Department of Computer Architecture and Technology
Konputagailuen Arkitektura eta Teknologia saila (KAT)
Departamento de Arquitectura y Tecnología de Computadores (ATC)

Universidad del País Vasco
Euskal Herriko Unibertsitatea

University of the Basque Country

INFORMATIKA FAKULTATEA
FACULTAD DE INFORMÁTICA

Behaviour modelling with data obtained from the Internet and contributions to cluster validation

Ph.D. Dissertation presented by
Iñigo Perona Balda

Supervised by
Olatz Arbelaitz Gallego
Javier Muguerza Rivero

Donostia, December 2015

Department of Computer Architecture and Technology
Konputagailuen Arkitektura eta Teknologia saila (KAT)
Departamento de Arquitectura y Tecnología de Computadores (ATC)

Universidad del País Vasco
Euskal Herriko Unibertsitatea

University of the Basque Country

INFORMATIKA FAKULTATEA
FACULTAD DE INFORMÁTICA

Behaviour modelling with data obtained from the Internet and contributions to cluster validation

Ph.D. Dissertation presented by
Iñigo Perona Balda

Supervised by
Olatz Arbelaitz Gallego
Javier Muguerza Rivero

Donostia, December 2015

This work was carried out with the grant of the Basque Government (ref.: BFI08.226) and with the grant of Spanish Ministry of Economy and Competitiveness (ref.: BES-2011-045989). The grant for the research stay was funded by the Spanish Ministry of Economy and Competitiveness (ref.: EEBB-I-14-08862).
Moreover, this work was funded by the University of the Basque Country UPV/EHU (BAILab, grant UFI11/45); by the Department of Education, Universities and Research of the Basque Government (grant IT-395-10); and by the Ministry of Economy and Competitiveness of the Spanish Government and by the European Regional Development Fund - ERDF (eGovernAbility, grant TIN2014-52665-C2-1-R).

"Sudur puntan jartzen zaiona egin beza
inor kaltetu gabe."

Erleak, zizta egitean hiltzen badira ere
erlauntza eta hurkoa lehenesten dute.
Bakezale eta gerlariak. Nortasuna.

Abstract

This PhD thesis makes contributions in modelling behaviours found in different types of data acquired from the Internet and in the field of clustering evaluation. Two different types of Internet data were processed, on the one hand, internet traffic with the objective of attack detection and on the other hand, web surfing activity with the objective of web personalization, both data being of sequential nature. To this aim, machine learning techniques were applied, mostly unsupervised techniques. Moreover, contributions were made in cluster evaluation, in order to make easier the selection of the best partition in clustering problems.

With regard to network attack detection, first, gureKDDCup database was generated which adds payload data to KDDCup99 connection attributes because it is essential to detect non-flood attacks. Then, by modelling this data a network Intrusion Detection System (nIDS) was proposed where context-independent payload processing was done obtaining satisfying detection rates.

In the web mining context web surfing activity was modelled for web personalization. In this context, generic and non-invasive systems to extract knowledge were proposed just using the information stored in webserver log files. Contributions were done in two senses: in problem detection and in link suggestion. In the first application a meaningful list of navigation attributes was proposed for each user session to group and detect different navigation profiles. In the latter, a general and non-invasive link suggestion system was proposed which was evaluated with satisfactory results in a link prediction context.

With regard to the analysis of Cluster Validity Indices (CVI), the most extensive CVI comparison found up to a moment was carried out using a partition similarity measure based evaluation methodology. Moreover, we analysed the behaviour of CVIs in a real web mining application with elevated number of clusters in which they tend to be unstable. We proposed a procedure which automatically selects the best partition analysing the slope of different CVI values.

Laburpena

Doktorego-tesi honek internetetik eskuratutako datu mota ezberdinetan aurkitutako portaeren modelugintzan eta multzokatzeen ebaluazioan egiten ditu bere ekarpenak. Zehazki, bi mota ezberdinetako interneteko datuak prozesatu dira: batetik, interneteko trafikoa, erasoak hautemateko helburuarekin; eta bestetik, web nabigazioen jarduera, weba pertsonalizatzeko helburuarekin; bi datu motak izaera sekuentzialekoak direlarik. Helburu hauek lortzeko, ikasketa automatikoko teknikak aplikatu dira, nagusiki gainbegiratu-gabeko teknikak. Testuinguru honetan, multzokatzeen partizio onenaren aukeraketak dakartzan arazoak gutxitzeko multzokatzeen ebaluazioan ere ekarpenak egin dira.

Sareko erasoen hautemateari dagokionez, lehenik gureKDDCup datubasea eratu da KDDCup99-ko konexio atributuei payload-ak (sareko paketeen datu eremuak) gehituz, izan ere, ez-flood erasoak (pakete gutxi erabiltzen dituzten erasoak) hautemateko ezinbestekoak baitira. Ondoren, datu hauek modelatuz testuinguruarekiko independenteak diren payload prozesaketak oinarri dituen sareko erasoak hautemateko sistema (network Intrusion Detection System (n-IDS)) bat proposatu da maila oneko eraso hautemate-tasak lortuz.

Web meatzaritzaren testuinguruan, weba pertsonalizatzeko helburuarekin web nabigazioen jarduera modelatu da. Honetarako, web zerbizarietako lorratz fitxategietan metatutako informazioa soilik erabiliz ezagutza erabilgarria erauziko duen sistema orokor eta ez-inbasiboak proposatu dira. Ekarpenak bi zentzutan eginaz: arazoen hautematean eta esteken iradokitzean. Lehen aplikazioan sesioen nabigazioa adierazteko atributu esanguratsuen zerrenda bat proposatu da, gero nabigazioak multzokatu eta nabigazio profil ezberdinak hautemateko. Bigarren aplikazioan, estekak iradokitzeko sistema orokor eta ez-inbasibo bat proposatu da, eta berau, estekak aurresateko testuinguruan ebaluatu da emaitza onak lortuz.

Multzokatzeak balioztatzeko indizeen (Cluster Validity Indices (CVI)) azterketari dagokionez, gaurdaino aurkitu den CVI-en konparaketa zabalena burutu da partizioen antzekotasun neurrian oinarritutako ebaluazio metodologia erabiliz. Gainera, CVI-en portaera aztertu da egiazko web meatzaritza aplikazio batean normalean baino multzo kopuru handiagoak dituena, non CVI-ek ezegonkorrak izateko joera baitute. Arazo honi aurre eginaz, CVI ezberdinek partizio ezberdinetarako lortzen dituzten balioen maldak aztertuz automatikoki partiziorik onena hautatzen duen prozedura proposatu da.

Resumen

Esta tesis doctoral hace contribuciones en el modelado de comportamientos encontrados en diferentes tipos de datos adquiridos desde internet y en el campo de la evaluación del clustering. Dos tipos de datos de internet han sido procesados: en primer lugar el tráfico de internet con el objetivo de detectar ataques; y en segundo lugar la actividad generada por los usuarios web con el objetivo de personalizar la web; siendo los dos tipos de datos de naturaleza secuencial. Para este fin, se han aplicado técnicas de aprendizaje automático, principalmente técnicas no-supervisadas. Además, se han hecho aportaciones en la evaluación de particiones de clusters para facilitar la selección de la mejor partición de clusters.

Respecto a la detección de ataques en la red, primero, se generó la base de datos gureKDDCup que añade el payload (la parte de contenido de los paquetes de la red) a los atributos de la conexión de KDDCup99 porque el payload es esencial para la detección de ataques no-flood (ataques que utilizan pocos paquetes). Después, se propuso un sistema de detección de intrusos (network Intrusion Detection System (IDS)) modelando los datos de gureKDDCup donde se propusieron varios preprocesos del payload independientes del contexto obteniendo resultados satisfactorios.

En el contexto de la minería web, se ha modelado la actividad de la navegación web para la personalización web. En este contexto se propondrán sistemas genéricos y no-invasivos para la extracción del conocimiento, utilizando únicamente la información almacenada en los ficheros log de los servidores web. Se han hecho aportaciones en dos sentidos: en la detección de problemas y en la sugerencia de links. En la primera aplicación, se propuso una lista de atributos significativos para representar las sesiones de navegación web para después agruparlos y detectar diferentes perfiles de navegación. En la segunda aplicación, se propuso un sistema general y no-invasivo para sugerir links y se evaluó en el contexto de predicción de links con resultados satisfactorios.

Respecto al análisis de índices de validación de clusters (Cluster Validity Indices (CVI)), se ha realizado la más amplia comparación encontrada hasta el momento que utiliza la metodología de evaluación basada en medidas de similitud de particiones. Además, se ha analizado el comportamiento de los CVIs en una aplicación real de minería web con un número elevado de clusters, contexto en el que los CVIs tienden a ser inestables, así que se propuso un procedimiento para la selección automática de la mejor partición en base a la

pendiente de los valores de diferentes CVIs.

Acknowledgements

Niretzako, hitzak ez du nahikoa indarrik ibilbide honen partaide izan zaretenoi merezi duzuen adina esker on azaltzeko, baina zerbait borobiltzen saiatuko naiz.

- Lehenik eta behin eskerrik asko tesi honen bi zuzendariei, bai Olatzi eta baita Javiri ere, zareten bezelakoak izateagatik.
- Eskerrik asko Ibai, zuk urratutako bidean asko ibili bainaiz, asko zor dizut.
- Eskerrik asko Aizea, zure uztatik ere baibaitu lan honek baina batik bat emaniko konfiantzagatik.
- Eskerrik asko Igor, Ainhoa eta Aizea, tesi hau berrikusken harturiko lanengatik, eta bulegokide apartak izate arren.
- Eskerrik asko hiri ere Joseba, hi ere bulegokidetzat baihaugu, eta lan honen inprimatze aukerei buruzko xehetasunak errazte arren.
- Eskerik asko Mendi eta Momori tesi aurkezkiei buruzko zalantzak argitzen laguntzeagatik.
- Eskerrik asko Iñaki eta Txus, CTC kafeen hastapenetan izandako elkarrizketa interesgarriak ez zaizkit berehalakoan ahaztuko.
- Laburrean eskerrik asko ALDAPA talde osoari, Olatz, Javi, Ibai, Txus, Natxo, Agus, Joseba, Igor, Ainhoa eta Aizeari. Lan hau zuena ere bada.
- Eskerrik asko bulegokide-ohiei, Nikos, Lierni, Otzeta, Javi, Aritz, Iñaki, Carlos, Unai eta Ana; baita Ugaitzi ere; eta ahaztu ditudanei. Baita egokituz eta dsg taldeko partaideei, bereziki Juliori.
- Eskerrik asko Patrasen egin nuen egonaldiko nire bi gain-begiratzaileei, eskerrik asko Evangelos eta eskerrik asko Christo. Aipamen berezia egin nahi nieke Evangelos eta Alexiari etxekotzat hartzeagatik, eskerrik asko.

  Many thanks to my supervisors of Patras, thank you Evangelos and thank you Christo. And thank you very much to Evangelos and Alexia for taking me as a member of your family. I wish you all the best.

- Egunero-egunero bazkaltzera deitzen nauten guztiei, baina batik bat, Unai eta Davidi. Eta oro har, bazkalkide eta kafekide zaituztedanoi eta fakultatean hartu-emana, edo hobeto esanda, hala baita gizalegea, lehenik eman eta gero hartu, eman-hartua izan dudan orori.
- Koadrilakoei, bereziki Mendez eta Gurruri. Hor zaituztedala badakidalako.
- Abaurrea eta konpainiari, Euskal Herriko herri garaienetik itsasertzeraino, garai bateko ipin-oholak bezela ibiltzeko aukera emateagatik, baina batik bat emaniko konfiantzagatik.
- Eskerrik asko etxekoei, ama, aita eta arrebari. Bihotz-bihotzez eskerrik asko.
- Aiton-amonei eta zeruko eguzki pozkidaz beteari.

Contents

Part I

Introduction

Chapter 1

Introduction to Behaviour Modelling

A behaviour is the way that a person, an animal, a substance, a machine, etc. has to act in a particular situation or under particular conditions. If we record all the conditions with their corresponding behaviours over time, we will have stored data to research upon the behaviours. Discovering patterns in this data and analysing it, will facilitate us to create a model to explain how the subject behaves providing us with the ability to predict future actions. This process of making sense of data is known as behaviour modelling and the humans are keen on doing it.

Consciously humans try to find explanations to many observed events or behaviours and unconsciously too, we are very skilful at making abstractions and decisions in new contexts using our historical experiences or the experiences of other people. We can even imagine the future of our decisions and predict presumably the best one. In other words, we try to find or understand the models or logics that lie under these behaviours. Hence, people have been seeking patterns in data since human life began. For example, hunters seek patterns in animal migration behaviour or farmers seek patterns in crop growth. A scientist's job is to make sense of data, to discover the patterns that govern how the physical or psychological world works and encapsulate them in theories that can be used to predict future behaviours.

However these theories are usually not perfect, they could have some not modelled exceptions which, in the best cases, adding them to the model could fix the theory, or in the worst case, could invalidate the model. This model shift can be seen clearly in how we have modelled the cosmos during the history. In the ancient era the prevalent model was geocentric, including a flat Earth in the middle of the universe. However, this model had flaws, for example, it was not able to explain why stars move as they do. Afterwards, the model of the spherical Earth revolving around the sun was proposed and it explained better the observational data of stars. Nevertheless, it was not accurate enough and

improvements were added such as considering the orbit of the Earth around the sun elliptical instead of spherical. Later on, the general theory of relativity was proposed to explain better the gravitational interaction of these spatial bodies.

Through history humans have observed all types of behaviours and we have analysed them to discover patterns on them and model them. Therefore by interpreting this information humans tried to infer logical explanations of those observed behaviours. Those models and explanations would be validated as far as possible, and lastly, new knowledge would be created. All this process of manual knowledge generation was quite time-consuming or it needed some enlightened people to infer theories. However, when Alan Turing designed a general purpose computer around the year 1940, new ways of knowledge generation were created. He defined the basis for machine computation making possible to automatically discover patterns that humans were not able to discover. A new science branch was born: computer science.

During the next decades computer science evolved and in the 1990 decade personal computers and with them, the Internet was spread all over the world. Nowadays, broadly used technological devices, from desktop to pocket or wrist size, are computers with their Central Processing Unit (CPU) and all are interconnected into the Internet. Thence, technology allows us to capture vast quantities of data as never before did. This rapid growth of the amount of data and its dimensionality makes it impossible to analyse it manually. For example, many applications have too many features that are very difficult to assess simultaneously and correlate making appropriate abstractions. Consequently the need of using machines' computational power rose. The idea is to build computer programs that shift through databases automatically, seeking regularities or patterns. Afterwards if strong patterns or simple models are found, which it make possible to understand how the raw data behaves, they are used to make accurate predictions on future data.

In a nutshell, the aim is turning data into information and turning information into knowledge, which is known as the "Knowledge Discovery in Databases" (KDD) process [68]. In other words, it is the process of converting low-level data into more useful forms that might be more compact or more abstract. The main steps of this process are data preparation, data selection, data cleaning, incorporation of prior knowledge, data mining and proper interpretation and validation of the results to ensure that useful and correct knowledge has been discovered. This process usually needs to return to the previous steps and execute them again making more accurate decisions in each run.

The core step of the KDD process is the data mining step [199] which tries to discover patterns (and models) in the databases involving computational methods such as statistics and machine learning. Machine learning is a branch of artificial intelligent which explores the algorithms that can learn models or discover patterns to describe data or make predictions. It is composed by practical techniques that can often extract useful information from raw data. Machine learning techniques emerge the structure underlying the data which is used to facilitate the understanding of the analysed data mining application and are usually also able to analyse bigger quantities of data than the traditional sta-

tistical models. It is believe that every time the size of the data increases 10 times (one magnitude order) a new era of models and algorithms have to be considered.

In this dissertation two data mining applications are researched: (1) modelling network traffic to detect intruders and (2) modelling users' behaviours on websites for web personalization. Being these applications' data very unstructured, the modelling process is quite challenging.

The first data mining application emerges from the popularity of computer networks. According to Cisco [46], the annual global IP traffic will surpass the zettabyte threshold in 2016, globally, Internet traffic will grow 3.2 times from 2014 to 2019, a compound annual growth rate of 26%. This fact broadens the scope for network attackers and increases the damage these attacks can cause. Network attacks compromise the stability of the network or the security of the information stored in the computers connected to it. Therefore, it is very important to build systems that are able to detect attacks before they can cause damage. Intrusion Detection Systems (IDS) form part of any complete security package and their goal is to detect any action that violates the security policy of a computer system. More precisely we will focus on a particular type of IDS that works by scanning the network traffic and is able to automatically detect intrusions: network Intrusion Detection System (nIDS). The detection of network attacks by human analysis, where memorization, looking up description libraries or searching sample collections is required, is not effective; it is too time consuming and subjective. As a consequence, automated and robust nIDSs are required in order to successfully confront the problem.

The need of the second data mining application emerges because the information on the web has increased drastically and this often makes the amount of information intractable for users. To have an idea about the dimensions of the web information we are generating we could analyse, The Wayback Machine [109] of the Internet Archive, which stores publicly available webpages over time, by 01/12/2015 contained 13 petabytes of data and is currently growing at a rate of 50-60 terabytes per week. Although we are working in each individual website, the need for websites to be useful in an efficient way for users is specially important. Larger quantities of data do not provide added value for web visitors; there is a need for easier access to the required information and adaptation to their preferences or needs. That is, web personalization becomes essential. In this direction, systems were proposed based on logs of websites of general interest such as NASA (National Aeronautics and Space Administration) [145], EPA (United States Environmental Protection Agency) [63], and SDSC (San Diego Supercomputer Center) [176]; and based on more specialized websites' logs such as "Bidasoa Turismo" a tourism website and "Discapnet" a website aimed at visually impaired people which are the people who most need the web personalization.

In this type of applications, as we do not have any prior knowledge about the type of behaviours the users have, we can only obtain unlabelled data, and thus, unsupervised machine learning techniques are used to discover the structure underlying the data. The most used techniques are clustering and

apriori [3]. The first group of techniques divides the input space into clusters. The goal of a clustering algorithm is to obtain a partition where objects within a cluster are similar and objects in different clusters are dissimilar. The second group of techniques finds frequent itemsets in transactional databases which can be used to determine association rules to highlight general trends in the database. Their most famous application is the market basket analysis. In this work an evolution of apriori techniques has been used, which takes into account the time when the events were generated, creating a new family of algorithms: Sequential Pattern Mining (SPM).

Although we used both unsupervised machine learning techniques, the use of clustering techniques has been more intensive. As defined before, clustering breaks the unlabelled examples into clusters or assigns an abstract label to each of them. Afterwards these clusters need to be analysed or interpreted by experts (or remodelled by supervised machine learning algorithms) to link the cluster with the corresponding behaviour.

When using clustering algorithms a relevant question arises: how well does the proposed partition fit the input data? The process of estimating how well a partition fits the structure underlying the data is known as cluster validation [82]. The most commonly used evaluation process is based on Cluster Validity Indices (CVIs). However, there is no CVI that can be considered best performing in all cases. In this direction, a more theoretic analysis was carried out: an extensive comparison of CVIs and the analysis of their usability for databases with elevated number of cluster was carried out and a method to use the combination of some CVIs in this context proposed.

1.1 Organization of this dissertation

This thesis has been divided into five different parts: Introduction, Background, Contributions, Conclusions and Appendixes.

Part I, Introduction, consists of a single chapter (Chapter 1) in which a brief introduction to the core of this thesis has been done, introducing the work carried out, recalling the motivation of this work, as well as marking the main objectives of this thesis.

Part II, Background, is devoted to explain the fundamental concepts and techniques used along this dissertation. Likewise the notation used in this dissertation is introduced. This part is composed by two chapters: Machine learning (Chapter 2) and Unsupervised learning techniques (Chapter 3). The former introduces the machine learning field and how the techniques in this field can be evaluated, whereas the latter focuses in unsupervised machine learning techniques, explaining all the unsupervised techniques used in this dissertation.

In Part III the three main contributions are presented in detail divided in chapters: Contributions to network security (Chapter 4), Contributions to web mining (Chapter 5) and Contributions to cluster validation (Chapter 6). These three chapters have a similar structure: first, an introductory section where the context and the problem are presented; in the second section, the related

material is prepared to experiment with it. Finally, next sections are devoted to describe different solutions to the presented problems and experiments to validate them. These three chapters compose the core of this dissertation.

Part IV, Conclusions, is composed by a single chapter of the same name (Chapter 7). In this chapter, the main contributions of the dissertation are outlined and the main conclusions are drawn, as well as the open lines we hope to address in the future. After this chapter the referenced bibliography is presented and next, the last part includes (Part V) a series of appendixes. They include additional information about network security database we generated and published (Appendix A and Appendix B).

Part II

Background: Structuring data

Chapter 2

Machine learning

2.1 Introduction

Machine learning is a subfield of computer science that evolved from the study of pattern recognition and computational learning theory in artificial intelligence. Machine learning explores the study and construction of algorithms that can learn from data and make predictions on data. Such algorithms operate by building a model from example inputs in order to make data-driven predictions or decisions, rather than following strictly static program instructions.

Machine learning techniques are usually part of the data mining process. Data mining is the computational process of discovering patterns in large datasets involving methods at the intersection of artificial intelligence, machine learning, statistics, and database systems. The overall goal of the data mining process is to extract information from a dataset and transform it into an understandable structure for further use. Aside from the raw analysis step, it involves database and data management aspects, data preprocessing, model and inference considerations, interestingness metrics, complexity considerations, post-processing of discovered structures, visualization, and online updating. Data mining is the analysis step of the "Knowledge Discovery in Databases" process, or KDD.

Any machine learning algorithm needs data to train the models. This data, is composed by cases, also called examples or instances. These examples can be represented in different forms, such as vectors of features (see Table 2.1) or sequences of events. These examples can be labelled or unlabelled, i.e., they can have an associated class or not.

Once the database is ready, machine learning algorithms can be applied. However, traditionally the most used models were statistical models which found the underlying distribution in the observable data. For example, having a set of numbers we can decide that those numbers are generated by a Gaussian distribution and thus, we can find the most likely parameter values (mean and standard deviation) of this Gaussian distribution which would become the model

Instance	A_1	A_2	...	A_F	Class
X_1	x_{11}	x_{21}	...	x_{F1}	θ_2
X_2	x_{12}	x_{22}	...	x_{F2}	θ_K
...	...	...	...	...	...
X_N	x_{1N}	x_{2N}	...	x_{FN}	θ_1

Table 2.1: An abstract example of a database, where the database has N examples with F attributes, and X_i represents the i-th example attribute vector and θ_i represents the class the example belongs to.

of the data.

Machine learning uses data to train algorithms and create models which are a structured representation of the data. For example, it can be used to predict the score one user will give to a certain product, for spam filtering, optical character recognition (OCR), search engines, computer vision, etc.

Machine learning algorithms are traditionally divided into two groups: (1) supervised learning and (2) unsupervised learning. In the next sections both groups are explained respectively.

2.2 Supervised learning

Supervised learning is also called classification or inductive learning [49]. Supervised learning is the task of inferring a function from labelled training data. Training data needs to include the input instances with their desired results. At least, for some examples the correct results (or dependent variable, class, labels) are known and are given as input to the model during the learning process.

The goal of the supervised learning is to build an artificial system that can learn a function for mapping between the input and the output, and can predict the output of the system given new inputs. If the output takes a finite set of discrete values that indicate the class labels of the input, the learned mapping leads to the classification of the input data. If the output takes continuous values, it leads to a regression of the input [129].

There are different supervised learning techniques and some of the most widely used ones are: k Nearest Neighbours (kNN) [69], decision trees [166], bayesian networks [152], Naive Bayes [138], neural networks [135], and Support Vector Machines (SVM) [37, 47]. The adequacy of each model depends on the objective pursued in the learning process and the type of data stored. In the behaviour modelling works carried in this thesis out we used kNN.

2.2.1 k Nearest Neighbours (kNN)

kNN [69, 48] assigns to the new example the class which is most frequently represented among its k nearest neighbours in the training set. There are three key elements of this approach: (1) a set of labelled training examples, (2) a distance metric to compute the distance between examples, and (3) the value

of k, the number of nearest neighbours. To classify an unlabelled example, the distances of this example to the labelled examples are computed, its k nearest neighbours are identified, and the class labels of these nearest neighbours are then used to determine the class label of the object. The kNN algorithm is showed in Algorithm 1.

Algorithm 1 kNN algorithm.

```
1: neighbourhood = {};
2: x the new example;
3: for each example y in the training set do
4:    Calculate the distance d(x, y);
5:    Add y to neighbourhood if d(x, y) is into the k smallest distances;
6: end for
7: x.class = SelectClass(neighbourhood);
```

There are several issues that affect the performance of kNN. One of them is the selection of k. If k is too small, then the result can be sensitive to noisy points. On the other hand, if k is too large, then the neighbourhood may include too many points from other classes. Another issue is the approach used to combine the class labels. The simplest method is to use the majority vote, but this can be a problem if the nearest neighbours vary widely in their distance and the closer neighbours indicate more reliably the class of the object. A more sophisticated approach, which is usually much less sensitive to the choice of k, weighs each example's vote by its distance.

The choice of the distance metric is another important issue. Although various metrics can be used to compute the distance between two points, the most desirable distance metric would be one for which a smaller distance between two objects implies a greater likelihood of having the same class. Different distances are analysed in Section 3.1.

2.3 Unsupervised learning

The other subgroup of machine learning techniques is called unsupervised learning. In this case the input data is not labelled and as a consequence, a model is prepared by deducing structures present in the input data. Generally the unsupervised learning methods are divided into association rule learning, sequential pattern mining (that is a further extension to the concept of association rule learning) and clustering.

The work presented in this dissertation makes intensive use of unsupervised learning methods, mainly of clustering algorithms and sequential pattern mining methods which are going to be described thoroughly in Chapter 3.

2.4 Performance metrics

Classifiers need to be evaluated and usually some performance metrics are used with this goal. But before explaining the various metrics, it is important to explain what a confusion matrix is (also referred to as cross tabulation or crosstab), since much of the metrics are based on that table.

The confusion matrix [115] is a table where rows show the number of instances belonging to each class and columns the number of instances classified as belonging to each class. In bi-classical problems (problems with only two classes), the minority class or class with fewer examples is also known as positive and the majority class as negative. In these cases, the matrix will have two rows and two columns, and therefore, four possible values:

- **True Positive (TP)**: number of positive instances classified as positive.
- **True Negative (TN)**: number of negative instances classified as negative.
- **False Positive (FP)**: number of negative instances classified as positive.
- **False Negative (FN)**: number of positive instances classified negative.

The matrix is completed as in Table 2.2:

		Prediction	
		P	N
Reality	T	TP	FN
	F	FP	TN

Table 2.2: Confusion matrix

Different metrics to evaluate the performance of a classifier can be calculated using the mentioned values (the range of possible values is $[0-1]$, 1 being the best value and 0 the worst):

- **Accuracy (acc)**: or hit rate; percentage of correctly classified instances.

$$acc = \frac{TP + TN}{TP + FP + FN + TN} = \frac{T}{T + F} \tag{2.1}$$

- **Error rate (err)**: percentage of wrongly classified instances.

$$err = \frac{FP + FN}{TP + FP + FN + TN} = \frac{F}{T + F} = 1 - acc \tag{2.2}$$

Analysing the confusion matrix two types of errors can be seen: FP and FN, which are known as type I error and type II error respectively. Usually those two errors do not have the same importance, as it is illustrated in the following example. In a medical test, at the beginning, with no evidence to the contrary,

one is considered to be healthy (null hypothesis, that is, he/she is a member of the negative class). In type I error the test shows a patient to have a disease when in fact the patient does not have the disease (incorrect rejection of true null hypothesis or FN error). In the type II error the test fails detecting the disease, in a patient who really has the disease (failure to reject the false null hypothesis or FP error). In this example, it is not the same to make one error or the other. The objective of designing this medical test is to make FP error zero (because it is the crucial one) while keeping the FN error as low as possible (but some failures are admissible because it is not so crucial). However, the presented accuracy and error rate metrics use the total number of failures or mistakes (no type of errors taken into account). Therefore, these metrics do not provide enough information to approve the analysed test.

As a result, new metrics were proposed which take into account different types of errors:

- **Precision (pr)**: percentage of instances that are actually positive among those who have been classified as such.

$$pr = \frac{TP}{TP + FP} \tag{2.3}$$

- **Recall (re)**: or sensitivity or True Positive Rate (TPR); percentage of correctly classified positive instances.

$$re = \frac{TP}{TP + FN} \tag{2.4}$$

- **F-measure (Fm)**: it is a metric that tries to balance the successes in both metrics, precision and recall. It is calculated using the harmonic mean of the precision and the recall.

$$F\beta - m = (1 + \beta^2)\frac{pr \cdot re}{(\beta^2 \cdot pr) + re} \tag{2.5}$$

  The usual β value is 1 (F1-m), but depending on the characteristics of the classifier one want to analyse it is common to find other variations of the F-measure, trying to give more importance to precision or to recall. For instance, F2-m gives more weight to recall, while F0.5-m, gives it to precision.

- **Specificity (sp)**: percentage of negative cases classified as such.

$$sp = \frac{TN}{TN + FP} \tag{2.6}$$

- **Combining two metrics in a plot**: this method graphically combines two previously mentioned metrics over threshold values. First, a test is carried out and it gives a probability of being a member of the positive

class or the negative class for each example. Therefore, fixing a threshold we can decide if each example is classified positively or negatively. And moving the threshold from 0 to 1 different classifications can be obtained for the same test. Analysing these plots the best threshold value is selected keeping the crucial error to zero and the other as low as possible.

