University of Porto
Faculty of Engineering

Email Classification: a case study

André Ricardo Azevedo Gonçalves da Silva

July 2016

Scientific Supervision by
Ademar Aguiar, Assistant Professor
Department of Informatics Engineering

Scientific Co-supervision by
Hugo Ferreira, Assistant Professor
Department of Informatics Engineering

In partial fulfillment of requirements for the degree of
Master in Computer Engineering
by the EUR-ACE Programme

Contact Information:

André Ricardo Azevedo Gonçalves da Silva
Faculdade de Engenharia da Universidade do Porto
Departamento de Engenharia Informática

Rua Dr. Roberto Frias, s/n
4200-465 Porto
Portugal

Tel.: +351 913 152 186
Email: ei12133@fe.up.pt
URL: https://www.linkedin.com/in/andreragsilva

Acknowledgments

I must start by thanking my family for giving me the opportunity to reach this big step in my life. Starting with Maria da Gloria for being the best grandmother one could ever have, by listening to all my thoughts and nodding the head even when she was not follow anything I said, not forgetting all the lunches and steaks cooked in favour of my greediness. Following with António Silva, Alexandra Silva and Andria Silva for all the noise made while one was working and for being great parents and sister. One could never repay you for this opportunity, but one can show its gratefulness by continuing to grow as a person and as an engineer, thriving through the game of life.

A special thanks goes to Ana Teresa Amorim for all the patience, caring, interest, confidence and motivation showed during the process of this work. Just like my family, one could not ask for a better girlfriend. Thank you for everything, specially the house that served as the perfect work place during the rough times. All the snacks and meals too!

Finally, one big hail to the person responsible for supervising my dissertation work and for pointing me in the right direction when needed, Professor Ademar Aguiar. Another hail goes to my co-supervisor, Professor Hugo Sereno Ferreira for all the expertise in machine learning. The help provided was fundamental in structuring the dissertation approach to this specific problem.

André Gonçalves Silva

This page was intentionally left mostly blank.

...to everyone who believes in me

This page was intentionally left mostly blank.

Abstract

Internet dependence on email has been frequent since its early days. In the present days, electronic mail is widely used in a professional and personal context. Although this service was developed as a way of communication, nowadays it serves many other purposes. The majority of services available online require an email address in order to authenticate or as a bridge of communication between the user and the service.

The average number of emails sent and received, by the average user, is in the order of the hundreds per day, and these emails can be of varying categories: social, professional, notifications, marketing, transactional, emails which warrant no response, emails to send files, emails requiring response, among others with different purposes. This originates an information overload problem, that proves difficult to be completely solved manually by the email address owner.

Therefore, there is a growing need to develop systems that can automatically learn and recommend users effective ways to organize their email information, which can aggregate emails into user defined groups, expediting the process of reading and consulting the mailbox. To alleviate this information overload problem there are several possible approaches and techniques, such as machine learning to help on email classification, in order to store new emails in the best fit folder of the massive inboxes we all have, now or in the future.

After a careful review of the state of the art on different email text classification approaches, this work elaborates on a modular system that is capable of several preprocessing configurations and takes advantage of a classifiers ensemble, in order to better solve the problem of email classification.

Afterwards, the system is be adapted to a very concrete case study, a desktop email client under development at Mailcube Lda. The case study tests and analyse different pre-processing configurations using three text classifiers for several users mailboxes from the Enron Corpus dataset. The final results are compared with work from the scientific community with identical configuration as validation.

At the end, is expected that the resulting system adapts well to the case study, automatically suggesting the user where to store the incoming messages, continuously

adapting to the arrival of new emails, improving the overall user experience and saving precious time for the users.

Resumo

A dependência da Internet no email tem se mantido constante desde os seus dias iniciais. No presente, o correio eletrónico é bastante utilizado tanto em contexto pessoal, como profissional. Embora o serviço tenha sido desenvolvido com o objetivo de servir como um meio de comunicação, hoje em dia este serve muitos outros propósitos. A maioria dos serviços disponíveis online utilizam o endereço de email como meio de autenticação ou como uma ponte de comunicação entre o utilizador e o serviço.

Em média, o número de emails enviados e recebidos, pelo utilizador corrente, encontra-se na ordem das centenas por dia e estes emails podem ser das mais diversas categorias: social, profissional, notificações, publicidade, transações, emails que não requerem resposta, emails como meio para enviar ficheiros, emails que requerem resposta, entre outros com diferentes propósitos. Toda esta diversidade encontra-se na origem de um problema de informação excessiva, difícil de resolver manualmente pela pessoa responsável pelo endereço eletrónico.

Como tal, existe uma crescente necessidade de desenvolver sistemas que sejam dotados de aprendizagem automática, capazes de recomendar ao utilizador formas eficientes de organizar a informação presente nas suas contas de email e ainda agregar os emails em grupos manualmente criados, de forma a facilitar a sua interpretação por parte do utilizador, agilizando todo o processo de leitura e consulta da caixa de correio eletrónica. Para atenuar o problema de informação excessiva, existem diversas abordagens e técnicas, como aprendizagem de máquina, para ajudar na classificação dos emails, de forma a atribuir automaticamente novas mensagens a pastas da massiva caixa de correio que todos temos, agora e no futuro.

Após uma revisão cuidada ao estado da arte sobre as diferentes abordagens existentes para a classificação de email recorrendo ao conteúdo de texto, este trabalha trata de apresentar um sistema modular, capaz de diferentes configurações de pré-processamento e que tira partido de um sistema que faz uma ponderação do resultado de diversos classificadores, para chegar a uma classificação final, com o intuito de obter melhores resultados para o problema de classificação de correio eletrónico.

De seguida, o sistema é adaptado a um caso de estudo muito especifico, um cliente de

email em desenvolvimento na empresa Mailcube Lda. O caso é testado e analisado para diferentes configurações, usando três classificadores e recorrendo a sete caixas de correio provenientes do corpo de dados do Enron Corpus.

No fim, é esperado que o sistema criado se adapte harmoniosamente ao caso em questão, sugerindo ao utilizador onde guardar as mensagens que vão chegando e ainda que seja capacitado de se adaptar continuamente aos emails que vão entrando na conta. Com este sistema, é expetável que a experiência de utilização melhore, rentabilizando o tempo dos utilizadores.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Users' mailboxes keep growing in size due to all the electronic correspondence that is exchanged between people and services. The same email account, is usually used for many different purposes: work and personal environment, events, tasks, subscribing services or newsletters, authentication and others. Different usages of email are continuously emerging. A user should not have the necessity to create several email accounts in order to compartmentalize the information, one should be able to filter and organize the data heterogeneity, contained in each email, to facilitate the reading and consulting interaction. Can machine learning, through the classification of unlabelled data and by adding the user interaction history to create new knowledge, help with the treatment of this information? A human does not have the ability to manually treat thousands of emails within a reasonable time, will the machine be able to do it, in a meaningful and valuable way for the user?

1.1 Email

Electronic mail is one of the most successful services yet created. In a simple way, email provides a bridge of communication between points over the network. It allows users to send messages from an email address to another, or multiple addresses, asynchronously. Email was actually created before internet creation, it became bigger within the military network operated, ARPANET and was used to exchange information and sharing and storage of documents. The most significant technical evolution of the service was to the protocols used to move emails between systems and it had to be completely replaced several times [1]. The interface of the mailboxes also changed to cope with the continuous evolution of design and with the appearance of new forms of email information.

It is one of the most popular services available on the internet and continues to grow mainly due to its high efficiency, its compatibility with many types of information, and low cost of usage and maintenance. Nowadays, email serves many different purposes that it was not initially design for, contributing to the service becoming one of the most used tools in the business and personal world, so far.

The worldwide usage of email is also the main cause of the problems still looking to be solved today.

1.2 Current Email Usage

As a broadly used technology, email is extensively used in our daily lives. To reinforce the success of electronic mail, the number of emails being exchanged has been in constant growing ever since the service appeared. The Radicati Group, Inc, a technology market research company based in Palo Alto, has been conducting studies on email since 1993, annually releasing a statistical report that provides information on the service usage, together with a forecast for the next four years.

Table 1.1 compiles email usage data present in the most recent report on email statistics, released by Radicati [2].

	2015	**2019**
Emails sent/received per day (B)	205.6	246.5
Worldwide email accounts (M)	4.353	5.594
Worldwide email users (M)	2.586	2.943

Table 1.1: Worldwide email traffic, accounts and users forecast

The report states that, by the end of 2019, around one-third of the worldwide population will be using email. It also shows that the number of accounts is expected to set a bigger

increase than the number of email users, mostly due to many consumers having multiple email addresses.

The main reason behind the escalating email usage is the increase of connectivity, with all the available technology in the market people can obtain access to the Internet from almost anywhere on the planet. The dependency on email from upsurge services like online shopping, banking, instant messaging, VOIP and many others that require an email to obtain access to the service, will also end up contributing to the continuous rise of email users.

Although the forecast indicates that the service will not stop expanding, many new applications, containing some of the email functionalities, keep urging. It may also lead to an evolution of the service, to catch up with the new technologies.

1.3 Information Heterogeneity

Initially, email was just used for plain text messaging. At the moment, it is still widely used has a tool of communication between people, although with many more formats and finalities.

Users tend to use electronic mail as an archiving tool for documents, conversations and other types of files like photos. The service is also being used to exchange tasks between users or by themselves, companies even use email to distribute work and assign jobs. Shopping services use email as an alert or notification tool for price changing. Airline companies also uses it to report flights changes to the user. Event emails integrate with the consumer calendar applications and can be shared among users with the finality of setting up a meeting or a trip, for example.

Marketing, publicity and newsletter emails, that the user subscribes to, are more diversified and graphical. Many of the spam[1] continues to pass the security filters, ending up creating more trash in the users' mailboxes, augmenting the overload problem.

Email may also possess a status, it can have a pending action to be performed such as, to be read or to be answered. The to do emails are inserted in the domain of task/action emails.

1.4 Overload Problem

The email overload problem was initially tackled in a paper by Whittaker and Sidner, back in 1996 [4]. The term was given as a result of the enormous quantity of emails, users

[1] spam was defined by Cormack and Lyam, in 2005, as being "unsolicited, unwanted emails that were sent indiscriminately, directly or indirectly, by a sender having no current relationship with the recipient" [3]

had in their mailboxes, leading to a time consuming problem, since they had to spend more time digging their mailboxes.

Ten years later, Microsoft researchers Fisher et al. [5] decided to compare their email reality with 1996 and they realised that mailboxes did not change much in size, but the number of email users had grown. Consequentially, the overload problem was still present, but in a wider range, since more people were using the service.

More recently, in 2014, Grevet et al. [6] did a research that ended up also concluding that the problem still persists and it has evolved into a two faced problem. On one side you have overload, meaning a large volume of emails in the users' mailbox and, on the other side, you have the term being justified with the many different types of email status, such as to do, to read, to answer, etc.

Only two years have passed and the problem is still on sight. It also seems that the problem is not just the number and the different status of emails, the variety of emails described in section 1.3 also creates an overload of information due to the data being so diversified.

Summing up, the overload problem can be created from mailboxes having an enormous number of emails, many different states and activities over it (to do), or many different types of content.

1.5 Mailcube

The dissertation was developed in Mailcube, Lda. based in Porto, Portugal.

Mailcube is a privately held startup company currently developing an email client for desktop. The purpose of this product is to improve the email workflow by making it easier for users to consult, read and manage their mailboxes.

The application focuses on improving the experience and reduce the time and patience people need to put in email management by using new concepts such as faces, cubes and contexts. These cubes work as storage boxes for the mailbox correspondence and faces are a way to visualise and slice information from each email in the cube.

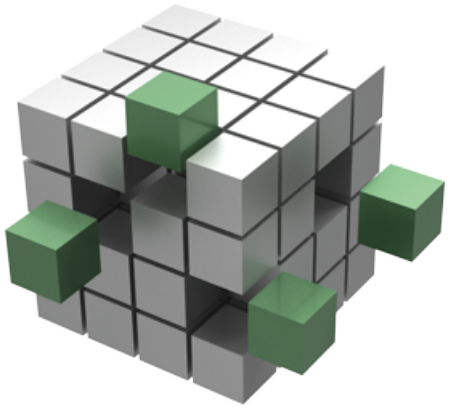

Figure 1.1: Graphical conceptualization - cube

This research aims to ensure that users do not need to do multiple clicks to organize the emails in cubes, the system will do it automatically and the users can overpass the system by deliberated manual actions.

1.6 Motivation

As described in section 1.4, the overload problem still persists and with the crescent usage of email and also its variety, information problems will become even more serious. Due to emails from different categories and types continuing to pile up the users' mailboxes, making it harder and harder to use it, important messages will be lost or delayed to be answered since many people use the mailbox as documents and files storage. The retrieval will also be more difficult and tasks will not be done. In general, problems of miscommunication and time consumption will be aggravated.

With so much correspondence being exchanged, users will not be able to efficiently deal with the electronic mail clog manually, therefore the necessity of creating a system that can automatically organize all the mixture of information that continues to arrive.

Some email clients, like Gmail from Google, try to filter emails into smart folders such as forums, social, updates, etc., but if you are someone who works in the social area, which is a growing area, this will not be enough because your social folder will also be filled with messages from different places and with divergent purposes, leading to the same problem. Also, they miss to adapt to each user mailbox which differ in variety and size, if one does not have emails from forums, one does not have the necessity of a forum folder, capability of adaptation is very important.

Mailcube is a new mail client with a different concept from the rest of the email clients currently available. Mixing a new concept with the ability to self organize mailboxes and

the capacity of adaptation to each individual, is one step further in the email management that brings value and a better and less frustrating experience for the users.

1.7 Goals

This dissertation main goals start with the creation of a system that is able to classify the new emails for a user. This classification must be significantly correct and fast, reducing user frustration and augmenting the productivity.

Second, the system needs to be sensitive to user interaction, adapting with the deliberated changes the user will impose to it.

Third, the system needs to be easily integrated with other applications and consume as less resources as possible.

Finally, the resulting system must be successfully validated using a case study, by Mailcube desktop application.

1.8 General Approach

Due to the nature of the problem, a supervised learning algorithm to perform text classification is used. It takes advantage of the already stored emails, belonging to the user, to create and update the artificial intelligence knowledge model with the incoming correspondence and with the users manual actions, such as drag a message to a different folder.

All the textual contents of the messages may be used to boost the classification output.

1.9 Expected Results

The resulting system uses an intelligent approach that adapts differently to each user mailbox and improves the more it is used. More specifically, the system classification was significantly increased from other approaches, producing a more accurate output leading to a less frustrating experience. With all this is expected that the email reading and consulting will be much easier and time consumption will be diminished. Communication and productivity will increase and fewer important messages will be lost, forgotten, or simply wrongly organized.

1.10 How to Read this Dissertation

The remainder of this dissertation is organized into three parts. For a more comprehensive understanding on this study, further reading should follow the bellow presented structure:

Part 1: Background & State of the Art provides a better understanding of commonly used concepts and elaborates on studies relevant to the dissertation:

- Chapter 2, "Machine Learning and Email"(p. 9), offers an extensive survey on supervised learning algorithms, preprocessing techniques and datasets available for email classification, followed by a review on multiple approaches that implement them.

Part 2: Problem & Solution clarifies the research problem and challenges, followed by the proposed solution:

- Chapter 3, "Research Problem"(p. 19), covers the general problem, elaborating on the specific research topics of this study.

- Chapter 4, "Proposed Solution"(p. 23), presents the resulting system, elaborating on its architectural division and specifications.

Part 3: Validation & Conclusions presents the case study validation and the dissertation conclusions.

- Chapter 5, "Mailcube Application: Case Study"(p. 31), covers the conducted studies on multiple scenarios, created to evaluate the classification performance while following the case restrictions, finalizing with a results comparison with approaches from the scientific community.

- Chapter 6, "Conclusions"(p. 45), describes the main contribution and conclusions of this dissertation, and provides a direction for future work.

Chapter 2

Machine Learning and Email

Machine learning, according to Mohri, Rostamizadeh and Talwalkar [7], is defined as "computational methods using experience to improve performance or to make accurate predictions", where experience is "the past information available to the learner, which typically takes the form of electronic data collected and made available for analysis".

This chapter gives a better understanding on the email composition, the multiple techniques of preprocessing and representation used to adapt emails into a format that is more suitable to the machine learning algorithms and the most used classifiers and datasets are also enumerated. The state of the art related to email on the different topics are also covered.

This dissertation research is not related to data mining, some of the algorithms described in this chapter may be shared across both these major areas, due to the solving problem's nature.

2.1 Email Format

At the present days, email text messages format is defined by the RFC 5322 [8] and is extended by the internet standard MIME [9, 10, 11, 12, 13, 14], in order to support non-ASCII text, non-text attachments (image, audio, video, application, messages, etc.) and multi-part message body support.

The research reviewed in this dissertation, takes advantage of a few fields from the email format structure.

2.2 Email Preprocessing

The email preprocessing is a step that processes the raw email content and helps cleaning, normalizing and transforming the input data. It is one fundamental step that can change completely the classifier outcome and is usually applied when one wants to remove noise from the terms that are extracted in order to send only the more relevant ones to the algorithm.

