphorical biology. *Biology and Philosophy*, 7(3):279–294, 1992.

[59] J.P. Pedroso. Simple Metaheuristics Using the Simplex Algorithm for Non-linear Programming. *LECTURE NOTES IN COMPUTER SCIENCE*, 4638:217, 2007.

[60] P. Pereira, L. Forni, EL Larsson, M. Cooper, C. Heusser, and A. Coutinho. Autonomous activation of B and T cells in antigen-free mice. *Eur J Immunol*, 16(6):685–8, 1986.

[61] J.D.M. Rennie. Derivation of the F-measure. *In other words*, 1:4.

[62] M. Sahami, S. Dumais, D. Heckerman, and E. Horvitz. A Bayesian approach to filtering junk e-mail. *Learning for Text Categorization: Papers from the 1998 Workshop*, 62, 1998.

[63] J. Schneider. K-Fold Cross Validation. `http://www.cs.cmu.edu/~schneide/tut5/node4` Access date August 2008.

[64] A. Schwarz. *SpamAssassin-The Open Source Solution to SPAM*. OReilly, 2004.

[65] A. Somayaji. Immunology, diversity, and homeostasis: The past and future of biologically inspired computer defenses. *Information Security Technical Report*, 12(4):228–234, 2007.

[66] DS Soper. Daniel Soper Statistics Calculator. `http://www.danielsoper.com/default.aspx`. Access date July 2008.

[67] Spamhaus Project. The Definition of Spam. `http://www.spamhaus.org/`. Access date January 2008.

[68] J. SpammerX, Posluns. *Inside the Spam Cartel*. Syngress Publishing, Inc, 2004.

[69] Task Force on Spam. DSTI/CP/ICCP/SPAM(2005)6/FINAL. 2005.

[70] R Tomlinson. The First Network Email. `http://openmap.bbn.com/~tomlinso/ray/firstemailframe.html`. Access date July 2008.

[71] K. Tretyakov. Machine Learning Techniques in Spam Filtering. *Data Mining Problem-oriented Seminar, MTAT*, 3:60–79, 2004.

[72] UNIS - United Nations International School. http://showcase.unis.org/UNIScienceNet/IBHbio2_frameset.html. Access date March 2008.

[73] US Senate and House of Representatives. Controlling the Assault of Non-Solicited Pornography and Marketing Act of 2003 (CAN-SPAM 2003). November 2003. http://www.spamlaws.com/pdf/pl108-187.pdf.

[74] Wikipedia. Arpanet. http://en.wikipedia.org/wiki/ARPANET. Access date January 2008.

[75] Wikipedia. Internet. http://en.wikipedia.org/wiki/Internet. Access date January 2008.

[76] Wikipedia. Kevin Mitnick. http://en.wikipedia.org/wiki/Kevin_Mitnick. Access date January 2008.

[77] Wikipedia. N-gram. http://en.wikipedia.org/wiki/N-gram. Access date January 2008.

[78] Wikipedia. Peter Steiner's Cartoon. http://en.wikipedia.org/wiki/On_the_Internet Access date August 2008.

[79] Wikipedia. Thomas Bayes - Bayes and Bayesianism. http://en.wikipedia.org/wiki/Thomas_Bayes. Access date January 2008.

[80] Wikipedia. Timeline of Hacker History. http://en.wikipedia.org/wiki/Timeline_of_hacker_history. Access date January 2008.

[81] Y. Yang and X. Liu. A re-examination of text categorization methods. In *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 42–49. ACM New York, NY, USA, 1999.

[82] X. Yue, A. Abraham, Z.X. Chi, Y.Y. Hao, and H. Mo. Artificial immune system inspired behavior-based anti-spam filter. *Soft Computing-A Fusion of Foundations, Methodologies and Applications*, 11(8).

[83] B. Zafer. The PDF List of Standard Amino-Acids. http://www.unc.edu/~bzafer/aminoacids/. Access date March 2008.

[84] J.H. Zar. *Biostatistical analysis*. Prentice Hall Upper Saddle River, NJ, 1999.

[85] J. Zdziarski. Bayesian noise reduction: Contextual symmetry logic utilizing pattern consistency analysis. In *Proceedings of the MIT Spam Conference, Cambridge, MA, USA, January*, 2005.

[86] J.A. Zdziarski. *Ending Spam: Bayesian Content Filtering and the Art of Statistical Language Classification*. No Starch Press, 2005.

[87] H. Zhang. The optimality of naive Bayes. *Proceedings of the Seventeenth Florida Artificial Intelligence Research Society Conference*, pages 562–567, 2004.