The following graphics are the most used ones:

- **Receiver Operating Characteristic (ROC) curve**: is a graphical representation of the relationship between the True Positive Rates (TPRs) and False Positive Rates (FPRs). In the plot, the X axis represents FPRs and the Y axis represents TPRs.

  Many times the ROC curve is summarized to Area Under ROC Curve (AUC). In the case of the ideal classifier, the area under the curve would be 1, while in the case of a random classifier it would be 0.5. Generally, the classifier with greater AUC is chosen as the best classifier.

- **Precision-Recall (PR) curve**: is a graphical representation of the relationship between the precision and the recall. In the plot, the X axis represents recall and the Y axis represents precision. It is mostly used with skewed datasets.

2.5 Validation methods

One of the important issues when designing effective machine learning models is the validation and refinement of the acquired knowledge. So, evaluation is one of the key points in any data mining process. The goal of every classifier is to correctly classify new cases submitted once the learning is done. The future behaviour of a classifier is estimated using statistical sampling tools and then the possible bias in the estimation (optimistic, pessimistic) and its variability are analysed. The process can serve two purposes: the prediction of how well the final model will work in the future, and as an integral part of many learning methods, to find the model that best represents the training data.

When the dataset of labelled instances is available but there are no independent test samples, there are several strategies, known as classification error estimators or validation methods that allow to calculate an estimation error given a classifier. These methods consist mainly on creating two sets: training and test-sets. The training set, used for training or building the model, and the test-set, a subset of unseen examples used for assessing the performance of the model. In addition to that, sometimes another subset of the dataset called validation-set is used which provides a testing platform to tune model's parameters and selecting the best-performing model.

Although these methods are presented as the estimation of the error, all the performance metrics commented in the previous section (see Section 2.4) could be estimated by the following methods.

Depending on how the database is used to make the estimation, two types of errors are differentiated: (1) Apparent error and (2) True error.

- **Apparent error rate.** Apparent error rate is the error rate obtained by the classifier after classifying the same cases that have been used for training. Normally it uses all data available to create the model. In other words, the same data used for training the model is used for estimating the performance of that model. This generates a non-realistic and overoptimistic or over-fitted prediction considered unacceptable by the data mining community [184]. If the training sample was infinite (or the whole population) the apparent error would be equal to the true error, but it is common to have a sample much smaller than the population.

- **True error rate.** For a more realistic estimation, for the training set some instances from the sample must be selected randomly. For the test-set, the cases not used in the learning phase must be used. When these assumptions are carried out, it is said that a true error estimation is obtained. There are different methods to perform this estimation such as, hold-out, K-fold cross-validation, leave-one-out or bootstrapping methods.

2.5.1 Hold-out

The hold-out method [121] partitions the dataset into two mutually exclusive subsets: one for training and the other one for testing. The proportion of the data in each subset can vary, but normally, the 2/3 of the data are used for training and 1/3 for testing.

The hold-out method has two basic drawbacks. On the one hand since it is a single training set experiment, the hold-out estimation of the error rate will be misleading if we happen to get an "unfortunate" split. On the other hand, if limited amount of data is available, the chances of the split to not to be a representation of the real behaviour are high. Therefore, more complex methods are necessary to achieve an unbiased estimation of the model performance over unseen examples.

To reduce the effect of these limitations this estimation method is repeated several times, using randomly generated training-test partitions. The final estimation will be the average of the values obtained from each run. This repeated process is called random subsampling or repeated hold-out.

2.5.2 K-fold cross-validation

In K-fold cross-validation [91], the original dataset is randomly divided into K disjoint subsets (folds) approximately of the same size and used to carry out K different experiments. For each of the K experiments (training and testing process), the method uses $K-1$ folds for training and the remaining one for testing. Commonly the K parameter is given a value of 10 [114]. The final estimate will be the average of all the estimates obtained.

K-fold cross-validation is similar to random subsampling. However, the advantage of K-fold cross-validation is that all the examples in the dataset are evenly used for both training and testing. On the other hand, the disadvantage is that this method presents a higher computational cost than the hold-out method because the process is repeated K times but it obtains less biased estimations.

As in random subsampling, this process can be repeated iterating a random K-fold cross-validation process several times. The final estimation will be the average of the values obtained from each run.

More exhaustive methods exist such as the leave-one-out or bootstrapping [61] but in this dissertation hold-out and K-fold cross-validation methods have been used. These methods are the most used ones because they are computationally more viable than the leave-one-out method and bootstrapping method [61]. The latter repeat the computation of partial validation metrics around N times, being N the number of cases in the training set, while the former compute fewer partial validations.

2.6 Statistical comparison of classifiers

It is often necessary to compare several classifiers over the same problem in order to decide which of them achieves better results. However, in order to obtain a meaningful decision, it is necessary to carry out a statistical analysis.

The statistical tests consider as null hypothesis that all the methods being compared obtain similar results with non-significant differences. Then the rejection of the null hypothesis means that statistical differences among the classifiers compared exist.

Statistical tests are divided in two branches depending on the assumptions they make:

- **Parametric tests.** This type of tests assume that the data follows a type of probability distribution and makes inferences about the parameters of the distribution.

- **Non-parametric tests.** This type of tests does not rely on data following any particular distribution. Therefore, order statistics, which is based on the ranks of observations is used.

Generally speaking, parametric methods make more assumptions than non-parametric methods. If those extra assumptions are correct, parametric methods can produce more accurate and precise estimates. They are said to have more statistical power. However, if assumptions are incorrect, parametric methods can be very misleading.

Another way to classify statistical tests is depending on the number of classifiers they compare. Two families of tests can be differentiated: (1) tests that compare two classifiers and (2) test that compare multiple classifiers.

In this dissertation, when we use statistical tests for comparing different options, we follow the guidelines presented by Demšar [53]. Demšar recommends to use the non-parametric Wilcoxon signed ranks test to compare two classifiers and the non-parametric Friedman test with the corresponding post-hoc tests to compare more classifiers over multiple datasets. Precisely, we used the latter.

2.6.1 Friedman test

The Friedman test [71] and the Iman-Davenport [98] extension are non-parametric tests to observe if statistical differences exist among several classifiers over several datasets.

The Friedman test assumes that k learning algorithms have been tested on N datasets. Let c_i^j be the performance score of the j-th algorithm on the i-th dataset. The task is to decide whether, based on the values c_i^j, the algorithms are significantly different and, in the case of more than two algorithms, which are the particular algorithms that differ in performance. We will not record the variance of these scores, $\sigma_{c_i^j}$, but we will assume that the measured results are "reliable"; to that end, we require that enough experiments were done on each dataset (for example, considerable cross-validation splits) and, preferably, that all the algorithms were evaluated using the same random samples.

The Friedman test ranks the classifiers for each dataset separately and compares the average ranks of them:

$$R_i = \frac{1}{N} \sum_{j=1}^{N} r_i^j \tag{2.7}$$

being r_i^j the rank of the i-th of k algorithms and the j-th of N datasets. Knowing this, Friedman's statistics is distributed as follows:

$$\chi_F^2 = \frac{12N}{k(k+1)} \left[\sum_{i=1}^{n} R_i^2 - \frac{k(k+1)^2}{4} \right] \tag{2.8}$$

under the null hypothesis, a chi-square distribution with $(k-1)$ degrees of freedom.

Iman and Davenport demonstrated that Friedmans χ_F^2 presents a conservative behaviour and proposed a better statistic:

$$F_F = \frac{(N-1)\chi_F^2}{N(k-1) - \chi_F^2} \tag{2.9}$$

which follows an F-distribution with $(k-1)$ and $(k-1)(N-1)$ degrees of freedom.

Under the null-hypothesis, these tests state that all the classifiers are equivalent. On the other hand, the rejection of the null-hypothesis implies the existence of statistical differences among the classifiers. In order to find the specific pair-wise comparisons which produce differences, a post-hoc procedure

must be carried out. These procedures for comparing the i-th and j-th classifiers are based on the following statistic test:

$$z = \frac{R_i - R_j}{\frac{k(k+1)}{6N}} \tag{2.10}$$

where R_i is the average rank computed through the Friedman test for the i-th classifier, k is the number of classifiers to be compared and N is the number of datasets used in the comparison. The z value is used to find the corresponding probability (p-value) from the table of normal distribution, which is then compared with an appropriate level of significance α.

Below the post-hoc procedures used in this dissertation are explained.

- **Nemenyi post-hoc procedure.** The Nemenyi test [147] adjusts the value of α in a single step by dividing the value of α by the number of comparisons performed ($p_i < \alpha/m$, where $m = k(k-1)/2$).

- **Shaffer post-hoc procedure.** García and Herrera [73] suggested using the Shaffer post-hoc procedure [180]. This procedure is an extension of Holm's procedure. Holm's adjusts the value of α in a step down method. Let $p_1, ..., p_m$ be the ordered p-values (from smallest to largest) and $H_1, ..., H_m$ be the corresponding hypothesis. Holm's procedure rejects H_1 to $H_{(i-1)}$ if i is the smallest integer such that $p_i > \alpha/(m-i+1)$. While Shaffer's procedure rejects H_i if $p_i \leq (\alpha/ti)$, where t_i is the maximum number of hypotheses which can be true given that any $(i-1)$ hypotheses are false.

Chapter 3

Unsupervised learning techniques

Unsupervised techniques are machine learning techniques designed to work with unlabelled data. These are the most used techniques through this dissertation and this chapter is devoted to describe the techniques about unsupervised learning required to understand the rest of the dissertation.

3.1 Finding Similar Items

A fundamental data mining problem is to examine data for "similar" items. For this aim many similarity measures and distances have been defined. Depending on the nature of the data different metrics are used. For example, Jaccard similarity [99, 101] can be used to compare sets; numerical vectors can be compared using euclidean and Manhattan distances or their angle can be computed using cosine similarity; nominal vectors can be compared using Hamming distance [85]; Hellinger distance [89] is used to compare probability distributions; and two sequences can be compared using edit distance [125].

Due to the diverse nature of data used in this dissertation, many different distances were used, the most important ones are described in the next subsections.

3.1.1 Distances for vectors

3.1.1.1 Numerical vectors

Euclidean distance. This is the most commonly chosen type of distance because it is the one we normally think of as "distance" (straight-line distance). An n-dimensional euclidean space is one where points are vectors of n real numbers and the euclidean distance to compare vectors X and Y is defined as

in Equation 3.1.

$$d(X,Y) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2} \tag{3.1}$$

where n is the number of elements of the vector and x_i and y_i represent the values taken by vectors X and Y in position i.

This specific distance in the euclidean space, it is also known as L_2-norm. However, the L_r-norm distance, the generic one, for any constant r, is defined as follows:

$$d(X,Y) = (\sum_{i=1}^{n}|x_i - y_i|^r)^{1/r} \tag{3.2}$$

Therefore, r=2 refers to the euclidean distance.

Manhattan distance. Another common distance metric is the L_1-norm, or Manhattan distance. In this case the distance line between two points is constrained to travel along grid lines.

Maximum distance. Another interesting distance measure is the L_∞-norm, which is the limit as r approaches infinity of the L_r-norm. As r gets larger, only the dimension with the largest difference matters, so formally, the L_∞-norm is defined as the maximum of $|x_i - y_i|$ over all dimensions i.

3.1.1.2 Nominal vectors: Hamming distance

Given a space of vectors (n nominal values), we define the Hamming distance between two vectors as a number of components in which they differ.

3.1.2 Distances for sequences

3.1.2.1 Edit distance

Edit distance [81, 44] measures distances between strings or more generically sequences, and it quantifies how dissimilar two sequences (e.g., words) are by counting the minimum number of operations required to transform one sequence into the other. The allowed operations are the deletion and insertion of a single character, and in some cases the substitution of one character could be included. The operations could have the same or different costs depending on the environment where it is used. For example, the edit distance between x =‘abcd’ and y =‘abc’ is 1 because we have to delete the character ‘d’ from x to convert it to y or otherwise insert ‘d’ in y to convert it to x.

Another way to define and calculate the edit distance is to compute a longest common subsequence (LCS) of x and y. LCS consists in deleting positions from

x and y, to obtain the longest common string that underlies. Therefore, edit distance can be calculate as follows:

$$d(x, y) = length(x) + length(y) - 2LCS \tag{3.3}$$

3.1.2.2 Alignment methods

The alignment methods try to find similarities or align sequences. Global alignment methods try to align the complete sequence, whereas the local methods focus on partial sequences.

Global alignment method (GA). The Needleman-Wunsch algorithm [146], attempts to align every element of the two sequences using the dynamic programming method (see Algorithm 2). This algorithm is more useful when the sequences are quite similar and more or less of equal size. The match is awarded by a positive number and the mismatch and the gap/spaces by negative numbers. The edit distance is equivalent to the global sequence alignment's cost score.

Algorithm 2 Global alignment algorithm.

```
1: Read the two sequences to be aligned: A and B;
2: Read the gap penalty (d) usually -1;
3: Initialize the matrix F of (length(A)+1) x (length(B)+1);
4: for i = 0 to length(A) do
5:    F(i,0)= d * i;
6: end for
7: for j = 0 to length(B) do
8:    F(0,j)= d * j;
9: end for
10: for i = 1 to length(A) do
11:    for j = 1 to length(B) do
12:       Match = F(i − 1, j − 1) + S(A(i), B(j));
13:       Delete = F(i − 1, j) + d;
14:       Insert = F(i, j − 1) + d;
15:       F(i,j) = max(Match, Insert, Delete);
16:    end for
17: end for
18: FinalScore = F(length(A),length(B));
```

Local alignment methods (LA). The Smith-Waterman algorithm [182], attempts to align similar sequence regions within their larger sequence context using the dynamic programming method (see Algorithm 3). This algorithm is used with dissimilar sequences. It differs from global alignment methods in the initialization phase of the score matrix, which is not biased to start the alignment from the beginning.

Algorithm 3 Local alignment algorithm.

```
Read the two sequences to be aligned: A and B;
Read the gap penalty (d) usually -1;
Initialize the matrix F of (length(A)+1) x (length(B)+1);
for i=0 to length(A) do
   F(i,0)= 0;
end for
for j = 0 to length(B) do
   F(0,j)= 0;
end for
for i = 1 to length(A) do
   for j = 1 to length(B) do
      Match = F(i − 1, j − 1) + S(A(i), B(j));
      Delete = F(i − 1, j) + d;
      Insert = F(i, j − 1) + d;
      F(i,j) = max(Match, Insert, Delete);
   end for
end for
FinalScore = F(length(A),length(B));
```

Combination of GA and LA (GL). GA and LA methods make use of dynamic programming and the score matrix in which we get the cost of aligning two sequences. After the score matrix is calculated, the backtracking process starts from the bottom-right corner moving up-vertically (↑), up-left (↖) or left-horizontally (←), always selecting the move to the minimum score. Having A and B sequences, the up-left or back-diagonal move (↖) means that the two sequences match in that position; the up-vertical move (↑) means that in the A sequence a gap has to be added in that position; and left-horizontal move (←) means that in the B sequence a gap has to be added in that position. When we arrive to the start-cell (0,0) the path drawn in the score matrix indicates one of the optimum alignments of the two strings (A' and B').

Once the two aligned sequences are backtracked, gapped versions of A and B of the same length are generated, A' and B'. Upon this alignment more ad-hoc scores can be defined by first defining the cost of the match (w_m), mismatch (w_{ms}) and the gap (w_g), and secondly counting the number of matches (#matches), mismatches (#mismatches) and gaps (#gaps). The final score can be calculated as follows:

$$\begin{aligned} FinalScore = & w_m * (\#matches) \\ & - w_{ms} * (\#mismatches) \\ & - w_g * (\#gaps) \end{aligned} \tag{3.4}$$

Therefore, having two sequences A and B, and the mentioned two alignment methods global alignment (GL) and local alignment (LA) the combined score

is defined as:

$$\begin{aligned} CombinedScore = (1-p) * score^{LA}(A,B) \\ +p * score^{GA}(A,B) \end{aligned} \tag{3.5}$$

where $score^{LA}(A,B)$ is the final local alignment score of the sequences A and B; $score^{GA}(A,B)$ is the final global alignment score for these sequences; and p is a parameter that expresses the importance that we give to the scores of the two different alignments in the sum. One option is to make p relative to the ratio of the lengths of the two sequences. Thus, assuming that $|A| \geq |B|$ we can define p as $p = |A|/|B|$.

Multiple Sequence Alignment (MSA). This method attempts to align more than two sequences [81]. In its implementation, an iterative pair-wise alignment is done in order to produce the alignment of all the sequences. This iterations are based on the idea that the solution can be computed by modifying an already suboptimal solution. The used approach first selects the sequence with the maximum average similarity (the medoid of the set). Afterwards, the medoid is aligned iteratively with other sequences and a new version of the medoid is created, that is, a medoid with gaps. Finally, the new medoid with gaps is aligned with other sequences, giving as the output a set of aligned sequences all together. Algorithm 4 describes the steps in more detail.

Algorithm 4 Multiple Sequence Alignment (MSA) algorithm.

```
1: Compute the medoid of the given sequences using global alignment score;
2: medoidGap = medoid;
3: for each sequence (seq) not the medoid do
4:     gAlign = global(medoidGap, seq);
5:     medoidGap = gAlign.getSeqA
6: end for{Prepare the medoid with gaps}
7: msaSeq = medoidGap
8: for each sequence (seq) not the medoid do
9:     gAlign = global(medoidGap, seq);
10:    seqGap = gAlign.getSeqB;
11:    msaSeq = msaSeq ∪ seqGap;
12: end for{Align the medoid with gaps with other sequences}
13: Return all sequences aligned: msaSeq;
```

3.1.2.3 Normalized Compression Distance (NCD)

The Normalized Compression Distance (NCD) [126, 198] is an appropriate distance to measure similarity relations between sequences, which is based on the non-computable notion of Kolmogorov complexity.

First, lets define the Normalized Information Distance (NID) which is the distance that uses Kolmogorov complexity (K). The distance between sequences

x and y is the length of the shortest program P that computes x from y and vice versa. This shortest program is in a fixed programming language and uses the Turing machine. Therefore, when the program is longer the K will be higher and vice versa. For normalization, the length of the shortest program between the length to create x from zero or y from zero is used.

$$NID = \frac{max\{K(x|y), K(y|x)\}}{max\{K(x), K(y)\}} \tag{3.6}$$

Since in practice Kolmogorov complexity cannot be computed, it is approximated with real compression algorithms (C). Therefore, in this case $C(x)$ is defined as the length of the string obtained by compressing x with compressor C, and $C(xy)$ as the length of the string obtained by compressing the concatenation of x and y. The NCD distance is defined in Equation 3.7.

$$NCD_C(x, y) = \frac{C(xy) - min(C(x), C(y))}{max(C(x), C(y))} \tag{3.7}$$

Actually, NCD is a family of distances parametrized with the compressor Z. The compressors used in this dissertation are:

- Gzip which is based on the deflate data compression algorithm.
- Bzip2 which is based on the Burrows-Wheeler transformation algorithm.

3.2 Clustering data

Clustering is the unsupervised pattern classification method that partitions the input space into clusters. The goal of a clustering algorithm is to perform a partition where examples within a cluster are similar and examples in different clusters are dissimilar. Therefore, the purpose of clustering is to identify natural structures in a dataset. Obviously the concept of similarity or dissimilarity between examples depends on the selected similarity metric (Section 3.1). Consequently the used metric is one of the key factors in clustering.

Clustering algorithms can be applied in many fields, for example in marketing, finding groups of customers with similar preferences; in biology, classification of plants and animals given their features; in libraries, for book ordering; in document and weblog classification, discovering groups of similar access patterns, among others.

Before interpreting, the partition done by the clustering algorithm needs to be close to the correct number of natural clusters to not to complicate the human task of making sense of those clusters. In practice each cluster represents an abstract behaviour which humans may be able to relate to a real behaviour. This is a challenging task and has led to many clustering algorithms [102].

There are two main types of clustering algorithms. If the algorithm assigns one cluster to each example, it is denominated a crisp clustering algorithm. However, if the algorithm assigns a probability of being in each cluster to each

example, it is denominated a fuzzy clustering algorithm. In this dissertation the first group of algorithms has been used.

Depending on the cluster generation strategy used, we can divided clustering algorithms in two groups: (1) Hierarchical clustering algorithms and (2) Partitional clustering algorithm.

Clustering algorithms can also be differentiated by the space in which the points are placed: (1) algorithms that assume an euclidean space and (2) algorithms that work with an arbitrary distance measure. The key distinction is that in an euclidean space it is possible to summarize a collection of points by their centroid –the average of the points. In a non-euclidean space, there is no notion of centroid, and we are forced to develop another way to summarize clusters.

3.2.1 Hierarchical clustering

These algorithms build a hierarchy of clusters from a database. Two types of strategies exist:

- **Agglomerative:** this is a "bottom up" approach: each observation starts in its own cluster, and pairs of clusters are merged as one moves up the hierarchy.
- **Divisive:** this is a "top down" approach: all observations start in one cluster, and splits are performed recursively as one moves down the hierarchy.

The one used in this dissertation is of agglomerative type an it is explained below.

3.2.1.1 Hierarchical agglomerative clustering algorithm

As Sequential Agglomerative Hierarchical Non-overlapping (SAHN) hierarchical clustering algorithm is the most used agglomerative clustering algorithm, many times it is simply called the hierarchical agglomerative clustering algorithm, but following the guidelines of Sneath and Sokal [183] we will call to this method Sequential Agglomerative Hierarchical Non-overlapping (SAHN).

The algorithm starts by creating one cluster with each case. Afterwards, iteratively, it joins the two nearest cases into a new cluster until all cases are in one cluster. Consequently, it creates a hierarchy with as many partitions of clusters as cases in the training set which is a binary tree (and is graphically represented in a dendrogram structure). See Algorithm 5 for more details.

There are different linkage criterias to measure the distance between clusters, below are some of them explained:

- **Single-linkage criteria:** the distance between clusters A and B is considered the distance between the two nearest instances $a \in A$ and $b \in B$.

Algorithm 5 SAHN algorithm.

1: Create one cluster with each case in the training set (the initial set of clusters (partition) with N elements: S_1);
2: Compute the distance matrix of all N clusters in S_1 (the initial distance matrix: M_1);
3: **for** $i = 2$ **to** N **do**
4: Join the nearest two clusters into a new cluster (and update the set of clusters (partition) with one cluster less: S_i);
5: Update the distance matrix with the clusters in S_i;
6: **end for**
7: Output: hierarchy of clusters (S_1, S_2, ..., S_N);

- **Complete-linkage criteria:** the distance between clusters A and B is considered the distance between the two farthest instances $a \in A$ and $b \in B$.

- **Average-linkage criteria:** also known as Unweighted Pair Group Method Average (UPGMA). The distance between clusters A and B is considered the average distance between all the $a \in A$ and all the $b \in B$ instances.

- **Centroid-linkage criteria:** also known as Unweighted Pair Group Method Centroid (UPGMC). The distance between clusters A and B is considered the distance between the centroid of A (c^A) and the centroid of B (c^B).

- **Medoid-linkage criteria:** the distance between clusters A and B is considered the distance between the medoid of A (m^A) and the medoid of B (m^B).

- **Ward-linkage criteria [196]:** in this criteria, in each iteration of the algorithm an objective function is calculated taking all clusters of the current level into account and the best pair of clusters are merged. In the case of Ward-linkage criteria the objective function is the sum of square errors.

The hierarchical clustering process provides a hierarchy of clusters but the aim of a clustering algorithm is usually to extract a single partition of clusters. In the following lines two of the methods to do so will be explained.

- **Height based partition extractor:** (also known as number of clusters based or minimum dissimilarity threshold based partition extractor): it is the *de facto* used one. This extractor selects a level from the hierarchy of clusters (graphically it cuts the dendrogram horizontally), obtaining a partition of clusters.

Therefore, the level parameter needs to be indicated, and to do so, there are three options. (1) When the height percentage is provided, its equivalent value in the range 1 to N is computed and the corresponding hierarchy level is selected. (2) When the number of clusters (K) is provided, the K-th level from the top of the hierarchy is selected. (3) When the dissimilarity value is provided, the first level with greater merge distance than the provided value is selected.

- **SEP/COP partition extractor**: it is a self-regulated method to extract a partition from a hierarchy of clusters proposed by our research group [77]. The idea behind this method is that in some cases just taking a determined level from the hierarchy (a horizontal cut of the dendrogram) is not an optimal solution, even though the optimal partition is hidden in the hierarchy of clusters. Graphically speaking, the dendrogram can be cut horizontally or in a more flexible way. To do so, we used SEP (Search over the Extended Partition set) which looks for the best partition among all the possible partitions of the dendrogram, including those that need a non-horizontal cut. This search space is called the extended partition set. The results obtained from the extended partition set must be at least as good as the results obtained from the hierarchy. The use of SEP combined with COP would have the advantage of not having to explore a wide range of values of K looking for the best partition as it is required with most clustering algorithms. The algorithm for SEP can be seen in Algorithm 6 where CVI is the cluster validity index used to measure the optimality of a partition. CVIs evaluate partitions according to the cohesion of the objects inside the cluster and the difference or distance of the objects among clusters. Yet, SEP needs a special CVI so that the index can also evaluate partial partitions. Because of that, we use COP (Context-independent Optimality and Partiality). This index is defined in Section 6.2.2.

3.2.2 Partitional clustering

This type of clustering algorithms find a single partition in the data. Below, the clustering algorithms used in this dissertation are explained: K-means, K-medoids and fixed-width clustering algorithms.

3.2.2.1 K-means algorithm

It is the most used clustering algorithm [200]. Taking a dataset and a K value as input, the algorithm breaks the dataset into K clusters, creating a partition of at least K clusters. The objective of K-means is to minimize the sum of squares errors (objective function) by selecting the optimal K centroids. The centroid is the average point of all the instances in a cluster. Next equation

Algorithm 6 SEP(CVI,Node)

```
C ← cluster in Node
if Node is a leaf node then
    return C
else
    Union ← ∅
    for all Child ∈ Node do
        Union ← Union ∪ SEP(CVI, Child)
    end for
    if CVI({C}) < CVI(Union) then
        return {C}
    else
        return Union
    end if
end if
```

defines the objective function:

$$S = \underset{\mu_1,\mu_2 \ldots \mu_K}{\arg\min} \sum_{i=1}^{K} \frac{1}{|S_i|} \sum_{x \in S_i} (x - \mu_i)^2 \tag{3.8}$$

where the partition has K clusters and S_i refers to the i-th cluster, x is an instance of that cluster and μ_i the centroid of the cluster S_i. Finally, S is the set of centroids defining a particular partition.

For the application of K-means, a dataset and a K value (number of clusters) needs to be given as input. The algorithm consists of several different steps. First of all the K centroids are selected (usually randomly). After that, the instances are grouped with the nearest centroid using the euclidean distance because it allows to calculate the centroid. Then, the centroids are recalculated, and if some centroid differs from the previous set, the process is repeated again as shown in Algorithm 7. This process is repeated until the centroids do not move.

Algorithm 7 K-means algorithm.

```
Read the parameter number of clusters (K);
Initialize the set of centroids (S); {For example randomly.}
repeat
    Assign each instance to the group with the closest centroid;
    Recalculate the positions of the K centroids;
until The centroids no longer move or maximum number of iterations.
```

3.2.2.2 K-medoids algorithm

The K-medoids algorithm is an adaptation of the K-means algorithm and therefore it is also a partitional clustering algorithm. Rather than calculating the mean of the instances in each cluster as a representative item, the centroid, a data-point is chosen: the medoid. One of the main differences between them is that in this case, there is no need for repeated calculation of distances to new centroids, since the K-medoids algorithm can simply look up distances from a distance matrix. The K-medoids algorithm is composed of the steps showed in Algorithm 8.

Algorithm 8 K-medoids algorithm.

1: Read the parameter number of clusters (K);
2: Initialize the set of medoids (S); {For example randomly.}
3: **repeat**
4: Assign each instance to the group with the closest medoid;
5: Recalculate the positions of the K medoids;
6: **until** The medoids no longer move or maximum number of iterations.

Partitioning Around Medoids (PAM) algorithm. The instantiation of the K-medoids algorithm we used is the Partitioning Around Medoids (PAM) algorithm [111] and it consists of two phases: build and swap.

In a first phase, called 'build', an initial partition is obtained by the successive selection of medoids until K medoids have been found. The first medoid is the one for which the sum of the distances to all other objects is as small as possible. That is, this first medoid is the most centrally located in the set of objects. Subsequently, at each step another medoid is selected. This medoid is the one which decreases the objective function as much as possible. Moreover, it works in non-euclidean spaces, since there is not need to calculate the average vector of several instances (the centroid). So other distances could be used such as, edit distance, cosine distance, Manhattan distance, etc. To find the initial medoids, the steps shown in Algorithm 9 are followed.

The second phase of the algorithm (called 'swap') attempts to improve the set of medoids and therefore to also improve the partition. This is done by considering all possible pairs of objects (i, h) where object i is a medoid and object h not. It is determined what effect is obtained on the objective function when a swap is carried out, i.e., when the medoid i is no longer selected as a representative object but object h is. The process is shown in Algorithm 10.