2.2.1 Word Tokenizer

A word tokenizer breaks text into word tokens. These tokens are separated for characters that are called delimiters. These delimiters usually are punctuation marks, digits, symbols such as the at symbol (@), space character (␣) and escape codes like newline and others.

Table 2.1 displays some examples of the word tokenizer. Given an input text, it shows the output array of tokens for the used delimiters.

Delimiters	Input	Output
␣,.@	My␣new␣email,␣is␣john@mail.com	My, new, email, is, john, mail, com
␣!-	My␣number␣is␣San-Francisco-93!	My, number, is, San, Francisco, 93
␣!?012	My␣username␣is␣alpha2001!?	My, username, is, alpha

Table 2.1: Word tokenizer examples

2.2.2 N-Gram Tokenizer

N-gram tokenizer creates tokens with n words from a text sentence, all the delimiters mentioned in the word tokenizer 2.2.1 are ignored when creating tokens. Normally the algorithms implementation support the definition of a minimum and maximum n value, leading to repeated words inside different tokens.

Table 2.2 displays some examples of the n-gram tokenizer using different simple configurations. Given an input text, it shows the output array of tokens for the used configuration.

In 2015, Alsmadi and Alhami [15] conducted a study where they compare several clustering and classification approaches for email contents and show that N-Gram treatment, in large multi-linguistic collections, gives the best result.

n-min:n-max {Delimiters}	Input	Output
1:2 {,}	My name is, John	My name, name is, is John, John
2:2 {,}	My name is, John	My name, name is, is John
3:3 {,}	My name is, John	My name is, name is John

Table 2.2: N-Gram tokenizer examples

N-Gram models may also be used for frequency analysis, speech recognition and many other applications.

2.2.3 Stop Words Removal

The stop words removal technique uses the occurrence of words in documents to represent them. Stop words are known to be the most common words of a language. Usually they are removed from the data representation because they normally serve a grammatical function, therefore adding little value for pattern matching and content identification. Both tokenizers mentioned (2.2.1, 2.2.2) can integrate a stop words removal in order to ignore certain words when creating tokens.

In 1992, Wilbur and Sirotkin [16] published a technique that automatically identifies the high percentage of words in documents, by measuring document-document similarity, that can be identified as stop words.

Kumar et al. [17] proposed a framework for email clustering that removes the stop words - defined manually as a list of words - previously to the algorithm's execution, reducing significantly the number of words to describe each email, making similarity analysis more accurate.

Alsmadi et al. [15] also presented the different results with and without stop words removal, proving that removing the stop words changes the top 100 most frequent words completely, resulting in a significantly accuracy increase.

2.2.4 Stemming

Stemming is a technique that tries to reduce words to their stem form. In order for the algorithm to understand how the fundamental word is composed, usually the stemming

needs to be implemented with a dictionary from a specific language that helps in finding the different words stem form.

A stemming algorithm reduces the words "stems", "stemming", "stemmer" and "stemmed" to their stem word "stem".

The first English stemming algorithm was introduced in 1968 by Julie Lovins [18], and it was already able to reduce all words to a common reduced form. Latter, in 1980, Martin Porter [19] created an English stemming algorithm that has a new way of automatically remove suffixes from words, not resorting to a set of rules has Lovins.

2.3 Feature Representation

In order to apply machine learning techniques to datasets of emails, the data present in them needs to be represented in a way that is more comprehensible for the algorithms.

The present section describes the techniques more frequently used in the area of ML, for data representation.

2.3.1 Vector Space Model

Salton, Wong and Yang [20], in 1975, proposed an algebraic model that allows to represent objects in a vector of identifiers for the object in question. Each of the vector dimension, represents a feature of the object. Later, in 1988, Salton et al. [21] presented a term-weighted approach that adds a weight value to each term of the 1975 model, this approach proved to obtain better results, heavily depending on the selected weighting system, as stated by the authors "These results depend crucially on the choice of effective term weighting systems".

2.3.2 Bag-of-Words Model

The BoW model represents documents data as a set of unordered words with the corresponding frequency of each others. Citing Turney and Pantel [22], "The bag of words hypothesis is the basis for applying the VSM to information retrieval" [20].

2.3.3 Term Frequency-Inverse Document Frequency

TF-IDF is a weighting technique created in 1972, by Spärck Jones [23], that is being widely used in the scientific community. This technique tries to give more importance to unexpected changes than expected ones. Turney and Patel [22] refer to this approach as being "The most popular way to formalize this idea for term-document".

In 1996, Singhal et al. [24] present another weighting technique based on TF-IDF called length normalization. Since TF-IDF tends to favour longer documents, this approach takes into account the analysed document length, in order to normalize the results.

In 1999, Segal and Kephart [25] presented MailCat, an assistant for organizing email and they use the TF-IDF before applying the text classifier.

2.4 Feature Selection

Feature selection is an important step of classification, what it does is to reduce the number of features provided to the classifiers by means of an evaluator and a search algorithm. The objective is to remove attributes that are not so relevant to the classification process, sometimes increasing the performance and decreasing the execution time consumption.

2.4.1 Evaluators

The evaluators or scoring algorithms take care of creating a weighted analysis of a certain attribute relatively to its class. Some algorithms resort to the information entropy function H(X) or others, using multiple combinations to offer different evaluation scenarios.

Information Gain evaluates the amount of chaos reduction in terms of information entropy. It tries to find groups of attributes with the most information gain to the correspondent class.

$$InfoGain\left(Attribute, Class\right) = H\left(Class\right) - H\left(Attribute|Class\right)$$

Gain Ratio is a modification of the information gain evaluator that reduces its tendency towards multi-valued attributes.

$$GainRatio\left(Attribute, Class\right) = \frac{InfoGain(Attribute,Class)}{H(Attribute)}$$

Correlation evaluators calculate the correlation between a feature and its class, by means of the Pearson product-moment correlation function. Guyon and Elisseeff [26] article offers an extensive clarification on the subject. In 1998, Mark Hall [27] introduced a now known correlation-based evaluator has his thesis project.

2.4.2 Search Algorithms

The search algorithm is responsible for retrieving and select the attributes distribution, based on the evaluators score. In 2010, Richard Korf [28] wrote an article that gives a further insight on search algorithms.

Greedy algorithms may also be used as a search algorithm. They follow the greedy theory which is to evaluate an heuristic at each stage, in order to select the most promising local path to follow next.

Best-first is an algorithm that uses an heuristic function to discover the best attributes. In 1993, Korf [29] presented a linear-spaced version of the algorithm, reducing the space complexity of the algorithm.

Ranker is a simple ranking algorithm that ranks the attributes by their evaluation. A fine suit for any evaluator.

2.5 Classification Algorithms

Classification is considered an instance of supervised learning. It tries to categorize a new observation into a predefined set of categories. The classifiers can be of different types, not exclusively, such as:

- Linear (probabilistic or not);
- Support vector machines;
- Kernel estimation;
- Decision trees.

In this work case study simulation, specifically the Section 5.3, the first three algorithms presented are analysed and compared for multiple preprocessing techniques combinations.

2.5.1 Naive Bayes

Naive Bayes is a probabilistic classifier that is based on the Bayes' theorem. It was described by Lewis as being "long a favorite punching bag of new classification techniques" [30] and it was still the most used text classification technique.

In 2000, Jason Rennie [31], presented an application for email filtering, called ifile, that uses Naive Bayes to perform the classification.Training is done based on a predefined dataset of emails with specific folders and, as the mail arrives, they are attributed to a folder from the learning set.

Rennie et al. [32] tried to fight the negative assumptions of Naive Bayes as a text classifier by presenting heuristics that can handle some of the algorithm problems. In 2004, Zhang [33] presents a new explanation for the algorithm and shows the "superb classification performance" of the same.

2.5.2 Support Vector Machine

SVM was first introduced in 1995, by Cortes and Vapnik under the name, support vector networks [34].

In 1998, John Platt [35] presented an algorithm for training support vector machines, capable of achieving fast execution times. The approached was denominated as Sequential Minimal Optimization (SMO). In 2001, Keerthi et al. [36] published an article with improvements to the algorithm, making it more able to perform classification tasks.

Drucker, Wu and Vapnik [37] studied the validation of the algorithm for email spam categorization and even compared with other algorithms realizing that, for binary decisions, SVM is the best, for more dimension it is similar in accuracy and speed. In 2002, Tong and Koller [38] presented a modified version of SVM that is able to do active learning, this means it does not use a selected training set to learn. This is useful if such approach is to be integrated into an application.

Shawe-Tayler and Sun [39], presented a review of the state-of-the-art on techniques for optimizing the training of SVM classifiers.

The algorithm was also reviewed and tested using a library in the research by Aslamadi et al. [15].

2.5.3 Random Forest

Random Forest was created by Leo Breiman [40] as a decision tree classifier. It was based on the random selection of features discussed by Ho [41][42] and also by Amit and Geman [43].

In 2004, Rios and Zha accomplished a research where they compared SVM with Random Forest for spam detection [44]. They concluded that both algorithms have similar performances.

2.5.4 K-Nearest Neighbour

Cunningham and Delany [45], did a more detailed review of the algorithm as a classifier. They also state that it "is very simple to understand and easy to implement", therefore "it should be considered in seeking a solution to any classification problem" and they follow up with an enumeration of the algorithms advantages.

2.6 Ensembling

Ensembling is the process of using methods for combining multiple classifiers, in order to obtain better classification results. The book from Kuncheva [46] presents a very complete

bibliography on methods for combining classifiers.

2.6.1 Bagging

Bagging, or bootstrap aggregation, was initially proposed in 1994 and published in 1996, by Leo Breiman [47]. The method extracts multiple samples from the training data and creates new classifiers for each sample. The multiple classifiers are aggregated and the average is calculated in order to provide a classification (other rules, beside the average, may be used to combine the distributions). This ensemble method tries to improve the accuracy and reduce the variance of statistical classification algorithms.

2.6.2 Boosting

Boosting is an ensemble method that starts with any base classifier that is trained and then resorts to another classifier to better classify instances, from the training data, that the first classifier could not properly classify. The process may replicate the creation of classifiers until a certain stopping criteria is reached, a fixed number of models or a specific accuracy are usually imposed.

The AdaBoost algorithm initial implementation was presented in 1995, by Freund and Schapire [48] and is, by far the most popular boosting algorithm and was even awarded with the Gödel Prize. After presenting it, the authors made a second paper [49] where they compare the algorithm with bagging method. Latter, in 1999, Freund et al. [50] published a short paper describing the concept of boosting and the AdaBoost more clearly.

2.6.3 Voting

Voting is the process of combining multiple classifiers probability distribution using a combination rule, in order to estimate a classification. The most commonly used combination methods are:

- Majority voting
- Probabilities average
- Probabilities product
- Minimum or maximum probability

This work approach main contribution takes advantage of a voting ensemble using the minimum probability combination rule. The results can be consulted in Chapter 5, more specifically from Section 5.4 to 5.8.

2.6.4 Stacking

Finally, the stacking method, combines multiple classifiers by resorting to the stacking generalization introduced to the scientific community in 1992, by Wolpert [51].

2.7 Available Datasets

Due to privacy issues involved with the contents of email messages, large and realistic email corpora is very hard to find.

Popular email datasets used through the scientific community are enumerated. The use of such databases, in several approaches, creates a bridge that enables performance analysis and comparison.

2.7.1 Enron Corpus

Enron Corpus is a popular email dataset used through the scientific community. Its usage creates a bridge that enables performance analysis and comparison between approaches.

In 2004, an article introducing the database was published by Klimt and Yang [52]. In the same year, a more detailed analysis on its "suitability with respect to email folder prediction" was released, by the same authors [53].

The dataset contains about half a million messages from one hundred and fifty different users. It is available, free of charge, at the Carnegie Mellon University Enron dedicated page [54].

This approach uses the Enron Corpus due to its popularity and the datasets used can be consulted in Appendix A.

2.7.2 TREC Public Spam Corpus

The Text Retrieval Conference (TREC) created an email corpus [55] to help with the creation of techniques to identify spam mail. The dataset contains around seventy five thousand emails, from which fifty are labelled as being spam.

With the creation of this database in 2005, the techniques elaborated by the scientific community started using the corpus as a reference and to enable results comparison between different approaches.

2.8 Approaches Using Enron Corpus

The present section enumerates several approaches for the email categorization problem, that take advantage of the Enron Corpus dataset to obtain results and compare them with

other research. The proposed solution is compared with some of the following research, by comparing the accuracy achieved, validating the system capacity in handling the problem being studied.

Bekerman, McCallum and Huang [56] published a technical report on automatic email foldering that offers extensive accuracy results comparison between different classifiers.

In 2011, Carmona-Cejudo et al. [57] present an open framework for on-line email classification called GNUsmail. They use known software frameworks with implemented algorithms to perform classification on several users from the dataset. In the same year, Carmona-Cejudo et al. [58] developed a study that compares different feature extraction and selection techniques.

Bermejo et al. [59] article propose a new method based on learning and sampling probability distributions to improve the classification of a Naive Bayes Multinominal classifier.

Carmona-Cejudo et al. [60], published a study where they present ABC-DynF, an adaptive learning framework created to solve the email foldering high dimensionality and dynamic nature problem.

In 2015, Boryczka, Probierz and Kozak [61] published an article that uses an adaptive ant colony decision forest to perform categorization of emails into folders, comparing the results with multiple ensemble methods and algorithms, achieving relatively higher accuracy values.

More recently, in 2016, Dehghani et al. [62] present ALECSA as an attentive learning approach for email foldering, displaying promising results when compared to older approaches, being able of achieving high efficiency when performing the categorization task.

Chapter 3

Research Problem

No different from main research areas, the email classification has several problems that may not have a scientific answer and are mainly subjective. One of the research objectives is to integrate the resulting system with an email client, which means many different types of people will be using the application. Email rookies will not have their mailboxes filled with emails, if any. Many different problems will arise from this.

3.1 General Problem

Ever since the email service initial years, new purposes to it have been appearing. The mail is used to exchange many types of information and this trade of emails is growing larger by the day. With higher data complexity and dimensionality, the classification task will become harder and the current algorithms may not perform good enough when integrated with an email client that offer features that resort to artificial intelligence.

This dissertation will address the general problem of email classification and the challenges that arise from it, trying to fit an intelligent system with several classification techniques available, to simulate a specific case study, the application under development by Mailcube .

3.2 Specific Research Topics

Although there are many possible approaches on preprocessing text, this work focus some of those and tries to analyse different combination of techniques. To perform classification many algorithms are available, making it difficult to test all of them. With that said, this study uses only three popular classifiers in the scientific community. Finally, to try and boost the accomplished results one step further a voting ensemble is also tested and analysed.

With these plans in mind, this dissertation focus the following research topics and correspondent subtopics:

- Text preprocessing techniques:
  - Stopwords removal
  - Tokenization
  - Stemming
  - Text representation
  - Attribute selection
- Supervised classifiers:
  - Naive Bayes
  - Support Vector Machine
  - Random Forest
- Voting Ensemble

3.3 Email Classification Challenges

There are multiple problems related with the classification and treatment of email due to its dynamic, multi-topical and changing nature. An email contains many different properties and it is a hard task to select the relevant ones, conversations topics may change along the way. In order to perform classification, it is expected that the user already has some emails that may be used for training, those might not be enough for a proper classification output.

3.3.1 Email Properties Relevance

As shown in section 2.1, the email comprises many different properties that one can consider. The weight of each one is relevant for the grouping achieved with the algorithms used. Every mailbox is different which makes it difficult to choose and attribute weights prior to

the algorithm's execution, meaning a dynamic weighting approach must be followed. Such method needs to take into account the properties available within the data and define the weight accordingly.

3.3.2 Email Conversations

Another important aspect that needs attention is the email conversations. Threads topic may drift along the communication interaction. Should the system group messages from the same conversation together? What if the topic drifts so much that the emails would fit another group? This is a difficult problem to solve and a perfect solution may not exist.

3.3.3 User Organization

Email users may already have emails organized in folders. The organization pattern they have for their mailbox might be really hard to discover with a classifier. This is a problem that proves difficult to solve, if not impossible. Users with no organization at all, should not be considered.

One approach is to dynamically learn and adapt the model to try and find the best fit configuration along the mailbox evolution.

3.3.4 Classes Unbalance

A user might have folders containing thousands of emails while other just have a few. This may lead to a shadowing effect where classes with many instances end up being more accurately classified then classes with much less instances, sometimes leading to false accuracy results.

This particular problem results from the diversification and organization of the user mailbox and relates to a well known problem with classification in general, which was presented by Japkowicz and Stephen in their study [63].

In 2012, Lemnaru et al. [64] extensively covers the imbalance issues and enumerates a group of best practices to attain the solution for this problem.

3.4 Solution Perspective

Due to the nature of these problems, a supervised system will need to be implemented. It will be constituted by a parser module that cleans and prepares the messages to be processed, a set of preprocessing techniques that help in extracting the most relevant features from the email text content, evolving the extracted output when there are

changes to the mailbox emails, and a classification module that is capable of high accurate predictions.