3.2.2.3 Fixed-width algorithm

Fixed-width is a partitional clustering algorithm and it divides the space in hyperspheres of radius w (cluster width). It is a very efficient clustering algorithm because in a single pass through the database it is able to find a partition. The complexity is of $O(K \times N)$ and normally $K << N$.

Algorithm 9 Build phase of PAM algorithm.

```
1: Read the parameter number of clusters (K);
2: Initialize the set of clusters (S) with the most centric case in training set;
3: while K iterations do
4:    Initialize to 0 the maximum contribution (c_max) an instance could introduce.
5:    for each instance (i) in the training set and not in S do
6:       for each instance (j) in the training set and not in S do
7:          Initialize to 0 the contribution of j (c_j);
8:          for each instance (h) in the training set and not in S do
9:             Find the nearest medoid (m) to h and the distance (d(m, h));
10:            if d(m, h) − d(j, h) is positive add it to c_j;
11:         end for
12:         if c_max < c_j then save the c_j in c_max and save the representant j (r);
13:      end for
14:   end for
15:   Add r to S;
16: end while
```

To measure the distance between the cluster and a data instance, the centroid linkage criteria is used.

The algorithm starts with an empty set of clusters and for each data instance it computes the distance between it and each of the centroids of the clusters in the set. The cluster with the shortest distance is selected, and if that distance is less than a given constant w (cluster width) then the instance is assigned to that cluster. Otherwise, a new cluster is created with the instance as its centre. For a more precise explanation see Algorithm 11.

This algorithm has some drawbacks. On the one hand, it is sensitive to the order in which the data is read. Moreover, it is very possible that some cases are not assigned to the nearest centroid because those centroids were not generated yet. This problem can be solved by doing another pass to the dataset and assigning the nearest case to each case. On the other hand, this algorithm is not suitable to find clusters of different sizes because the parameter w fixes the size of all the clusters. Furthermore, as the shape of all the clusters is a hypersphere the algorithm would have the difficulty or impossibility to find many shapes of another kind.

3.2.3 Cluster validation

Once a clustering algorithm has processed a dataset and a partition of the input data is obtained, a relevant question arises: How well does the partition fit the data? This question is important for two reasons. First, an optimal clustering algorithm does not exist. That is to say, different algorithms, or different configurations of the same algorithm, produce different partitions and

Algorithm 10 Swap phase of PAM algorithm.

```
1: Read the set of clusters (S) with K medoids selected in the 'build' phase;
2: while true do
3:    Initialize to infinity the minimum contribution (c_min)
4:    for each medoid (i) in S do
5:       for each instance (h) in the training set and not in S do
6:          Initialize to 0 the contribution of h (c_h);
7:          for each instance (j) in the training set and not in S do
8:             if i is the nearest medoid of j then
9:                if the second nearest medoid of j (i2) is farther than d(h,j)
                  then
10:                  Add d(h,j) − d(i,j) to c_h;
11:               else
12:                  Add d(i2,j) − d(i,j) to c_h;
13:               end if
14:            else
15:               if d(h,j) < d(i,j) add d(h,j) − d(i,j) to c_h;
16:            end if
17:         end for
18:         if c_min > c_h then save the c_h in c_min and save the minimum medoid
            i (mi) and save the minimum instance h (mh);
19:      end for
20:   end for
21:   if c_min < 0 then
22:      Swap mi in S for mh;
23:   else
24:      Break;
25:   end if
26: end while
```

Algorithm 11 Fixed-width algorithm.

```
1: Read the parameter cluster width (w);
2: Initialize the set of clusters (S) with the first instance (x_1);
3: for i = 2 to N do
4:    Find the nearest element (c) of x_i in S (c = arg min_{s∈S} d(x_i, s));
5:    Calculate the distance (d) between x_i and c (d(x_i, c));
6:    if d is less or equal than w then
7:       Assign x_i to c;
8:    else
9:       Add x_i to S;
10:   end if
11: end for
```

none of them have proved to be the best in all situations [149]. Thus, in an effective clustering process we should compute different partitions and select the one that best fits the data. Secondly, many clustering algorithms are not able to determine the number of natural clusters in the data, and therefore they must initially be supplied with this information. Since this information is rarely previously known, the usual approach is to run the algorithm with different values and select the partition that best fits the data. The process of estimating how well a partition fits the structure underlying the data is known as cluster validation [82].

To validate data partitions we must examine the clusters determined by the evaluated partition and measure the compactness of the clusters and their separation. Many authors have proposed different indices, called internal cluster validity indices (see Section 6.2 for some examples) to perform this validation. Unfortunately, no internal Cluster Validity Index (CVI) has proved to be efficient in all conditions. Hence, a method to compare different CVIs, that is, evaluation of cluster validity indices, is necessary.

3.2.3.1 CVI evaluation methodology based on the number of clusters

In this methodology, in order to evaluate a group of CVIs we need a set of datasets, a clustering algorithm and the true number of clusters for each dataset. The algorithm is run over the dataset with a set of m different values for the K parameter, $K = k_1, k_2, ..., k_m$. In this way, a set of m partitions is obtained, $S = P_1, P_2, ..., P_m$, but just one of them has partitioned the data with the correct number of clusters (nc). We refer to this particular partition as the P^{nc} partition:

$$P^{nc} = P_i \mid nc(P^*) = nc(P_i),\ P_i \in S \tag{3.9}$$

where P^* is the correct partition of the analysed dataset, and $nc(P)$ the number of clusters of a partition P.

The CVI is computed for all the partitions in S and the partition obtaining the best value for the evaluated CVI will serve to predict the actual number of clusters. If the function $CVI(P)$ computes the value obtained by the evaluated index over the partition P, it is said that the cluster validity index proposes partition P^{CVI} as the best partition in S:

$$P^{CVI} = \arg\max_{P_i \in S}(CVI(P_i)) \tag{3.10}$$

It is said that the index has predicted that the dataset contains $nc(P^{CVI})$ clusters and consider that it has made a successful guess if $nc(P^{CVI}) = nc(P^*)$. In the evaluation, the more times a CVI guesses the number of clusters of the different datasets the better it is considered to be.

However, the assumption made is not always true. That is, among m partitions, the P^{nc} partition is not necessarily the one that best fits a particular dataset. For example, in a given P^{nc} partition the algorithm can split one natural cluster into two, and join two naturally apart clusters into one, roughly

	C_1	C_2	$\cdots$	C_K	
C'_1	n_{11}	n_{12}	$\cdots$	n_{1K}	$n_{1.}$
C'_2	n_{21}	n_{22}	$\cdots$	n_{2K}	$n_{2.}$
$\vdots$	$\vdots$	$\vdots$		$\vdots$	$\vdots$
$C'_{K'}$	$n_{K'1}$	$n_{K'2}$	$\cdots$	$n_{K'K}$	$n_{K'.}$
	$n_{.1}$	$n_{.2}$	$\cdots$	$n_{.K}$	$n_{..} = N$

Table 3.1: Contigency matrix between partitions P and P'.

speaking the algorithm makes two mistakes. With this methodology, the natural number of clusters is found ($nc(P^{nc}) = nc(P^*)$) but not the natural clusters (P^*). The P^{nc+1} partition may make less mistakes, that is, it may fit the data better than the P^{nc} partition. As a consequence, the evaluation of CVIs taking into account just the number of clusters does not seem to be adequate.

3.2.3.2 CVI evaluation methodology based on partition similarity measures

Our research group proposed an alternative CVI evaluation methodology [80] to overcome the mentioned problem. The underlying idea in the new methodology is to change the definition of a successful guess of a CVI. Instead of considering successful guesses to be the ones proposing the P^{nc} partition as best partition, we consider successful guesses to be the ones proposing the most similar partition to the perfect partition, $\hat{P}$.

$$\hat{P} = \arg\max_{P_i \in S}(sim(P^*, P_i)) \tag{3.11}$$

Therefore, in the new methodology a successful guess will be the one that proposes the $\hat{P}$ partition as the best partition; that is, a successful guess exists if $\hat{P} = P^{CVI}$. The functions that measure the similarity between partitions are known as external cluster validity indices or partition similarity measures. Below, used partition similarity measures are defined divided into two groups being partition P and P' with K and L clusters (C) respectively ($P = C_{1,2}, ..., C_k$ and $P' = C'_1, C'_2, ..., C'_l$).

- **Counting pairs:** This type of measures are based on the estimation of the similarity in a contingency matrix between both partitions. Each row of the matrix corresponds to a cluster of one partition while the other column corresponds to a cluster of the other partition. The value of each cell is the number of cases that the clusters corresponding to the row and column have in common. Table 3.1 shows a contingency matrix between P and P' where n_{ij} value is the number of cases the cluster C_i and C_j have in common, $n_{ij} = |C_i \cap C_j|$.

  However, many similarity measures use a simplified contingency matrix of 2 rows and 2 columns. In this matrix all pair of cases of the database

		P'	
		Same cluster	Different cluster
P	Same cluster	a	b
	Different cluster	c	d

Table 3.2: Contingency matrix between pair of cases of the partitions P and P'.

and their situations in P and in P' are counted. For each pair of cases there are 4 possibilities: (a) both cases are in the same cluster in both partitions (P and P'); (b) both cases belong to the same cluster in P, but not in P'; (c) both cases belong to the same cluster in P', but not in P; and (d) both cases belong to different clusters in both partitions (P and P'). Table 3.2 shows this type of contingency matrix where a, b, c and d are the counts of the four situations mentioned before.

Intuitively two partitions are similar if they have high values of a and d and low values of b and c. These values can be obtained using the values of Table 3.1:

$$a = (1/2)\sum_{i=1}^{K}\sum_{j=1}^{K'} n_{ij}^2 - (N/2) \tag{3.12}$$

$$b = (1/2)\sum_{j=1}^{K'} n_{\cdot j}^2 - (1/2)\sum_{i=1}^{K}\sum_{j=1}^{K'} n_{ij}^2 \tag{3.13}$$

$$c = (1/2)\sum_{i=1}^{K} n_{i\cdot}^2 - (1/2)\sum_{i=1}^{K}\sum_{j=1}^{K'} n_{ij}^2 \tag{3.14}$$

$$d = \frac{N(N+1)}{2} - (1/2)\left[\sum_{i=1}^{K} n_{i\cdot}^2 + \sum_{j=1}^{K'} n_{\cdot j}^2\right] \tag{3.15}$$

Below are described the two partition similarity measures used in this dissertation which are based on counting pairs:

- Rand [167]: $(a+d)/(a+b+c+d)$. It represents the portion of pairs of cases in the same situation in both partitions.
- Jaccard [100]: $a/(a+b+c)$. It is similar to Rand metric but ignores the pairs of cases that have been assigned to different clusters in both partitions (d). Therefore, in a partition of many clusters as the d value can be very high Rand measure can be saturated, however, the Jaccard measure evades this problem.

– Adjusted Rand [95]:

$$\frac{\sum_{i=1}^{K}\sum_{j=1}^{K'}\binom{n_{ij}}{2} - \left[1/\binom{N}{2}\right]\sum_{i=1}^{K}\binom{n_{i.}}{2}\sum_{j=1}^{K'}\binom{n_{.j}}{2}}{(1/2)\left[\sum_{i=1}^{K}\binom{n_{i.}}{2} + \sum_{j=1}^{K'}\binom{n_{.j}}{2}\right] - \left[1/\binom{N}{2}\right]\sum_{i=1}^{K}\binom{n_{i.}}{2}\sum_{j=1}^{K'}\binom{n_{.j}}{2}}$$

It is a adjusted version of the Rand index. It eliminates Rand's problem when high values of d are probable. Thus, it obtains values near zero when two random partitions are compared.

- **Information theory:** This type of measures use concepts of information theory, thus first, some related concepts are explained:

  – Entropy:

$$H(P) = -\sum_{C_i \in P} p(C_i)\log p(C_i) \tag{3.16}$$

  where $p(C_i) = \frac{|C_i|}{N}$.

  – Mutual information:

$$I(P, P') = \sum_{C_i \in P}\sum_{C'_i \in P'} p(C_i, C'_i)\frac{\log p(C_i, C'_i)}{p(C_i)p(C'_i)} \tag{3.17}$$

  The used partition similarity measure is Variation of Information (VI) [136] and it combines the explained concepts:

$$VI(P, P') = H(P) + H(P') - 2I(P, P') \tag{3.18}$$

3.3 Mining Sequences

3.3.1 Sequential Pattern Mining

Sequential Pattern Mining (SPM) discovers a set of attributes shared across time among a large number of examples in a given database. As a consequence, it has emerged as an important data mining task since it is broadly applicable to market and customer analysis, weblog analysis, intrusion detection systems, mining proteins and genes, and in DNA sequence patterns [191]. SPM algorithms address the problem of discovering the existent frequent sub-sequences from a set of sequential dataset [5, 181].

SPM is very similar to Association Rule Learning (ARL) [142] the difference between them is that in the case of SPM the items are linked by time. A simple example could be the case of a market basket problem, if normally people who buy bread and beans also buy pepper, in ARL it does not matter if the bread

is bought first or the beans are first. In the case of the SPM it is important to pay attention to the order.

The algorithms for SPM mainly differ in the way in which candidate sequences are generated and stored, and in the method of testing if the candidates have enough support. Based on these conditions SPM is divided into two different branches [191, 154]. In the first category we can find the apriori based algorithms such as Generalized Sequential Pattern algorithm (GSP) [186] and Sequential PAttern Discovery using Equivalence classes (SPADE) [202], which follow the generate-and-test approach. They generate candidate itemsets and test if they are frequent. This approach is computationally expensive, especially when a large number of patterns exist. Therefore, the algorithms in the second category are based on Frequent Pattern growth (FP growth) algorithm, for example, PrefixPan [22] and they find patterns more efficiently. In this approach first a compact data structure called FP-tree is built from data and second, the frequent sub-sequences are extracted directly from the FP-tree.

However, according to Just [108] the most used algorithms in sequential pattern mining are still the apriori based algorithms. Possibly this happens because in massive datasets, candidate generation is a more divisive task and with more options to make the algorithm parallel than the tree structure.

The one used in this dissertation is SPADE (explained next) because as it is more flexible it finds more sub-sequences.

3.3.1.1 Sequential Pattern Discovery using Equivalence classes

Sequential Pattern Discovery using Equivalence classes (SPADE) [202] is a sequential pattern mining algorithm based on the popular apriori algorithm. It was proposed to mine the market basket sequences. In this case, a sequence is composed by events and each event by items. The events are ordered by time but the items in each event do not have an order. In this dissertation, we used particular sequences with this algorithm, sequences of events with one item, that is, sequence of items.

SPADE sequences represent the most frequently occurring sub-sequences and more hidden patterns than other sequential mining algorithms such as GPS (Generalized Sequential Pattern algorithm), apriori and PrefixPan. The extracted sub-sequences are composed by events ordered in time but not strictly in consecutive positions as in the training sequences. For example, if, after the A item in seq_1 comes Z, and seq_2 needs 2 steps to reach Z, and seq_3 3 steps and so on, SPADE sequences would be able to capture that comes Z after A although this was not so evident in the original sequences. The minimum support determines the number of occurrences of SPADE sequences in the training set.

The SPADE process is explained in the following lines. In the first step of SPADE all events are candidate (1-length sequences) and in a single database scan their support is computed choosing the ones with equal or greater support than the minimum support chosen. In the second step all the 2-length sequences are created (candidates) by using 1-length sequences and in another database

scan the ones with enough support are selected. In the k-th step, k-length sequence candidates are formed by joining $(k-1)$-length sequences and then, another scan is done to ensure their minimum support. The algorithm stops when no supported sequences can be found any more. The algorithm can use a breadth-first or a depth-first search method to find new sequences [202] and follows the steps showed in Algorithm 12.

Algorithm 12 SPADE algorithm.

1: Read the parameter minimum support (minsup);
2: Scan the training set and select the 1-length sequences with al least minsup support (S_1);
3: Candidate generation: using the selected 1-length sequences create all 2-length sequences;
4: Test: scan the training set and select the 2-length sequences with at least minsup support (S_2);
5: $k = 3$;
6: **while** ($S_{k-1} \neq \{\}$) **do**
7: Candidate generation: merge the selected $(k-1)$-length sequences to create k-length sequence candidates.
8: Test: scan the training set and select the k-length sequences with at least minsup support (S_k);
9: $k = k + 1$;
10: **end while**
11: Return the selected: $S = S_1 \cup S_2 \cup ... \cup S_k$;

3.3.2 Modelling sequences

Modelling sequences refers to the process which takes as an input a set of sequences and usually summarizes those sequences into a compact and organized structure to exploit or analyse it. In this dissertation the following ones have been used: Markov model, weighted sequence and suffix tree model.

3.3.2.1 Markov model

Markov model is a stochastic model used to model sequences where it is assumed that future states depend only on the present state (or in higher order Markov models the n previous steps) and not on the sequence of events that preceded it (that is, it assumes the Markov property). Generally, this assumption enables reasoning and computation with the model that would otherwise be intractable.

In this dissertation the simplest Markov model is used: Markov chain. It models the state of a system with a random variable that changes through time. In this context, the Markov property suggests that the distribution for this variable depends only on the distribution of the previous state.

The model is composed by the initial probability vector and the transition matrix which it is calculated by counting transitions into the training sequences.

Multiple Sequence Alignment								
Aligned sequence 1	A	B	C	-	A	-	C	D
Aligned sequence 2	C	B	-	B	A	A	C	D
Aligned sequence 3	C	-	C	-	A	A	-	D

Weighted Sequence								
Page/Positions	1	2	3	4	5	6	7	8
A	1/3	0	0	0	1	1	0	0
B	0	1	0	1	0	0	0	0
C	2/3	0	1	0	0	0	1	0
D	0	0	0	0	0	0	0	1

Table 3.3: Multiple sequence alignment (MSA) to weighted sequence (Wseq).

3.3.2.2 Weighted Sequence (Wseq)

The weighted sequence represents a set of sequences in a special sequence in which each position can emit many elements in the corresponding proportions (the structure is similar to Hidden Markov Model (HMM)). An example of a weighted sequence can be seen in Table 3.3. In this example, the weighted sequence of a set of aligned sequences (see Section 3.1.2.2) is calculated. In this work, the representation of aligned sequences by weighted sequences is shortened as MSA.Wseq.

3.3.2.3 Suffix Tree model

Suffix Trees (ST) store all suffixes of the input sequences in a tree structure, and they are useful to capture patterns in data sequences. The ST over an alphabet is a non empty tree, whose nodes vary in degree of offspring edges between zero (for leaves), and the size of the alphabet at most. Each edge in the tree is labelled by a single symbol of the alphabet. The ST has many virtues: linear construction space, linear construction time and many well designed practical implementations such as effective string operations. This structure is essentially a high order Markov condition.

The term Suffix Tree (ST) refers to the tree structure that models all the suffixes in a single string, while generalized Suffix Tree (gST) makes reference to the tree structure that models all the suffixes of a set of words. In the following lines the construction of the gST (as gST is equal or larger ST, we will refer to them as ST) from a database of sequences is described (Algorithm 13).

Weighted suffix tree. The weighted Suffix Tree (wST) is a ST where each edge stores the frequencies of the modelled suffixed. In this dissertation we will also refer to it as ST. To find easily and quickly all apparitions of the given suffix, in each node of the tree is labelled by the string, generated by walking up the tree from that node to the root.

Algorithm 13 Generalized Suffix Tree (ST) algorithm.

```
1: Create the root node (r) with null value and pointing to nothing.
2: for each sequence (x) in the training set do
3:    for each suffix (s) in x do
4:       Initialize the pointer (p) pointing to the root;
5:       if s does not exist starting from p then
6:          Move p as many nodes as s's initial part makes possible.
7:          Add the elements of the second part of s starting from p;
8:       end if
9:    end for
10: end for
```

Probabilistic Suffix Tree (PST). The PST [169, 134, 190] is a ST where each node of the tree stores a vector of transition probabilities, which gives information of probabilities of a transition from the state of one symbol to the next symbol. As in weighted STs, in this case it is also convenient to label the nodes to easily find all probabilities of a given sequence.

In the PST building phase the depth of the tree can be limited to reduce the size of it. Therefore, several pruning mechanisms have been proposed such as Pmin by Bejerano [29] and MinCount by Yan and Wang [201].

3.4 Unsupervised anomaly detection

Anomaly detection, which is very related to outlier detection [92], is defined as the identification of items, events or observations which do not conform to an expected pattern or other items in a dataset [41]. Typically the anomalous items will translate to some type of problem such as bank fraud, a structural defect, medical problems or errors in a text.

In this dissertation, we used anomaly detection to detect network intrusions and web navigation problems. In these contexts, the interesting objects are often not rare objects, but unexpected bursts in activity. These patterns do not adhere to the common statistical definition of an outlier as a rare object, and many outlier detection methods will fail on such data, unless it has been aggregated appropriately. Instead, a cluster analysis algorithm may be able to detect the micro clusters formed by these patterns.

In the following lines, the models used for anomaly detection are explained: cluster models and probabilistic suffix tree models.

3.4.1 Anomaly detection using cluster partitions

Clustered data can be used to determine anomalous behaviour. The idea is to first perform the clustering over the points in the feature space and assign a score to each point based on its cluster size. This score will be used to determine

the degree of anomalousness of the point. The points with lower scores will be labelled as anomalous.

Although many clustering algorithms could be used, based on the experience of other authors [64, 124], fixed-width clustering algorithm [64], also known as the leader algorithm [185] is well suited to detect anomalies (see Section 3.2.2.3). This algorithm does not accurately fit to databases with clusters of different sizes: it over partitions the largest clusters. Nevertheless, in the unsupervised anomaly detection context we are interested just in the small clusters or no dense groups, so this drawback of the algorithm is not a real problem.

First, the clustering over the points in the feature space is performed, and an anomalousness score is assigned to each cluster based on its size. Then, this score is used to determine the degree of anomalousness of the point or example. Therefore, the points in small clusters are labelled as anomalous because they were assigned a low score.

3.4.2 Anomaly detection using Probabilistic Suffix Trees

Probabilistic Suffix Trees (PST) can be used for anomaly detection [169, 190]. We assume that the PST trained over the dataset represents the normal behaviour and the behaviours deviating from it are considered anomalous. Once the PST is built (see Section 3.3.2.3), the next step for anomaly detection is to compute the similarity of a sequence with the PST, that is, to define a similarity measure.

As a consequence the similarity measure will be a key issue. Two examples of similarity measures are Odds (SIM_o) and Length Normalized (SIM_N). Given a sequence $s = s_1 s_2 ... s_l$ and a PST T, the similarity value of s with respect to T is computed as follows:

- **Odds similarity** (SIM_o): the probability of seeing each element of s in its position, normalized by dividing the probability of seeing them individually. See Equation 3.19.

$$SIM_o(s,T) = \frac{P^T(s_1)P^T(s_2|s_1)...P^T(s_l|s_1...s_{l-1})}{P^T(s_1)P^T(s_2)...P^T(s_l)} \tag{3.19}$$

- **Length normalized similarity** (SIM_N): the probability of seeing each element of s in its position, normalized by the square root of the sequence length. To avoid the value becoming a very small number, as a consequence of multiplying probabilities, its equivalent sum of logarithms in base 10 value is calculated. This similarity explicitly captures the length of the sequence in the similarity calculation as shown in Equation 3.20.

$$SIM_N(s,T) = \frac{1}{l}(\log(P^T(s_1) + \sum_{j=2}^{l} \log(P^T(s_j|s_1..s_l))) \tag{3.20}$$

Therefore, the two main steps to carry out anomaly detection with a PST are: one, to build the PST and two, to define an anomalousness threshold and compute the similarity measure of new sequences to the built PST and decide if it is anomalous or not.

3.5 Topic Modelling

In machine learning and Natural Language Processing (NLP), a topic model is a statistical model for discovering the abstract themes that occur in a collection of documents. It is a text mining technique, which divides a collection of text documents into K topics assigning to each document a distribution vector indicating the probabilities of belonging to each topic (mixtures of topics) and it links words to each topic with their corresponding importance. It can be seen as a fuzzy clustering algorithm but for collections of text documents. Topic models are a suite of algorithms whose aim is to discover the hidden thematic structure in a large number of documents [35]. The simplest topic model is Latent Dirichlet Allocation (LDA). The idea behind LDA is that documents exhibit multiple topics. LDA is a statistical model of document collections that tries to capture this situation and in this model the topic is defined as a distribution over a fixed vocabulary. For instance a "computer science" topic would have words about computer science with high probability.

One easy way to understand how LDA topic modelling tool works, is to imagine working through a research paper with a set of highlighters [35]. As the readers read through the article, they use a different colors for the key words of themes within the paper. When they finish reading the paper they could copy out the words as grouped by the color they assigned them. That list of words would be a topic, and each color represents a different topic (see Figure 3.1).

The example of Figure 3.1 [35] uses an article titled "Seeking Lifes Bare (Genetic) Necessities". This article is about using data analysis to determine the number of genes that an organism needs to survive. The topics extracted from the example are: words about data analysis like "computer" and "prediction" highlighted in blue, words about evolutionary biology, such as "life" and "organism", highlighted in pink; words about genetics, such as "sequenced" and "genes", highlighted in yellow. So that, the main sense of the document is extracted.

The LDA topic modelling algorithms take as input a set of documents and a number of topics to find and as output they provide a list of topics represented by the words belonging to them sorted by importance (topic-word lists) and a matrix with the probability that each document has to belong to each of the topics (document-topic matrix). The LDA algorithm is showed in Algorithm 14.

The main core of the algorithm is the phase when the model is improved. For resampling the topic of the word two different phases are required. The first one is concerned to the topics occurring in the document and the second one is concerned to how many times the word is related with each topic.

The output of the LDA process, the document-topic probability vector can

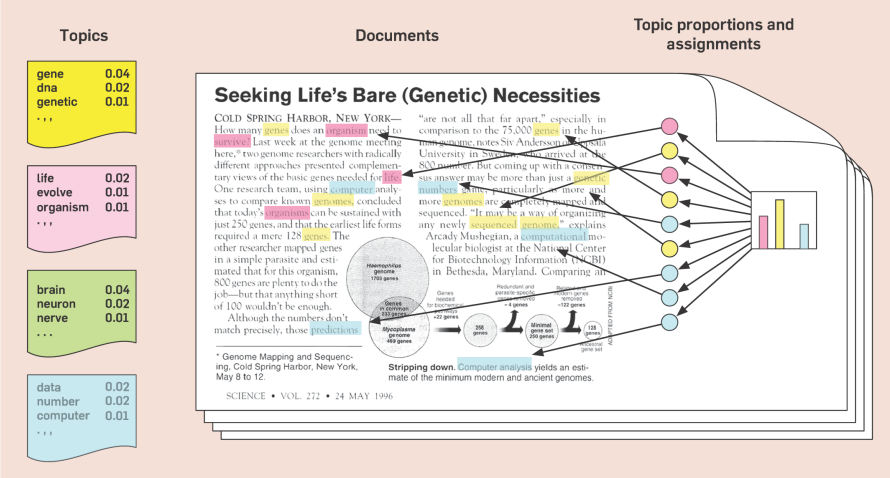

Figure 3.1: Illustration of the intuition behind latent Dirichlet allocation.

Algorithm 14 Latent Dirichlet Allocation (LDA) algorithm.

Read the number of topics to discover (K);
for each document (d) in the collection (c) **do**
  for each word (w) in the document **do**
    Assign to w random distribution over topics;
  end for
end for{Initialise the model. }
Compute topic-words list (the initial random model (m)).
repeat
  for each document (d) in the collection (c) **do**
    for each word (w) in the document **do**
      Randomly choose a topic (t) from the distribution over topics in the model m.
      Randomly choose an other word from the distribution of t over the vocabulary.
      Resample w's topic distribution.
    end for
  end for
until no word's topic distribution is modified; {Improving the model.}
return topic-words list and doc-topic distribution;

be used to deduce the bias of the content of each document. This information combined with a distance metric could be used to compare documents with each other and also to group them by thematic structure.

Part III

Contributions

Chapter 4

Contributions to network security

4.1 Introduction

There has been a huge increase in the use of computer networks. This fact broadens the scope for network attackers and increases the damage these attacks can cause. Network attacks compromise the stability of the network or the security of the information stored in computers connected to it. Therefore, it is very important to build systems that are able to detect attacks before they cause damage. Intrusion Detection Systems (IDS) constitute part of any complete security package and their goal is to detect any action that violates the security policy of a computer system before it causes damage. In this chapter we will focus on a particular type of IDS that works by scanning the network traffic and is able to automatically detect intrusions: network Intrusion Detection System (nIDS).

The detection of network attacks can be done by human analysis or automatically. The detection by human analysis requires memorisation, looking up description libraries or searching sample collections and so it is not effective; it is too time consuming and subjective. As a consequence, in order to successfully confront the problem, the security systems require automated and robust nIDSs. In this sense, a very popular option is the use of data mining techniques, mainly trained on labelled data, to detect attacks.

However we focus on unsupervised anomaly detection methods for detecting non-flood attacks in nIDSs. The characteristics of the attacks change depending on the type of attack and as a consequence, the suitability of a tool to detect them will also change. For example flood attacks generate a lot of traffic and most of them can be successfully detected by scanning the TCP/IP headers of network packets. On the contrary, non-flood attacks only need few connections to generate damage and as a consequence, the headers information is not enough to detect most of them. To detect them another source of information is used:

the transferred information or payload. The payload can be seen as a sequence of ASCII characters or bytes, but generally its features vary depending on the type of network connection and service. As a consequence, most payload based nIDSs we found in bibliography were service-specific and thus, very context dependent [118, 195, 122]. However, new attacks and services appear every day and this makes it important to be able to build a system that works in any environment independently of the type of service or machine.