Such system will enable learning with already classified user mail and be capable of accurately predict the different possible classifications for new incoming email. It will also be able to adapt from user manual actions, by retraining the system model.

3.5 Validation Methodology

In order to validate the outcome work of this dissertation, the proposed system will have to successfully adapt to a very specific case study. The adapted system will be tested and the results compared with approaches from the scientific community with identical configuration for the same problem.

Chapter 4

Proposed Solution

This chapter presents the solution proposed by this dissertation project. The resulting system is presented starting with the conceptualization and architecture of the modular design. Each module used is extensively covered and described, when possible with examples mentioning the input and output of the piece.

We also cover ways of integrating the system and how it can be updated to better cope with the application integrating it.

4.1 Proposed System

The dissertation proposed solution consists in a system highly dynamic and volatile, easily integrated with an external application. The system is capable of achieving high accuracies, even when dealing with high dimensional data.

The main objective of developing a simple system is to make it easier to integrate with external applications and to replace sections of it, with this said, there is a need for the system to follow a simple modular architecture, expediting the integration and update process, making it able to support more then just the provided algorithms and preprocessing techniques. The parsing and output modules may also be changed to meet the application requirements.

4.2 System Architecture

The system architecture overview is illustrated by Figure 4.1, the design is distinguished by its modular separation. The entire system also works as a big module at the parent application disposal.

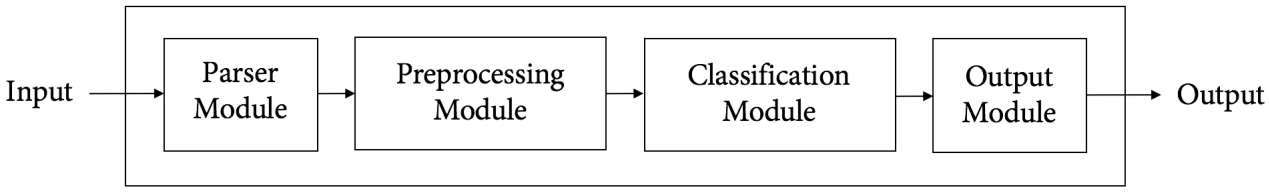

Figure 4.1: Architecture overview

Figure 4.2 exhibits the system high-level architecture to get a more expanded view over each module interaction.

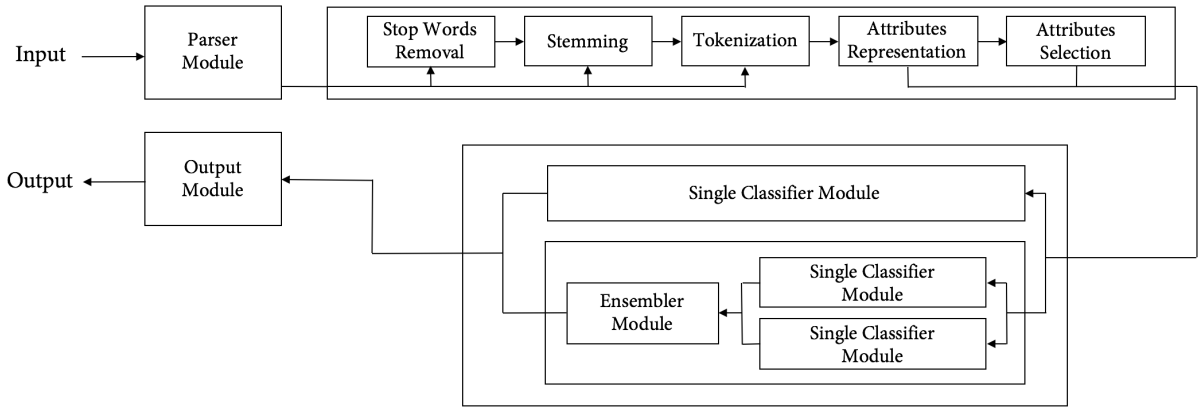

Figure 4.2: High-level architecture

The system is fully replaced and some modules support multiple configurations. These settings allow for the system to be adapted to the application using it, in order to obtain better performance results or to meet the application requirements.

4.2.1 Parser Module

The parser module receives the raw emails input, cleaning and normalizing the data that is needed into a single line of text.

The date field gets a specific treatment to transform it into a more useful format. The remaining fields are parsed by removing punctuation and other special characters

from the text, ".com" and ".net" are also removed since they do not add value to the classification. Finally, all the words are converted to lowercase and spaced by one space character, consequentially merged into a single text line.

Table 4.1 displays some parsing examples. The "Contact" refers to fields with email addresses such as *From* or *To*.

Field	Input	Output
Date	Date: Thu, 9 Nov 2000 10:44:00 -0800 (PST)	thursday 9 november 2000
Contact	From: david.delainey@enron.com	david delainey enron
Other	Guys, ENA 2001 expense budget "attached".	guys ena 2001 expense budget attached

Table 4.1: Parsing example

The parsing significantly reduces the memory used to represent emails, speeding up the preprocessing process.

4.2.2 Preprocessing Module

The preprocessing module receives the parser output as input and proceeds with the data processing using multiple configurations available. Figure 4.3 illustrates the preprocessing module architecture overview.

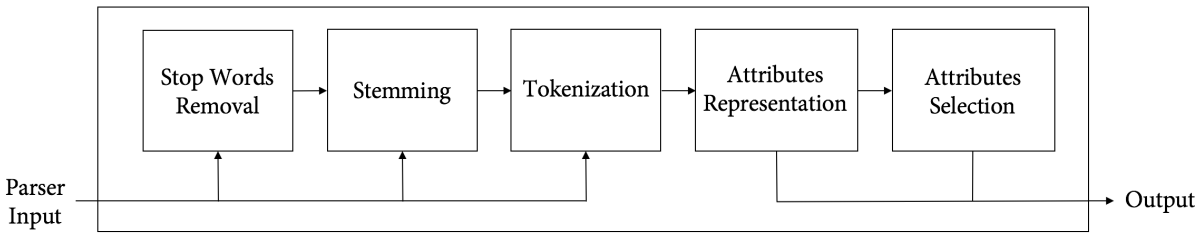

Figure 4.3: Preprocessing module architecture

The module has three optional steps, which are the removal of stop words, stemming and attributes selection. The tokenization and representation steps are mandatory, since the classification module algorithms need a specific structure in order to function properly.

Stop Words Removal unit is responsible for removing words that are listed in a configuration file, subsection 2.2.3 provides a more extensive review on stop words removal. The unit iterates through all the words from each message content and removes words that exist in the list of words. Normally the most used words in a certain linguistic are removed but specific words, not only common, may also be ignored by adding them to the configuration file.

Stemming unit is responsible for transforming the words into stems, subsection 2.2.4 offers a detailed review on the stemming technique. For this system, only the Porter stemmer was considered, others stemmer should be added if multilingual support is needed.

Tokenization unit creates tokens with the data so far processed. The system supports either a default word tokenizer 2.2.1 or n-gram 2.2.2. The delimiters should be configured before using.

Attributes Representation unit transforms each message content into a weighted vector of words. In the unit configuration is possible to select the amount of words used to represent each message and/or to limit the number of words per class. TF and/or IDF may also be applied to the vector. A more complete understanding on feature representation is offered by section 2.3.

Attributes Selection unit filters the extracted features from the representation unit, selecting the most relevant ones based on a scoring algorithm working with a search algorithm to perform the ranking 2.4.

4.2.3 Classification Module

The classification module is responsible for the model training and for classifying unlabelled instances. Figure 4.4 illustrates an overview of the classification module.

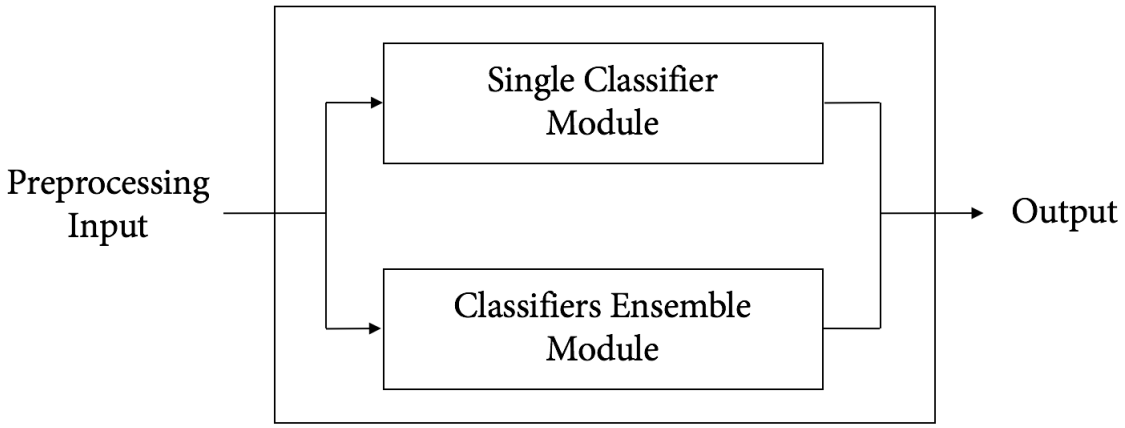

Figure 4.4: Classification module overview

Although there exists two classification modules, single classifier and classifiers ensemble module, the system may only use one, this means the user shall choose the fittest approach,

the decision should be held considering time-cost proficiency since in order to obtain better accuracy results, time and memory may be sacrificed.

Figure 4.5 clarifies the classification module providing a more detailed view over its architecture.

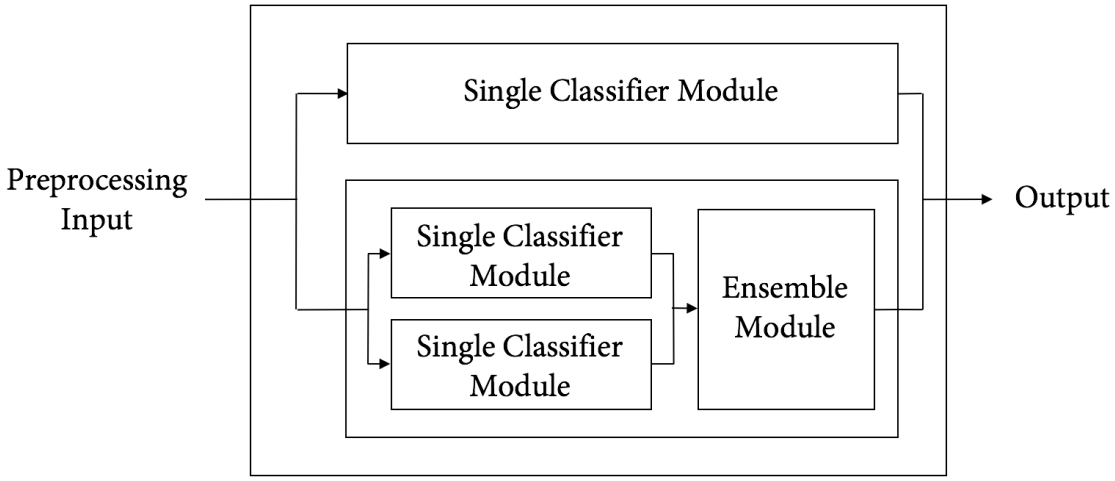

Figure 4.5: Classification module architecture

Single Classifier module takes care of the classification algorithm implementation and execution, using data originated from the preprocessing process. For this dissertation, only a standard Naive Bayes [65], SVM (Table 4.2) and Random Forest [40] were integrated. The module returns each instance classification distribution.

Ensemble model uses a voting technique with the results of two classifiers and tries to merge them in order to produce better results. Simply put, an instance is classified twice, one for each classifier, and the final classification is decided by means of a voting function. Different combination rules are supported (Section 2.6.3).

Implementation	SMO [35] with improvements [36]
Calibrator	Logistic Regression
Kernel	Polynomial
Complexity	0,56

Table 4.2: SVM configuration

4.2.4 Output Module

The output module is the simpler one and its existence is justified by the need to have a way of customizing the application outputs, without interfering with the remainder system

modules. The default is to return the final classification for each non classified message you pass into the system.

In some cases, a different output might be required. For instance, the module may be configured to retrieve the three most likely classifications instead of just one.

4.3 System Integration

The system being external to the application has only two endpoints of communication, one for input and another for output. This way we guarantee the system does not directly interfere with the application performance.

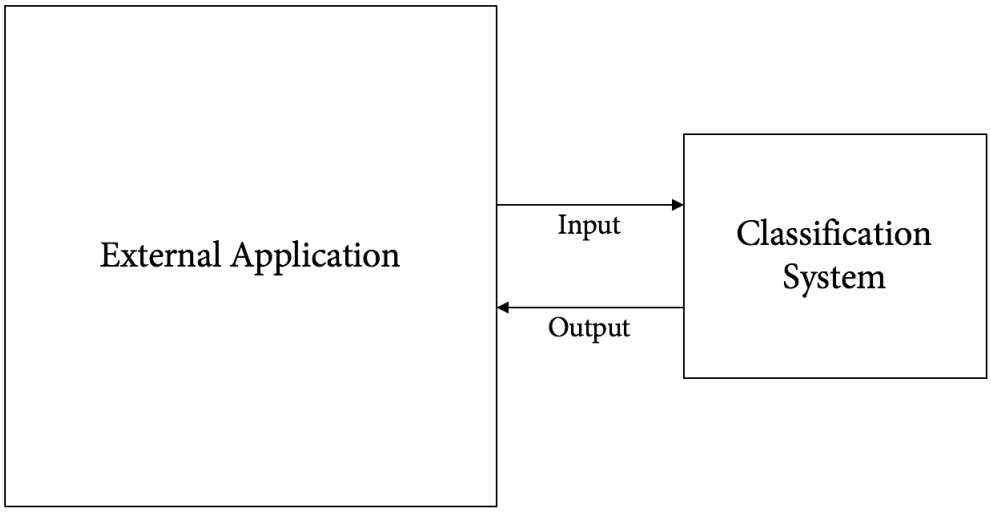

Figure 4.6: Integration overview

Input is expected to be an array containing the already classified messages from the user and new emails to be classified. The mails may be represented as a string of text content and an extra field for the class.

Output is controlled by correspondent model 4.2.4 and may be configured to meet any imposed standard as long as it is possible to create such output with the data provident from classification module.

Taking into account performance issues, the system shall always run in background and the application should not depend immediately on the classification output.

On a last note, since the system supports different configurations of several modules it should be tuned to further improve the outcome results.

4.4 Implementation Details

The system was implemented mainly in Java with a small resort to Python for the parser module. The programming languages choice is justified by the integration of WEKA software [66], written in Java, and Python due to its huge text processing ability.

Chapter 5

Mailcube Application: Case Study

This chapter covers the entire Mailcube application case study. The cube concept is briefly described to better understand how it fits to classification techniques. A simulation of the system integration with the application is adapted to take into account the case study implications.

During the simulation, several tests evaluate the proposed system performance. The Enron Corpus is adapted to serve as the entry dataset. Results were analysed and compared along all the simulation phases and with other approaches found in the scientific community.

It is expectable for the system to adjust properly to the case study at hand, performing with high accuracy and precision.

5.1 Introduction

Mailcube is an email client that created a concept of cube to bundle messages, contacts, and other typical email entities you find in most mail clients. In the context of this work,

a cube can be seen as a message storage unit that tries to separate the concept of mailbox from the folder. A cube named Science may contain messages relative to the science topic but these messages may be in any mailbox like the inbox, outbox, sent, spam, archive, trash or drafts. On the contrary of the folders, the cubes do not support sub-cubes, this brings implications to the simulation.

The cube also supports following specific contacts or conversations. If a cube is following a certain contact, at the arrival of new emails that have the contact in question, the messages are automatically redirected to the cube. This feature requires no classification and it was not attained during this case study. The incoming emails that already have a cube by means of this feature, should be used to retrain the model in order to better perform the classification of other mail.

5.2 Case Study Simulation

The main objective of the simulation is to see if the system can be adapted to the application and still perform accurately and with the enough precision not to become frustrating towards the user. Due to Mailcube application concepts, several steps were accomplished in order to meet the case study restrictions as best as possible.

The simulation serves to test multiple configurations that might increase the system output. Therefore multiple classifiers and preprocessing techniques were simulated.

Mailcube also wishes to be able to enable the user with multiple possible choices for the message classification, this means the system output module was adapted, in order to return the number of choices given, instead of a final classification.

5.2.1 Technical Specifications

The simulation tests were executed in a Macbook Pro running OS X El Capitan with the specifications listed in Table 5.1.