4.1.1 State of the art

In bibliography we can find three main approaches for nIDSs. The first two are the misuse detection approach [122] and the anomaly detection approach [197]. Due to the problems the previous approaches have, a third one appeared: unsupervised anomaly detection [165].

The misuse detection approach is used in systems such as MADAM/ID [123] where machine learning techniques are used on labelled data. The classifier learns from a set of labelled connections, where there is normal traffic and attacks, and in subsequent uses it recognizes known attacks. These methods have two main problems. On the one hand, it is very difficult to obtain completely labelled network traffic and, on the other hand, these methods are not able to detect new attacks. They need to be revised each time a new type of intrusion appears and this happens every day. These methods can not solve the "zero-day" problem and as a consequence, the new attacks will always succeed in damaging the system.

The second approach, anomaly detection, more precisely, supervised anomaly detection was originally proposed by Denning [54] and a survey can be found in Warrender et al. [197]. This method profiles normal network traffic behaviour and is able to successfully detect attacks when the observed traffic deviates from the modelled behaviour.

When the anomaly detection approach is used, classifiers learn how normal traffic behaves and any anomalous connection is considered to be an attack. As a consequence, if not all the types of normal traffic are modelled, high false positive rates can appear in the system. Moreover, they need purely normal data in order to model normal traffic and it is not usual in real systems to have neither purely normal data nor labelled data. If any attack is left in the hypothetical purely normal data, this attack will be learned as normal traffic and the IDS will never produce an alert related to it.

Due to the problems of the two previous approaches, a third one rose: unsupervised anomaly detection [165]. This option works under the assumption that (1) the volume of normal traffic is much greater than the traffic containing attacks, and, furthermore, (2) the intrusions' behaviour is different from normal data's behaviour. Therefore, it does not need purely normal data and it uses unlabelled data, which is easy to obtain. Under these two assumptions the intrusion detection problem can be confronted in terms of outlier or unsupervised anomaly detection. This approach can be used as a stand-alone system, or, even more effectively, it can be combined with a misuse detection or anomaly

detection process.

Unsupervised anomaly detection methods usually build probabilistic models of the data that will help them decide whether or not the connections are attacks. In this context, clustering methods can be used as a tool for anomaly detection. The connections will be clustered and instances appearing in small clusters, i.e. anomalies, will be considered intrusions.

Although it has been long since the first anomaly detection approaches appeared, it is a successful approach being used in many systems. An example of the use of this methodology is the number of papers mentioning it at the Recent Advances in Intrusion Detection (RAID) conference. Ashfaq et al. [20] for example presented a comparative evaluation of eight lately developed anomaly detectors under portscan attacks from the accuracy, scalability, complexity and detection delay points of views. The authors built two independently collected datasets for the evaluation, both of them including packet header information since all the evaluated systems are based on this information. On the other hand, Dagorn presented in [50] an anomaly-based intrusion detection system for web applications and Rehák et al. [168] presented a way to improve error rate in anomaly detection by collective trust modelling.

However, most of these works only used packet header information to detect intrusions; they did not use the transferred information or payload. The transferred information usually depends on the type of service, and, as a consequence the payload of different network connections can be very different. This is probably why, there are few works where the payload is used to model network traffic and detect the possible intruders.

Unsupervised anomaly detection approaches work under the mentioned assumptions which make it inadequate to detect flood attacks. These types of attacks usually need to send a large number of packets in a short time, and are used for many types of "Denial of Service" (DoS) and "Probe" attacks. Since they are based on the emission of many similar packets they will naturally form large groups and, as a consequence, the clustering process will group them in large clusters. Nevertheless, flood attacks are easy to detect and high detection rates can often be achieved by simpler systems that scan network traffic or analyse headers [148].

Although most of the flood attacks can successfully be detected by scanning the TCP/IP headers of the network packets, this information is not enough to detect most of the non-flood attacks. In these types of attack, such as U2R (User to Root, its goal is to get root or administrative privileges without having them) and R2L (Remote to Local, its goal is to use resources without permission), the intruder only has to send very few packets (often, a single one is sufficient) and, as a consequence, it is nearly impossible for systems to use traffic flow models to detect such anomalies. Notice that R2L and U2R attacks are actually the only ones that allow the intruder to obtain complete control of the attacked system and therefore, they can lead to catastrophic consequences.

In this context some authors proposed [118, 195, 122] the use of another source of information: the transferred information or payload.

The features of the payload vary depending on the type of network connec-

tion and service. As a consequence, most payload based IDSs we can find in bibliography are service-specific [118, 195, 122]. These service-specific methods have the disadvantage that they are very context dependent. That is to say, as they are moved to machines offering different services or as new services appear, the system will need to be rebuilt. In this context it would be important to be able to build a system that is able to work in any environment independently of the type of services or machines.

Krügel et al. [118] presented a work that focuses on R2L attacks and uses service-specific knowledge to increase the detection rate of intrusions. They implemented a prototype that can process HTTP and DNS traffic although they only presented results for DNS.

Wang and Stolfo [195] based their work on profile byte frequency distribution and they computed the standard deviation of the application-level payload flowing to a single host and port during a training phase. The Mahalanobis distance was used during the detection phase and if the distance exceeded a certain threshold the system generated an alarm. This model is also host- and port-specific and also conditioned by the payload length.

In a different context, Waizumi et al. [194] also processed the payload as byte histograms for early worm detection. But they did a different work since, instead of concentrating only on the reduction of false positives, they proposed a payload processing methodology to detect worms in different networks; they only experimented with a worm, Beagle_AV.

Another example of payload processing can be found in the content variables of KDDCup99 [122]. In this case, the payload was used to obtain some information based on the experts' experience. This type of processing is very context dependent and it can only be done for some well known services and protocols. The processing is totally static; it has no learning capability at all. In order for it to be adapted to new situations the experts need to manually analyse the network data and adapt their knowledge to new attacks.

Each type of method has its advantages and disadvantages: signature-based methods have the advantage of generating few false alarms, whereas anomaly-based systems generally produce a lot of false alarms for unusual but authorized activities, which is not recommendable at all. Anyway, the false negatives (attacks not detected) generated by signature-based methods are far more problematic than the false alarms generated with anomaly detectors.

In order to obtain the best nIDS, a combination of some of the approaches is usually the best option. For example, a flood detecting firewall could first filter most flood attacks; a signature-based IDS could then be used to remove the known attacks and unsupervised anomaly detection could finally focus on detecting the unknown attacks.

4.2 Data preprocessing: the gureKDDCup database

As mentioned in Section 4.1 the most of the used techniques to detect attacks were based on signatures, and, as a result, the Unsupervised Anomaly Detection for Intrusion detection system (UADI) approach [64, 23] became necessary. That is, the generation of a system that does not only use signatures, and models the payload part of network packages.

In this context we generated the gureKDDCup database and developed a classifier that detects intrusions or attacks in network based systems. To develop this classifier we used unsupervised classification techniques. The main distinctive feature of this approach is that it uses the payload to detect attacks in network connections. The analysis of the payload to classify the connections is not a deeply analysed field, however, it seems that it is essential to detect attacks such as R2L and U2R.

In the classification process we had to handle a huge number of connections and discover useful patterns among them, what led us to the data mining field. Moreover, we wanted our UADI system to be able to discover patterns or generate the model of network traffic automatically, that is, we wanted the learning process to be automatic, and to make that possible, we used machine learning techniques [21]. In order to generate the system we first needed to generate the database to build this system.

4.2.1 The KDDCup99 database

We generated a database with similar characteristics to KDDCup99 [90] which is a broadly used database in the scientific environment, and was generated taking as starting point the DARPA98 (DARPA Intrusion Detection Data Sets) [140]. The generated database is called gureKDDCup and it has similar features to the ones appearing in KDDCup99, but we added the payload information to it. The following lines explain the steps followed to generate the KDDCup99 database because our aim was to repeat those steps as accurately as possible, to create gureKDDCup, an extension of KDDCup99 including payload, the database we needed to implement the UADI approach.

The source data of the KDDCup99 database was the DARPA98 dataset. To create DARPA98 the Cyber Systems and Technology Group (formerly the DARPA Intrusion Detection Evaluation Group) of the MIT Lincoln Laboratory, under the Defense Advanced Research Projects Agency (DARPA ITO) and the Air Force Research Laboratory (AFRL/SNHS) sponsorship, created a closed military network environment to simulate real traffic, normal traffic, attacks and intrusions. They recorded all network movements using the tcpdump sniffer program. All the simulated traffic was captured in tcpdump files. Thereafter from these files they extracted information such as the outcome of executed commands in the simulation or network packages' information. Therefore, they created a standard corpora for evaluation of computer network intrusion detec-

tion systems and to provide researchers with extensive examples of attacks and background traffic. This dataset is called DARPA98 and was generated in 1998.

Afterwards, from DARPA98 they generated a database in the same format the UCI machine learning repository [70] uses (to use it in the machine learning field), that is, a comma separated data matrix and they called it 'KDD Cup 1999 Data Set' (KDDCup99). This database was used for The Third International Knowledge Discovery and Data Mining Tools Competition (KDD99). The competition task was to build a network intrusion detector, a predictive model capable of distinguishing between 'bad' connections, called intrusions or attacks, and 'good' or normal connections.

To generate the KDDCup99 they had to identify the connections and their values for each attribute. These attributes were computed analysing the files generated by the tcpdump command in the simulated network environment. They computed 41 attributes divided in three groups (See Appendix A):

- **Intrinsic attributes**: the attributes obtained by analysing the headers of the network packages, such as service-name, protocol, etc.
- **Content attributes**: the attributes obtained by analysing the payloads of the network packages, such as, if the root user has got the shell or not, number of logins, etc.
- **Traffic attributes**: the attributes computed analysing the previous connections, such as the number of connections accessing to the same IP address in the last two seconds or between the last 100 connections, etc.

4.2.2 The gureKDDCup database

Our first objective was to extend KDDCup99, that is, to compute all the attributes KDDCup99 has for each connection and add the basic information (IP address, port numbers, etc.) and the payload for each connection, and to label the connection with the same label it has in KDDCup99. As the starting point were the files generated by the tcpdump sniffer program, the sniffed files, contained all the network packages transmitted in the simulated network, so, we could identify a connection and compute its attributes but we could not know the class of each connection, that is, if it belonged to an attack or to a normal connection.

To identify the connection class DARPA98 provides the tcpdump.list file with the class of each connection for each sniffed tcpdump file, i.e., if it is an attack (and type of the attack) or not. This is possible because all the connections, normal traffic and attacks, were generated in a controlled environment. Although not every connection appearing in the tcpdump files appears in the tcpdump.list files they ensure that all the attacks are included.

To create our database, gureKDDCup, we repeated the attributes computation process carried out for KDDCup99. Hence, we used the explanations Lee gives in his Ph.D. thesis [122].

Briefly this is the database generation process: First, we used Bro-IDS (The Bro Network Security Monitor) [151] to extract basic attributes, intrinsic attributes, content attributes and payloads from the tcpdump files, based on writing policies or scripts. Then, we extracted traffic attributes. And finally, to label (normal/attacks) the connections obtained with bro-ids we matched them with the connections in the tcpdump.list file and validated them using KDDCup99.

4.2.3 The gureKDDCup generation process

In this sub-section explanations about the gureKDDCup generation process will be given. To start, the steps followed to create the gureKDDCup database are enumerated:

4.2.3.1 Getting the connections

To ensure that the connections were identified correctly, at first we only extracted the basic information of the connection such as starting time of the connections, source's & destination's IP addresses, and source's & destination's port numbers.

4.2.3.2 Matching the connections

To ensure that the connections were identified correctly, we matched them with the tcpdump.list file and checked the obtained matching degree. Further steps were only executed when the matching degree was acceptable.

Two connections were matched when they had the same starting time and the same source and destination IPs and ports. However to increase the matching rate we relaxed the matching criteria to match the remaining unmatched ones assuming that tcpdump.list had some activity logging confusions. Therefore, we matched those connections interchanging connection's origin and destination and those connections which were established in the same time window ($\pm$3 seconds).

Table 4.1 shows the matching rates between the connections obtained by our preprocessing and the tcpdump.list. The table shows the number of connections matched with each matching criterion (rows) and in each day (column). The values are the sum of matching numbers obtained in each particular weekday during the seven weeks in which the simulated network activity was sniffed. The matching numbers are accumulated values, that is to say, the matching values obtained with the current criteria plus the addition of the previous ones too.

The 86% of the connections in tcpdump.list are matched by the strict comparison of the connections. The 92% of the connections in tcpdump.list were matched relaxing the matching criteria. Therefore, as most of the matchings are strict ones and the matching rates obtained are high, we considered them acceptable.

Besides, each connection must be labelled as attack or normal traffic. This was done by matching each connection with one connection in the tcpdump.list

WEEK[1-7]		Monday	Tuesday	Wednesday	Thursday	Friday	SUM
Match	directly	198,204	200,884	432,786	1,138,572	631,746	2,602,192
	interchanging	219,630	219,760	460,038	1,162,005	665,305	2,726,738
	time-window						
	directly	222,121	221,977	467,679	1,164,812	668,061	2,744,650
	interchanging	225,898	226,500	471,661	1,165,873	669,562	2,759,494
gureKddcup.list		663,823	508,591	756,726	1,770,361	1,149,026	4,848,527
tcpdump.list		343,529	233,998	554,210	1,182,099	700,026	3,013,862

Table 4.1: Matching rates between tcpdump.list and our connections for all Weeks and for days; and the total number of connections in the databases (SUM).

file and taking its class to label it.

4.2.3.3 Computing attributes

The KDDCup99 database has 41 predictor attributes for each connection and the class attribute defining if the type of the connection is attack or normal. These attributes are divided in three groups.

- Intrinsic attributes: compute the attributes for each connection, using the network's packages headers.
- Content attributes: compute the attributes for each connection, using the networks' packages payloads.
- Traffic attributes: compute the attributes for each connection, using the previous connections. Two options were computed: a fixed time-window and a fixed number of connections.

See Appendix A or the internal report "Generation of the database gureKDDCup" [156] for more details on the calculation of these attributes.

4.2.3.4 Adding the payload

Once we defined the 41 attributes that constitute the KDDCup99, we added the payloads to each connection. For each connection three files were stored with the number of connection as file name and differentiated by their extensions which are ".a", ".b" and ".c", i.e., "connectionNumber.a", "connectionNumber.b" and "connectionNumber.c" with the following meanings:

- **'connectionNumber.a'** files accumulate payloads sent by the client side.

- **'connectionNumber.b'** files accumulate payloads sent by the server side.
- **'connectionNumber.c'** files accumulate all the payloads in sequential order according to their timestamp.

4.2.3.5 Validation comparing with KDDCup99

The database gureKDDCup has 4,848,527 connections where 3,218,889 (66.4%) are normal and 1,629,638 (33.6%) are attacks. There are two types of normal connections, the matched ones (35.1% of normals) and the non-matched ones (64.9% of normals). The latter are considered normal because the connections not appearing in tcpdump.list are considered normal. On the other hand, there are two types of attack connections, the flood ones (99.8% of the attacks) and the non-flood ones (0.2% of the attacks).

To compare and validate the obtained values we used a sample of 10% of the KDDCup99 database (KDDCup99-10%), containing: 494,021 connections where 97,278 (20.7%) are normal connections, 394,301 (79.8%) are flood attacks, and, 2,442 (0.5%) are non-flood attacks.

The comparison process was done by comparing the values obtained for each attribute in the gureKDDCup and the KDDCup99-10% databases. KDDCup99-10% is the most used and analysed version of KDDCup99 database by the machine learning community even though, it could not represent the behaviour of all the KDDCup99 attributes. The decision of comparing with KDDCup99-10% was taken mainly because we wanted to ensure that we were finding the values appearing in KDDCup99-10% because our final intention was to create a similar sample.

When the attribute was nominal we computed Pearson's correlation coefficient between the frequencies of nominal values; and when the attribute type was a positive integer or real we calculated the average value of the attribute. However, in most cases the latter takes few values being the zero value predominant. This happens because the usual behaviour is not having alarming activity. Therefore, since the values are too biased to zero the average value was not meaningful (the values were almost zero) and thus, we correlated values of those positive integer values, i.e., frequencies.

The most critical attributes to compare are the content ones because their definitions are not so clear and exact, i.e., they are more subjective than the intrinsic and traffic attributes, thus, intuition and common sense had to be used to define the exact calculation rules.

We finally came out to achieve similar values for all KDDCup99 attributes in both compared databases, either for intrinsic and traffic attributes, and for content attributes obtaining correlation values near 1. In addition to this, we validated it by creating a fixed-width model (see Section 4.3.2.1) from both databases and comparing the attack detection rates, obtaining satisfying results. Therefore, we considered the gureKDDCup database generation process validated and finished. An extensive comparison and the complete process de-

scribed can be found in the report "Generation of the database gureKDDCup" [156].

4.2.4 Database for data mining

Due to the huge size of the original KDDCup99 database (about 5,000,000 connections), most authors performed their experiments using a sample of the original dataset. This sample contains about the 10% of the connections. Similarly, we extracted a stratified sample of similar size from the gureKDDCup. Since our goal is to find the non-flood attacks, and the DARPA98 is overloaded with flood attacks, we filtered all the flood attacks in the dataset. Thus, we worked with a database of 178,810 examples, where 3,937 examples belong to intrusions of non-flood type of 27 different types. We show in Table 4.2 the information about the types of attack and their frequency.

4.2.5 Impact in the scientific community

Since we published the gureKDDCup database in our website (in the year 2008), hundreds (505) of people have downloaded some part of the gureKDDCup and half (247) of them have fulfilled the welcome form used for statistical purposes (checked on 01/12/2015).

According to the provided data, people from many places and universities have been interested with the project (see the Appendix B). Summing up, the forms were filled from 86 entities distributed all around the world.

Moreover, as many people have emailed us asking for the source code used for the database generation. We published the code in the following URL:

`https://github.com/inigoperona/tcpdump2gureKDDCup99`

4.3 Anomaly Detection in Intrusion Detection Systems

Network packages can be divided in two main parts: the package's header information and the transferred information or payload. Since the information of the TCP/IP headers is well-known it can be processed to obtain a tabular representation with intrinsic (I) and traffic (T) variables. On the contrary, payload processing can be difficult because its format in a packet depends on the application and used protocol. Payload data can generally be seen as a sequence of bytes, so it could either be processed to obtain a tabular representation with some type of information or it could be used with a method that is appropriate for sequences.

The aim of this section is to use this payload information as a tool to help detecting attacks in nIDS systems. In this context, it is important for the selected payload processing method to be efficient in detecting attacks and to achieve low false positive rates, but, it also requires having some other characteristics such as:

Attacks	**#con.**	**%**
1. anomaly	9	0.23
2. dict	879	22.33
3. dict_simple	1	0.03
4. eject	11	0.28
5. eject-fail	1	0.03
6. ffb	10	0.25
7. ffb_clear	1	0.03
8. format	6	0.15
9. format_clear	1	0.03
10. format-fail	1	0.03
11. ftp-write	8	0.20
12. guest	50	1.27
13. imap	7	0.18
14. land	35	0.89
15. load_clear	1	0.03
16. loadmodule	8	0.20
17. multihop	9	0.23
18. perl_clear	1	0.03
19. perlmagic	4	0.10
20. phf	5	0.13
21. rootkit	29	0.74
22. spy	2	0.05
23. syslog	4	0.10
24. teardrop	1085	27.56
25. warez	1	0.03
26. warezclient	1749	44.42
27. warezmaster	19	0.48
SUM	3937	1.00

Table 4.2: The number of connections (#con.) and percentage (%) of non-flood attacks in the final database.

$hash_1$	8A	F1	05	AE	87	...	91
	1	2	3	4	5	...	N
$hash_2$	F1	8A	05	AE	90	...	8C
	1	2	3	4	5	...	N

Figure 4.1: Hash representation of two payloads.

1. **To be automatic**: not requiring human intervention.
2. **To be general**: to be service-independent, and, as a consequence, usable in different environments and adaptable to changing situations.
3. **To be computationally efficient**: so that it can operate in real time, in environments with large bandwidth.

It is not easy to build a system with all the required characteristics; it seems that more complex or computationally expensive systems would better model the payload, however the preferred characteristics pull in the other direction.

4.3.1 Payload representation and metrics

Payload data can generally be seen as a sequence of bytes, thus we investigated simple and efficient ways to process the payload.

4.3.1.1 Byte sequence representation (seq)

It does not require any payload processing. However, as it is a sequential representation, suitable distances or similarity metrics have to be used. In this way the use of sequence alignment methods (see Section 3.1.2.3) would be a good options if they were not computationally so expensive and so, incalculable because of the long length of the payloads. Instead, we used a faster distance, Normalized Compression Distance (NCD) using the Gzip compression algorithm (see Section 3.1.2.3).

4.3.1.2 Most Frequent Bytes representation (hash)

Based on the byte frequency idea, we thought that the sequence of the most frequent bytes could in some sense be more significant to represent the payload. Similar connections will probably have similar payload patterns and as a consequence we could also expect the most frequent bytes to be similar. That is why the 1-gram representation of the payload was calculated and the 256 possible values ordered from the most frequent to the least frequent. The information referred to the N most used bytes was used (we tried 3 different values for N: 15, 30 and 50). The byte-ordered payload vectors $hash_1$ and $hash_2$ could be two examples of the representation of two different payloads ($payload_1$ and $payload_2$) as shown in Figure 4.1.

In this way, a new representation of the payload was obtained based on a vector of nominal values, thus we needed to use suitable distances for them: Hamming distance (see Section 3.1.1.2) shortened as hash.ham. Moreover, we proposed a new distance: Ordinal-hamming distance.

- **Ordinal-hamming distance**: the hash representation can be considered a sequence where the position in which each character appears influences the distance. In this context, we defined a distance to compare the representation of two payloads. If two payloads are compared and their corresponding payload character-ordered vectors ($hash_1$ and $hash_2$) are obtained (see Figure 4.1), the distance could be computed with the expression shown in Equations 4.1 and 4.2. We shortened the name to hash.oham.

$$Dist(hash_1, hash_2) = 1 - \frac{\sum_{i=1}^{N} Sim(i)}{N^2} \tag{4.1}$$

where

$$Sim(i) = \begin{cases} 0 & \text{if } hash_1(i) \neq hash_2(j), \\ & \forall j 1 \leq j \leq N, j \neq i \\ N - |i - j| & \text{if } \exists j \mid hash_1(i) = hash_2(j), \\ & 1 \leq j \leq N, j \neq i \end{cases} \tag{4.2}$$

where N is the number of elements of the hash.

4.3.1.3 n-gram or histogram representation (hist)

The well known n-gram analysis is another option that could be used to model payload. To make it computationally efficient a 1-gram representation, where the frequency of each one of the 256 byte values was used. This method was also used by some other authors [195]. In this way, we represented the ASCII characters (0-255 bins) in the x-axis and their frequencies, normalised with the payload's length, in the y-axis. Afterwards, we compacted these histograms using the histogram signatures proposed by Serratosa and Sanfeliu [178] which is explained next.

Sparse long vectors. In many applications it is usual to have histogram representations of some objects, for example, in text mining it is usual to count the apparition of n-grams in different texts and to represent them as an histogram.

The histogram could be very long and difficult to work with. Therefore, if the histogram is sparse, that is, if it has many grams with the value zero it could be interesting to represent it in a more compressed way to save space.

In this direction, Serratosa and Sanfeliu [178] proposed the use of histogram signatures, a loss-less representation of histograms. The histogram signature is

a vector that contains the non-zero bins of the corresponding histogram. Thus, the signature does not lose information.

First, lets define a T-length histogram $H = [h_1, ..., h_T]$ where h_i, $1 \leq i \leq T$ are the discrete values (bins). Let us define the Z-length signature $S^H = [s_1, ..., s_Z]$ of the histogram H, composed by non-empty bins, where each s_k, $1 \leq k \leq Z \leq T$, is composed of a pair of terms: $s_k = \{w_k, m_k\}$. The first term, w_k contains the value of the original histogram index value (i) or the bin number. Thus, if w_k is i, then m_k will be the value h_i (bin height).

To compute the distance between two histogram signatures they need to have the same length. The authors proposed the use of extended signatures: a signature with the minimum number of empty bins added so that, given a pair of signatures to be compared, the number of bins are the same in both of them. Moreover, each bin in both signatures represents the same bin in the histograms.

$$\begin{aligned} A^B &= ES(S(A), S(B)) = [ES_1(A), ..., S_{Z'}(A)] \\ B^A &= ES(S(B), S(A)) = [ES_1(B), ..., S_{Z'}(B)] \end{aligned} \tag{4.3}$$

where A and B are the original T-length histograms, and their corresponding Z-length signatures are $A' = S(A)$ and $B' = S(B)$. $A^B = ES(S(A), S(B))$ and $B^A = ES(S(B), S(A))$ are the Z'-length extended signatures (ES) of the pair $S(A)$ and $S(B)$. The length of the extended signatures is the same and it will be in the range of $1 \leq Z \leq Z' \leq T$.

Once the extended signatures are obtained different distances can be computed using the w and m terms to compare each element of the extended signatures.

In our case we used the well known Manhattan distance shortened as hist.manh and euclidean distance shortened as hist.eucl (see Section 3.1.1.1). Moreover, we proposed a new distance which is able to work with histogram signatures: landmover distance shortened as hist.landm.

- **Landmover distance**: the landmover distance between histograms is the minimum number of unit-bin movements needed to transform one histogram to the other. This distance takes the sky-line of the histograms into account and can be seen as a way of comparing smoothed histograms [190, 110]. Formally described in Equation 4.4.

$$D_{landm}(A^B, B^A) = \sum_{i=1}^{z^{AB}-1} (w_{i+1}^{A'} - w_i^{A'}) \left| \sum_{j=1}^{i} (m_j^{A'} - m_j^{B'}) \right| \tag{4.4}$$

  where A and B are two histograms and $A' = S(A)$ and $B' = S(B)$ are their corresponding signatures which are composed by the non-empty bins of A and B respectively. $A^B = ES(A', B')$ and $B^A = ES(B', A')$ are

the extended signatures of A' and B' which are z^{AB} long. w_i is the i-th element position in the histogram and m_i it is the value of the i-th element.

4.3.2 Payload based modelling

As mentioned before, the aim of this work was to use the payload for built nIDS systems based on unsupervised anomaly detection techniques i.e., data needs to be modelled. We could perform the modelling process based only on the payload or combining it with information from other sources. Different options were analysed in this work.

4.3.2.1 Cluster model

To model the payloads the fixed-width (FW) clustering algorithm (see Section 3.2.2.3) with several distances (see Section 3.1 and 4.3.1) was used. The fixed-width clustering algorithm needs the parameter weight (w) which is determined by analysing the distances to the nearest examples and running the algorithm with some different w values.

The clustering algorithms can be used for anomaly detection. First, clustering over the points in the feature space was performed (over the connections), and a score was assigned to each cluster based on its size. Then we scored the examples in each cluster, and we used this score to determine the degree of anomalousness of the example. We labelled the points in small clusters, i.e., with lower scores as anomalous and we considered them attacks (see Section 3.4.1). Using this ideas we built the different models listed below:

- Modelling headers based intrinsic and traffic variables (IT) defined in KDDCup99

  with euclidean distance (eucl): 1:IT.eucl+FW.

- Modelling payload based specific content (C) variables defined in KDDCup99

  with euclidean distance (eucl): 2:C.eucl+FW.

- Modelling of the payload represented as byte sequences directly (seq)

  with Normalized Compression Distance (NCD): 3:seq.NCD+FW.

- Modelling of the payload represented as the Most Frequent Bytes (hash)

  with Hamming distance (ham): 4:hash.ham+FW.

  with ordinal-Hamming distance (oham): 5:hash.oham+FW.

- Modelling of the payload represented as histogram representation (hist)

  with Manhattan distance (manh): 6:hist.manh+FW.

  with euclidean distance (eucl): 7:hist.eucl+FW.

  with landmover distance (landm): 8:hist.landm+FW.

4.3.2.2 Probabilistic Suffix Tree (PST) model

The PST can work directly with the byte sequences corresponding to the payload. This way, the need of the processing disappeared and we used Probabilistic Suffix Trees (PST) as used in Sun et al. [190] for anomaly detection.

We built a PST based on the payloads of the whole sample. This is a cheap process because as Sun et al. [190] stated, the first levels of the PST are enough to detect the outliers. Based on their experience we only developed the trees to a maximum of 5 layers. Then, we tested all the payloads against the built model, and calculated the length normalized similarity measure (SIM_N) between the suffix tree we trained and the payloads one by one. For a detailed description of similarity measures see Section 3.4.2. In our experiments we used SIM_N because Sun el al. [190] proved in their work that this measure achieves better results than (SIM_o) when PSTs are used to find outliers. The obtained similarity value measures the deviation of the evaluated sequence from the built model. As a consequence, we could use it to determine the level of anomalousness of the corresponding connection; we used it as a anomalousness score.

This model can be of great help to the expert since it will give a clue about the structure of the payload of different types of connections, it could be used to find intrusion signatures.

We added this model to the previous list:

- Modelling of the payload represented as byte sequences directly (seq): 9:seq+PST.simN.