Disk	512GB SSD
RAM	16GB (used 8GB)
CPU	2,5GHz Intel Core i7

Table 5.1: Machine technical specifications

5.2.2 Implementation Design

Since the simulation uses the Enron Corpus dataset, some pre-treatment is necessary in order to not induce in false results.

```
Message-ID: <5482922.1075855813971.JavaMail.evans@thyme>
Date: Thu, 26 Oct 2000 09:21:00 -0700 (PDT)
From: ted.bland@enron.com
To: janet.dietrich@enron.com, wes.colwell@enron.com, sally.beck@enron.com,
        kevin.presto@enron.com, thomas.martin@enron.com,
        hunter.shively@enron.com, scott.neal@enron.com, w.duran@enron.com,
        jeff.donahue@enron.com, brian.redmond@enron.com
Subject: Super Saturday Interviewers for October 28, 2000
Cc: david.delainey@enron.com, charlene.jackson@enron.com, david.oxley@enron.com,
        shelly.jones@enron.com, jana.giovannini@enron.com
Mime-Version: 1.0
Content-Type: text/plain; charset=us-ascii
Content-Transfer-Encoding: 7bit
Bcc: david.delainey@enron.com, charlene.jackson@enron.com, david.oxley@enron.com,
        shelly.jones@enron.com, jana.giovannini@enron.com
X-From: Ted C Bland
X-To: Janet R Dietrich, Wes Colwell, Sally Beck, Kevin M Presto, Thomas A Martin, Hunter S
Shively, Scott Neal, W David Duran, Jeff Donahue, Brian Redmond
X-cc: David W Delainey, Charlene Jackson, David Oxley, Shelly Jones, Jana Giovannini
X-bcc:
X-Folder: \Sally_Beck_Dec2000\Notes Folders\Analyst_assoc program
X-Origin: Beck-S
X-FileName: sbeck.nsf

We are making progress but we still need 12 additional Manager and above
interviewers for Satrurday's Super Saturday (October 28).  Please canvas your
people again so we finish this weekend off with a bang.  Thanks for all your
support.
```

Figure 5.1: Enron email example (from *beck-s* user)

First, all the dataset emails contain the *x-folder* field that needs to be removed because it already has the folders name on it.

Concerning the cube concept, it is necessary to flatten the users subfolders, since hierarchy is not supported by the application, and also to remove folders that represent mailboxes or of non-topical characteristics, for this the following folders were removed: *all_documents*, *calendar*, *contacts*, *deleted_items*, *discussion_threads*, *inbox*, *notes_inbox*, *sent*, *sent_items*, *_sent_mail* and *outbox*.

Since fewer than three messages do not represent a decent sample to train, all the folders with less than three emails were not considered in the simulation.

5.2.3 Simulation Structure

In order to analyse several different configurations, the simulation is divided into five phases, figure 5.2 illustrates the simulation flow.

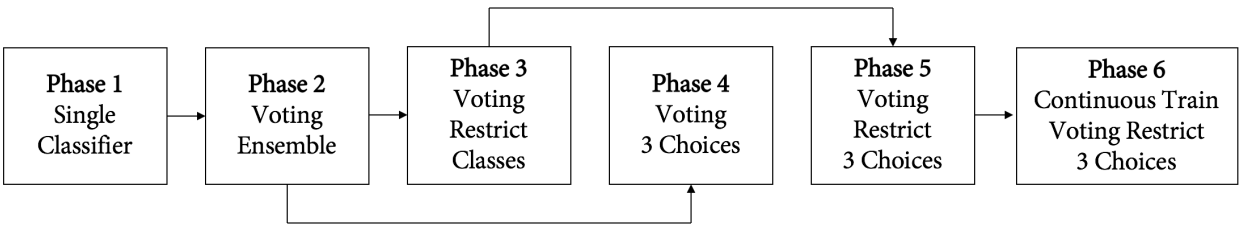

Figure 5.2: Simulation structure

Phase 1 serves as a base to decide each user best pre-processing configuration and algorithms to use in the following phases. A set of eight different configurations was tested for three algorithms.

Phase 2 uses the **Phase 1** best configuration achieved, for each user, and tests the voting module to verify if ensemble classifiers increases the performance.

Phase 3 uses the same setup as **Phase 2** but the dataset is restricted to include only classes with a more significant amount of messages, at least twenty.

Phase 4 also uses **Phase 2** setup but the system evaluates its accuracy when it outputs three possible classifications for each message.

Phase 5 uses **Phase 3** restricted configuration setup to test the three choices output, like in **Phase 4**.

Phase 6 uses **Phase 5** setup but it incrementally trains the classification model as messages are classified, giving the most realistic implementation scenario for the system integration with Mailcube.

The parsing module, described in 4.2.1, takes care of the message parsing and the possible preprocessing configurations are enumerated in table 5.2.

Code	Configuration
WordVector0	50 words per class, stopwords removal, word tokenizer
WordVector1	TF-IDF, 50 word per class, stopwords removal, word tokenizer
WordVector2	50 words per class, Porter stemming, stopwords removal, word tokenizer
WordVector3	TF-IDF, 50 words per class, Porter stemming, stopwords removal, word tokenizer
AttributeSelection	Information gain using Ranker with 0 threshold

Table 5.2: Preprocessing configurations

5.3 Phase 1

The first phase of the simulation takes advantage of the single classifier module to discover the configuration and algorithm that gives the best classification results, for each user.

The testing datasets from the seven users to be used in this phase are listed in Table 5.3.

User	Emails Number	Classes (Folders)
beck-s	1971	101
farmer-d	3672	25
kaminski-v	4477	41
kitchen-l	4015	47
lokay-m	2493	11
sanders-r	1188	30
williams-w3	2769	18

Table 5.3: Dataset 1 (Section A.1)

Table 5.4 displays the different configurations of preprocessing techniques used.

Code	Configuration
Config. 0	WordVector0
Config. 1	WordVector1
Config. 2	WordVector2
Config. 3	WordVector3
Config. 4	WordVector0 + AttributeSelection
Config. 5	WordVector1 + AttributeSelection
Config. 6	WordVector2 + AttributeSelection
Config. 7	WordVector3 + AttributeSelection

Table 5.4: Phase 1 preprocessing configuration (using Table 5.2)

For each one of the seven users listed in Table 5.3, the test structured is clarified in Table 5.5.

Number Runs	Method	Preprocessing Config.	Classifier
5	10-folds cross validation	Config. 0	Naive Bayes
			SVM
			Random Forest
		Config. 1	Naive Bayes
			SVM
			Random Forest
		Config. 2	Naive Bayes
			SVM
			Random Forest
		Config. 3	Naive Bayes
			SVM
			Random Forest
		Config. 4	Naive Bayes
			SVM
			Random Forest
		Config. 5	Naive Bayes
			SVM
			Random Forest
		Config. 6	Naive Bayes
			SVM
			Random Forest
		Config. 7	Naive Bayes
			SVM
			Random Forest

Table 5.5: Phase 1 structure

After performing the tests for each user, Table 5.6 presents the accuracy, standard deviation, maximum and minimum value achieved. The cells follow a **value (classifier - configurations)** format, presenting only the best values. The complete set of results are listed in Appendix B.

User	Accuracy	Std. Deviation	Max	Min
beck-s	61,461 (Naive Bayes - 0, 1) **67,458 (SVM - 0, 1)** 61,370 (Random Forest - 1)	2,880 (Naive Bayes - 4, 5) **2,759 (SVM - 6, 7)** 2,761 (Random Forest - 5)	71,066 (Naive Bayes - 0, 1) **73,604 (SVM - 0, 1)** 67,005 (Random Forest - 0)	48,731 (Naive Bayes - 0, 1, 2, 3) **56,853 (SVM - 0, 1)** **56,853 (Random Forest - 0, 1)**
farmer-d	73,203 (Naive Bayes - 2, 3) **81,471 (SVM - 2, 3)** 81,035 (Random Forest - 4)	1,856 (Naive Bayes - 6, 7) 1,638 (SVM - 6, 7) **1,506 (Random Forest - 0)**	76,839 (Naive Bayes - 0, 1) **86,376 (SVM - 0, 1)** 85,831 (Random Forest - 4)	67,935 (Naive Bayes - 0, 1) **78,261 (SVM - 6, 7)** 77,929 (Random Forest - 0, 4, 5)
kaminski-v	60,607 (Naive Bayes - 0, 1) 67,555 (SVM - 0, 1) **67,867 (Random Forest - 0)**	2,311 (Naive Bayes - 0, 1) 2,061 (SVM - 2, 3) **2,015 (Random Forest - 6)**	65,548 (Naive Bayes - 0, 1) 73,602 (SVM - 0, 1) **74,330 (Random Forest - 0, 1)**	55,357 (Naive Bayes - 0, 1) 63,170 (SVM - 0, 1) **63,393 (Random Forest - 0)**
kitchen-l	57,694 (Naive Bayes - 0, 1) **63,153 (SVM - 0, 1)** 62,775 (Random Forest - 5)	2,025 (Naive Bayes - 2, 3) **1,734 (SVM - 4, 5)** 1,775 (Random Forest - 5)	62,687 (Naive Bayes - 0, 1) **68,579 (SVM - 0, 1)** 67,662 (Random Forest - 4)	52,985 (Naive Bayes - 4, 5) **59,204 (SVM - 0, 1)** **59,204 (Random Forest - 4, 5)**
lokay-m	76,214 (Naive Bayes - 0, 1) 85,118 (SVM - 4, 5) **86,161 (Random Forest - 4)**	2,097 (Naive Bayes - 2, 3) 1,755 (SVM - 6, 7) **1,561 (Random Forest - 1)**	82,329 (Naive Bayes - 0, 1) 89,960 (SVM - 4, 5) **90,361 (Random Forest - 4, 5)**	70,400 (Naive Bayes - 4, 5) 79,920 (SVM - 0, 1) **82,329 (Random Forest - 1, 4)**
sanders-r	78,756 (Naive Bayes - 0, 1) **85,270 (SVM - 4, 5)** 85,018 (Random Forest - 4, 5)	3,758 (Naive Bayes - 4, 5) 3,033 (SVM - 0, 1) **2,669 (Random Forest - 1)**	88,136 (Naive Bayes - 0, 1) **93,277 (SVM - 4, 5)** 90,756 (Random Forest - 5)	71,186 (Naive Bayes - 0, 1) **78,151 (SVM - 0, 1, 2, 3, 4, 5)** 75,630 (Random Forest - 5)
williams-w3	91,592 (Naive Bayes - 2, 3) **96,735 (SVM - 6, 7)** 95,847 (Random Forest - 4)	1,457 (Naive Bayes - 2, 3) 0,925 (SVM - 0, 1) **0,907 (Random Forest - 3)**	94,585 (Naive Bayes - 2, 3, 6, 7) **98,556 (SVM - 2, 3)** 97,834 (Random Forest - 1, 2, 5, 6, 7)	88,043 (Naive Bayes - 2, 3) **94,585 (SVM - all)** 93,502 (Random Forest - 6)

Table 5.6: Phase 1 results

We can see from the results that SVM and Random Forest classifiers stand out from Naive Bayes, clearly obtaining better results. From all the multiple configurations possible

there is not a best one that fits every user, this is the reason a tune is need to better cope with the dataset being used.

Class	Naive Bayes	SVM	Random Forest
bill	0,176	**0,412**	0,176
bill_williams_iii	0,841	0,973	**0,983**
california_messages	**0,875**	**0,875**	0,500
el_paso	0,833	**0,917**	0,667
enron_messages	0,000	**0,250**	0,000
gwolfe	**0,250**	**0,250**	0,000
hr	**0,919**	**0,919**	0,849
human_resources	0,978	**1,000**	**1,000**
operations_committee_isas	**1,000**	0,737	0,632
personal	**0,875**	**0,875**	**0,875**
preschedule	**0,800**	0,733	0,200
rt_cuts	**0,250**	**0,250**	0,000
rt_strat	0,667	**0,722**	0,444
schedule_crawler	0,995	**0,999**	**0,999**
settlements	**0,167**	**0,167**	0,000
symesees	0,827	**0,901**	0,827
tie_meter_multipliers	1,000	**0,667**	**0,667**
timbelden	**0,727**	**0,727**	02545

Table 5.7: Phase 2 *williams-w3* classes true positive rate

Table 5.7 gives the detailed true positive rate per class from one cross validation. It is possible to see that, despite of the high accuracy achieved with the Random Forest for the *williams-w3* user, some folders were completely missed (*enron_messages*, *gwolfe*, *rt_cuts* and *settlements*). This is mainly due to the algorithms nature, being a decision tree instead of constructing a hyperplane like SVM does, and also the unbalance present in the *williams-w3* classes with some having just a few instances while others have more then a thousand, experiencing the shadowing problem covered in 3.3.4.

5.4 Phase 2

Phase 2 uses the initial phase dataset 5.3 and focus on using the best configuration from each user and the two more accurate classifiers, provided from the results 5.6. With this in mind, this phase uses an ensemble through voting using the minimum probability from the SVM and Random Forest algorithms classification. Table 5.8 displays the test structure that is followed.

User	Number Runs	Method	Configuration	Classifier
beck-s	5	10-folds cross validation	Config. 1	Voting (SVM:Random Forest)
farmer-d			Config. 3	
kaminski-v			Config. 1	
kitchen-l			Config. 1	
lokay-m			Config. 5	
sanders-r			Config. 5	
williams-w3			Config. 7	

Table 5.8: Phase 2 structure

Table 5.9 displays the phase 2 test results. Only the accuracy results are being displayed, the complete set is listed in Section C.1.

User	Accuracy	Std. Deviation	Max	Min
beck-s	67,316	2,882	73,604	56,853
farmer-d	82,320	1,729	86,376	78,747
kaminski-v	68,439	2,073	73,826	64,063
kitchen-l	63,831	1,917	68,657	59,453
lokay-m	87,012	1,839	90,763	83,936
sanders-r	86,146	3,123	93,277	78,151
williams-w3	96,822	0,964	98,195	94,585

Table 5.9: Phase 2 results

With the exception of *beck-s* user, that experienced a decrease, the remainder users had a small accuracy increase of around one percent 5.10. This happens because the two algorithms do not compensate each other much, since they obtain similar results. This means that when one misses, the other is also likely to miss, compensating only when one of them has the correct answer. If time restriction is not a problem, the ensemble by voting should be used, since this 1% increase might translate to a significant amount of instances being correctly categorized.

User	Phase 1	Phase 2	Phase 2 - Phase 1
beck-s	67,458	67,316	**-0,142**
farmer-d	81,471	82,320	0,849
kaminski-v	67,867	68,439	0,572
kitchen-l	63,153	63,831	0,678
lokay-m	86,161	87,012	0,851
sanders-r	85,270	86,146	0,876
williams-w3	96,735	96,822	0,087

Table 5.10: Accuracy comparison between phase 1 and 2

5.5 Phase 3

Phase 3 follows phase 2 setup but uses a restricted dataset 5.11. The purpose is to see if there exists a significant increase when each class has, at least twenty messages.

User	Emails Number	Classes (Folders)
beck-s	1379	30
farmer-d	3560	15
kaminski-v	4347	27
kitchen-l	3852	31
lokay-m	2471	9
sanders-r	1097	17
williams-w3	2632	5

Table 5.11: Dataset 2 (Section A.2)

Using the voting configuration from phase 2 and the restricted dataset, the results are enumerated at Table 5.12. The complete set of results is presented in Section C.2.

User	Accuracy	Std. Deviation	Max	Min
beck-s	79,797	3,277	86,232	71,739
farmer-d	83,433	1,727	86,517	78,090
kaminski-v	69,648	1,748	74,194	65,438
kitchen-l	65,208	1,930	70,909	61,818
lokay-m	87,212	1,613	90,688	83,401
sanders-r	87,384	3,158	93,636	80
williams-w3	98,936	0,662	100	96,958

Table 5.12: Phase 3 results

As we can see from the comparison results in Table 5.13, there has been a significant increase in the accuracy of *beck-s* (12%), probably because of the huge reduce in dimensionality, from 101 to 30 classes. The user *williams-w3* experience a maximum of 100% meaning at least once, every instance was successfully classified. Overall, the accuracy was increased between 0,2% and 12%, for each user. This increase is small for the users data that has not suffered a dramatic change of dimensionality and size.

User	Phase 2	Phase 3	Phase 3 - Phase 2
beck-s	67,316	79,797	12,481
farmer-d	82,320	83,433	1,113
kaminski-v	68,439	69,648	1,209
kitchen-l	63,831	65,208	1,377
lokay-m	87,012	87,212	0,200
sanders-r	86,146	87,384	1,238
williams-w3	96,822	98,936	2,114

Table 5.13: Accuracy comparison between phase 2 and 3

5.6 Phase 4

Phase 4 follows the configuration and setup of phase 2, but allows the algorithm to answer three possible classifications for each instance, the ones with higher probability. The structure, present in Table 5.8, was changed for this test using a data split method after randomizing the data, resulting in the structure displayed by Table 5.14.

User	Number Runs	Method	Configuration	Classifier
beck-s	100	66% split	Config. 1	Voting - 3 Best Classifications
farmer-d			Config. 3	
kaminski-v			Config. 1	
kitchen-l			Config. 1	
lokay-m			Config. 5	
sanders-r			Config. 5	
williams-w3			Config. 7	

Table 5.14: Phase 4 structure

The results are displayed at Table 5.15 with the exception of the measured accuracy standard deviation.

User	Accuracy	Max	Min
beck-s	79,507	83,283	76,418
farmer-d	96,014	96,955	94,631
kaminski-v	84,959	87,385	83,114
kitchen-l	85,151	87,839	82,417
lokay-m	96,721	97,995	95,165
sanders-r	93,978	96,535	91,337
williams-w3	99,012	99,681	97,981

Table 5.15: Phase 4 results

Table 5.16 shows the results comparison between phase 2 and 4. Verifiying that when the classifier gives three possible correct classifications (the three possibilities with the higher value), the accuracy was significantly increased for all users, detaining a correct

classification percentage above 79%. Although *williams-w3* did not experience a full correct classification, the average accuracy achieved was still higher.

User	Phase 2	Phase 4	Phase 4 - Phase 2
beck-s	67,316	79,507	12,191
farmer-d	82,320	96,014	13,694
kaminski-v	68,439	84,959	16,520
kitchen-l	63,831	85,151	21,320
lokay-m	87,012	96,721	9,709
sanders-r	86,146	93,978	7,832
williams-w3	96,822	99,012	2,190

Table 5.16: Accuracy comparison between phase 2 and 4

5.7 Phase 5

Phase 5 serves as a complementary study for phase 3 and 4. This test follows phase 4 structure (Table 5.14) using the restricted dataset (Table 5.11) to validate a possible increase in accuracy, due to the three choices output and the data morphological change. Table 5.17 shows the experiment results.

User	Accuracy	Max	Min
beck-s	91,842	95,096	88,699
farmer-d	97,375	98,430	96,446
kaminski-v	86,158	88,701	83,424
kitchen-l	86,658	88,855	84,885
lokay-m	96,855	98,095	95,595
sanders-r	95,903	98,391	91,957
williams-w3	99,931	100	99,665

Table 5.17: Phase 5 results

As expected (Table 5.18) there was a small increase in almost every user accuracy, with *beck-s* experiencing the most accentuated positive change. The *williams-w3* average accuracy went up almost 1% and the difference between the observed maximum and minimum value, was drastically reduced to 0,3%.

User	Phase 4	Phase 5	Phase 5 - Phase 4
beck-s	79,507	91,842	**12,335**
farmer-d	96,014	97,375	1,361
kaminski-v	84,959	86,158	1,199
kitchen-l	85,151	86,658	1,507
lokay-m	96,721	96,855	0,134
sanders-r	93,978	95,903	1,925
williams-w3	99,012	99,931	0,919

Table 5.18: Accuracy comparison between phase 4 and 5

All the results are now above the 86%. This means that if the system is integrated with an application, it can suggest three correct classifications for each unclassified email, using folders with at least twenty messages, with a high rate of correct classifications.

5.8 Phase 6

The final phase is the best approximation to an integration of the system with an external application. The phase configuration is identical to phase 5 with the particularity of performing a continuous update on the entire classification model after classifying one instance, simulating a real situation of incoming email.

Table 5.19 enumerates the experience results. Phase 5 structure was followed, using the test split instances to incrementally train and classify the classification model.

User	Accuracy	Max	Min
beck-s	93,639	94,456	93,177
farmer-d	97,563	98,347	96,287
kitchen-l	87,568	88,701	85,927
kaminski-v	87,412	88,473	86,565
lokay-m	97,327	97,759	96,905
sanders-r	96,458	96,783	96,247
williams-w3	99,799	100	99,464

Table 5.19: Phase 6 results

Table 5.20 comprises the results comparison for the two phases. It was possible to verify that, continuously updating the model has a small increase in the performance, with the exception of *williams-w3*.

User	Phase 5	Phase 6	Phase 6 - Phase 5
beck-s	91,842	93,639	1,797
farmer-d	97,375	97,563	0,188
kaminski-v	86,158	87,568	1,410
kitchen-l	86,658	87,412	0,754
lokay-m	96,855	97,327	0,472
sanders-r	95,903	96,458	0,555
williams-w3	99,931	99,799	**-0,132**

Table 5.20: Accuracy comparison between phase 5 and 6

This phase validates the system reliability when working with a real application. Having a feature capable of providing three possible classifications, even with the class quota restrictions, can be of great value to the end user. This means, the application being evaluated in this case study could benefit greatly from the proposed system.

5.9 Results Comparison

The results obtained in this simulation using the proposed system were compared with other approaches that have an identical configuration. Since the results obtained in phase 3, 4, 5 and 6 have too many restrictions to meet the case study evaluation, no identical configuration was found. Although, for phases 1 and 2, there are many approaches that can be used as an accuracy validation, but only phase 2 results were compared since they show an higher accuracy.

Table 5.21 shows an accuracy comparison between the proposed system and other approaches best results, for the dataset configuration presented in Table 5.3.

Approach	beck-s	farmer-d	kaminski-v	kitchen-l	lokay-m	sanders-r	williams-w3
Beckerman et al. [56]	56,4	77,5	57,4	59,1	83,6	73,0	94,6
GNUsmail [57]	-	76,4	66,6	-	78,8	73,5	95,5
Bermejo et al. [59]	50,6	70,3	56,7	46,5	73,2	75,9	89,0
ABC-DynF [60]	49,5	61,1	65,8	59,9	87,5	76,3	88,3
Boryczka et al. [61]	51,7	77,5	65,7	58,3	84,6	75,9	94,4
Alecsa [62]	63,8	70,8	63,1	58,3	**91,6**	74,3	**96,8**
Proposed approach 5.8 5.9	**67,3**	**82,3**	**68,4**	**63,8**	87,0	**86,1**	**96,8**

Table 5.21: Accuracy comparison between approaches

From the comparison is very clear that this work approach gets higher accuracies, with the exception of the *lokay-m* user, where Dehghani et al. Alecsa [62] managed to obtain an higher value then the rest remainder of the approaches, with a value above 90%. This may be duo to the dynamic concept behind the framework leading to a raise in values for that specific distribution (the *lokay-m* user mailbox).

In the remainder distributions, the resulting system surpassed the other approaches with a difference ranging from 2% to 10%. The *williams-w3* user is difficult to improve any further since the results are already near the maximum value and mainly thanks to the unbalanced nature of the dataset class distribution.

Chapter 6

Conclusions

There is much research work in the area and many ways of classifying emails into folders. Most of the study techniques focus on text retrieval and obtaining knowledge from the email text attributes, which shows that a lot of work is still to be accomplished in this area. There is a wide space of improvement for approaches who consider email attachments and contacts network of connections.

The main effort for companies like Google or Microsoft is in the spam-detection methodologies, although a system that can organize mailboxes and adapt accordingly to user interaction and information arrival, may be of great interest since it will save the users precious time, therefore improving the productivity and communication that represent areas of focus for these companies.

6.1 Main Results

This dissertation work provides the conceptualization of a classification system, capable of adapting to an external application without the necessity of great modifications. The system allows to configure the preprocessing and classification techniques used, allowing it to be optimized towards the dataset being classified. The ability to achieve high accuracy derivates from the methods used, presenting a classification ensemble that proved to be effective in improving the overall performance.

Besides validating the classification capability, it was possible to test different use case scenarios for the system, validating the possibility of integrating it into an existing

application, in this case being the Mailcube case study, furthermore analysing how the integration could be done, in order to obtain valuable performance for the end user.

6.2 Future Work

There are many different directions one can take to further extend the work presented in this dissertation. Although it seems the text classification is reaching a limit in the email categorization problem, there is still many topics that need more research. If it is indeed reaching its limit, new ways of classifying email need more focus, such as network analysis and other approaches.

The scientific community is also in need for an up-to-date email dataset, with a more complete message content, containing attachments, html and others. This way, new patterns to approach this problem can be studied.

The next sections provide a more extensive explanation on the future work that may be accomplished in this area.

6.2.1 Preprocessing Techniques and Classifiers

The proposed system was analysed with eight different preprocessing configurations and only three most well know classifiers, which led to good results, but there is almost an infinite number of possible combinations and algorithms available, making it very hard to test every single one of them or making a meta-algorithm learn them all. With this in mind, some techniques are very similar and only aim at improving certain parts of the classification.This leaves space to conduct a study that aggregates similar approaches, selecting only the most efficient, and comparing them with not so similar ones, or even try to combine them.

6.2.2 Model Update

The adaptability of a system to changes in the data size and dimension is fundamental for any real case scenario. Dividing and classifying a static dataset produces results that may not be verified along the system usage, as the user mailbox keeps getting bigger.

All the literature work reviewed in this dissertation, does not take into account the classification model evolution during its integration with an application. Section 5.8 study aims at reproducing an approximation to what would be a real integration with an external application, updating the model every time it classifies a message, but such heavy approach might not be necessary.

This calls for a more extensive analysis on how one can update the model. Should it be updated every time a new message arrives, or should it work in a batch way, only updating when a certain number of messages are classified? Is it worthy to continuously update it? If a new message arrives and is classified as belonging to the class with the highest number of messages, rebuild the model might not be necessary, since it already has a decent amount of training data.

6.2.3 Non-text Attributes

In the conducted study, only the messages text content was considered to perform the classification. Although very good accuracy results were achieved, there is still space for improving. Is very likely that the classification of email through their text content may not evolve any further but one may take more patterns into account. For instance, email messages with the same categorization share more then just similar test, they usually have a network of contacts that can be extracted and used as an attribute in the classification process.

Future work could take advantage of this system text classification and merge it with a classifier that treats non-text features, where ensembling methods could be use.

6.2.4 Meta Parametrization

The results achieved from the simulation helped concluding that the dynamics of a user mailbox directly affects the output of a classification system and there is no good-for-all approach that gives the best possible results. With this in mind, although the system is capable of supporting different configurations, it is still depending on a manual tuning to reach optimized results.

Using meta programming to dynamically tweak the preprocessing methods used, would make the system less dependent on manual configuration, increasing its volatility and adaptability to different environments.

The classification module may also take advantage of such automated learning, changing the different parameters of the classifiers, with the intent of maximizing the entire system classification.

The system modular architecture makes it easier to integrate a meta programming module that can automatically tune the system, by exchanging information with the configurable modules (preprocessing and classification module). Figure 6.1 illustrates an overview on the integration of such module with the proposed system.

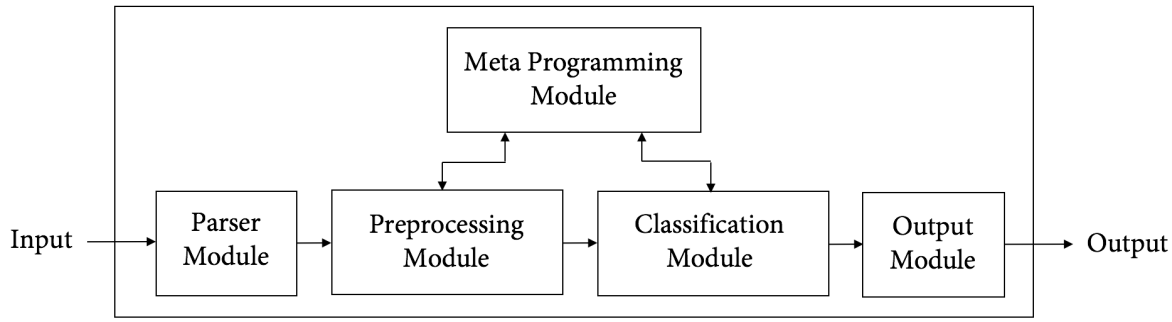

Figure 6.1: Meta programming module integration overview

To perform this meta parametrization the scientific community tends to use bio inspired algorithms, such as genetic algorithms and ant colony optimization approaches that create populations. Basically, each population is a different possible configuration that is tested to find the best parameters.

6.2.5 Data Considerations

The Enron Corpus did not have any attachments that could be used in the classification. If the attachment contains text, it is easily adapted to fit the proposed system. What if the attachment is a picture? There are already a lot of graphic mining techniques that extract features from images.

The case study did not require taking into account folders hierarchy, since Mailcube cubes do not support it. Flatten the subfolders results in a higher accuracy error, since folders inside folders tend to have the same overall topic. A possible research approach is to create a classifier that initial checks if a given message is inside a parent folder and afterwards performs classification using only the sub-folders categories, continuing iterating until the hierarchical tree can not be more expandable.

6.3 System Integration with Mailcube

One of the main goals of this dissertation project was to validate the system integration with an email client. This validation was successful, remaining to be accomplished the proposed system integration with the application. In order to do that one needs to export the system in a library format — since there is only the need for the app to call the system — and properly merge it with the desktop client implementation.

The Mailcube application is written in Objective-C, one of the two native languages of macOS, which might prove difficult to integrate with a Java library, this means the system

will likely have to be adapted and re-written in C++, Objective-C or Swift using other machine learning libraries available. Since those languages are supported by the operating system, using them will fully enable the integration.

Nomenclature

API	Acronym for **A**pplication **P**rogramming **I**nterface.
ARPANET	Acronym for **A**dvanced **R**esearch **P**rojects **A**gency **Net**work.
ASCII	Acronym for **A**merican **S**tandard **C**ode for **I**nformation **I**nterchange.
BoW	Acronym for **B**ag **o**f **W**ords.
EGM	Acronym for **E**mail **G**rouping **M**ethod.
HTML	Acronym for **H**yper**T**ext **M**arkup **L**anguage.
MIME	Acronym for **M**ultipurpose **I**nternet **M**ail **E**xtensions.
ML	Acronym for **M**achine **L**earning.
OS	Acronym for **O**perating **S**ystem.
RFC	Acronym for **R**equest **f**or **C**omments.
SMO	Acronym for **S**equential **M**inimal **O**ptimization.
SVM	Acronym for **S**upport **V**ector **M**achines.
TF-IDF	Acronym for **T**erm **F**requency-**I**nverse **D**ocument **F**requency.
TREC	Acronym for **T**ext **Re**trieval **C**orpus.
VOIP	Acronym for **V**oice **O**ver **I**nternet **P**rotocol.
VSM	Acronym for **V**ector **S**pace **M**odel.
WEKA	Acronym for **W**aikato **E**nvironment for **K**nowledge **A**nalysiss.

References

[1] C. Partridge, "The technical development of internet email," *IEEE Annals of the History of Computing*, vol. 30, no. 2, pp. 3–29, 2008. Cited on p. 2.

[2] G. Radicati, "Email Statistics Report, 2015-2019," 2015. Cited on p. 2.

[3] G. Cormack and T. Lynam, "Spam corpus creation for TREC," *Second Conference on Email and Anti-Spam (CEAS2005)*, 2005. Cited on p. 3.

[4] S. Whittaker and C. Sidner, "Email overload: Exploring personal information management of email," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '96, pp. 276–283, 1996. Cited on p. 3.

[5] D. Fisher, A. J. Brush, E. Gleave, and M. A. Smith, "Revisiting whittaker & sidner's "email overload" ten years later," in *Proceedings of the 2006 20th Anniversary Conference on Computer Supported Cooperative Work*, CSCW '06, pp. 309–312, 2006. Cited on p. 4.

[6] C. Grevet, D. Choi, D. Kumar, and E. Gilbert, "Overload is overloaded: Email in the age of gmail," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '14, pp. 793–802, 2014. Cited on p. 4.

[7] M. Mohri, A. Rostamizadeh, and A. Talwalkar, *Foundations of Machine Learning*. The MIT Press, 2012. Cited on p. 9.

[8] E. P. Resnick, "Internet message format," 8 2008. RFC 5322. Cited on p. 10.

[9] N. Freed and N. Borenstein, "Multipurpose internet mail extensions(mime), part one: Format of internet message bodies," 11 1996. RFC 2045. Cited on p. 10.

[10] N. Freed and N. Borenstein, "Multipurpose internet mail extensions(mime), part two: Media types," 11 1996. RFC 2046. Cited on p. 10.

[11] K. Moore, "Multipurpose internet mail extensions(mime), part three: Message header extensions for non-ascii text," 11 1996. RFC 2047. Cited on p. 10.

[12] N. Freed and J. Klensin, "Media type specifications and registration procedures," 12 2005. RFC 4289. Cited on p. 10.

[13] N. Freed and J. Klensin, "Multipurpose internet mail extensions(mime), part four: Registration procedures," 12 2005. RFC 4289. Cited on p. 10.

[14] N. Freed and N. Borenstein, "Multipurpose internet mail extensions(mime), part five: Conformance criteria and examples," 11 1996. RFC 2049. Cited on p. 10.

[15] I. Alsmadi and I. Alhami, "Clustering and classification of email contents," *J. King Saud Univ. Comput. Inf. Sci.*, vol. 27, pp. 46–57, Jan. 2015. Cited on pp. 11 and 15.

[16] W. J. Wilbur and K. Sirotkin, "The automatic identification of stop words," *J. Inf. Sci.*, vol. 18, pp. 45–55, Jan. 1992. Cited on p. 11.

[17] P. Kumar, H. Kumar, and R. Joseph, "A Framework for Email Clustering and Automatic Answering Method," *International Journal of Advanced Research in Computer Engineering & Technology (IJARCET)*, vol. 1, no. 9, pp. 52–59, 2012. Cited on p. 11.

[18] J. B. Lovins, "Development of a Stemming Algorithm," *Mechanical Translation and Computational Linguistics*, vol. 11, no. June, pp. 22–31, 1968. Cited on p. 12.

[19] M. Porter, "An algorithm for suffix stripping," *Program*, vol. 14, no. 3, pp. 130–137, 1980. Cited on p. 12.

[20] G. Salton, A. Wong, and C. S. Yang, "A vector space model for automatic indexing," *Commun. ACM*, vol. 18, pp. 613–620, Nov. 1975. Cited on p. 12.