4.3.3 Combination

The score assigned to each of the patterns by different models was used to determine whether the pattern is anomalous or not. We combined the anomalousness scores obtained with different payload based models and the scores obtained from the header information, intrinsic and traffic variables (IT).

The most direct way of combining the different techniques is to combine the different scores obtained for each connection. Unfortunately, since the distribution of the score values assigned by each technique depends on the number and size of the clusters, it varies from one model to another. This fact makes the direct combination of the score values probably not adequate (this option was tried and suspicions were confirmed). As a consequence, some normalization is required to situate the scores of the different classifiers in the same situation. Two options for normalizing the scores in each one of the examined techniques were tried:

1. **Linear 0-1 normalization**: Normalize all scores linearly according to the maximum score value assigned, so that the minimum value is 0 and the maximum is 1. As shown in Equation 4.5.

$$score_{norm} = \frac{score}{score_{max}} \tag{4.5}$$

2. **Rank normalization**: In this normalization all the connections are sorted by their score. The rank for each connection will be its new score. If more than one connection has the same score, the average rank value of the group of connections with equal score is computed and the result assigned as the new rank score for all the connections in the group.

The second option showed to be more appropriate in the results. This probably happens because it is more independent of the specific scores obtained by each of the classifiers: it depends on the position each connection has in the ranking. Independently of the values used to compare the scores, how should we combine the different values? We could probably think of many different combination strategies but for this work we tried three of them:

1. Select the **minimum score** for each connection.
2. Select the **maximum score** for each connection.
3. **Average the score** of the combined techniques for each connection.

The three proposed combining methodologies achieved reasonable results, but in general, better results were obtained by averaging the scores.

4.3.4 Proposed system

The proposed system was a combination of individual models as shown in Figure 4.2. Therefore, when a new network packet was detected, on the one hand we will preprocessed the header part to obtain intrinsic and traffic attributes (IT) and find the nearest cluster to it in the IT cluster partition, and then, assign an anomalousness score to it. On the other hand, we can preprocess the body part of the packet, the payload, for example to extract content attributes (C) and find the nearest cluster to it in the C cluster partition, and then, assign an anomalousness score to it. To use the payload more generically and automatically we can preprocess it to get its histogram representation (hist) and find the nearest cluster to it in the corresponding cluster partition with the corresponding distance, and then, assign an anomalousness score to it. In the same way, we can use the PST model to discover how anomalous the payload is. After getting different anomalousness scores we will combine them to compute the final anomalousness score and decide if it is an attack or not.

Therefore, the last point to decide at this stage is to select which classifiers' results to combine. We could combine all of them or select pairs or trios and combine them. We tried to select the best ones as shown in next section through experiments carried out upon the generated dataset.

4.4 Experimental setup

The aim of this research line was to build a payload based general system as a tool for unsupervised anomaly detection and evaluate its performance. And

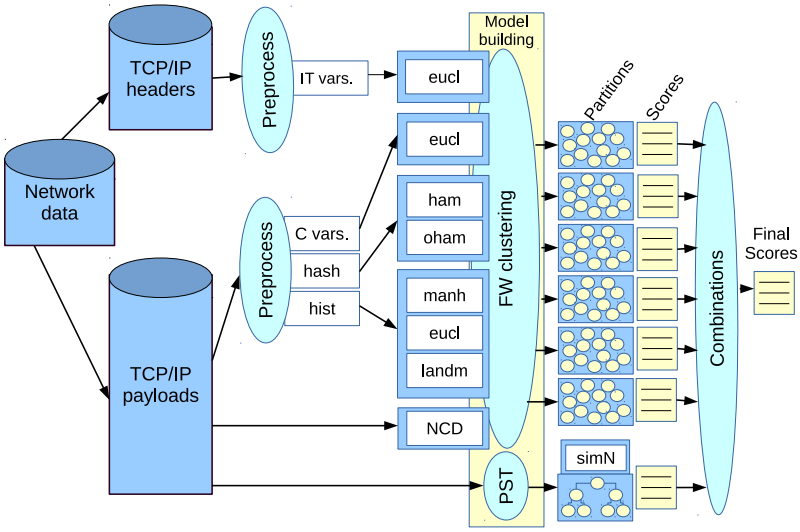

Figure 4.2: The different options of the proposed system.

then, to build an efficient nIDS combining it with the system built based on packet header information (IT). As a first approach we used only the payload information to build classifiers and we also built classifiers using the specific content variables (C). The classifier inferred from C attributes was used as a reference system.

Validation strategy. The system to detect non-flood attacks was evaluated based on the used database. The model was created using all the connections of the database but without using their class in the process (unsupervised learning). This made possible to use their class to evaluate the attack detection accuracy.

Model comparison. The first approach was to use just the payload information to build classifiers and evaluate them separately with no more information than just the payload. For comparison, the payload model extracted from manually created content variables (C) was computed as a reference model. We also generated and included results obtained from TCP/IP header information (IT). Finally we combined payload models with the IT model to asses the systems' overall prediction ability.

We combined the different payload representations and algorithms to finally evaluate the nine different systems mentioned in Sections 4.3.2.1 and 4.3.2.2.

Validation metrics. The experiments were carried out with the whole database, that is, normal traffic data plus data from 27 different types of attacks.

But, in order to present the results obtained in our experiments we chose to examine the detection rates of each type of attack separately instead of examining a global result for all of them. We noticed that in many previous works where KDDCup99 was used for experiments, a single global value was used to measure the performance of a particular detection system. Due to the high number of connections belonging to certain types of attacks (mostly Smurf and Neptune) the results are heavily biased to the detection level the system obtained for those particular attack types. Even so we removed all the flood attacks from our dataset, differences in the number of attacks of each type exist as shown in Table 4.2. Our results were always analysed for each attack type, and, therefore, all attack types had the same weight when presenting the final results. Otherwise the overall results would practically ignore the attack types other than warezclient, teardrop and dict.

The evaluation was done by analysing the obtained ROC curves and the Area Under ROC Curve (AUC) values [67]. To compute the ROC of just a single attack type, the examples belonging to other attack types were ignored.

4.5 Results and Analysis

4.5.1 Payload based modelling

The first analysis was devoted to evaluate the systems built using only the payload. The results are shown in Table 4.3. On the one hand we built models from raw payloads, without preprocessing (3 and 9); on the other hand we built them by preprocessing payloads to hash-like nominal vectors (4 and 5) or histogram-like numeric vectors (6, 7 and 8). The rows in Table 4.3 belong to different attack types whereas the columns belong to different payload modelling techniques. The second column shows AUC values achieved using packet header information (1:IT.eucl+FW). Then, the third column shows AUC values achieved using attributes extracted from the packet content information with ad-hoc signatures (2:C.eucl+FW) which we used as a reference (REF) system. And finally, the rest of columns in Table 4.3 show the results achieved with each general and automatic methodology based only in payload information.

In general terms, the first conclusion that can be drawn from Table 4.3 is that although no context knowledge is used and simple processing is performed, the seven options for modelling payload are, to a certain extent, able to differentiate between normal traffic and intrusions; the average AUC values are 0.77 for the NCD based system, 0.73 for the hash based systems, the histogram based systems have more variability from 0.85 to 0.93, and the PST based system has 0.95; the best value. Furthermore, the model built using header information, 1:IT.eucl+FW, achieved an AUC value of 0.80, and the model built using attributes extracted with ad-hoc signatures from the payload (2:C.eucl+FW) obtained 0.84, which are good values but they can be improved as the previous AUC values indicate.

The histogram based options 6, 7 and 8, achieved better AUC values than the

Attacks	**1:IT.eucl+FW**	**2:C.eucl+FW (REF)**	**3:seq.NCD+FW**	**4:hash.ham+FW**	**5:hash.oham+FW**	**6:hist.manh+FW**	**7:hist.eucl+FW**	**8:hist.landm+FW**	**9:seq+PST.simN**
1. anomaly	0.76	1.00	0.88	0.35	0.74	0.98	0.64	1.00	1.00
2. dict	0.76	0.99	0.82	0.64	0.83	0.93	0.92	1.00	1.00
3. dict_simple	0.65	1.00	0.81	0.69	0.83	0.98	0.99	1.00	0.99
4. eject	0.76	0.98	0.82	0.80	0.80	0.99	0.35	0.98	1.00
5. eject-fail	0.99	0.80	0.48	0.99	0.58	0.95	0.95	0.97	0.98
6. ffb	0.80	0.85	0.88	0.72	0.93	0.95	0.65	0.96	0.92
7. ffb_clear	0.65	1.00	0.81	0.71	0.67	0.99	0.97	0.96	1.00
8. format	0.79	0.75	0.93	0.81	0.95	0.93	0.82	0.96	0.89
9. format_clear	0.52	1.00	0.81	0.88	0.83	0.99	0.21	0.95	1.00
10. format-fail	0.98	1.00	0.81	0.80	0.67	0.99	0.75	0.95	1.00
11. ftp-write	0.88	0.73	0.88	0.76	0.56	0.80	0.87	0.90	0.71
12. guest	0.77	1.00	0.85	0.81	0.83	0.93	0.92	0.89	0.99
13. imap	0.90	0.80	0.70	0.97	0.68	0.97	0.86	0.89	0.99
14. land	0.92	0.80	0.48	0.99	0.58	0.95	0.95	0.88	0.98
15. load_clear	0.65	1.00	0.81	0.12	0.14	0.93	0.99	0.87	0.99
16. loadmodule	0.70	0.84	0.71	0.69	0.68	0.97	0.77	0.87	0.93
17. multihop	0.72	0.74	0.78	0.63	0.71	0.93	0.94	0.84	0.89
18. perl_clear	0.95	1.00	0.81	0.52	0.87	0.99	0.99	0.81	0.99
19. perlmagic	0.66	1.00	0.83	0.86	0.86	0.99	0.99	0.77	0.99
20. phf	0.90	0.50	0.71	0.99	0.72	0.98	0.98	0.77	0.93
21. rootkit	0.88	0.81	0.77	0.86	0.77	0.94	0.96	0.76	0.94
22. spy	0.71	0.80	0.81	0.66	0.52	0.99	1.00	0.76	1.00
23. syslog	0.82	0.80	0.48	0.97	0.58	1.00	0.90	0.75	0.99
24. teardrop	0.96	0.65	0.48	0.76	0.58	0.89	0.48	0.69	0.88
25. warez	0.82	0.31	1.00	0.12	0.85	0.93	0.97	0.69	0.98
26. warezclient	0.81	0.68	0.86	0.86	0.86	0.94	0.92	0.69	0.77
27. warezmaster	0.94	0.75	0.87	0.96	0.88	0.82	0.87	0.69	0.95
min	0.52	0.31	0.48	0.12	0.14	0.80	0.21	0.69	0.71
Average	0.80	0.84	0.77	0.74	0.72	0.93	0.92	0.85	0.95

Table 4.3: AUC values achieved for all the attack types in different payload only based classifiers: 2:C.eucl+FW (REF, reference), 3:seq.NCD+FW, 4:hash.ham+FW, 5:hash.oham+FW, 6:hist.manh+FW, 7:hist.eucl+FW, 8:hist-.landm+FW and 9:seq+PST.simN. Likewise, detection rates for the header based classifier (1:IT.eucl+FW) are presented. The best system for each attack type appears coloured.

hash based ones (4 and 5) and also better than 3:seq.NCD+FW. In the same way, histogram based models behaved even better than the model built with data obtained using context specific knowledge for processing payload (2:C.eucl-+FW) or the packet's header information (1:IT.eucl+FW). Taking a closer look at the histogram based models, specifically to the row with minimum AUC value within them, we can see that mainly two of the options are most interesting: 6:hist.manh+FW and 8:hist.landm+FW. They achieved AUC values of 0.69 or greater for all the types of attacks. The payload-based options that behave worse (2, 3, 4, 5 and 7 options) have minimum values smaller than 0.5 what means that for some type of attack they achieved worse results than a random classifier would. However they achieved higher detection rates than the system based on packet header information (1:IT.eucl+FW).

In the last column we modelled the raw payload in a general way, 9:seq-+PST.simN, and it differentiated between normal traffic and intrusions better than many of the other payload based options: 2, 3, 4, 5, 7 and 8. This option works similar to 6:hist.manh+FW and moreover, the PST based system (9) does not need payload preprocessing. It achieved higher detection rates that the system based on packet header information (1:IT.eucl+FW). It achieved AUC values of 0.71 or bigger for all the types of attacks and, for example, the header based 1:IT.eucl+FW option achieved a minimum AUC of 0.52 which means that at least for one of the attacks it behaves similarly to a random classifier.

In order to take the conclusions a bit further we made a pair-wise comparison of the nine different techniques. In each comparison we compared the AUC values obtained for each attack type and counted the number of times each technique obtained greater values than the other. The cells in Table 4.4 show the number of attacks (out of 27) the technique of the corresponding row detected better than the technique of the corresponding column. The last columns show the number of times the corresponding technique behaved better than any of the rest (SUM) and the raking (Ranking).

In order to interpret the results it is important to notice that if a certain technique was always better than another one, the value in the cell corresponding to their comparison would be 27 (values of around 13-14 would mean similar behaviour), and if that happened for all nine comparisons that can be made for each classifier, the total value (column Total) would be 216. Values in Table 4.4 vary, so, there are differences in the number of attacks each classifier is able to detect better. In this way, if we observe the general behaviour of different options it seems that two of them, 9:seq.PST and 6:hist.manh+FW, behave better than the rest. This means that, we were able to detect more non-flood attacks using general and automatic options than by analysing header information (1:IT.eucl-+FW) and by analysing the signatures on payloads (2:C.eucl+FW).

4.5.2 Combination of different approaches

Analysing the AUC values obtained in Table 4.3, it can be observed, that, each one of the built classifiers specializes better in detecting some types of attacks.

Systems	1:IT.eucl+FW	2:C.eucl+FW (REF)	3:seq.NCD+FW	4:hash.ham+FW	5:hash.oham+FW	6:hist.manh+FW	7:hist.eucl+FW	8:hist.landm+FW	9:seq+PST.simN	SUM	Ranking
1:IT.eucl+FW	0	12	10	13	16	4	10	12	4	81	6
2:C.eucl+FW (REF)	15	0	18	16	21	10	14	11	6	111	3
3:seq.NCD+FW	16	9	0	16	12	3	8	6	4	74	8
4:hash.ham+FW	13	11	10	0	14	4	11	10	6	79	7
5:hash.oham+FW	11	6	13	10	0	2	7	6	3	58	9
6:hist.manh+FW	23	17	23	22	25	0	14	20	9	153	2
7:hist.eucl+FW	17	13	17	16	20	8	0	14	5	110	4
8:hist.landm+FW	15	13	20	17	21	7	13	0	4	110	5
9:seq+PST.simN	23	17	23	21	24	15	17	21	0	161	1

Table 4.4: Pair-wise comparison of the different techniques. Summary of the number of times each technique behaves better than the other. For the systems behaving better than the reference one, the ranking appears coloured.

As an example, we observed that teardrop attack is best detected by 1:IT.eucl-+FW approach; warez attack by 3:seq.NCD+FW approach; and phf attack by 4:hash.ham+FW approach. This led us to think that a good option could be to combine the knowledge acquired as explained in Section 4.3.3. We combined them by averaging scores improving the results considerably.

The option 1:IT.eucl+FW was always maintained in the combination, that is, 1:IT.eucl+FW was not ignored in any of the combinations because our aim was to provide our system with both header and payload information. For the rest, we tried all possible combinations between the presented techniques using the same payload representation because once the payloads are preprocessed it is efficient to apply different distances to them. The effect of the combination was in most cases positive but generally it became more positive when more than two techniques were combined. Hence, in Table 4.5, for each of the attacks (rows) and each of the relevant combinations (columns) the AUC values are shown.

Generating those combined systems, we integrated the knowledge of the payload based techniques and scores obtained from modelling Intrinsic and Traffic variables (1:IT.eucl+FW), to improve the original results. First, as a reference option, we combined the header based scores with the scores obtained from the Content variables (1+2). And then we combined the header based model with the proposed generic payload based models.

As it can be observed, all the combinations contributed to increase the overall

AUC values. Above all, the minimum AUC value increased in all cases and this means that the system using this nIDS will be better protected against any type of attack.

Average AUC values show that the general behaviour of any of the combinations improves the behaviour of the classifiers obtained with each independent option. We could consider as the best options the last two combinations, 1+6+8 and 1+9, because they obtained the best average and minimum AUC values.

Results obtained with 6:hist.manh+FW and 8:hist.landm+FW, were the best two options of histogram processing (see Table 4.3). The combinations contributed to increase the overall AUC values in both cases. The combination of 1+6+8 is the one that achieved the best results with an average AUC of 0.96 and a minimum AUC value of 0.89.

The combination of 1:IT.eucl+FW and 9:seq+PST.simN also achieved very good results with an average AUC of 0.95. Furthermore, it was able to detect any type of attack because the minimum AUC, taking into account all types of attacks, was 0.85. Whereas the reference 1+2 option achieved an average AUC of 0.91 and a minimum value of 0.83.

The same type of pair-wise comparison presented in Table 4.4 was performed for all the evaluated combined techniques. The global results for the seven combined techniques evaluated in Table 4.5, are presented in Table 4.6. The maximum value each system can obtain against another system is 27 attacks; and the maximum SUM value each system can obtain is 27 attacks x (7 systems - 1 itself) = 162. The results confirmed the conclusions drawn from Table 4.5.

It is important to analyse the computational efficiency of the best combination (the system was programmed in Java and run in an Pentium 4 with 2.8 GHz and 1 GB RAM). In this sense, we should take into account two phases: the modelling phase and the exploitation phase. The modelling of IT information (1) in both cases (1+6+8 and 1+9) is the same. Therefore, we focused on comparing the payload modelling part: histogram based model (6+8) and PST based model (9). In this phase, the bottleneck was the PST; it took 12 minutes to build the PST (9) based on 178,810 examples and 9 minutes to divide the same sample in clusters based on Fixed-Width algorithm (6+8).

The second phase is the exploitation phase, where new connections need to be tested with the models in order to obtain the corresponding score or anomalousness degree. In this phase the PST clearly outperforms the fixed-width; more than 8,000 connections can be tested per second whereas in the case of the clusters only 200 connections per second are tested. This means that once the models are built, the best performance would be achieved with the PST model.

4.6 Summary

Modelling of Internet traffic is motivated by the increase in the use of computer networks. This fact broadens the scope for network attackers and increases the damage these attacks can cause. Therefore, it is very important to build systems

Attacks	**1+2 (REF)**	**1+3+4**	**1+3+5**	**1+4+5**	**1+3+4+5**	**1+6+8**	**1+9**
1. anomaly	1.00	0.72	0.90	0.68	0.77	0.96	0.96
2. dict	0.95	0.83	0.93	0.84	0.89	0.95	0.96
3. dict_simple	1.00	0.81	0.91	0.81	0.88	0.96	0.94
4. eject	0.98	0.90	0.90	0.88	0.92	0.96	0.96
5. eject-fail	1.00	0.94	0.80	0.98	0.90	0.96	1.00
6. ffb	0.93	0.90	0.97	0.92	0.94	0.95	0.95
7. ffb_clear	1.00	0.82	0.84	0.73	0.84	0.95	0.95
8. format	0.89	0.95	0.98	0.96	0.98	0.93	0.95
9. format_clear	1.00	0.84	0.85	0.84	0.90	0.92	0.90
10. format-fail	1.00	0.98	0.95	0.93	0.96	1.00	1.00
11. ftp-write	0.87	0.94	0.88	0.82	0.89	0.89	0.85
12. guest	0.94	0.92	0.94	0.92	0.95	0.96	0.97
13. imap	0.92	0.94	0.85	0.96	0.92	0.94	0.98
14. land	0.94	0.90	0.76	0.95	0.87	0.94	0.99
15. load_clear	1.00	0.54	0.56	0.19	0.43	0.92	0.94
16. loadmodule	0.87	0.77	0.79	0.77	0.80	0.89	0.92
17. multihop	0.83	0.76	0.80	0.73	0.77	0.98	0.91
18. perl_clear	1.00	0.87	0.98	0.91	0.93	0.94	0.99
19. perlmagic	1.00	0.88	0.91	0.90	0.93	0.96	0.94
20. phf	0.88	0.96	0.89	0.97	0.95	0.93	0.97
21. rootkit	0.87	0.93	0.90	0.94	0.93	0.96	0.97
22. spy	0.86	0.80	0.77	0.68	0.77	0.98	0.95
23. syslog	0.85	0.87	0.70	0.92	0.84	0.94	0.98
24. teardrop	0.85	0.83	0.78	0.88	0.81	0.91	0.98
25. warez	0.98	0.68	0.98	0.64	0.82	0.96	0.97
26. warezclient	0.83	0.95	0.96	0.96	0.97	0.95	0.88
27. warezmaster	0.96	0.98	0.98	0.98	0.99	0.95	0.97
min	0.83	0.54	0.56	0.19	0.43	0.89	0.85
Average	0.91	0.86	0.87	0.84	0.87	0.96	0.95

Table 4.5: AUC values achieved for all the attack types in different combined classifiers, where the single classifiers are: 1:IT.eucl+FW 2:C.eucl+FW, 3:-seq.NCD+FW, 4:hash.ham+FW, 5:hash.oham+FW, 6:hist.manh+FW, 7:hist-.eucl+FW, 8:hist.landm+FW and 9:seq+PST.simN. And REF indicates the reference system which uses header and signature based content information. The best systems for each attack type appear coloured.

Systems	1+2 (REF)	1+3+4	1+3+5	1+4+5	1+3+4+5	1+6+8	1+9	SUM	Ranking
1+2 (REF)	0	19	18	18	18	11	10	94	3
1+3+4	8	0	10	9	10	4	3	44	7
1+3+5	7	15	0	18	8	6	6	60	5
1+4+5	9	13	7	0	7	7	3	46	6
1+3+4+5	8	16	16	20	0	4	4	68	4
1+6+8	13	21	21	20	22	0	7	104	2
1+9	15	23	21	23	22	15	0	119	1

Table 4.6: Pair-wise comparison of the different combined techniques. The values indicate the number of times each technique behaves better than the other. The systems which are better than the reference one, the ranking appears coloured.

that are able to detect attacks before they cause damage. In this direction we focused on a system that works by scanning the network traffic and is able to automatically detect intrusions: network Intrusion Detection System (nIDS). In this context, we focused on non-flood attacks in which the intruder sends very few packets and it takes the complete control of the attacked system. To detect this attacks it is necessary to analyse the content part of the network packages, called payload. Since the payload's format depends on the service it is used for, most works are directed to find non-flood attacks proposing service dependant payload based methods. We proposed a nIDS system that is able to work in any environment by using unsupervised anomaly detection techniques and the payload transferred in network connections. Our experiments showed that general payload processing and analysis can also be effective to detect non-flood attacks.

The first step in this process was to build and publish the gureKDDCup database with normal and attack connections. This database includes 41 attributes of the KDDCup99 plus the connections payloads. There were not any public database that contained payload, thus, with gureKDDCup database which included payloads we facilitated the research in general payload processing for intrusion detection.

Afterwards, we proposed an automatic and context independent payload-based system. We proposed different generic payload processing approaches, on the one hand, three ways to represent the payload (sequence, hash and histogram), and on the other hand, two payload modelling approaches for anomaly detection (based on fixed-width clustering algorithm and on Probabilistic Suffix Tree (PST)).

The results showed that each technique was more appropriate than others to detect specific types of attacks, we have decided to combine them. The

combined results showed that it is possible to integrate the knowledge of different payload-based techniques and the packet-header-based technique and improve the previous results. All the tested combinations contributed to increase the overall AUC values. Above all, the minimum AUC value increased in all cases and this means that the system using this nIDS system will be better protected against any type of attack.

With regard to the methods we showed that fixed-width clustering algorithm is able to detect attacks by scoring connections as outliers when located in small clusters. To build the payload based systems three general representations of the payload with their proper distances were presented: the payload as a byte sequence, as a histogram of byte frequencies (1-gram) and as a hash of most frequent bytes.

Regarding the hash payload representation it did not work as well as we expected. It seems that the order of the most frequent bytes is not as significant as we hypothesized. Furthermore, analysing the distances used between hashes the hamming distance worked better than the ordinal-hamming. This suggests us that the frequent byte position does not have a valuable meaning for intrusion detection because when we added more order information we got worse results. The hash representation with hamming distance is the best one detecting some types of attacks: 5:eject-fail, 14:land, 20:phf, and 7:warezmaster. Therefore, for some attacks the most frequent bytes have importance.

Regarding the use of the NCD distance (with gzip) to compare pairs of byte sequences, it is not as effective as we expected. The idea was that if two payloads shared many sub-sequences of bytes the compression algorithm would compress a lot and so, the two sequences would be similar. We used the gzip compression algorithm and it did not get good compression rates. However, the NCD distance got the best results for the 25:warez attack.

Regarding the histogram payload representation, it works well, thus, it seems that the frequency of each byte is meaningful. Analysing the used distances, it seems that the best are Manhattan and landmover. The euclidean was the worst, so, it seems that this application prefers 1-norm distances like Manhattan and landmover than 2-norm ones like euclidean. However, all the three distances were the best performing for some types of attacks.

Hence, although all the payload processing options were able to detect attacks in the range of payload based ad-hoc content variables (C), the histogram representation outperforms. And even more when combined with the information of intrinsic and traffic variables (IT). The best option was the combination of 1:IT.eucl+FW, 6:hist.manh+FW and 8:hist.landm+FW, it achieved an average AUC of 0.96 whereas the average AUC achieved with the best option for ad-hoc processing, 1:IT.eucl+FW and 2:C.eucl+FW was 0.91. Furthermore, it was able to detect any type of attacks because the minimum AUC, taking into account all types of attacks, was 0.89.

On the other hand we showed that the tree like structure of PST is able to detect attacks. Furthermore, without any payload processing and besides, the structure generated with the PST could be used by security experts to identify

new signatures of known attacks.

Moreover PST discovered outliers in shallow, that is to say, the limiting the depth of the PST tree showed that shallow depths were enough to detect many types of non-flood attacks. Concretely, we used a maximum depth of 5, that is, the model remembered length 5 suffixes. Summing up, short sub-sequences of the payload are enough to detect the attack.

The experiments carried out showed that PST was able to detect attacks more efficiently than the payload based ad-hoc content variables in both cases: when used on its own and when it is combined with the information of intrinsic and traffic variables. 1:IT.eucl+FW, and 9:seq+PST.simN combined system achieved an average AUC of 0.95 whereas the average AUC achieved with the best option for ad-hoc processing, 1:IT.eucl+FW and 2:C.eucl+FW combination was 0.93. Furthermore, the 9:IT+PST.simN option was able to detect any type of attack because the minimum AUC, taking into account all types of attacks, was 0.85. Moreover, this option was computationally cheap since there is not need of preprocessing for the payload and the PST is a computationally efficient model.

Chapter 5

Contributions to web mining

5.1 Introduction

During the last decades, the information on the web has increased drastically and this often makes the amount of information intractable for users. Moreover, users access web data constantly in various circumstances: locating useful information, sharing knowledge, performing online transactions, etc. In this web navigation they either use web search engines or online indexes providing navigation routes. However, since, larger quantities of data do not provide added value for web visitors, there is a need of easier access to the required information and adaptation to their preferences or needs. That is, web personalization becomes essential.

Web personalization [163] can be defined as the set of actions that are useful to adapt dynamically the presentation, the navigation schema and the contents of the Web, based on the preferences, abilities or requirements of the user. Brusilovsky et al. [39] described many research projects focused on this area. One of these web personalization applications is based on user modelling and aims suggesting links, mostly in the context of e-Commerce [39] and e-learning [72].

In order to the web being adaptable, it is compulsory to first obtain user models that allow the characterization of their different needs. In other words, adaptations of the web environments to specific users require a previous phase of generating user profiles. A user profile is a description that contains the most important facts about the user. The motivation of building user profiles is to gather the different preferences, interests, backgrounds, goals and skills of the users. The content of a user profile and how the information it contains is acquired varies from one application domain to another.

User profiles can either be explicitly provided by the user or learned using some intelligent techniques. Although the first option might seem easier, in

general, users are not willing to fill in long forms to provide information about them, they do not always tell the truth, and, moreover, they might not know how to express their interests. Thus, a better way for obtaining information about users is observing their actions [174]. User profiling implies inferring unobservable information about users from their observable actions. In adaptive systems, the user profile is used to provide the adaptation effect, that is to behave differently for different users. Individual user profiles can inform about user interests, knowledge, background, goals, behaviour, interaction preferences or context. Whereas, group profiles aim at combining individual profiles to model a group and they are vital in contexts where the adaptations have to be done based on the experience of previous groups of users.

Among the different techniques to collect and process the information for web personalization we can distinguish three approaches: (1) manual decision rule approach that uses manually defined profiles; (2) content-based filtering approach that uses the history of each concrete user to build a profile; and (3) collaborative filtering approach that uses the experience acquired from other users with similar characteristics.

Although the three approaches mentioned can be complementary, the first two can be excessively biased to the concrete user, too subjective and very often static. The third approach does not have the mentioned drawbacks, however, it has a scalability problem. That is, it might be difficult to perform the process in a reasonable time interval when the amount of data increases. One of the ways to make a system scalable based on a collaborative filtering approach is to first use data mining techniques to generate profiles in a batch process or off-line, and then use the outcome to accelerate the on-line personalization process [39].

The use of data mining techniques in order to find patterns in the context of web applications is called web mining [116, 187]. Depending on the nature of the data mined, three categories can be found: web usage mining [141], web structure mining [55] and web content mining.

The objective of this chapter is contributing to web personalization [163] based mostly on web usage information and focusing specially on:

- **Problem detection:** we tried to detect sets of users with specific profiles and the problematic ones.
- **Link suggestion:** we generated links to be suggested to the users from early stages of the navigation, as a list of links they would probably use, and could help them reaching the aim faster.

Therefore, our global aim is to personalize the web browsing experience of the users making it easier and more comfortable. User modelling is a very important phase of this process. To model different users' navigation we used the collaborative filtering approach, that is, we estimated the users' needs based on the experience acquired from other users with similar characteristics. Hence, this research area belongs to the area web mining, more precisely to web usage mining [141] which refers to the application of data mining techniques to the web usage data.

Statistical analysis over web usage data can be performed extracting information about the most frequently accessed pages, average time spent viewing pages, average lengths of paths, etc. However, the knowledge extracted from this type of analysis can be very limited. On the contrary, machine learning techniques are in general able to extract more general and abstract information from data which is more valuable knowledge. In addition, machine learning techniques model data in an automatic way. In our context, for collaborative filtering, one of the most used techniques is the clustering.

One of the most widely explored applications of web access patterns' prediction has been webpage prefetching [42, 4, 130]. Common characteristics of many of the systems are the use of clustering and/or markovian models to predict the next link to be accessed. Likewise, the URL access order is of high interest, which can be considered using either sequence alignment methods [44], or sequential pattern mining algorithms [202] when generating user profiles, for example.

Link suggestion could be considered an extension of prefetching where not only the next link is predicted. We built web mining systems to contribute to web personalization. Our model focuses on web navigation patterns and attempts to interpret them in order to build personalized navigation. The idea is that we can successfully model the users' web navigation patterns and learn/understand their information interests in the sites they regularly visit. The novelty of the approach lies on modelling through the combination of diverse algorithmic approaches and inclusion of data of diverse nature in the same structure: the users' navigational patterns enriched by the site's contents, using techniques of web usage mining and web content mining respectively. In this modelling process, the proposed method estimates the degree to which every page on a site can be of use of the user needs. Based on this model, the proposed method generates suggestions about the site's navigational paths that the user should follow in order to find the desired information instantly.

Summing up, we built a system based on the collaborative filtering approach that takes as input the minimum information stored in a webserver: server log files stored in Common Log Format (CLF) [193]. This fact makes the system generic and applicable to any web environment. Afterwards it blends the unsupervised and supervised machine learning techniques and pattern mining techniques to group and build user profiles, i.e., the profiles are constructed without disturbing the user. These profiles will be used in the future to adapt the navigation of new users, for example, providing them with possible links that they will probably use in the future or detecting specific navigation problems they are having. In this process, we tried to answer the following research questions: what type of knowledge can be extracted using just webserver logs? can just the information in webserver logs be enough for web personalization? if not, how could the process be improved? are the selected machine learning techniques and their combination appropriate for modelling navigation sequences?

5.1.1 State of the art

5.1.1.1 Problem detection

Although predefined user models can be used in the design processes, models built based on in-use information where behaviours emerge from the obtained data, will provide more realistic information about the usage characteristics of the site. Weblogs are the most simple in-use techniques and are thus applicable to a wider range of users.

As previously mentioned, a part of this work focuses on automatically discovering special or problematic navigation profiles to be used to improve the site and, hence, the browsing experience of the user, making the website easier to use and more convenient. We consider that web mining techniques are appropriate to achieve our objectives and that the combination of the knowledge extracted from web usage information with knowledge extracted from the content and structure of the site makes it possible to detect not only human problems but also content or structure-related problems.

In this sense, to the best of our knowledge, nobody has previously proposed such work for automatically detecting navigation problems using webserver log files.

5.1.1.2 Navigation mining

Although web access patterns' prediction has many applications such as improving web cache performance, personalizing the browsing experience of the users, suggesting related pages, etc. the most widely explored application in the web research community has been webpage prefetching. Many years ago, in 1997, Kroeger et al. [117] already showed that the performance improvement achieved by combining caching and prefetching can be twice that of caching alone. Since that date, many approaches have been published that, taking as starting point a click sequence, concentrate on predicting the next page that will be accessed in order to prefetch it before the user requests it, and, as a consequence, reduce web access latency [42, 43, 164, 4, 130, 131]. Common characteristics of those approaches are in general the use of clustering and/or Markov models to predict the next link to be accessed but, unfortunately, the presented results differ from one to another and they are difficult to compare.

Some different approaches can be found in the works presented by Makkar et al. [130] and Bhawsar et al. [33]. The former proposes a prefecthing strategy that combines the user log information and the structure of the website where Petri nets are used for extracting the structure. On the other hand the later proposes the use of Page Rank (PR) to compute the importance of each page for the cases where the obtained Markov model's prediction ability is not accurate.

Undoubtedly, the order the users access the pages is important to differentiate the usage patterns. In fact, this is probably the reason of the popularity of sequence analysis methods to predict web access. Another interesting approach would be to take into account the access order in the clustering process, for

example the work by Chordia and Adhiya [44] proposes an efficient implementation of sequence alignment methods for grouping web access sequences that combine global and local alignment techniques. They use open access NASA [144] webserver logs for experiments.

Likewise, there are many other web usage mining approaches as Facca and Lanzi showed showed in their survey [66]. Those approaches are mainly based on association rules, sequential patterns and clustering. Furthermore, effective mining of sequences can be seen in many works [143, 107, 153, 106, 119]. These mined sequences are the basis for producing association rules or they can be stored in suitable tree structures or Markov chains in order to represent navigation patterns.

Moreover, techniques of string modelling have been applied to web access sequences [204, 171]. Using string modelling, web user navigations are represented as strings, thus permitting the efficient use of suitable string comparison algorithms, such as, longest common subsequence [26] or sequence alignment [87].

More complete methods have been proposed [76, 128] that use content based knowledge (usually employing N-gram techniques) by means of clustering the webpages by content, in order to enhance the quality of the association rules.

Dimopoulos et al. [58] presented a method for modelling users' navigation history and webpage content with weighted suffix trees, a tree like data structure that stores weighted patterns [97]; an approach that has been extended in [132] with semantic information.

5.1.2 Application environments

5.1.2.1 General websites: EPA, SDSC, NASA

First of all to acquire experience, we started extracting useful knowledge from public domain server logs. We selected three databases from "The Internet Traffic Archive" [51]. To be precise, we selected the databases SDSC-HTTP [175, 176], a supercomputing centers' webpage; EPA-HTTP [62, 63], an environment related webpage; and NASA-HTTP [144, 145], related to space issues. Since these three databases belong to different application contexts, the obtained outcomes were applicable to many environments.

The main reason to select these databases was the similarity they have on the contained information with the standardized text file format, i.e. Common Log Format [193], used by webservers to accumulate traces of requests arrived to their machines. Common Log Format file is the minimum information saved on a webserver. Therefore, the data mining process designed for these environments will be applicable to the information collected in any other webserver.

These logs were generated in the year 1995 and the only public opportunity to obtain the corresponding web structure of that time was the Internet Archive's wayback machine [109]. In this webpage we could try to obtain websites stored approximately in the same period of time the logs were taken. Unfortunately, it does not have records as old as 1995 and therefore, for this work we only used

the information accumulated in the server log files.

Now, we will briefly describe these three databases:

- **The SDSC-HTTP database** [175, 176] (San Diego Supercomputer Center): it has logs of the server located at the San Diego Supercomputer Center in San Diego, California. The logs were collected from 00:00:00 PDT until 23:59:41 PDT on Tuesday, August 22 1995, a total of 24 hours.

- **The EPA-HTTP database** [62, 63] (United States Environmental Protection Agency): it has logs of the server located at Research Triangle Park, NC. The logs were collected from 23:53:25 EDT on Tuesday, August 29 1995 until 23:53:07 on Wednesday, August 30 1995, a total of 24 hours.

- **The NASA-HTTP database** [144, 145] (National Aeronautics and Space Administration): it has logs of the server located at NASA Kennedy Space Center in Florida. The logs where collected from 00:00:00 July 1, 1995 until 23:59:59 August 31, 1995, a total of 62 days.

By mining these logs we took the first steps in predicting the navigation and in detecting navigational problems. Afterwards, this experience was enhanced in two real applications: a tourism website (included in this dissertation) and a website aimed at visually impaired people (not included in this dissertation).

5.1.2.2 Tourism website: Bidasoa Turismo website (BTw)

The use of web mining techniques could help the web tourism personalization [74, 75, 188], making the navigation for travellers more satisfactory [65, 150, 94] and providing useful information to service providers [2] in order to make more interesting and suitable offers to the travellers and facilitating the understanding between each other.

The tourism data we mined comes from the website which promotes the Bidasoa-Txingudi bay, which is located at the western tip of the Pyrenees, straddling two countries (France and Spain) and linking the Basque provinces of Gipuzkoa and Lapurdi. The Bidasoa River has had the effect of socially and culturally linking the three towns surrounding the bay (Hendaye, Hondarribia and Irun). The area offers the opportunity of a wide range of tourism activities and the bidasoa turismo website[1] (BTw) includes all sorts of practical tourist information on the area: thematic tourism, professional tourism, gourmet tourism, agenda, recommendations, etc. A screen shot of the website is shown in Figure 5.1. The usage data for BTw was provided by the staff of the Destination Management Organization (DMO). The information contained on this database consists in the webserver logs of requests stored in Common Log Format [193]. Apart from the usage data, we also acquired and used the content information of the website; i.e. the text appearing in the website. In

[1] `www.bidasoaturismo.com`

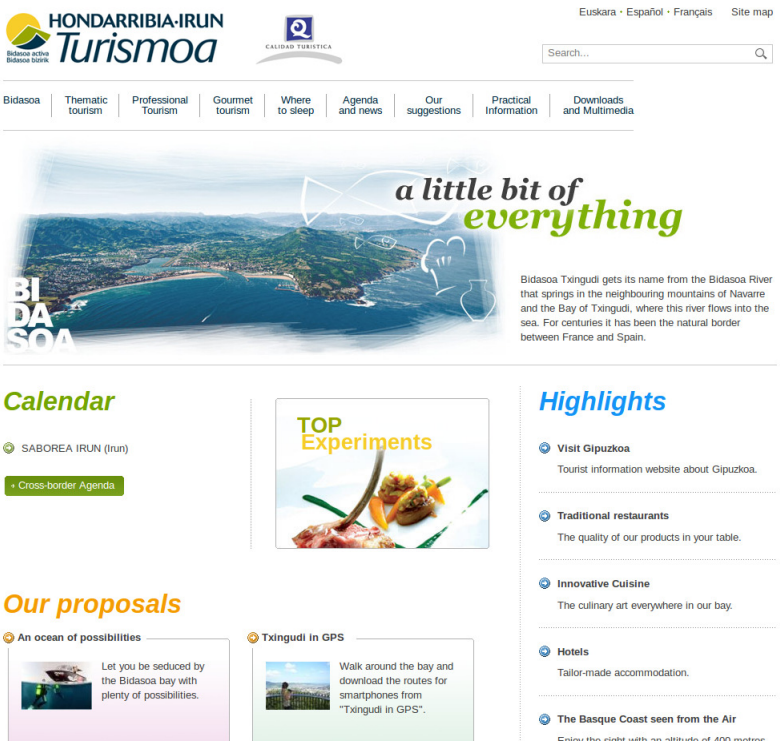

Figure 5.1: Appearance of the home page of BTw website.

addition, we used a third source of information: the language used when accessing the website. The webpage can be accessed in four different languages (Basque, Spanish, French and English), which provides information about the origin of the users.

5.2 Data preprocessing

Webservers in all cases perform thorough logging of HTTP requests by appending them in data files. Thus raw data files are produced with the requests linked to the user click mixed with all other browser's requests, thereby hiddenly capturing the interests of users. Webserver log files store all HTTP requests received by a webserver in Common Log Format [193]. These files store data about the request in different fields. The fields we used are the following ones:

- **IP address field**: client side IP address.
- **Timestamp field**: when the request was recorded.

- **Method field**: HTTP method used. GET, HEAD and POST are the most used ones.
- **URL field**: requested URL.
- **Status field**: informing about the success or failure when processing the request.
- **Bytes field**: number of bytes sent to the client side.
- **User agent information**: the identification string of the software used by the user. It is not always available.
- **Reference field**: the URL from which the request was made. It is not always available.

A sample of the used webserver log is shown in Figure 5.2. In these raw data files, among the mixed requests, hidden and implicit information is captured.

```
65.52.108.999 - - [08/Jul/2012:15:38:01 +0200] "GET /index.php?option=com_agenda&
view=day&id=20120301&Itemid=280&lang=en&el_mcal_month=8&el_mcal_year=2012&ch=1
HTTP/1.1" 200 30233 "-" "Mozilla/5.0 (compatible; bingbot/2.0; +http://www.bing.com/
bingbot.htm)"

146.0.74.999 - - [08/Jul/2012:15:38:05 +0200] "GET /administrator/index.php?
option=com_login HTTP/1.1" 200 4761 "-" "Mozilla/5.0 (Windows; U; Windows NT 5.1;
en-US; rv:1.9.1.3) Gecko/20090824 Firefox/3.5.3 GTB5"

66.249.72.999 - - [08/Jul/2012:15:39:19 +0200] "GET /index.php?option=com_agenda&
view=day&id=20120215&Itemid=280&lang=es HTTP/1.1" 200 30049 "-" "Mozilla/5.0
(compatible; Googlebot/2.1; +http://www.google.com/bot.html)"
```

Figure 5.2: Webserver log sample in Common Log Format (with masked IP address for privacy issues).

5.2.1 General preprocessing steps

The logs are not easy to handle in their raw form in machine learning algorithms. As a consequence, we have preprocessed the log files to obtain information from different users and sessions. It is noteworthy saying that this part of the proposed system is probably the one that is more tightly coupled to the concrete database. For the rest of the phases we propose general procedures that could be applied with little changes to any web environment.

Therefore, it is quite important to first clean-up raw datasets of noisy requests and then carry out "quantization" of the dataset into user sessions and webpage topics in order to facilitate the extraction of patterns and model them. This data preparation phase is usually quite laborious, affected in many cases by the type of targeted website (i.e. news portals, blogs, and sites that constantly change structure and topics as opposed to websites with static structure and content categories). All steps of the data preparation phase are highlighted in the following Algorithm 15. Then depending on the specific application, all of them or some of them can be applied.

Algorithm 15 Data Preparation.

Input:
(1) Webserver log files,
(2) Webpages HTML sources.
Output:
Annotated (content/role) web access sequences (sessions).

Step 1: Filter out erroneous requests.
Step 2: Identify webpages (URLs) and user session.
Step 3: Filter out non-user requests (web crawlers, web admin requests, etc.).
Step 4: Filter out requests to unpopular webpages (URL string with scarcely any requests).
Step 5: Filter out requests to not interesting webpages (webpages that expire fast).
Step 6: Filter out short and long access sessions.
Step 7: Annotate webpages of access sequences with topics based on their content.
Step 8: Annotate webpages of access sequences with roles based on user behaviour.

Filtering out erroneous requests and identifying user sessions (Steps 1 and 2): The first step is to remove erroneous requests, those that have an erroneous status code (client error (4xx) and server error (5xx)). Therefore, the process takes into account successfully processed requests (2xx); as well as those HTTP methods (GET, POST) that have, in fact, requested complete pages; and those requests which were directly related to the user activity according to their extension, such as ".html".

To identify the user session we rely on IP addresses and the time interval as a specific IP is making requests [127]. After analysing the distribution of the elapsed time in each click (consecutive requests by the same IP) we fixed a threshold of inactivity [88] between requests to consider a new session. Within each session, consecutive identical requests were joined assuming that they are generally created automatically by the browser (or by the user pressing reload on dynamic pages).

Filtering out requests not useful for user suggestions (Steps 3 to 6): First of all, we formatted the URL strings carrying out the following steps:

- We analysed the URL string, if the language in which the page was requested could be identified we identified it and made the URL language transparent.
- In general, as modern portals have URLs with a lot of posting parameters it is quite important to order and eliminate non-influential parameters so

that every URL is mapped to the same URL string.

The next steps consisted on identifying the nature of each click. Afterwards, the section of the website to which the URL belongs was extracted in order to focus on website parts which are appropriate for link prediction and, as a consequence, we removed the clicks for which modelling was not interesting due to their particular nature. The following list summarizes this process:

- Identify administrators' activity and navigations in private zones by analysing the URL string; and then remove navigation in private zones of the website which are not accessible for everyone (Step 3).
- Identify the web robots activity by analysing the user agent string, i.e., activity generated by crawlers, spiders and search engine indexers; and then remove web robot activity (Step 3).
- The most popular URLs, those appearing at least 10 times in the logs, are identified; and then infrequent URLs removed to ensure that the modelled URLs are well-formed (Step 4).
- Remove URLs that are changing content extremely frequently (hourly or daily, such as *agenda* or *headline news* pages) (Step 5).
- Remove search result pages, because they are not fixed pages (Step 5).

Finally, from the obtained sessions, those with a minimum activity level were selected; three or more clicks, and the longest sequences were removed (those with a sequence length outside the 98% percentile), with the assumption that long sequences are outliers and might be caused by some type of undetected web robots, such as, crawlers, spiders or web indexers.

Identifying webpage topics (Step 7): The introduction of content information in the data mining process can enhance the acquired knowledge [11]. The extraction of content information was done by using Natural Language Processing (NLP) techniques to automatically extract the semantic structure of a website. Specifically we used topic modelling techniques (see Section 3.5) and an alternative URL string based option.

- **Topic Modelling.** Topic modelling techniques take as input a set of documents and yield as output a list of topics represented by the words belonging to them and a vector expressing the probability that each text item or document has of belonging to each of the topics. The document-topic probability vector could be used to compare documents with each other and also to group them by thematic structure. Several statistical models have been developed to automatically extract the topical structure of large document collections [34]. For this work, we used the Stanford

Topic Modelling Toolbox (STMT) [2] which uses Latent Dirichlet Allocation (LDA) [36] as its topic model.

In order to obtain this thematic structure of the website, first the HTML files of the whole website need to be downloaded recursively. Secondly we applied an HTML parser to obtain the content of each page and filtered the menus of the webpages so that we were only working with the real content. We gave a dataset as input to STMT containing all the content information about each URL. After running STMT we obtained a list of topics in which each topic is represented by the keywords related to it (topic-keyword list) and a matrix containing the probabilities each topic has in each of the URLs in the database (document-topic probability matrix).

- **URL generalization.** Sometimes the URL addresses can be meaningful and useful to abstract the topical structure of a site. Therefore, we proposed a generalization of the URL string and represented URLs in a higher level of abstraction.

  The generalization step consists on erasing a fraction of the URL segments (parts separated by '/' appearing in the URL) from the right side of the path to diminish their specificity. For each one of the visited URLs, we obtained the length of the generalized URL based on next expression:

$$length^{gen}(URL) = max\{MinNSegment, \alpha * NSegments\} \quad (5.1)$$

  where $NSegments$ represents the number of segments separated by '/' appearing in the URL and α and $MinNSegment$ are parameters that can be varied depending on the structure of the site. $MinNSegment$ represents the minimum number of segments starting from the root an URL can have after the generalization step whereas α represents the fraction of the URL that will be kept in the generalized version. This generalization process will allow to work with the general structure of the site avoiding the confusion that too specific zones could generate.

Identifying webpages roles in the user navigation (Step 8): It is possible to estimate the level of interestingness of a webpage for a user by the amount of time that the user spends on it [58]. Based on this intuition, we discretized the elapsed time of each request, and deduced the importance of the accessed page for the user. After some trials, we observed that the following discretization boundaries for characterizing pages are fairly representative, but different websites could have its discretization boundaries.

- **Unimportant (U)**: it is not important for the user because few seconds were spent on it (0 sec. - 5 sec.). The pages which are mandatory to visit, such as the homepage, could be of this type.

[2] http://www-nlp.stanford.edu/software/tmt/tmt-0.4/

- **Hub (H)**: it routes the user to another page, i.e., it has functioned as an index page for the user. Users in those pages locate the link to the desired page and relatively short time is spent on them (5 sec. - 20 sec.).
- **Content (C)**: users find these pages interesting and they stay longer (20 sec - 10 min.).

In Figure 5.3, we show a sample of the created database after the data preprocessing phase which contains session identifier with its corresponding sequence of URL identifiers and the role (Unimportant (U), Hub (H) and Content (C)) each URL-request has complied.

```
470001,000011U,000323C,000328H,000329U,000302H,000197C,000331H,000059H,000332H
500001,000334H,000011U,000331U,000334C
580001,000074H,000420C,000074H,000424H,000074U,000427H
590001,000074H,000011U,000089U,000344C
600001,000089U,000453H,000089U,000454H,000089C,000011U,000295H,000127U
630001,000328U,000464H,000412H,000467H,000412U,000464U,000127U,000295U,000011U
640001,000011U,000386U,000011U,000386H,000011U,000471C,000011U,000474H,000480H
650001,000487U,000070U,000382U,000070U,000382H,000491H,000382U,000152H,000011U
660001,000504H,000500U,000059H,000196U,000059C
710001,000011U,000127C,000007H
```

Figure 5.3: A sample of the sequences after the preprocessing phase.

5.2.2 Logs from the Internet Traffic Archive

In this case, since we downloaded public logs, the data acquisition phase was not been part of our work. Therefore, we started from the data preprocessing step. The preprocessing of the server logs of SDSC, EPA and NASA was more or less as explained in Section 5.2.1 with the following specificities:

- **Step 1: Filter out erroneous requests.** Equally processed.
- **Step 2: Identify webpages and user session.** The expire time of each session was fixed to 30 minutes.
- **Step 3: Filter out non-user requests.** Equally processed.
- **Step 4: Filter out requests to unpopular webpages.** All webpages were taken into account.
- **Step 5: Filter out requests to not interesting webpages.** All webpages were taken into account.
- **Step 6: Filter out short and long access sessions.** In order to discard sessions with very few requests, as a first approach we selected the users with the highest number of requests. We selected for each database the 90% of the users with the highest number of requests (we removed the short ones). Likewise we removed the outliers in sequence length, the ones out of 90% percentile (we removed the long ones).

	SDSC	**EPA**	**NASA**
#requests (in logs)	28,338	47,748	3,461,612
#sessions (in DB)	439	984	31,853
#clicks (in DB)	5,796	20,882	347,731

Table 5.1: Number of requests in logs and after preprocessing (in the database (DB)) with how many sessions and clicks we ended.

- **Step 7: Annotate webpages with topics based on their content.** We annotated webpages of access sequences with a more abstract URL representation based on the URL string generalization.
- **Step 8: Annotate users' clicks with roles based on their behaviour.** Not done.

After this process the size of the interesting requests was reduced to the numbers shown in Table 5.1.

5.2.3 Logs from Bidasoa Turismo

The log files used in this work contained the requests recorded by Bidasoa Turismo web portal[3] (in short BTw) server over a period of 10 months: from 09/01/2012 to 19/11/2012. They contained 3,835,086 requests and the main data cleansing and data preparation steps applied are described in Section 5.2.1.

We first grouped the correctly processed user clicks into sessions and removed the sequences containing a single click (50% of the total requests were removed). The following steps differ form experiment to experiment, therefore the exact steps are explained at the beginning of each experiment done with BTw.

5.2.3.1 Content of BTw

The topic identification process was done using the STMT topic modelling tool provided by Stanford University. After several fine-tuning experiments to test different values and analysing the coherence of the keywords related to each of the topics proposed by the STMT tool, we determined that the website has 10 main topics or abstract themes. Some examples of the list of words assigned to each topic could be: "Topic A (Nature)" for mountain, river, beach, bay, etc., "Topic B (Historical monuments)" for church, chapel, castle, history, and "Topic C (Cuisine)" for cuisine, restaurant, cider-house, etc. Once the STMT tool extracted the different topics from the URL collection, we named them manually, inferring a title for the topic based on the keywords grouped under each topic. The titles we selected for the 10 topics proposed by STMT are: Nature, Historical monuments (HistMon), Cuisine, Accommodation Camping (AccCamp), Accommodation Hotel (AccHotel), Events, Culture, Sea and Sports (Sea&Spo), Sports and Tradition.

[3] http://www.bidasoaturismo.com/

5.3 Problem detection

The aim of this section is to propose the use of web mining techniques combined with information from other sources to adapt the web to users. As a preliminary work in that direction we tried to detect user profiles based just on the information of the server log files. Therefore, in this section, we tried to detect sets of users with special profiles based on some preprocessed web navigation logs we found in the Internet Traffic Archive [51].

5.3.1 Data preprocessing: Feature extraction

Once the requests and sessions were filtered we aggregated the information of each user session in a set of meaningful features that we expected to be representative of the users' characteristics. Thus we obtained a vector representation of each user session.

- **User activity level (#clicks)**: total number of clicks.
- **Average speed (time/page)**: average time the user spent in each page.
- **Temporary distribution of the users' activity**: number of clicks at night (**#N**) from 00:00 to 08:00; number of clicks in the morning (**#M**) from 08:00 to 16:00); and number of clicks in the evening (**#E**) from 16:00 to 24:00.
- **How deep the user navigates (navigation depth)**: average depth of the URLs the user has visited.
- **How persistent/disperse the user is**: in this direction four different attributes were computed:
  - Number of different pages visited (**U.#pagdiff**).
  - From the most visited URL by each user (U.top1): number of clicks on it (**U.#top1**).
  - From the five most visited URLs by each user (U.top1, U.top2, ..., U.top5 URLs): sum of the number of clicks on them (**U.#top1-5**).
  - The five most visited URLs (**U.top1to5**).

We expected these features to be useful to identify different groups of users that share navigation patterns such as similar difficulties, same interests, etc.

5.3.2 Pattern discovery: Clustering of data vectors

We used K-means clustering algorithm [103] to divide the users in groups with similar characteristics. But before this, we normalized the database.

The used database is composed by 10 attributes with different range of values: (1) #clicks, (2) time/page, (3) #N, (4) #M, (5) #E, (6) navigation depth,

(7) U.#pagdiff, (8) U.#top1, (9) U.#top1to5 and (10) U.top1to5. Therefore, before applying K-means to the training set we must normalize them, because we want all attributes to have the same effect in the classification and as a consequence, all the values to be in the same range. We have used a two-step normalization procedure:

1. **Normalize the numerical attributes**: first, we divided the number of requests by the total number of requests of the user, and then, we standardized all the numerical attributes x by the statistical normalization.

$$x'_i = \frac{x_i - \mu}{\sigma}, \forall i \in x \tag{5.2}$$

   where μ is the mean of the attribute x and σ is the standard deviation.

2. **Normalize the five most frequent URLs**: to normalize the five most frequent URLs the user has accessed in the navigation, first, we defined the similarity between two URLs as the number of common fields they have. We considered a field, the character sequence appearing between two slashes ('/') and counted them from left to right until they did not match. That is, we calculated the path's steps two URLs have in common in the web structure tree. However, when we computed the similarity between two instances that is, the similarity between two sets of 5 URLs, the similarity between each possible pair of URLs (at most 5×5 pairs were calculated) and we selected the maximum one for each URL in the set. This method can be used only when the URL string segments give an idea about the topic of real webpages hanging from them, otherwise similarity between the text of the pages needs to be computed.

   Next, the similarity needed to be converted into distance. We did the conversion based on the average of all the path lengths of the database, being the length the number of fields. That is, we divided this average by the maximum similarity computed in the previous paragraph to get the distance value.

Thanks to the normalization, since the user sessions are represented as normalized numerical vectors, euclidean distance could be to carry out the clustering.

5.3.3 Results and Analysis

We had not prior knowledge of the structure of the data in any of the databases, but we fixed the parameter number of clusters (K) to 10 after analysing partitions of different numbers of clusters. And we tried to evaluate how it adapted to data. Therefore, the application of the clustering algorithm with the described user representation, distances and parameters lead to a set of 10 clusters for each database.

Tables 5.2, 5.3 and 5.4 show the results belonging to the most meaningful clusters for each of the three databases used for the experiments. The rows represent the different features used to characterize the groups of users whereas the columns represent the values these features have for the selected clusters. We include in the first column (All) the values these features take in the whole database. This will be useful to decide how different the behaviour of the users in a cluster is from the average behaviour of the database. In order to make results more visual we shaded the cells corresponding to a concrete feature and cluster if they differ notoriously from the average behaviour. If the value in the cell is considerably larger (1.5 times greater or equal) than the average value for the whole database we used light grey to shade the cell. In contrast, we used dark grey when the value in the cell is considerably smaller (2 times less or equal) than the average value for the whole database.

We summarized each cluster for their further analysis with the attributes presented in Tables 5.2, 5.3 and 5.4. When the attributes were numeric we reduced them by average value; in the case of 'Distribution' we indicated the most frequent access period in the cluster; and in 'Dispersion' the portion of sequence covered by some particular URLs. In the following list we define the values appearing in the table that were not defined before. Being U_i the i-th user session, C_j the j-th cluster of sessions and n_j the number of sessions in C_j. The function $length(U_i)$ returns the given session's length. The function $pagdiff(U_i)$ returns the number of different pages visited by the given user session. The function $top1(U_i)$ returns the most used URL in the given session or set of sessions. The function $top1to5(U_i)$ returns the five most used URLs in the given session or set of sessions. The function $nmatch(U_i|urlset)$ indicates given a set of URLs ($urlset$) the number of clicks done in those URLs in the session U_i.