[21] G. Salton and C. Buckley, "Term-weighting approaches in automatic text retrieval," *Inf. Process. Manage.*, vol. 24, pp. 513–523, Aug. 1988. Cited on p. 12.

[22] P. D. Turney and P. Pantel, "From frequency to meaning: Vector space models of semantics," *J. Artif. Int. Res.*, vol. 37, pp. 141–188, Jan. 2010. Cited on p. 12.

[23] K. Sparck Jones, "A Statistical Interpretation of Term Specificity and its Retrieval," *Journal of Documentation*, vol. 28, no. 1, pp. 11–21, 1972. Cited on p. 12.

[24] A. Singhal, G. Salton, M. Mitra, and C. Buckley, "Document length normalization," *Inf. Process. Manage.*, vol. 32, pp. 619–633, Sept. 1996. Cited on p. 13.

[25] R. B. Segal and J. O. Kephart, "Mailcat: An intelligent assistant for organizing e-mail," in *Proceedings of the Third Annual Conference on Autonomous Agents*, AGENTS '99, pp. 276–282, 1999. Cited on p. 13.

[26] I. Guyon and A. Elisseeff, "An introduction to variable and feature selection," *J. Mach. Learn. Res.*, vol. 3, pp. 1157–1182, Mar. 2003. Cited on p. 13.

[27] M. A. Hall, *Correlation-based Feature Subset Selection for Machine Learning*. PhD thesis, University of Waikato, Hamilton, New Zealand, 1998. Cited on p. 13.

[28] R. E. Korf, "Algorithms and theory of computation handbook," ch. Artificial Intelligence Search Algorithms, pp. 22–22, Chapman & Hall/CRC, 2010. Cited on p. 13.

[29] R. E. Korf, "Linear-space best-first search," *Artif. Intell.*, vol. 62, pp. 41–78, July 1993. Cited on p. 14.

[30] D. D. Lewis, "Naive (Bayes) at Forty: The Independence Assumption in Information Retrieval," in *Proceedings of the 10th European Conference on Machine Learning*, ECML '98, (London, UK, UK), pp. 4–15, Springer-Verlag, 1998. Cited on p. 14.

[31] J. D. M. Rennie, "ifile: An Application of Machine Learning to E-Mail Filtering," in *Proc. KDD Workshop on Text Mining*, 2000. Cited on p. 14.

[32] J. D. M. Rennie, L. Shih, J. Teevan, and D. R. Karger, "Tackling the Poor Assumptions of Naive Bayes Text Classifiers," *Proceedings of the Twentieth International Conference on Machine Learning (ICML)-2003)*, vol. 20, no. 1973, pp. 616–623, 2003. Cited on p. 14.

[33] H. Zhang, "The Optimality of Naive Bayes," *Proceedings of the Seventeenth International Florida Artificial Intelligence Research Society Conference FLAIRS 2004*, vol. 1, no. 2, pp. 1 – 6, 2004. Cited on p. 14.

[34] C. Cortes and V. Vapnik, "Support-vector networks," *Mach. Learn.*, vol. 20, pp. 273–297, Sept. 1995. Cited on p. 15.

[35] J. Platt, "Fast training of support vector machines using sequential minimal optimization," in *Advances in Kernel Methods - Support Vector Learning* (B. Schoelkopf, C. Burges, and A. Smola, eds.), MIT Press, 1998. Cited on pp. 15 and 27.

[36] S. Keerthi, S. Shevade, C. Bhattacharyya, and K. Murthy, "Improvements to platt's smo algorithm for svm classifier design," *Neural Computation*, vol. 13, no. 3, pp. 637–649, 2001. Cited on pp. 15 and 27.

[37] H. Drucker, D. Wu, and V. N. Vapnik, "Support vector machines for spam categorization," *IEEE Transactions on Neural Networks*, vol. 10, pp. 1048–1054, Sep 1999. Cited on p. 15.

[38] S. Tong and D. Koller, "Support Vector Machine Active Learning with Applications to Text Classification," *J. Mach. Learn. Res.*, vol. 2, pp. 45–66, Mar. 2002. Cited on p. 15.

[39] J. Shawe-Taylor and S. Sun, "A Review of Optimization Methodologies in Support Vector Machines," *Neurocomput.*, vol. 74, pp. 3609–3618, Oct. 2011. Cited on p. 15.

[40] L. Breiman, "Random Forests," *Mach. Learn.*, vol. 45, pp. 5–32, Oct. 2001. Cited on pp. 15 and 27.

[41] T. K. Ho, "Random decision forests," *Proceedings of 3rd International Conference on Document Analysis and Recognition*, vol. 1, pp. 278–282, 1995. Cited on p. 15.

[42] T. K. Ho, "The random subspace method for constructing decision forests," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 20, pp. 832–844, Aug. 1998. Cited on p. 15.

[43] Y. Amit and D. Geman, "Shape Quantization and Recognition with Randomized Trees," *Neural Comput.*, vol. 9, pp. 1545–1588, Oct. 1997. Cited on p. 15.

[44] G. Rios and H. Zha, "Exploring Support Vector Machines and Random Forests for Spam Detection," *Conference on e-mail and anti-spam (CEAS)*, pp. 5–10, 2004. Cited on p. 15.

[45] P. Cunningham and S. J. Delany, "k-Nearest Neighbour Classifiers," *Technical Report UCD-CSI-2007-4*, pp. 1–17, 2007. Cited on p. 15.

[46] L. I. Kuncheva, *Combining Pattern Classifiers: Methods and Algorithms*. John Wiley and Sons, Inc., 2004. Cited on p. 15.

[47] L. Breiman, "Bagging predictors," *Mach. Learn.*, vol. 24, pp. 123–140, Aug. 1996. Cited on p. 16.

[48] Y. Freund and R. E. Schapire, "Experiments with a new boosting algorithm," in *Computational Learning Theory Second European Conference, EuroCOLT '95 Barcelona, Spain, March 13–15, 1995 Proceedings*, pp. 23–37, 1995. Cited on p. 16.

[49] Y. Freund and R. E. Schapire, "Experiments with a new boosting algorithm," in *International Conference on Machine Learning*, pp. 148–156, 1996. Cited on p. 16.

[50] Y. Freund and R. E. Schapire, "A short introduction to boosting," in *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence*, pp. 1401–1406, Morgan Kaufmann, 1999. Cited on p. 16.

[51] D. H. Wolpert, "Original contribution: Stacked generalization," *Neural Netw.*, vol. 5, pp. 241–259, Feb. 1992. Cited on p. 17.

[52] B. Klimt and Y. Yang, "Introducing the Enron Corpus," in *CEAS*, 2004. Cited on p. 17.

[53] B. Klimt and Y. Yang, "The Enron Corpus: A New Dataset for Email Classification Research," *Machine Learning: ECML 2004*, pp. 217–226, 2004. Cited on p. 17.

[54] C. Project, "Enron email dataset," 2004. Cited on p. 17.

[55] G. V. Cormack and T. R. Lynam, "Trec 2007 public corpus," 2007. Cited on p. 17.

[56] R. Bekkerman, A. McCallum, and G. Huang, "Automatic categorization of email into folders: Benchmark experiments on enron and sri corpora," *Center for Intelligent Information Retrieval, Technical Report IR*, vol. 418, 2004. Cited on pp. 18 and 43.

[57] J. M. Carmona-Cejudo, M. Baena-García, J. del Campo-Ávila, R. Morales-Bueno, and A. Bifet, "Gnusmail: Open framework for on-line email classification," 2011. Cited on pp. 18 and 43.

[58] J. M. Carmona-Cejudo, G. Castillo, M. Baena-García, and R. Morales-Bueno, "A comparative study on feature selection and adaptive strategies for email foldering," in *Intelligent Systems Design and Applications (ISDA), 2011 11th International Conference on*, pp. 1294–1299, IEEE, 2011. Cited on p. 18.

[59] P. Bermejo, J. A. Gámez, and J. M. Puerta, "Improving the performance of naive bayes multinomial in e-mail foldering by introducing distribution-based balance of datasets," *Expert Systems with Applications*, vol. 38, no. 3, pp. 2072–2080, 2011. Cited on pp. 18 and 43.

[60] J. M. Carmona-Cejudo, G. Castillo, M. Baena-GarcíA, and R. Morales-Bueno, "A comparative study on feature selection and adaptive strategies for email foldering using the abc-dynf framework," *Know.-Based Syst.*, vol. 46, pp. 81–94, July 2013. Cited on pp. 18 and 43.

[61] U. Boryczka, B. Probierz, and J. Kozak, "Adaptive ant colony decision forest in automatic categorization of emails," in *Asian Conference on Intelligent Information and Database Systems*, pp. 451–461, Springer, 2015. Cited on pp. 18 and 43.

[62] M. Dehghani, A. Shakery, and M. S. Mirian, "Alecsa: Attentive learning for email categorization using structural aspects," *Knowledge-Based Systems*, vol. 98, pp. 44–54, 2016. Cited on pp. 18 and 43.

[63] N. Japkowicz and S. Stephen, "The class imbalance problem: A systematic study," *Intell. Data Anal.*, vol. 6, pp. 429–449, Oct. 2002. Cited on p. 21.

[64] C. Lemnaru and R. Potolea, *Imbalanced Classification Problems: Systematic Study, Issues and Best Practices*, pp. 35–50. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012. Cited on p. 21.

[65] G. H. John and P. Langley, "Estimating continuous distributions in bayesian classifiers," in *Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence*, UAI'95, (San Francisco, CA, USA), pp. 338–345, Morgan Kaufmann Publishers Inc., 1995. Cited on p. 27.

[66] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, "The WEKA Data Mining Software: An Update," *SIGKDD Explor. Newsl.*, vol. 11, pp. 10–18, Nov. 2009. Cited on p. 29.

Appendices

Appendix A

Enron Datasets Distribution

The present appendix enumerates the two datasets distributions used in this dissertation work. Both sets contain the mailboxes of seven Enron Corpus users. Dataset 1 A.1 only considers folders with at least three messages, while dataset 2 A.2 needs, at least twenty.

A.1 Dataset 1

User	Emails Number	Folders Number
beck-s	1971	101
farmer-d	3672	25
kaminski-v	4477	41
kitchen-l	4015	47
lokay-m	2493	11
sanders-r	1188	30
williams-w3	2769	18

Table A.1: Dataset 1 distribution

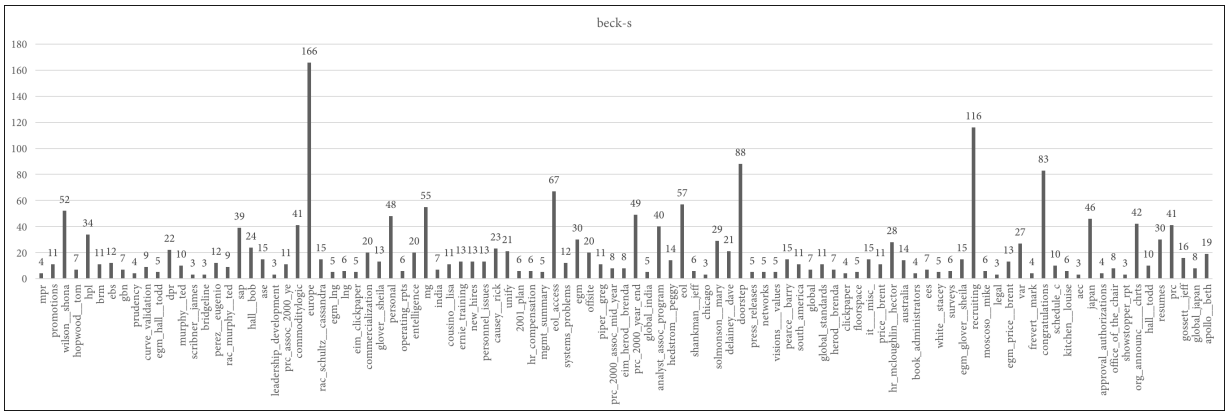

Figure A.1: *beck-s* class distribution

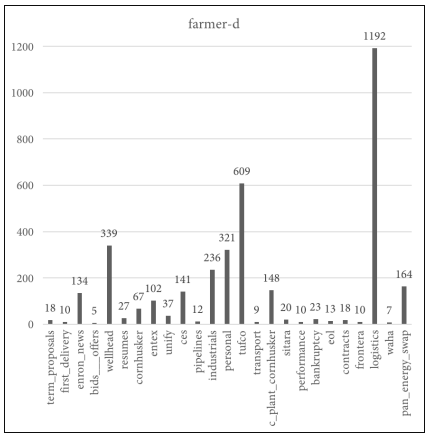

Figure A.2: *farmer-d* class distribution

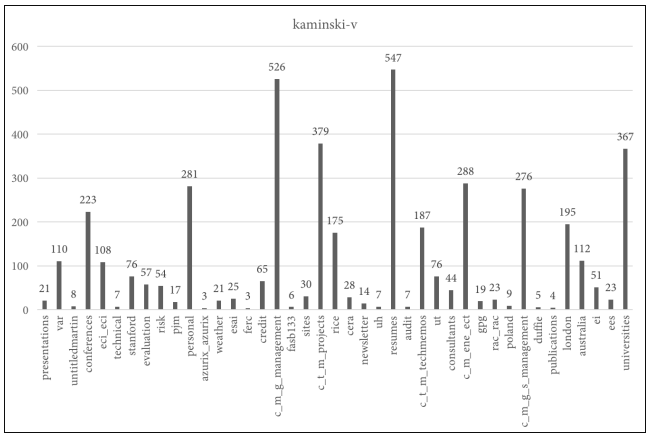

Figure A.3: *kaminski-v* class distribution

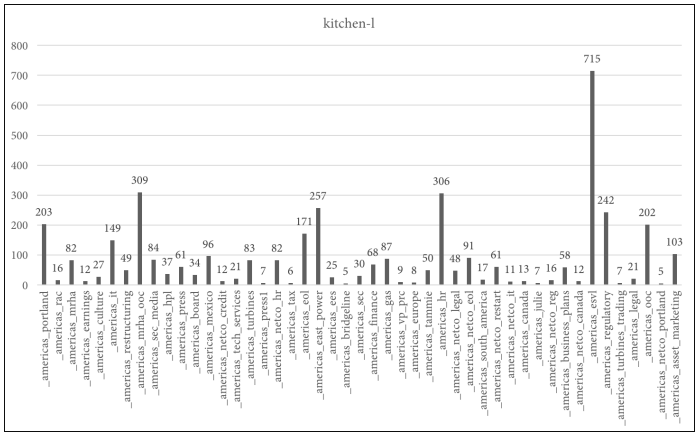

Figure A.4: *kitchen-l* class distribution

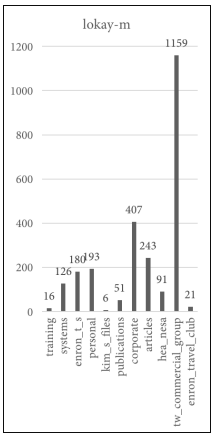

Figure A.5: *lokay-m* class distribution

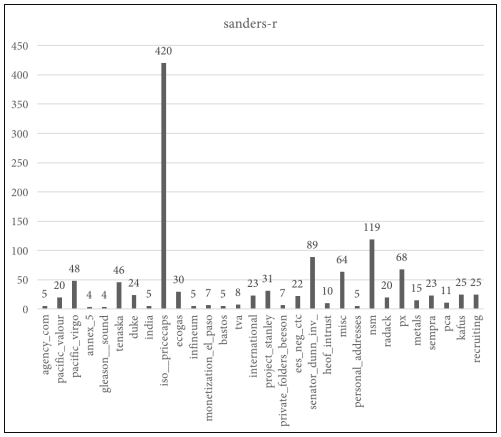

Figure A.6: *sanders-r* class distribution

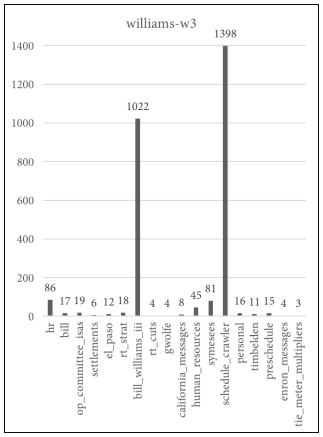

Figure A.7: *williams-w3* class distribution

A.2 Dataset 2

User	Emails Number	Folders Number
beck-s	1379	30
farmer-d	3560	15
kaminski-v	4347	27
kitchen-l	3852	31
lokay-m	2471	9
sanders-r	1097	17
williams-w3	2632	5