1. **%#pagdiff**: Average of the percentages of different (not repeated) pages accessed by the sessions of the cluster.

$$P_{C_j}(\#pagdiff) = \frac{1}{n_j} \sum_{U_i \in C_j} \frac{pagdiff(U_i)}{length(U_i)} \tag{5.3}$$

2. **%U.#top1**: Average of the percentage of clicks done by each session of the cluster in each session's most used URL.

$$P_{C_j}(U.\#top1) = \frac{1}{n_j} \sum_{U_i \in C_j} \frac{nmatch(U_i|top1(U_i))}{length(U_i)} \tag{5.4}$$

3. **%U.#top1to5**: Average of the percentage of clicks done by each session of the cluster in each session's five most used URLs.

$$P_{C_j}(U.\#top1to5) = \frac{1}{n_j} \sum_{U_i \in C_j} \frac{nmatch(U_i|top1to5(U_i))}{length(U_i)} \tag{5.5}$$

4. **%C.#top1**: Average of the percentage of clicks done by each session of the cluster in the cluster's most used URL.

$$P_{C_j}(C.\#top1) = \frac{1}{n_j} \sum_{Ui \in C_j} \frac{nmatch(U_i|top1(C_j))}{length(U_i)} \tag{5.6}$$

5. **%C.#top1to5**: Average of the percentage of clicks done by each session of the cluster in the cluster's five most used URLs.

$$P_{C_j}(C.\#top1to5) = \frac{1}{n_j} \sum_{U_i \in C_j} \frac{nmatch(U_i|top1to5(C_j))}{length(U_i)} \tag{5.7}$$

Clusters			All	Cl.1	Cl.2	Cl.3	Cl.5	Cl.6	Cl.7
#users			439	51	30	47	25	47	52
Activity (#clicks)			9	5	7.5	12	15	20	7
time/page			13.7	18.7	17.2	23.5	6.3	7.1	11.9
Distribution (M,E,N)			M	M	N	M	E	M	M
Depth			3.1	2.8	3.6	3.9	1.0	1.0	3.4
Dispersion	Activity (%#pagdiff)		66,7	80.0	53.3	100.0	13.3	25.0	57.1
	user	%#top1	22.2	40.0	40.0	8.3	73.3	75.0	42.9
		%#top1to5	77.8	100.0	93.3	41.7	100.0	95.0	100.0
	cluster	%#top1	0.0	0.0	20.0	8.3	73.3	75.0	28.6
		%#top1to5	33.3	40.0	80.0	25.0	80.0	75.0	85.7

Table 5.2: Most meaningful clusters and their characteristics for SDSC-HTTP database.

Clusters			All	Cl.0	Cl.1	Cl.2	Cl.4	Cl.5	Cl.7	Cl.8
#users			984	59	112	56	128	57	57	58
Activity (#clicks)			15	11	7	4	15.5	7	77	30.5
Time			14.7	20.6	16.1	11.9	17.8	12.7	12.1	11.6
Distribution (M,E,N)			M	E	M	M	M	E	M	E
Depth			2.8	3.3	2.2	1.3	3.1	2.0	3.4	3.5
Dispersion	Activity (%#pagdiff)		80.0	54.5	85.7	25.0	67.7	85.7	75.3	82.0
	user	%#top1	13.3	36.4	14.3	100.0	25.8	28.6	7.8	8.2
		%#top1to5	46.7	90.9	71.4	100.0	64.5	85.7	23.4	27.9
	cluster	%#top1	6.7	18.2	14.3	75.0	12.9	14.3	2.6	3.3
		%#top1to5	13.3	45.5	14.3	100.0	38.7	28.6	6.5	6.6

Table 5.3: Most meaningful clusters and their characteristics for EPA-HTTP database.

Clusters			All	Cl.0	Cl.1	Cl.4	Cl.5	Cl.6	Cl.7	Cl.9
#users			1224	194	33	193	207	14	6	150
Activity (#clicks)			116.5	127.5	2577	72	190	57.5	102	86
Time			14.0	14.5	15.4	12.2	14.9	7.1	26.4	13.6
Distribution (M,E,N)			M	E	E	M	M	N	M	M
Depth			3.2	3.4	3.2	1.1	3.3	3.0	3.8	1.9
Dispersion	Activity (%#pagdiff)		42.9	53.7	13.9	6.9	40.0	8.7	62.7	20.9
	user	%#top1	18.9	8.6	7.9	87.5	11.1	91.3	3.4	59.3
		%#top1to5	45.5	29.8	28.0	95.8	33.2	99.1	11.3	79.1
	cluster	%#top1	4.3	3.9	4.7	83.3	5.8	0.0	1.5	45.3
		%#top1to5	30.0	15.7	25.9	90.3	20.5	91.3	3.4	65.1

Table 5.4: Most meaningful clusters and their characteristics for NASA-HTTP database.

From the results obtained for SDSC database we selected 6 meaningful clusters (see Table 5.2). The users in clusters 2, 5, 6 and 7 are users that focus on a concrete task and also share interests with the rest of the users in the cluster. In the concrete case of clusters 5 and 6 the users also show high activity, low access time and very few different pages accessed. This suggests that the users on these two clusters are used to navigate in the web and they are searching for something very specific. The only difference between the two groups seems to be the time of the day they access to the web. A deeper analysis of the visited URLs (web structure, web content) would allow us to decide whether the users in these clusters share interests or not.

On the other hand, from the results obtained for EPA database we selected 7 meaningful clusters (see Table 5.3). The users in clusters 0, 1, 2, 4 and 5 are users that focus on a concrete task. Furthermore almost all the users in cluster 2 focus on the same task whereas in the other four groups there is more variety of objectives. On the other hand, the users in clusters 1, 2 and 5 show very low activity level in contrast to the users in clusters 7 and 8 which show very high activity level but without concrete objectives.

Finally, from the results obtained for NASA database we also selected 7 meaningful clusters (see Table 5.4). From the values in the table we could suspect that the activity of the users in cluster 1 belongs to some type of robots due to its surprisingly high activity but with very high dispersion in the accessed pages (low values in the last four rows). In this database, we can also find, in clusters 4, 6 and 9, a set of users that focus on a concrete task and besides they share the main objectives, they access mostly to the same URLs.

If we analyse the results in the three tables, we find that users in cluster 3 in SDSC database, in cluster 0 in EPA dataset and in cluster 7 in NASA dataset, are users that access the web slowly. This could happen because these users have difficulties to navigate or it could also be due to the high interest they have in the pages they are visiting.

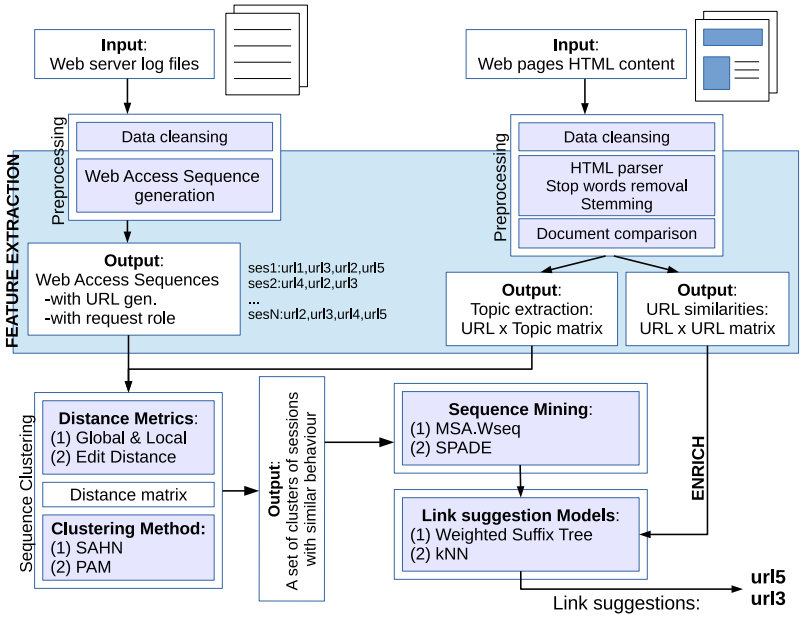

Figure 5.4: Overall methodology of the proposed systems.

5.4 Sequential navigation mining

In this section, we built a generic system based on the collaborative filtering approach that takes as input the minimum information stored in a webserver (webserver log files stored in Common Log Format (CLF) [193]) and combines unsupervised and supervised machine learning techniques to build user profiles and propose links drawn from those profiles to new users that they will probably be using in a short future. Furthermore, the profiles are built without any effort from the user, and besides, since we used the webserver log information available in all webservers, the system can be integrated in any website.

5.4.1 Overall schema of the proposed system

The user sessions extracted in the data preprocessing phase, represented as accessed URL sequences, will be used to build user navigation models for the analysed website. These models could be useful for many applications such as link prediction and page prefetching, trends and drift detection or helping to understand how people navigate within the website.

The proposed system, as depicted in Figure 5.4, constructs the required models by employing four major processes:

1. **Feature extraction:** where on the one hand the server log files (usage

data) are processed to create a database of web access sequences; and on the other hand the HTML text of the website (content data) is processed to create topic modelling based similarities between URLs.

2. **Sequence clustering:** whereby using properly defined sequence distance metrics (including topic distributions of URLs) and clustering methods the initially extracted sequences are partitioned.

3. **Sequence mining in the produced clusters:** where each one of the cluster is 'summarized' to its representative characteristics.

4. **Link suggestion model:** where the final models are produced and maintained for utilization (finally suggested links can be enriched with similar URLs).

Figure 5.4 also highlights the basic mechanics of each process. During the sequence clustering it is possible to employ different distance metrics either based on the combination of global/local alignment or on edit distance schemes. We propose alternative definitions for these metrics adding information about page topics, time spent on visited page (role), etc. The clustering methods employed are the agglomerative hierarchical clustering (SAHN) and the partitioning around medoids method (PAM), explained in Sections 3.2.1.1 and 3.2.2.2 respectively. During the cluster modelling process two main ways are used to model each cluster: (1) weighted sequences obtained from multiple sequence alignment (MSA.Wseq) and (2) SPADE sequences. Finally, during the link suggestion model construction phase the extracted sequences were modelled in suffix tree structures (ST) or the navigation model was constructed between the generated clusters and the extracted URLs which was exploited by computing kNN to the medoids.

In order to facilitate the readability of the proposed method, Figure 5.5 presents a summary of all the options evaluated and the nomenclature used to describe each process and their sub-steps respectively. Likewise, the developed configurations are shown. As an example, ED+SAHN+SPADE+ST means that we employed edit distance in hierarchical clustering (clustering), SPADE (cluster modelling) and suffix trees (link suggestion modelling). As another example, a label with prefix GL.wirol.notop means that as a distance metric a combination of global & local alignment was used, using page roles and not page topics. So the '+' mark is used for discriminating the process steps while the '.' mark is used for alternative options inside a step. In bold the default options.

5.4.2 Clustering the navigational sequences

We grouped users with similar navigation behaviour using clustering algorithms. Specifically we used two algorithms:

- **Agglomerative hierarchical clustering (SAHN)** with Ward linkage criteria (after testing different criterias, the one that behaved best). As an

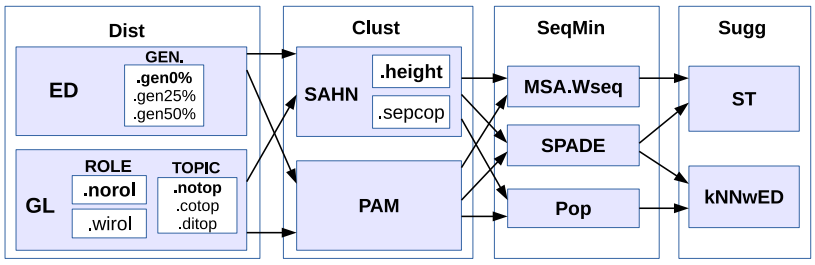

Figure 5.5: Naming scheme (nomenclature) of various setups of the proposed system.

output creates an hierarchy of clusters and to extract a particular partition of clusters we used two techniques: (1) the height based method and (2) the SEP/COP method (see Section 3.2.1.1). We adapted the SEP/COP method to be able to work with medoids instead of centroids.

- **Partitioning Around Medoids (PAM)**.

Both clustering algorithms were able to cluster sequential data. Next, the distances used in the clustering process are explained (see Section 5.4.2.1) and afterwards, how these clusters of sequences were modelled is explained (see Section 5.4.2.2).

5.4.2.1 Distances for sequences

In order to cluster sequences the use of appropriate distance functions is necessary, regardless of which clustering algorithms are used.

First, we created the distance matrix with all the training sequences selecting one of the following two metrics:

1. Edit Distance (ED): (see Section 3.1.2.1) the intuition behind is that when more operations are needed to convert one sequence into other, less common navigational patterns share the two sequences and vice versa. The generation of the distance matrix of training sequences is straightforward with this distance.

2. Hybrid metric combining Global and Local alignment methods (GL).

Global and Local alignment method (GL). The intuition behind is that common navigational preferences of the users are not captured only by aligning the sequences in their whole length; and that very similar subsections of the sequences can also capture common behaviour. Therefore, we propose the combination of global and local sequence alignment. This combination will bring the sequences of different length which have similarities closer than it would by simply using edit distance or its equivalent global alignment. In this way

the comparison between sequences will be more length independent which is a desirable feature.

With this method the creation of the distance matrix out of the training sequences is not so direct because it returns a score (similarity). Therefore, the process to create the distance matrix is explained next:

First the combination of global and local alignment (see Section 3.1.2.2) was computed using the classical approach of dynamic programming [146]. However, we applied some changes. Standard global and local algorithms score 1 for the match between two aligned elements and -1 for the mismatches. However, we introduced URL role and URL topic similarity in the scoring process.

- **Role**: in the case of time-roles, if two URLs were equal then we weighed the matching according to their time-role tag. We experimented with different weightings:
  - The basic one where the role is not taken into account (norol) and all values of Table 5.5 are one.
  - It includes weights taking into account role matches. The weights of the matches taking into account roles are shown in Table 5.5 (wirol).

Roles	U	H	C
U	0	0	0
H	0	1	0.75
C	0	0.75	1

Table 5.5: Weights of matchings using roles.

- **Topic**: in the global and local alignment calculation the topics are taken into account: when two URL's were equal we used the original match score, 1; in the case of mismatches, we propose the use of two strategies.
  - The similarity value of the two URL's topic-distribution was used as the match score (cotop: continuous topic).
  - A similarity threshold value was defined analysing the distribution of the URL similarity and those being above the threshold were given a 0.5 match score (ditop: discrete topic).

Finally, to combine role and topic similarity we multiplied them. Summing up, the alternative configurations for the proposed global and local alignment distance metric are GL.norol.notop, GL.wirol.notop, GL.norol.cotop, GL.wirol-.cotop, GL.norol.ditop and GL.wirol.ditop.

To obtain the score we used, first, the optimal sequence alignment was obtained by backtracking in the score matrix. Next, for two aligned sequences, the final score was computed according to the number of matches (#matches),

the number of mismatches (#mismatches), the number of gaps (#gaps) and the number of spaces, composed by many gaps simultaneously (#spaces). We computed the alignment score as follows:

$$\begin{aligned} FinalScore =& w_m * (\#matches) \\ & -w_{ms} * (\#mismatches) \\ & -w_g * (\#gaps) - w_s * (\#spaces) \end{aligned} \tag{5.8}$$

where w_m denotes the weight of the match, w_{ms} denotes the weight of the mismatch, w_g denotes the weight of starting a new gap and w_s denotes the weight of the alignment with a space. GL weighs negatively the alignments of symbols with spaces as very long spaces mean that the users have followed very different paths. On the contrary, it is more reasonable to favour small gaps. Therefore, we chose the scores '$w_g = 1$' and '$w_s = 5$' (and '$w_m = 1$' and '$w_{ms} = 1$' for match and mismatch). The gap opening penalty was intentionally kept smaller than the space penalty in order to favour frequent gap openings with small lengths instead of longer and more infrequent spaces [81].

The obtained similarity matrix was normalized based on the maximum and minimum values and converted to distance matrix so that it could be used in clustering algorithms.

Once the distance matrix was created we applied SAHN and PAM clustering algorithms to break the training sequences into clusters.

5.4.2.2 Cluster Modelling

Next, it is crucial to model each user group and to discover the associated navigation patterns or profiles, i.e., to find the common click sequences appearing in the sessions within a cluster. For profiling the group of sequences, we relied on three alternative schemes:

1. The most frequent URLs (Pop).

2. Profile based on sequential mining techniques (SPADE).

3. Profiling based on Multiple Sequence Alignment (MSA), Weighted Sequences (Wseq) and weighted Suffix Trees (ST).

Next, the four systems presented are explained:

Most frequent URLs to model sequences (Dist+Clust+Pop). Probably one of the easiest methods to summarize and profile each cluster's sequences for its future exploitation is to extract the most frequent URLs. We called it popularity base method (Pop). After this process the final model consists on clusters (which can be represented with its medoid) and their popular URLs.

SPADE URLs to model cluster sequences (Dist+Clust+SPADE1). To evaluate the generated clusters' quality, we mined the most frequent sequences using SPADE and use the 1-length ones (SPADE1). Therefore, the final model consists on clusters (which can be represented with its medoid) and their SPADE URLs.

Weighted sequences to create generalized weighted suffix trees (Dist+Clust+MSA.Wseq+ST). In this approach, we expressed the sequences of each cluster as a weighted sequence (see Section 3.3.2.2). The produced weighted sequence will implicitly capture most of the navigational behaviour grouped in the cluster.

First, the web access sequences within a cluster were aligned using a Multiple Sequence Alignment (MSA) method (see Section 3.1.2.2) and creating MSA aligned sequences. These aligned sequences of the cluster's sequences were used to build the representative weighted sequence of the cluster. Each position on the weighted sequence provides information regarding the appearance probability of each page (see Section 3.3.2.2).

Next, we used the weighted sequence (Wseq) to extract all the possible representative sequences using the algorithm used by Iliopoulos et al. [97] tuned by a parameter, r/p, being r the number of repetitions of each element-position and p the size of the set. For each weighted sequence we generated all possible artificial sequences that exceed the minimum ratio fixed in every position. Finally, each cluster's artificial sequences were used to build the final Suffix Tree (gWST or ST), which was the final model.

SPADE sequences to create generalized suffix trees (Dist+Clust+SPADE+ST). SPADE is used for mining the most common click sequences within each cluster. Taking the partition of clusters as an input, we calculated the SPADE sequences for each cluster, n-length SPADE sequences, where the only parameter is the sequences' minimum support. The application of SPADE provides a set of sequences of URLs for each cluster that are likely to be visited in the sessions belonging to it. Therefore, these SPADE sequences were used to build the final Suffix Tree (gWST or ST), which will be the final model.

5.4.3 Link suggestion models

Our hypothesis is that the navigation pattern of new users can be foreseen using previous user groups' navigations. Therefore, the system will make the user navigation more efficient since it would be proposing the links that he/she will probably use and so he/she could reach faster and more comfortably his/her objectives.

5.4.3.1 kNN to medoids (kNNwED)

The model is composed by clusters (represented by their medoids for example) and their representative URLs (Pop or SPADE1). In this case links can be

suggested to the user based on a kNN supervised machine learning technique. The use of this model in the link prediction context consists of first using a kNN algorithm to find the nearest clusters to the known part of the current sequence, and then, using the URLs proposed in that cluster to do the link proposals. There are many options to measure the distance between the current sequence and the cluster: the linkage criteria can be varied using single linkage (to the medoid for example), average linkage, etc. and the used distance can also be varied we used edit distance. The nearest two, three and so on clusters (2-NN, 3-NN, etc.) can also be used to generate the suggestion.

Our hypothesis is that the navigation pattern of that user will be similar to the navigation patterns grouped in its nearest clusters. Based on the most representative patterns obtained in these concrete clusters, the most interesting links can be proposed or outlined to the new user.

Enriching the suggestions with semantics. We proposed to enrich the suggested links, generated using the usage data, with semantic information to improve performance (see Figure 5.4). Obviously the navigation pattern of the users depends on their interests, and, as a consequence, URLs with similar or related content to the ones appearing in the navigation profile will also be interesting for the user.

Although the enrichment was applied in this kNN to medoids context, since the enrichment is done based on links suggested by the system, this enrichment can be applied to all link suggestion systems.

5.4.3.2 Generalized Weighted Suffix Trees (ST)

The ST can be used for link prediction at any point of the user navigation following the path made by the user in the tree and proposing the transitions available from the current node. If the current user session is not directly represented by the ST, transitions not modelled in it (failure situations) can appear. In these situations, as the path in the ST cannot be followed, two options exist: ignoring it or repositioning the pointer.

We applied failure-functions as in the Aho-Corasick Automaton [81]. Its preprocessing takes linear time to the size of the structure. When fails happen the procedure consists on backtracking towards the root to nodes with smaller paths. This backtracking cannot be very extensive because of the limited length of the WASes.

Our experiments showed that repositioning the pointer always enables a better performance and the performance depends on the implemented strategy:

1. **Go to Root (GST.R)**: restart from the root node.
2. **Go to Longest Suffix (GST.LS)**: place the pointer in the node that represents the longest runnable suffix of the current path.
3. **Go to Longest Prefix (GST.LP)**: place the pointer in the node that represents the longest runnable prefix of the current path.

4. **Go to Length-1 Suffix (GST.L1S)**: run the last step from the root and place the pointer in the obtained node.

5. **Go to Longest Suffix and enrich with Length-1 Suffixes (GST-.LS.L1S)**: keep two pointers, one in the longest suffix and the other one in the length-1 suffix and if the former does not propose enough URLs enrich with the ones obtained from the latter.

The number of times the failure function is triggered determines the suitability of the suffix tree: triggering the failure function often denotes that the suffix tree needs an update, because many of the transitions appearing in the database have not been modelled.

5.4.4 Results and Analysis: NASA database

To evaluate the created models we integrated them in a system able to dynamically propose links to the user navigating in the web. In this way we were able to validate them upon standard evaluation methods.

The evaluation of this type of systems is complex. We are adapting the presentation of the web data so that after a new user starts navigating, proposals of links that he/she will likely be using are done. The best validation strategy would be to perform an experiment with real users and analyse the improvement the system introduces but, unfortunately, this is not always possible. As a consequence, we measured the efficiency of the system based on the prediction ability of its proposed links upon the navigation sequences we already have in the database.

First, the evaluation methodology and the system's parameter values are listed:

- **Validation method**: We used the hold-out validation method, basing the division of the database in temporal criteria: we used the oldest examples for training (2/3, 66%) and the latest ones for testing (1/3, 33%). The statistics of the division are shown in Table 5.6.

Hold-out	nSessions	nClicks	clicks/ses.
train-set	21,235	235,771	11,1
test-set	10,618	111,960	10.5

Table 5.6: The number of sessions (nSessions), number of clicks (nClicks) and the average length of each session (clicks/ses.) in hold-out sets.

  The sessions of the database are composed of at least 6 clicks, with a median of 8 clicks long (the middle value in the ordered length list) and 1,591 different visited URLs.

- **Evaluated navigation stages**: we used the test sequences to simulate a real navigation where, seeing only the initial part of the sequence, we

try to predict the rest. Different navigation stages were evaluated: 10% (just one click more or less), 25%, 50% and 100% (only for the evaluation of the model, because in evaluating the prediction the second part would be empty).

- **Validation metrics.** The prediction ability was measured using precision (pr), recall (re) and F-measure (Fm). These metrics were explained in Section 2.4 but their definitions can be contextualized to link prediction as follows.

  Precision measures the probability of the suggested links to be correct or used by the users as indicated in Equation 5.9. For example, if there exists an URL accessed in every user session (lets say the homepage), then by proposing only that URL the highest precision value would be obtained. When fewer URLs are proposed it will be easier to get a 100% precision value because there is less risk of failure.

$$Precision = \frac{numberOfCorrectlyPredictedLinks}{numberOfProposedLinks} \tag{5.9}$$

  Recall measures the probability of the clicked URLs to be suggested as indicated in Equation 5.10, in this case if all possible URLs are proposed the recall value will be 100%.

$$Recall = \frac{numberOfCorrectlyPredictedUserClicks}{numberOfUserClicks} \tag{5.10}$$

  The best precision values will be obtained by proposing few links and vice versa for the best recall values. In addition, the precision takes into account the number of proposed links but not the number of clicks made by the user, and recall does the opposite. As a consequence a balance between precision and recall needs to be found, and F-measure is a good ally to achieve this aim.

  Furthermore we proposed the metric called *touched* (tou) which calculates the percentage of test examples that used at least one of the URLs proposed by the system. That is, the percentage of users that would get some type of satisfaction. Equation 5.11 shows how the *touched* metric is computed.

$$Touched = \begin{cases} 0 & \text{if } \nexists rec_i \mid rec_i = click_j, \\ & \forall i1 \leq |rec|, \forall j1 \leq |click| \\ 1 & \text{if } \exists rec_i \mid rec_i = click_j, \\ & \forall i1 \leq |rec|, \forall j1 \leq |click| \end{cases} \tag{5.11}$$

  We evaluated the performance of the link suggestion system, i.e., calculating the metrics comparing the suggested URLs to the really used

ones. Likewise, we evaluated how well the sequences were modelled, that is, we measured the model concordance with the test sequences using the complete test sequences: precision of the model (prMo), recall of the model (reMo), F-measure of the model (FmMo) and *touched* of the model (touMo).

When analysing the results we need to take into account that the number of clicks proposed could be more than, equal to, or, less than, the clicks the user makes in the navigation, thus, in general, the optimal maximum value for precision and recall is smaller than 100%.

Note that in general, the obtained prediction ability should be seen as a lower bound because, although not appearing in the user navigation sequence, the proposed links could be useful for them. Unfortunately, their usefulness/relevance could only be evaluated in a controlled experiment, by using user feedback.

Therefore, to overcome these two drawbacks of the mentioned metrics we proposed to use a higher abstraction of URLs or the URLs' content similarity instead of the exact match of the URLs to compute precision and recall.

- **System parameters:**
  - **URL generalization parameters**: we instantiated the URL generalization parameters as follows: $MinNSegment$ to 3 and generalization degree (gen, α) to three values: 0% (no generalization), 25% and 50% (see Section 15).
  - **Profiling the clusters**: to extract the SPADE URLs (SPADE1) from clusters a minimum support of 0.5 was used.
  - **kNN linkage criteria**: edit distance between the test sequence and the cluster by medoid-linkage criteria.
  - **Number of selected profiles**: the nearest one or two clusters were found (1-NN or 2-NN) and their corresponding profiles were selected.
  - **Number of proposed links**: the system decides according to the selected profiles. Generally between 3 and 4 links. In this type of link suggestion systems the proposal of a large number of links to the user would probably distract him/her and would not be very helpful. As a consequence, we proposed a small number of useful links so that the user is not confused.

5.4.4.1 Link prediction using cluster model SHAN + SEP/COP

We applied the combination of SHAN and the modified SEP/COP to the training data so that the sessions with similar navigation characteristics were clustered into the same group. The outcome of the process consisted on 2,818 clusters where 726 (25.8%) clusters contained just a single session and 638 (22.6%) clusters had just two sessions. From this outcome we could conclude that some of

minCS	1	2	3	5	10	20	30	40	50
#clusters	2818	2092	1454	847	359	156	89	57	45

Table 5.7: Number of clusters (#clusters) in the partition and their minimum size (minCS).

		S:100			**S:50**		
Options	**minCS**	**prMo**	**FmMo**	**touMo**	**prMo**	**FmMo**	**touMo**
Y:0	3	71.1	59.4	93.4	66.5	55.4	96.4
	5	74.1	60.5	97.5	69.8	56.1	98.3
	10	75.1	59.3	98.9	70.5	54.5	98.0
	20	72.4	55.8	97.4	68.8	51.4	95.5
Y:50	3	66.5	53.2	99.2	63.3	49.7	96.5
	5	69.8	54.1	92.7	69.3	51.9	90.1
	10	73.8	55.1	92.7	71.3	51.6	89.4
	20	71.5	52.2	90.5	68.7	48.9	90.0

Table 5.8: Average performance of the model concordance (prMo, FmMo, touMo) of the the system ED.gen0%+SHAN+SPADE1+kNNwED.gen((**Y**))%-:st**((S))**%.

the sessions belong to users with odd behaviour and they have not been grouped with any other session. The navigation patterns of these users do not seem to be very common or have enough repetitions to be taken into account. Therefore, we fixed the minimum cluster size to 3, and therefore we built a smaller model (fewer clusters). We explored the behaviour of the system using different values as minimum number of examples per cluster. Concretely, we experimented with 3, 5, 10, 20, 30, 40 and 50, and the number of clusters (#clusters) obtained for each minimum cluster size (minCS) is shown in Table 5.7.

First, we evaluated the model concordance, varying the cluster filtering threshold (minCS) and the navigation stage. Table 5.8 summarizes the performance results obtained for each configuration of the system when evaluating the model. We saw that if we fix the minimum size of the clusters to 30, 40 or 50 examples, the system performance worsens, thus we do not show them in the table.