Table A.2: Dataset 2 distribution

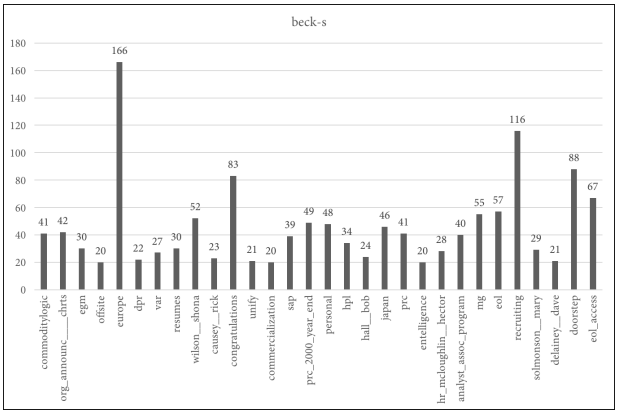

Figure A.8: *beck-s* class distribution

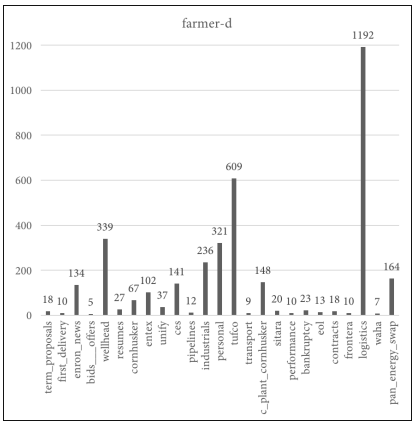

Figure A.9: *farmer-d* class distribution

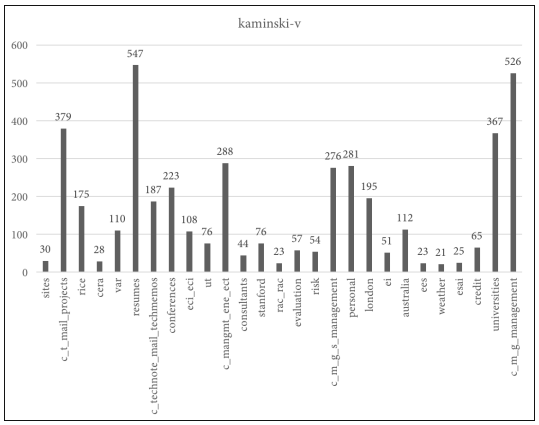

Figure A.10: *kaminski-v* class distribution

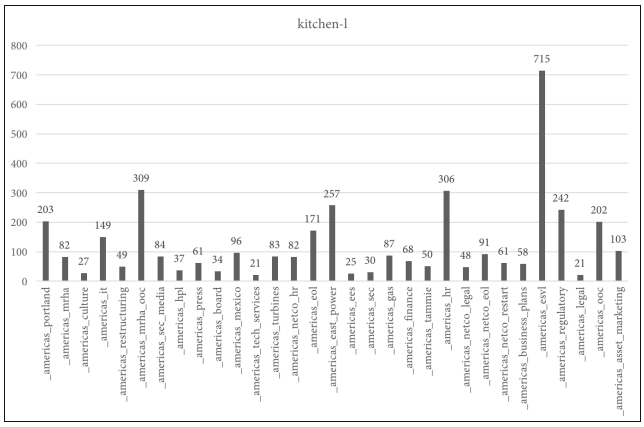

Figure A.11: *kitchen-l* class distribution

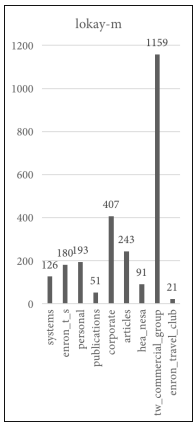

Figure A.12: *lokay-m* class distribution

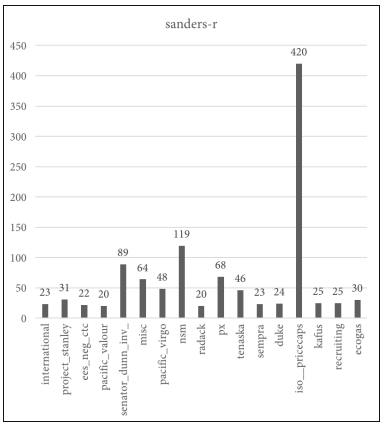

Figure A.13: *sanders-r* class distribution

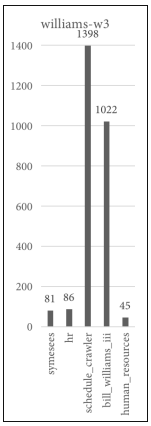

Figure A.14: *williams-w3* class distribution

Appendix B

Phase 1 Results

Phase 1 uses the Dataset 1 (Section A.1) with the following configuration:

Number Runs	Method	Preprocessing Config.	Classifier
5	10-folds cross validation	Config. 0	Naive Bayes
			SVM
			Random Forest
		Config. 1	Naive Bayes
			SVM
			Random Forest
		Config. 2	Naive Bayes
			SVM
			Random Forest
		Config. 3	Naive Bayes
			SVM
			Random Forest
		Config. 4	Naive Bayes
			SVM
			Random Forest
		Config. 5	Naive Bayes
			SVM
			Random Forest
		Config. 6	Naive Bayes
			SVM
			Random Forest
		Config. 7	Naive Bayes
			SVM
			Random Forest

Table B.1: Phase 1 structure

Configuration	Naive Bayes				SMO				Random Forest			
	Accuracy	Std. Dev.	Max	Min	Accuracy	Std. Dev.	Max	Min	Accuracy	Std. Dev.	Max	Min
Config. 0	61,461	3,245	71,066	48,731	67,458	2,869	73,604	56,853	61,258	2,954	67,005	53,299
Config. 1	61,461	3,245	71,066	48,731	67,458	2,869	73,604	56,853	61,370	2,873	66,497	51,269
Config. 2	61,177	3,182	68,528	48,731	66,585	2,899	73,096	56,345	58,569	2,803	62,944	48,223
Config. 3	61,177	3,182	68,528	48,731	66,585	2,899	73,096	56,345	58,772	2,909	64,975	48,731
Config. 4	51,720	2,880	57,868	45,685	56,266	2,807	61,421	48,731	56,012	2,866	62,944	46,701
Config. 5	51,720	2,880	57,868	45,685	56,266	2,807	61,421	48,731	55,992	2,761	62,437	48,223
Config. 6	51,618	3,031	58,883	41,624	55,495	2,759	62,437	50,254	55,769	2,798	62,944	49,239
Config. 7	51,618	3,031	58,883	41,624	55,495	2,759	62,437	50,254	55,961	2,886	62,437	49,239

Table B.2: Results for *beck-s*

Configuration	Naive Bayes				SMO				Random Forest			
	Kappa	T. P. Rate	T. N. Rate	Precision	Kappa	T. P. Rate	T. N. Rate	Precision	Kappa	T. P. Rate	T. N. Rate	Precision
Config. 0	0,604	0,615	0,990	0,625	0,665	0,675	0,991	0,659	0,599	0,613	0,985	0,584
Config. 1	0,604	0,615	0,990	0,625	0,665	0,675	0,991	0,659	0,600	0,614	0,985	0,585
Config. 2	0,601	0,612	0,989	0,613	0,656	0,666	0,990	0,645	0,570	0,586	0,982	0,558
Config. 3	0,601	0,612	0,989	0,613	0,656	0,666	0,990	0,645	0,572	0,588	0,982	0,557
Config. 4	0,505	0,517	0,991	0,547	0,550	0,563	0,989	0,537	0,547	0,560	0,989	0,545
Config. 5	0,505	0,517	0,991	0,547	0,550	0,563	0,989	0,537	0,547	0,560	0,989	0,543
Config. 6	0,504	0,516	0,991	0,546	0,542	0,555	0,989	0,531	0,545	0,558	0,989	0,544
Config. 7	0,504	0,516	0,991	0,546	0,542	0,555	0,989	0,531	0,547	0,560	0,990	0,545

Table B.3: Statistics for *beck-s* (T.P. - True Positive; T.N. - True Negative)

Configuration	Naive Bayes		SMO		Random Forest	
	Time Training (s)	Time Testing (s)	Time Training (s)	Time Testing (s)	Time Training (s)	Time Testing (s)
Config. 0	0,825	9,379	12,215	1,108	36,719	0,083
Config. 1	0,943	10,702	12,184	1,098	22,329	0,059
Config. 2	0,652	7,458	12,149	1,147	18,342	0,052
Config. 3	0,659	7,496	12,454	1,137	18,404	0,052
Config. 4	0,022	0,455	9,833	0,085	4,640	0,034
Config. 5	0,022	0,452	9,822	0,085	4,653	0,034
Config. 6	0,020	0,412	9,782	0,080	4,424	0,034
Config. 7	0,020	0,410	9,789	0,080	4,459	0,034

Table B.4: Performance for *beck-s* (s - seconds)

Configuration	Naive Bayes				SMO				Random Forest			
	Accuracy	Std. Dev.	Max	Min	Accuracy	Std. Dev.	Max	Min	Accuracy	Std. Dev.	Max	Min
Config. 0	73,160	2,056	76,839	67,935	81,209	1,741	86,376	77,929	80,675	1,506	84,469	77,929
Config. 1	73,160	2,056	76,839	67,935	81,209	1,741	86,376	77,929	80,735	1,795	85,831	76,839
Config. 2	73,203	2,001	76,567	67,302	81,471	1,676	85,559	78,202	80,610	1,785	84,741	77,657
Config. 3	73,203	2,001	76,567	67,302	81,471	1,676	85,559	78,202	80,305	1,855	85,014	76,022
Config. 4	72,441	2,000	76,022	67,575	81,356	1,728	85,286	77,929	81,035	1,564	85,286	77,929
Config. 5	72,441	2,000	76,022	67,575	81,356	1,728	85,286	77,929	80,975	1,648	85,014	77,929
Config. 6	72,500	1,856	76,567	67,030	81,395	1,638	86,104	78,261	80,768	1,788	84,741	77,112
Config. 7	72,500	1,856	76,567	67,030	81,395	1,638	86,104	78,261	80,550	1,897	84,741	76,839

Table B.5: Results for *farmer-d*

Configuration	Naive Bayes				SMO				Random Forest			
	Kappa	T. P. Rate	T. N. Rate	Precision	Kappa	T. P. Rate	T. N. Rate	Precision	Kappa	T. P. Rate	T. N. Rate	Precision
Config. 0	0,687	0,732	0,960	0,767	0,775	0,812	0,961	0,810	0,763	0,807	0,942	0,800
Config. 1	0,687	0,732	0,960	0,767	0,775	0,812	0,961	0,810	0,764	0,807	0,943	0,802
Config. 2	0,687	0,732	0,960	0,766	0,778	0,815	0,960	0,812	0,762	0,806	0,940	0,799
Config. 3	0,687	0,732	0,960	0,766	0,778	0,815	0,960	0,812	0,758	0,803	0,939	0,797
Config. 4	0,679	0,724	0,960	0,763	0,777	0,814	0,960	0,812	0,768	0,810	0,946	0,803
Config. 5	0,679	0,724	0,960	0,763	0,777	0,814	0,960	0,812	0,768	0,810	0,946	0,803
Config. 6	0,679	0,725	0,960	0,762	0,777	0,814	0,961	0,811	0,764	0,808	0,942	0,800
Config. 7	0,679	0,725	0,960	0,762	0,777	0,814	0,961	0,811	0,761	0,806	0,941	0,798

Table B.6: Statistics for *farmer-d* (T.P. - True Positive; T.N. - True Negative)

Configuration	Naive Bayes		SMO		Random Forest	
	Time Training (s)	Time Testing (s)	Time Training (s)	Time Testing (s)	Time Training (s)	Time Testing (s)
Config. 0	0,419	1,412	3,022	0,066	0,534	0,064
Config. 1	0,416	1,415	3,036	0,067	0,470	0,063
Config. 2	0,390	1,282	3,265	0,071	,857	0,062
Config. 3	0,393	1,288	3,275	0,071	,867	0,062
Config. 4	0,310	1,071	2,760	0,057	,231	0,060
Config. 5	0,310	1,069	2,758	0,057	,198	0,060
Config. 6	0,316	1,046	3,067	0,062	,862	0,059
Config. 7	0,310	1,034	3,060	0,062	,786	0,059

Table B.7: Performance for *farmer-d* (s - seconds)

Configuration	Naive Bayes				SMO				Random Forest			
	Accuracy	Std. Dev.	Max	Min	Accuracy	Std. Dev.	Max	Min	Accuracy	Std. Dev.	Max	Min
Config. 0	60,607	2,311	65,548	55,357	67,555	2,462	73,602	63,170	67,867	2,425	74,330	63,393
Config. 1	60,607	2,311	65,548	55,357	67,555	2,462	73,602	63,170	67,621	2,562	74,330	62,500
Config. 2	58,642	2,406	64,509	54,241	67,273	2,061	72,321	61,607	67,068	2,351	71,429	62,723
Config. 3	58,642	2,406	64,509	54,241	67,273	2,061	72,321	61,607	67,041	2,480	72,098	62,500
Config. 4	59,290	2,412	65,324	54,688	65,634	2,267	71,588	61,384	67,130	2,399	73,438	60,268
Config. 5	59,290	2,412	65,324	54,688	65,634	2,267	71,588	61,384	67,345	2,250	72,545	62,054
Config. 6	58,893	2,399	63,393	53,571	65,879	2,257	71,875	62,054	67,711	2,015	71,205	63,170
Config. 7	58,893	2,399	63,393	53,571	65,879	2,257	71,875	62,054	67,814	2,292	72,483	62,054

Table B.8: Results for *kaminski-v*

Configuration	Naive Bayes				SMO				Random Forest			
	Kappa	T. P. Rate	T. N. Rate	Precision	Kappa	T. P. Rate	T. N. Rate	Precision	Kappa	T. P. Rate	T. N. Rate	Precision
Config. 0	0,579	0,606	0,971	0,632	0,652	0,676	0,975	0,684	0,653	0,679	0,970	0,691
Config. 1	0,579	0,606	0,971	0,632	0,652	0,676	0,975	0,684	0,650	0,676	0,970	0,688
Config. 2	0,558	0,586	0,970	0,623	0,649	0,673	0,974	0,683	0,644	0,671	0,968	0,685
Config. 3	0,558	0,586	0,970	0,623	0,649	0,673	0,974	0,683	0,643	0,670	0,968	0,685
Config. 4	0,566	0,593	0,973	0,618	0,631	0,656	0,973	0,667	0,647	0,671	0,973	0,674
Config. 5	0,566	0,593	0,973	0,618	0,631	0,656	0,973	0,667	0,649	0,673	0,973	0,676
Config. 6	0,562	0,589	0,972	0,617	0,634	0,659	0,973	0,668	0,652	0,677	0,973	0,679
Config. 7	0,562	0,589	0,972	0,617	0,634	0,659	0,973	0,668	0,654	0,678	0,973	0,679

Table B.9: Statistics for *kaminski-v* (T.P. - True Positive; T.N. - True Negative)

Configuration	Naive Bayes		SMO		Random Forest	
	Time Training (s)	Time Testing (s)	Time Training (s)	Time Testing (s)	Time Training (s)	Time Testing (s)
Config. 0	1,546	8,000	16,110	0,585	18,955	0,126
Config. 1	1,495	7,375	69,926	0,487	30,738	0,168
Config. 2	1,049	63,993	19,120	0,330	15,894	0,108
Config. 3	0,715	3,795	9,922	0,320	15,893	0,108
Config. 4	0,236	1,433	5,653	0,113	12,597	0,100
Config. 5	0,238	1,438	5,686	0,114	12,584	0,099
Config. 6	0,209	1,245	5,844	0,114	11,844	0,098
Config. 7	0,209	1,234	5,842	0,114	11,127	0,095

Table B.10: Performance for *kaminski-v* (s - seconds)

Configuration	Naive Bayes				SMO				Random Forest			
	Accuracy	Std. Dev.	Max	Min	Accuracy	Std. Dev.	Max	Min	Accuracy	Std. Dev.	Max	Min
Config. 0	57,694	2,453	62,687	50,498	63,153	2,034	68,579	59,204	61,430	1,870	65,672	56,965
Config. 1	57,694	2,453	62,687	50,498	63,153	2,034	68,579	59,204	61,425	2,135	65,423	55,473
Config. 2	55,886	2,025	60,349	50,498	62,018	2,025	66,667	57,711	59,044	2,276	63,184	53,234
Config. 3	55,886	2,025	60,349	50,498	62,018	2,025	66,667	57,711	59,463	2,144	63,184	53,483
Config. 4	57,166	2,063	62,095	52,985	62,520	1,734	66,169	58,706	62,685	2,089	67,662	59,204
Config. 5	57,166	2,063	62,095	52,985	62,520	1,734	66,169	58,706	62,775	1,775	66,833	59,204
Config. 6	55,741	2,212	60,697	49,502	61,753	1,887	66,418	57,214	61,350	2,073	65,423	55,473
Config. 7	55,741	2,212	60,697	49,502	61,753	1,887	66,418	57,214	61,609	1,905	65,835	56,965

Table B.11: Results for *kitchen-l*

Configuration	Naive Bayes				SMO				Random Forest			
	Kappa	T. P. Rate	T. N. Rate	Precision	Kappa	T. P. Rate	T. N. Rate	Precision	Kappa	T. P. Rate	T. N. Rate	Precision
Config. 0	0,548	0,577	0,970	0,590	0,604	0,632	0,972	0,645	0,579	0,614	0,958	0,626
Config. 1	0,548	0,577	0,970	0,590	0,604	0,632	0,972	0,645	0,579	0,614	0,958	0,627
Config. 2	0,529	0,559	0,969	0,575	0,592	0,620	0,971	0,632	0,551	0,590	0,952	0,612
Config. 3	0,529	0,559	0,969	0,575	0,592	0,620	0,971	0,632	0,556	0,595	0,953	0,615
Config. 4	0,543	0,572	0,971	0,591	0,597	0,625	0,972	0,640	0,596	0,627	0,966	0,631
Config. 5	0,543	0,572	0,971	0,591	0,597	0,625	0,972	0,640	0,597	0,628	0,966	0,631
Config. 6	0,528	0,557	0,970	0,578	0,589	0,618	0,972	0,631	0,580	0,614	0,962	0,622
Config. 7	0,528	0,557	0,970	0,578	0,589	0,618	0,972	0,631	0,583	0,616	0,962	0,623

Table B.12: Statistics for *kitchen-l* (T.P. - True Positive; T.N. - True Negative)

Configuration	Naive Bayes		SMO		Random Forest	
	Time Training (s)	Time Testing (s)	Time Training (s)	Time Testing (s)	Time Training (s)	Time Testing (s)
Config. 0	0,753	4,542	7,973	0,460	33,387	0,132
Config. 1	0,752	4,432	7,892	0,454	23,420	0,104
Config. 2	0,588	3,438	8,560	0,429	19,209	0,095
Config. 3	0,586	3,460	8,567	0,433	19,233	0,096
Config. 4	0,447	2,833	6,648	0,310	18,919	0,098
Config. 5	0,447	2,835	6,642	0,309	18,929	0,098
Config. 6	0,374	2,279	6,984	0,305	26,588	0,127
Config. 7	0,807	4,842	9,766	0,452	16,147	0,092

Table B.13: Performance for *kitchen-l* (s - seconds)

Configuration	Naive Bayes				SMO				Random Forest			
	Accuracy	Std. Dev.	Max	Min	Accuracy	Std. Dev.	Max	Min	Accuracy	Std. Dev.	Max	Min
Config. 0	76,214	2,211	82,329	69,600	84,926	2,040	88,755	79,920	85,624	1,696	89,558	81,928
Config. 1	76,214	2,211	82,329	69,600	84,926	2,040	88,755	79,920	85,680	1,561	89,157	82,329
Config. 2	73,285	2,097	76,800	67,600	82,840	1,957	87,550	78,715	84,035	1,639	87,550	79,920
Config. 3	73,285	2,097	76,800	67,600	82,840	1,957	87,550	78,715	83,971	1,670	87,149	79,920
Config. 4	75,724	2,100	81,928	70,400	85,118	2,077	89,960	78,313	86,161	1,745	90,361	82,329
Config. 5	75,724	2,100	81,928	70,400	85,118	2,077	89,960	78,313	86,145	1,698	90,361	81,928
Config. 6	73,157	2,256	76,707	67,470	83,249	1,755	87,149	78,715	84,533	1,715	87,952	78,313
Config. 7	73,157	2,256	76,707	67,470	83,249	1,755	87,149	78,715	84,252	1,570	88,353	79,920

Table B.14: Results for *lokay-m*

Configuration	Naive Bayes				SMO				Random Forest			
	Kappa	T. P. Rate	T. N. Rate	Precision	Kappa	T. P. Rate	T. N. Rate	Precision	Kappa	T. P. Rate	T. N. Rate	Precision
Config. 0	0,687	0,762	0,954	0,795	0,793	0,849	0,947	0,849	0,797	0,856	0,927	0,858
Config. 1	0,687	0,762	0,954	0,795	0,793	0,849	0,947	0,849	0,797	0,857	0,927	0,859
Config. 2	0,650	0,733	0,947	0,768	0,765	0,828	0,942	0,828	0,773	0,840	0,920	0,845
Config. 3	0,650	0,733	0,947	0,768	0,765	0,828	0,942	0,828	0,772	0,840	0,920	0,843
Config. 4	0,681	0,757	0,953	0,791	0,795	0,851	0,946	0,850	0,805	0,862	0,933	0,860
Config. 5	0,681	0,757	0,953	0,791	0,795	0,851	0,946	0,850	0,805	0,861	0,933	0,860
Config. 6	0,649	0,732	0,947	0,766	0,769	0,832	0,941	0,832	0,781	0,845	0,924	0,846
Config. 7	0,649	0,732	0,947	0,766	0,769	0,832	0,941	0,832	0,777	0,843	0,925	0,843

Table B.15: Statistics for *lokay-m* (T.P. - True Positive; T.N. - True Negative)

Configuration	Naive Bayes		SMO		Random Forest	
	Time Training (s)	Time Testing (s)	Time Training (s)	Time Testing (s)	Time Training (s)	Time Testing (s)
Config. 0	0,167	0,254	1,514	0,009	5,217	0,046
Config. 1	0,142	0,220	1,349	0,009	4,459	0,037
Config. 2	0,152	0,218	1,551	0,009	4,199	0,035
Config. 3	0,139	0,205	1,589	0,009	3,924	0,037
Config. 4	0,114	0,177	1,120	0,006	3,705	0,031
Config. 5	0,114	0,177	1,125	0,007	3,710	0,032
Config. 6	0,111	0,164	1,323	0,007	3,594	0,031
Config. 7	0,111	0,163	1,322	0,007	3,966	0,033

Table B.16: Performance for *lokay-m* (s - seconds)

Configuration	Naive Bayes				SMO				Random Forest			
	Accuracy	Std. Dev.	Max	Min	Accuracy	Std. Dev.	Max	Min	Accuracy	Std. Dev.	Max	Min
Config. 0	78,756	3,979	88,136	71,186	85,220	3,033	90,756	78,151	82,089	2,939	88,136	73,950
Config. 1	78,756	3,979	88,136	71,186	85,220	3,033	90,756	78,151	81,719	2,669	86,555	73,109
Config. 2	76,685	3,773	86,441	69,492	84,867	3,218	91,525	78,151	79,160	3,350	85,593	70,588
Config. 3	76,685	3,773	86,441	69,492	84,867	3,218	91,525	78,151	79,311	3,054	84,746	71,429
Config. 4	74,731	3,758	83,051	66,387	85,270	3,095	93,277	78,151	85,019	3,161	90,678	74,790
Config. 5	74,731	3,758	83,051	66,387	85,270	3,095	93,277	78,151	85,018	3,289	90,756	75,630
Config. 6	73,957	4,011	83,051	64,706	84,093	3,232	89,916	74,790	84,328	2,974	89,076	74,790
Config. 7	73,957	4,011	83,051	64,706	84,093	3,232	89,916	74,790	84,075	3,036	89,076	73,950

Table B.17: Results for *sanders-r*

Configuration	Naive Bayes				SMO				Random Forest			
	Kappa	T. P. Rate	T. N. Rate	Precision	Kappa	T. P. Rate	T. N. Rate	Precision	Kappa	T. P. Rate	T. N. Rate	Precision
Config. 0	0,754	0,788	0,974	0,813	0,823	0,852	0,967	0,845	0,777	0,821	0,926	0,803
Config. 1	0,754	0,788	0,974	0,813	0,823	0,852	0,967	0,845	0,772	0,817	0,924	0,797
Config. 2	0,731	0,767	0,973	0,798	0,818	0,849	0,963	0,839	0,737	0,792	0,908	0,775
Config. 3	0,731	0,767	0,973	0,798	0,818	0,849	0,963	0,839	0,739	0,793	0,910	0,772
Config. 4	0,712	0,747	0,977	0,806	0,825	0,853	0,973	0,860	0,820	0,850	0,966	0,845
Config. 5	0,712	0,747	0,977	0,806	0,825	0,853	0,973	0,860	0,820	0,850	0,966	0,845
Config. 6	0,704	0,740	0,978	0,800	0,811	0,841	0,969	0,845	0,810	0,843	0,958	0,832
Config. 7	0,704	0,740	0,978	0,800	0,811	0,841	0,969	0,845	0,807	0,841	0,958	0,832

Table B.18: Statistics for *sanders-r* (T.P. - True Positive; T.N. - True Negative)

Configuration	Naive Bayes		SMO		Random Forest	
	Time Training (s)	Time Testing (s)	Time Training (s)	Time Testing (s)	Time Training (s)	Time Testing (s)
Config. 0	0,220	0,919	1,504	0,049	3,547	0,023
Config. 1	0,214	0,893	1,454	0,049	3,215	0,022
Config. 2	0,172	0,698	1,550	0,052	2,804	0,021
Config. 3	0,189	0,748	1,594	0,052	3,029	0,021
Config. 4	0,058	0,280	1,161	0,020	1,958	0,017
Config. 5	0,060	0,289	1,137	0,019	1,904	0,018
Config. 6	0,054	0,251	1,168	0,020	1,779	0,018
Config. 7	0,055	0,250	1,169	0,020	1,827	0,017

Table B.19: Performance for *sanders-r* (s - seconds)

Configuration	Naive Bayes				SMO				Random Forest			
	Accuracy	Std. Dev.	Max	Min	Accuracy	Std. Dev.	Max	Min	Accuracy	Std. Dev.	Max	Min
Config. 0	90,935	1,613	94,224	87,365	96,699	0,925	98,195	94,585	95,825	0,991	97,826	93,141
Config. 1	90,935	1,613	94,224	87,365	96,699	0,925	98,195	94,585	95,652	0,983	97,834	93,141
Config. 2	91,592	1,457	94,585	88,043	96,613	0,977	98,556	94,585	95,608	1,036	97,834	93,116
Config. 3	91,592	1,457	94,585	88,043	96,613	0,977	98,556	94,585	95,428	0,907	97,112	93,141
Config. 4	90,820	1,486	93,863	87,365	96,562	0,999	98,195	94,585	95,847	0,932	97,473	93,141
Config. 5	90,820	1,486	93,863	87,365	96,562	0,999	98,195	94,585	95,832	0,984	97,834	93,478
Config. 6	91,390	1,534	94,585	87,004	96,735	0,945	98,195	94,585	95,710	0,927	97,834	93,502
Config. 7	91,390	1,534	94,585	87,004	96,735	0,945	98,195	94,585	95,666	0,964	97,834	92,780

Table B.20: Results for *williams-w3*

Configuration	Naive Bayes				SMO				Random Forest			
	Kappa	T. P. Rate	T. N. Rate	Precision	Kappa	T. P. Rate	T. N. Rate	Precision	Kappa	T. P. Rate	T. N. Rate	Precision
Config. 0	0,855	0,909	0,990	0,940	0,945	0,967	0,990	0,963	0,930	0,958	0,983	0,942
Config. 1	0,855	0,909	0,990	0,940	0,945	0,967	0,990	0,963	0,927	0,957	0,982	0,941
Config. 2	0,865	0,916	0,990	0,941	0,944	0,966	0,990	0,961	0,926	0,956	0,981	0,939
Config. 3	0,865	0,916	0,990	0,941	0,944	0,966	0,990	0,961	0,923	0,954	0,980	0,937
Config. 4	0,853	0,908	0,989	0,940	0,943	0,966	0,990	0,962	0,931	0,958	0,983	0,944
Config. 5	0,853	0,908	0,989	0,940	0,943	0,966	0,990	0,962	0,930	0,958	0,983	0,944
Config. 6	0,862	0,914	0,990	0,941	0,946	0,967	0,991	0,964	0,928	0,957	0,982	0,941
Config. 7	0,862	0,914	0,990	0,941	0,946	0,967	0,991	0,964	0,927	0,957	0,981	0,941

Table B.21: Statistics for *williams-w3* (T.P. - True Positive; T.N. - True Negative)

Configuration	Naive Bayes		SMO		Random Forest	
	Time Training (s)	Time Testing (s)	Time Training (s)	Time Testing (s)	Time Training (s)	Time Testing (s)
Config. 0	0,312	0,801	0,819	0,031	3,567	0,026
Config. 1	0,296	0,771	0,732	0,027	3,437	0,025
Config. 2	0,265	0,670	0,780	0,029	3,102	0,025
Config. 3	0,268	0,678	0,760	0,028	3,099	0,025
Config. 4	0,224	0,592	0,697	0,022	2,804	0,023
Config. 5	0,225	0,592	0,699	0,022	3,049	0,026
Config. 6	0,472	1,202	1,305	0,049	6,601	0,049
Config. 7	0,590	1,534	1,217	1,878	2,813	0,025

Table B.22: Performance for *williams-w3* (s - seconds)

Appendix C

Phase 2 and 3 Results

C.1 Phase 2

Phase 2 uses the Dataset 1 (Section A.1).

User	Accuracy	Std. Deviation	Max	Min
beck-s	67,316	2,882	73,604	56,853
farmer-d	82,320	1,729	86,376	78,747
kaminski-v	68,439	2,073	73,826	64,063
kitchen-l	63,831	1,917	68,657	59,453
lokay-m	87,012	1,839	90,763	83,936
sanders-r	86,146	3,123	93,277	78,151
williams-w3	96,822	0,964	98,195	94,585

Table C.1: Phase 2 results

User	Kappa	T.P. Rate	F.P. Rate	T.N. Rate	F.N. Rate	Precision	Recall
beck-s	0,663	0,673	0,010	0,990	0,327	0,656	0,673
farmer-d	0,787	0,823	0,041	0,959	0,177	0,817	0,823
kaminski-v	0,661	0,684	0,025	0,975	0,316	0,692	0,684
kitchen-l	0,611	0,638	0,028	0,972	0,362	0,651	0,638
lokay-m	0,819	0,870	0,055	0,945	0,130	0,868	0,870
sanders-r	0,835	0,861	0,027	0,973	0,139	0,868	0,861
williams-w3	0,947	0,968	0,009	0,991	0,032	0,963	0,968

Table C.2: Phase 2 statistics

User	Time Training (s)	Time Testing (s)
beck-s	63,841	2,683
farmer-d	19,923	0,207
kaminski-v	41,688	0,737
kitchen-l	47,209	0,868
lokay-m	7,329	0,054
sanders-r	4,213	0,051
williams-w3	6,182	0,084

Table C.3: Phase 2 performance results (s - seconds)

C.2 Phase 3

Phase 3 uses the Dataset 2 (Section A.2).

User	Accuracy	Std. Deviation	Max	Min
beck-s	79,797	3,277	86,232	71,739
farmer-d	83,433	1,727	86,517	78,090
kaminski-v	69,648	1,748	74,194	65,438
kitchen-l	65,208	1,930	70,909	61,818
lokay-m	87,212	1,613	90,688	83,401
sanders-r	87,384	3,158	93,636	80
williams-w3	98,936	0,662	100	96,958

Table C.4: Phase 3 results

User	Kappa	T.P. Rate	F.P. Rate	T.N. Rate	F.N. Rate	Precision	Recall
beck-s	0,787	0,798	0,011	0,989	0,202	0,814	0,798
farmer-d	0,798	0,834	0,042	0,958	0,166	0,836	0,834
kaminski-v	0,673	0,696	0,026	0,974	0,304	0,709	0,696
kitchen-l	0,624	0,652	0,029	0,971	0,348	0,672	0,652
lokay-m	0,820	0,872	0,059	0,941	0,128	0,872	0,872
sanders-r	0,844	0,874	0,035	0,965	0,126	0,882	0,874
williams-w3	0,981	0,989	0,004	0,996	0,011	0,990	0,989

Table C.5: Phase 3 statistics

User	Time Training (s)	Time Testing (s)
beck-s	8,523	0,110
farmer-d	43,389	0,438
kaminski-v	262,203	3,152
kitchen-l	61,135	0,780
lokay-m	14,249	0,127
sanders-r	8,206	0,086
williams-w3	4,280	0,050

Table C.6: Phase 3 performance results (s - seconds)

Appendix D

Phase 4, 5 and 6 Results

D.1 Phase 4

Phase 4 uses the Dataset 1 (Section A.1).

User	Accuracy	Max	Min
beck-s	79,507	83,283	76,418
farmer-d	96,014	96,955	94,631
kaminski-v	84,959	87,385	83,114
kitchen-l	85,151	87,839	82,417
lokay-m	96,721	97,995	95,165
sanders-r	93,978	96,535	91,337
williams-w3	99,012	99,681	97,981

Table D.1: Phase 4 results

D.2 Phase 5

Phase 5 uses the Dataset 2 (Section A.2).

User	Accuracy	Max	Min
beck-s	91,842	95,096	88,699
farmer-d	97,375	98,430	96,446
kaminski-v	86,158	88,701	83,424
kitchen-l	86,658	88,855	84,885
lokay-m	96,855	98,095	95,595
sanders-r	95,903	98,391	91,957
williams-w3	99,931	100	99,665

Table D.2: Phase 5 results

D.3 Phase 6

Phase 6 uses the Dataset 2 (Section A.2).

User	Accuracy	Max	Min
beck-s	93,639	94,456	93,177
farmer-d	97,563	98,347	96,287
kitchen-l	87,568	88,701	85,927
kaminski-v	87,412	88,473	86,565
lokay-m	97,327	97,759	96,905
sanders-r	96,458	96,783	96,247
williams-w3	99,799	100	99,464

Table D.3: Phase 6 results