The results shown in Table 5.8 are for the system ED.gen0%+SHAN+SPADE1+kNNwED.gen((**Y**))%:st((**S**))%. Results at stage 50% and 100% (st50%, st100%) and with and without URL generalization (gen50%, gen0%) at the exploitation time. Where minCS means minimum cluster size. And the best values of each configuration (row) are coloured.

The first conclusion we can draw from this table is that even if the values of the measured parameters vary depending on the selected option, all of them can model in certain percentage the URLs a new user will be visiting.

Analysing all the results it seems that the option where the most similar clusters to the test examples are found is without URL generalization (in exploitation time) from the two points of view: F-measure and number of sessions *touched*. This makes us think that for the nearest cluster selection phase, more specificity of the accessed links is an important aspect and the proposed URL generalization step introduces too much generalization and so, loss of information.

Moreover, it seems that in the structure captured by the combination of SAHN and the modified SEP/COP, up to a point, not only large clusters are significant. Because, if we ignore clusters smaller than 30 examples, the system performance worsens. Nevertheless, it is not clear which is the best minimum size for the generated clusters. Independently of the used criteria, F-measure or number of sessions *touched*, it could be either 5 or 10 depending on the rest of parameters because the difference between the two options is small.

0-day problem. Finally, if we center the analysis in the 0-day problem, we realize that although the quality of results decreases when we use the initial 50% of the test sequence instead of using 100%, the results were still good, obtaining precision values up to 71.3% and being at least one of the proposed URLs useful for up to 98.3% of the test examples. As we mentioned before, we went further in this analysis and evaluated how the system models sequences when just 10% or 25% of the URLs in each test sequence are used to select the corresponding cluster.

Therefore, we analysed the presented system ED.gen0%+SHAN.seqcop+-SPADE1+kNNwED.gen0% in the navigation stage 10% and with its best options: without URL generalization and with minimum cluster size of 5 and 10. As it could logically be expected, the quality of the results decreased as we used a shorter initial part of the test sequence (early stages) because we know less about the user. However, results achieved at very early stages (10%) were still good obtaining precision values up to 59.2% and being at least one of the proposed URLs useful for up to 96.3% of the test examples. So we could say that the proposed system is definitely able to tackle the 0-day problem.

5.4.4.2 Link prediction using PAM cluster model

In this case we used PAM clustering algorithm for pattern discovery. As we concluded with the SHAN model, URL generalization in finding the nearest profiles worsens the quality of the results, thus, in the PAM model we did not use generalization in the exploitation phase. Regarding to the used performance metrics precision (pr, prMo) and F-measure (Fm, FmMo) are presented for model evaluation and in link prediction.

In PAM clustering algorithm we have to provide the number of clusters, K, to cluster the sessions with similar navigation characteristics into the same group, and so, we might be somehow obliging the clustering algorithm to assign patterns to a cluster even if their distance to the rest of the patterns in the same

Options	K:100			K:200			K:500		
	X:0	**X:25**	**X:50**	**X:0**	**X:25**	**X:50**	**X:0**	**X:25**	**X:50**
P:Pop; **st25%-pr**	34.1	33.8	31.7	35.9	35.8	32.5	35.4	34.9	33.4
P:Pop; **st50%-pr**	26.9	26.8	24.6	42.1	42.6	48.8	41.7	42.2	45.5
P:Pop; **st25%-Fm**	28.8	28.5	27.1	25.3	25.4	23.3	29.3	28.9	28.3
P:Pop; **st50%-Fm**	24.1	24.1	22.6	25.3	25.4	23.3	25.6	25.5	23.8
P:SPADE1; **st25%-pr**	41.8	42.2	47.0	28.5	28.7	25.5	29.4	29.3	26.5
P:SPADE1; **st50%-pr**	34.7	35.1	39.8	34.5	34.8	42.1	34.8	36.0	39.3
P:SPADE1; **st25%-Fm**	31.3	31.2	29.2	32.3	32.3	30.3	31.6	31.9	30.7
P:SPADE1; **st50%-Fm**	27.9	27.9	27.4	28.5	28.3	29.5	28.4	29.4	28.9
P:Pop; **st25%-prMo**	54.7	54.5	51.0	57.2	57.2	52.1	58.4	58.2	54.0
P:Pop; **st50%-prMo**	60.2	60.0	55.1	64.7	65.2	70.7	65.8	66.1	68.3
P:Pop; **st25%-FmMo**	47.1	46.9	43.8	49.3	49.3	44.8	50.4	50.2	46.6
P:Pop; **st50%-FmMo**	51.9	51.8	47.4	54.3	54.4	48.9	56.4	56.4	51.1
P:SPADE1; **st25%-prMo**	64.3	64.7	69.1	62.8	62.9	56.7	65.2	65.2	59.2
P:SPADE1; **st50%-prMo**	69.8	70.3	73.3	69.7	70.3	75.0	71.5	71.9	73.6
P:SPADE1; **st25%-FmMo**	51.4	51.2	47.6	52.9	52.9	49.4	54.1	54.2	50.9
P:SPADE1; **st50%-FmMo**	56.8	56.8	52.3	58.2	58.2	54.9	60.1	60.6	56.4

Table 5.9: F-measure (Fm, FmMo) an precision (pr, prMo) values achieved with the system ED.gen((**X**))%+PAM((**K**))+((**P**))+k1NNwED.gen0.

cluster is big. To avoid the noise this could produce we removed in each cluster the patterns with distance 1 (maximum distance) to every other pattern.

As a first stage we tried to determine the best parameter combination of the system: the number of clusters (K), the URL generalization in the clustering phase (X) and profiling strategy (P). Table 5.9 summarizes the results for K in the range of 100, 200 and 500 (PAM100, PAM200 and PAM500); different URL generalization degrees 0%, 25% and 50% (gen0%, gen25% and gen50% respectively); and the popularity based profiles (Pop) or the SPADE URLs based profiles (SPADE1). The metrics used are Precision (pr and prMo) and F-measure (Fm and FmMo) and have been calculated using 25% and 50% (evaluated navigation stages) of the test user sequences for finding the nearest cluster (st25% and st50%). Every result belongs to the 1-NN, nearest cluster option (k1NN). The best value for each row is coloured (dark gray) and the best for each K and for a row too (light gray).

The first conclusion we can draw form the results is that also in this case, even if the values of the measured parameters vary depending on the selected option, all the configurations are able to predict a certain percentage of the links a new user will be visiting.

A second clear conclusion that can be drawn from the table is that, keeping constant the rest of the parameters, the SPADE URLs based profiles (SPADE1) are more appropriate than the popularity based ones. Moreover, it seems that taking into account precision and F-measure values, the biggest K values seem

Options	k:1			k:2		
	X:0	**X:25**	**X:50**	**X:0**	**X:25**	**X:50**
st25%-pr	41.7	42.2	45.5	51.0	53.5	50.2
st25%-prMo	65.8	66.1	68.2	51.0	78.4	73.5
st25%-Fm	31.6	31.9	30.7	36.0	37.3	33.0
st25%-FmMo	54.1	54.2	50.9	48.3	65.2	55.3
st50%-pr	34.8	36.0	39.3	44.8	45.5	45.1
st50%-prMo	71.5	71.9	73.6	81.3	81.5	79.6
st50%-Fm	28.4	29.4	29.0	33.8	34.3	31.4
st50%-FmMo	60.1	60.6	56.4	69.3	69.6	61.5

Table 5.10: Precision and F-measure results of the system ED.gen**((X))**%-+PAM500+SPADE1+**((k))**nnED.gen0%.

to perform better, 200 and 500, than 100. Nevertheless, the difference between PAM200 and PAM500 does not seem too big and, as a consequence, it does not seem that the analysis of bigger K values will benefit the system because performance improvement would be small and the increase of the system complexity big.

Moreover, from the comparison of the achieved values we can conclude that the use of a certain URL generalization degree in the clustering stage improves the quality of the results. The generated user profiles seem to be more appropriate for the mid-range values of α parameter (gen25%) since those are the ones achieving lightly greater values. In addition, we can observe that the generalization procedure seems more important when bigger the number of generated clusters is. That is to say, when more diversity of clusters we have, URL generalization works better, what may indicate that the system wants more uniformity, less clusters. However the patterns found in this way are different from the ones obtained using smaller K values as the results indicate.

As a summary we could state that, when 1-NN strategy is used in exploitation, and independently of the part of the sequence seen for prediction (st25% or st50%), a good parameter combination is: ED.gen25%+PAM500+SPADE1-+kNNwED.gen0%.

Diversification when suggesting links. Next step is to analyse the effect of changing the exploitation approach; the effect of using the two nearest profiles (2-NN) instead of one (1-NN), that is, tuning the k parameter. Table 5.10 shows the comparison of precision and F-measure values obtained with 1-NN and 2-NN for the configurations showing the best performance in Table 5.9: PAM500 for clustering and SPADE1 for the profiling option. However, the URL generalization degree (X) and the number of nearest clusters (k). The best results for each option is coloured.

The results show that combining two profiles to propose links to the new user clearly benefits the performance of the system. Both parameters, preci-

sion and F-measure increase around 10 points. Another observable effect is that, URL generalization benefits the quality of results, and, also in this case the improvement is greater when mid-range values for URL generalization are used (gen25%). Concretely in both cases, according to values of model concordance and prediction values, the best performance is achieved for ED.gen25%-+PAM500+SPADE1+k2NNwED.gen0% option obtaining precision values up to 81.5 (prMo) and 53.5 (pr) respectively.

The evaluation method used can be considered very strict because proposed but not used URLs could be interesting for the user. Therefore we computed the performance metrics considering each URL as generalized URL or website zone, and we achieved precision values up to 97.0 in the model evaluation (prMo) and up to 92.4 in the prediction (pr). This is an important outcome since it means that the proposed generalized links (website zones) are located in interesting zones for the users in more than 90% of the times.

0-day problem. If we center the analysis in the 0-day problem, we realize that although the quality of results logically increases when longer has the user been navigating, the values are still acceptable in very early stages of the navigation. Even though the 10% of the user navigation sequence is known (one click in average), good precision values (pr = 51.6 and prMo = 65.2) are obtained. That is, the system is able to deal with the 0-day problem.

To conclude we must say that log files from The Internet Traffic Archive, and specially NASA (for its considerable number of requests) were very useful to learn and to take practical experience on preprocessing and mining the web usage data. From this experience we noticed that PAM clustering algorithm got better partitions than using SAHN, although it needed more computation.

5.4.5 Results and Analysis: BTw database

5.4.5.1 First approach: usage and content data based link suggestion system

In this approach as we worked in a real environment where the webpages' content was available, we combined it with the previously presented system. Therefore, we first generated navigation profiles by combining PAM with SPADE1 and we enriched the suggested links with models generated from content information.

First, the evaluation methodology and the system's parameter values are listed:

- **Log preprocessing.** Applying the experience acquired in processing the logs of The Internet Traffic Archive (see Section 5.2.2), we repeated the same process with the server logs of BTw making a few context adaptations. The preprocessing steps (see Section 5.2.1) are listed bellow:
  - **Step 1: Filter out erroneous requests.** Equally processed.

  - **Step 2: Identify webpages and user session.** The expire time of each session was fixed to 10 minutes.
  - **Step 3: Remove non-user requests.** Equally processed.
  - **Step 4: Filter out requests to unpopular webpages.** All webpages were taken into account.
  - **Step 5: Filter out requests to not interesting webpages.** Navigation in agenda, news and search pages were removed.
  - **Step 6: Filter out short and long access sessions.** those with a minimum activity level were selected (3 or more clicks); and the longest sequences were removed (those with a sequence length outside the 98% percentile).
  - **Step 7: Annotate webpages with topics based on their content.** Annotate webpages of access sequences with topic distributions obtained with Topic Modelling technique.
  - **Step 8: Annotate users' clicks with roles based on their behaviour.** Not done.

  After this process the size of the database was of 21,916 sequences with an average length of 7.7 clicks.

  The system evaluation process was also similar to the one used with data from the Internet Traffic Archive. The specific characteristics are summarized in the next points:

- **Validation method.** We used a 10-fold cross-validation methodology, dividing folds into a training set (7 folds, 15,341 examples), validation-set (2 folds, 4,380 examples) and test-set (1 fold, 2,195 examples). We used the validation-set to select K (number of clusters) and the test-set to evaluate the performance of the system.

- **Evaluated navigation stages.** The 25% stage was used. Analysing it we wanted to know the ability of the system to react in early stages of the navigation.

- **Validation metrics.** We evaluated the quality of the partitions according to the precision (pr), recall (re) and F-measure (Fm) metrics.

  As link prediction is a very strict evaluation strategy for link suggestion, we proposed to use a topic-distribution of URLs also to compute the prediction misses and hits.

- **System parameters:**

  - **Explored range of K.** As the internal structure of the data is completely unknown we experimented with a wide range of values for K to select the optimum number of clusters: we tried values ranging from 20 to 700. This range is much larger than the usually

explored one which is bounded by $\sqrt{n}$ (rule of thumb), in our case to 124 for 15,341 training examples. The exploration area was so extensive because we prioritized to find all navigation groups into the data.

- **Profiling the clusters.** We used a minimum support value of 0.2 for SPADE URLs (SPADE1).
- **kNN linkage criteria.** edit distance between the test sequence and the cluster by medoid-linkage criteria.
- **Number of selected profiles.** As the previous experience had indicated, a new user might not be identical to any of the profiles discovered in the training set, that is, their profile might have similarities with more than one profile and, as a consequence, the diversification helps; it is better to build the profiles of the new users dynamically based on some of their nearest profiles. We combined the profiles of the two nearest clusters.
- **Number of suggested links.** Mostly 3 links. Therefore, take into account that in the stage 25%, the user has done more or less 2 clicks, and he/she has more or less 6 left. Therefore, since we propose 3 URLs, the value of recall (re and reMo) supposing that all the URLs in the sequence are different would be at most $3/6 = 0.5$ and $3/8 = 0.38$ respectively.

The initial system. First, we applied directly the acquired experience to BTw server logs. Table 5.11 shows average results for the evaluation of the navigation profiles (pr, re and Fm) obtained for the validation-set and for different values of K. The prMoIncr, reMoIncr and FmMoIncr columns show the increment of the performance metrics per extra cluster. The second column, (#URL), shows the average number of links proposed in each profile.

The values of the performance metrics in Table 5.11 improve as the number of clusters increases, what means that the users of the system are very diverse and, as a consequence, smaller clusters capture better the different types of existing profiles. However, if we analyse the improvements in the performance metrics for each new cluster added, or normalized improvement differences, we detect an elbow, or change in order of magnitude of the improvement (shaded cells in Table 5.11), in K =300. We therefore fixed the number of clusters to 300. Thus, from this point on, we set the system to 300 clusters. The last row in Table 5.11 shows the average results of the 10 folds, for the test-set and K =300. We coloured the point where the increment becomes noticeably small and underlined the best partition of clusters considering the balance between results and complexity.

The results show that the values obtained for the test samples are similar to those obtained for the validation samples, thus confirming that the behaviour of the system is stable. Moreover, if we focus on the values of the performance metrics we can say that the profiles proposed to the new users fit in more than

K	#URL	PrMo	PrMoIncr	ReMo	ReMoIncr	FmMo	FmMoIncr
V:20	2.91	0.532		0.274		0.361	
V:40	2.93	0.556	0.00123	0.290	0.00081	0.381	0.00099
V:80	2.89	0.590	0.00085	0.309	0.00046	0.405	0.00060
V:120	2.93	0.603	0.00033	0.321	0.00032	0.419	0.00035
V:160	2.95	0.615	0.00029	0.331	0.00024	0.430	0.00027
V:200	2.89	0.633	0.00046	0.337	0.00016	0.440	0.00025
V:300	2.90	0.648	0.00015	0.347	0.00010	0.452	0.00012
V:400	2.90	0.656	0.00008	0.352	0.00005	0.458	0.00006
V:500	2.90	0.662	0.00006	0.356	0.00004	0.463	0.00005
V:700	2.96	0.663	0.00000	0.362	0.00003	0.468	0.00003
T:300	2.89	0.642		0.344		0.448	

Table 5.11: Profile evaluation: validation-set (V) results for the 10-fold CV. And the test-set (T) results for the selected partition.

60% with the real navigation sequence and that the generated profiles therefore have good quality.

Nevertheless, a qualitative analysis of the generated profiles showed that many of the users visited the homepage at intermediate points of the navigation. We consider this provides information about the users' navigation behaviour; it could suggest situations such as being lost or changing their mind about their interests, etc. so we still consider this as part of the profile. However, we consider that if the profiles are used to propose links to new users the proposal of the homepage would not provide any help for them and we therefore decided to remove the homepage from the generated profiles.

In this situation we generated new profiles in the same conditions used to generate the previous ones (K =300, maximum number of URLs per profile 4, minimum support = 0.2) but not admitting the homepage URL as a proposal. This will obviously affect the precision and recall values because having a forbidden URL reduces the highest achievable values. In order to obtain more realistic values, misses in the homepage were not counted when calculating recall. This process reduced the average number of URLs proposed per profile to 2.64, precision to 0.509 and recall and F-measure to 0.262 and 0.346 respectively. What this means is that half of the proposed URLs were used by the new users and the system was able to propose more than a quarter of the URLs that were actually used.

Evaluation using topic abstraction. The combination of usage and content data allowed us to evaluate the navigation profiles according to interests and it also allowed a further evaluation of the quality of the link prediction tool. It

Evaluation	Abstraction	#URLs	pr	re	Fm
Model (Mo)	URL	2.7	0.642	0.344	0.448
	Topic		0.909	0.730	0.810
Prediction	URL	2.7	0.267	0.155	0.196
	Topic		0.736	0.598	0.660

Table 5.12: Evaluation of profiles and link prediction according to interests.

allowed us to measure to what extent the links proposed to the new user are of interest for him/her.

With this aim, we first used the URL-topic probability vector provided by the STMT tool to automatically analyse the theme of each of the URLs. In the generated navigation profiles we replaced the URLs by their main topic and calculated precision, recall and F-measure in the same way we did it in the previous section, but using the new profiles. Table 5.12 shows the results obtained and compares them to the performance metrics calculated according to the URLs.

As it can be observed our intuition was correct. Precision and recall values, for example, soar up to 90.9% and 73% when evaluating profiles and up to 73.6% and 59.8% in the case of link prediction, what means that the generated navigation profiles are accurate according to interests and, moreover, that a high percentage of the links proposed by our system is of interest to the new users. This will obviously contribute to make their browsing experience more pleasant and will help them to achieve their objectives faster.

Enriching navigation profiles. The system we designed offers the option to enrich the link proposals by combining the profiles generated using clustering & SPADE1 with links proposed based on semantic information, i.e., the system would enrich profiles generated based on usage information with semantic information.

We implemented and evaluated this new option where in order to locate the system in the same point of the learning curve, we limited the usage profiles to two URLs and enriched them with a single URL; the semantically most similar URL to the one with greatest support among the ones obtained with usage information. When the proposed URL already appeared in the profile, we selected the next one with higher similarity.

This variant of the system for link prediction was evaluated by calculating the same performance metrics we calculated to evaluate the previous system. Table 5.13 shows the values of the performance metrics when the link proposal is diversified using semantic information and when link proposals are made based only on the navigation profiles. The number of proposed URLs is in the same range for both options although it is slightly greater for semantically enriched link proposals. The values of the three performance metrics calculated are also in the same range for both cases, although they are slightly better for the new system.

System	#URLs	Pr	Re	Fm
Usage	2.7	0.736	0.598	0.660
Usage+Semantic	2.9	0.736	0.603	0.663

Table 5.13: Evaluation of semantically-enriched link proposals according to interests.

Following the idea presented previously in the preliminary system, we could conclude that when using semantic information, based on topic modelling in this case, to diversify the links proposed to new users and introduce other products that might be of interest to the service provider, the system is still able to propose links to the new users that seem to be of great interest to them. In this case the system is achieving a double objective: it is helping the DMO staff in their campaigns and it is also providing a tool to enable the users to achieve their objectives faster and in a more pleasant way.

5.4.5.2 Second approach: combining link suggestion models

In this approach first we compare two previously published link suggestion systems and then we combine them to obtain a superior system. The two initial models which are listed below were shortly described in Section 5.4.2.2:

- **ED+PAM+SPADE1+kNNwED**: SPADE URLs to model cluster sequences and to exploit it the kNN to medoids technique was used.
- **GL.wirol+SHAN+MSA.Wseq+ST**: weighted sequences to create generalized weighted suffix tree models.

In the following lines, first, two base reference models are presented; afterwards, how to improve the initial systems is analysed; and finally, the combinations are analysed. In order to do a fair comparison all the systems were evaluated using the same data and evaluation criteria as described next:

- **Log preprocessing.** In this case, we did a slightly different log preprocessing to the one explained in Sections 5.2.1 and 5.2.3. For example, a deeper analysis of user agent was done for robot detection, a minimum frequency of pages was included and a time based role was assigned to each request.
  - **Step 1: Filter out erroneous requests.** Equally processed.
  - **Step 2: Identify webpages and user session.** The expire time of each session was fixed to 10 minutes.
  - **Step 3: Remove non-user requests.** Equally processed.
  - **Step 4: Filter out requests to unpopular webpages.** Equally processed.

  - **Step 5: Filter out requests to not interesting webpages.** Navigation in agenda, news and search pages were removed. Hence, for the modelling the sequences with at least 75% of appropriate URLs for link prediction (previously not eliminated URLs) were selected.
  - **Step 6: Filter out short and long access sessions.** those with a minimum activity level were selected (3 or more clicks); and the longest sequences were removed (those with a sequence length outside the 98% percentile).
  - **Step 7: Annotate webpages with topics based on their content.** Annotate webpages of access sequences with the topic distributions obtained with the STMT topic modelling toolbox.
  - **Step 8: Annotate users' clicks with roles based on their behaviour.** We annotated clicks with roles based on the time spent in each click: Unimportant, Hub and Content. Since, as opposed to hub-type and content-type clicks, the user has not shown any interest in unimportant requests, in order to ensure a minimum interestingness level in each sequence, we removed those sequences with more than 75% unimportant requests.

  In this way the database was reduced to 9,325 sessions with an average length of 10 clicks.

- **Validation method.** as the number of experiments carried out was big, we used the hold-out method, that is, the database was temporally divided so that the first part (2/3) was used for training the models and the last part (1/3) for testing them in a link prediction context.

- **Evaluated navigation stages.** We used the test sequences to simulate a real navigation where, seeing only the initial part of the sequence, we try to predict the rest. The prediction was made in different stages (evaluated navigation stages: 25%, 50% and 75%) but since the most preferable situation is to facilitate the navigation as soon as possible, we focused mostly in the 25% stage, i.e., we used the first 25% of the sequence to make link suggestions and calculated the prediction ability with the rest of the sequence (75%).

- **Validation metrics.** We evaluated the quality of the partitions according to precision (pr) and F-measure (Fm) metrics.

- **Number of proposed links.** We analysed the performance of the system for several numbers of proposed links: 3, 4, 5, 10, and 15. If the number of webpages provided by the system is relatively large, a selection of the most relevant webpages are presented (or prefetched) to the user. From a practical point of view the most desirable or appropriate number of proposed links are in the range of 3 to 5 links because too many links could confuse the user.

Base reference link suggestion models. We found interesting to compare the proposed systems with two baseline models: the Markov Chain (MC) and the Global Suffix Tree (GST).

Markov Chain. We generated a first order Markov Chain (MC), where each state represents one URL and the directed edges the transitions made by the users between URLs. We calculated the initial states' probability vector and the transition matrix using the URL sequences of the database. This model can easily be used to propose the next most probable URLs in the link prediction context. For each stage of the navigation, the next URLs with highest probabilities are proposed from the row corresponding to the most recently visited URL in the transition matrix. The results obtained are shown in Table 5.14.

Global Suffix Tree (GST). We denoted as Global Suffix Tree (GST) a weighted Suffix Tree (ST) [97] built using the complete training database. The nodes' depth of the suffix trees represent the relative navigation stage where the user is, and the edges of the suffix trees represent the URLs the users are clicking on. The suffix tree represents every suffix that appears in the training sequences; we weighed the edges according to the times the URLs appear in the training sequences.

Any suffix tree requires a failure function when used to facilitate next step in the navigation as described in Section 5.4.3.2. We selected the best failure function with this GST model. Five different failure functions (ff) were five failure functions were analysed: GST.R, GST.LS, GST.LP, GST.L1S and GST.LS.L1S. Therefore, we evaluated the performance of the aforementioned failure functions based on the GST. The results are shown in Table 5.14. We coloured the best precision and F-measure values for each number of recommendations (nrec).

	nrec	**3**	**4**	**5**	**10**	**15**
MC	**pr**	29.54	26.68	24.10	16.61	13.34
	Fm	24.66	24.05	22.90	17.73	14.88
GST.R	**pr**	32.50	29.39	26.95	20.10	17.33
	Fm	27.04	26.07	24.79	19.65	17.08
GST.LP	**pr**	28.77	26.35	24.19	18.97	16.83
	Fm	23.67	22.74	21.40	17.19	15.06
GST.LS	**pr**	31.79	29.14	26.87	21.04	18.61
	Fm	25.95	25.05	23.74	19.20	16.83
GST.L1S	**pr**	33.02	30.14	27.46	20.75	17.88
	Fm	27.28	26.46	25.00	20.06	17.39
GST.LS.L1	**pr**	33.13	30.01	27.14	19.09	15.29
	Fm	28.14	27.44	26.09	20.54	17.17

Table 5.14: Precision (pr) and F-measure (Fm) values obtained proposing 3 to 15 links (nrec) for the Markov Chain (MC) model and for the Global Suffix Tree (GST) model.

The results show that the suffix based failure functions are better than the prefix based ones, what makes sense because the navigation path's suffix contains the last steps the user has made. Logically to foresee the next steps, the suffix contains more valuable information than the first steps, i.e., the prefix.

Go to Longest Suffix and enrich with Length-1 Suffixes, GST.LS.L1S, is the failure function showing the best overall behaviour. As it can be observed in Table 5.14, GST.LS.L1S obtains the highest F-measure values in the most desirable range of number of proposed links (3-5 links). In precision values, especially when few links are proposed, its values are near the best performing option. GST.L1S behaves very well too, but comparing the F-measure values, GST.LS.L1S always behaves better.

Link suggestion models using sequence clusters. Once the baseline options were evaluated we proceed to evaluate our proposals. All our proposals need some parameter fixing and some were fixed using the previous experience but the other decisions made will be described in the following lines.

Systems based on SPADE1 for profiling. The ED+PAM+SPADE1-+kNNwED system was proposed by our research group [11] and combines clustering and SPADE1 for URL extraction and uses then kNN for link prediction. This system was evaluated based on two different distances: (1) edit distance (ED) and (2) combination of global and local alignment (GL) for its different variations introducing roles and topics; and using two different clustering methods: (1) SAHN and (2) PAM. Results are shown in Table 5.15 where the initial system is ED+PAM+SPADE1+kNNwED, is marked in bold, the best improved performance values to propose few links (3-5) are coloured for each clustering option where the best option is darker.

Results in Table 5.15 indicate that ED+PAM+SPADE+kNNwED behaves a little better than the reference model GST, up to 5 URL proposals. Otherwise, regarding the distance used to compare sequences, ED, behaves better than any of the variations combining global and local alignment. The results also indicate that generally PAM works better than SAHN for SPADE+kNNwED systems.

Small variations of the initial system ED+PAM+SPADE+kNNwED were tested, such as, the introduction of other distances and the use of topics and roles, other clustering algorithm, etc. Although for some numbers of URL proposals the initial system was improved, it was still the best for suggesting few links. When the number of suggested links is big, the introduction of topics seems to work better without roles, obtaining a good system in proposing many links.

Compared to the reference systems, it can be deduced that the link prediction ability of MC is easily improved, however the GST is only improved when few links are proposed.

Systems based on suffix trees built using weighted sequences. The system GL.wirol+Clust+MSA.Wseq+ST was proposed by the research group

Options		nrec	3	4	5	10	15
D:GL;	C:PAM	pr	34.06	29.42	26.82	19.35	15.75
r:norol;		Fm	29.63	27.46	26.26	21.01	17.82
t:notop;	C:SAHN	pr	31.41	26.97	23.83	17.00	13.58
		Fm	27.15	24.97	23.12	18.45	15.35
D:GL;	C:PAM	pr	33.07	30.44	27.24	19.16	15.49
r:wirol;		Fm	28.75	28.54	26.73	20.81	17.49
t:notop;	C:SAHN	pr	33.99	29.34	26.20	17.45	13.90
		Fm	29.70	27.41	25.62	18.95	15.72
D:ED;	**C:PAM**	pr	34.92	29.51	26.36	17.91	14.21
r:norol;		Fm	30.48	27.54	25.77	19.45	16.07
t:notop;	C:SAHN	pr	30.62	25.93	23.58	16.55	13.43
		Fm	26.45	23.95	22.88	17.90	15.14
D:GL;	C:PAM	pr	31.50	26.13	22.82	16.85	13.74
r:norol;		Fm	27.35	24.33	22.28	18.39	15.60
t:cotop;	C:SAHN	pr	31.94	26.98	23.80	16.29	13.41
		Fm	27.98	25.34	23.45	17.77	15.21
D:GL;	C:PAM	pr	33.60	28.90	25.72	16.93	13.74
t:cotop;		Fm	29.61	27.40	25.53	18.52	15.61
r:wirol;	C:SAHN	pr	32.11	27.93	25.03	17.44	13.85
		Fm	28.03	26.24	24.66	19.06	15.72
D:GL;	C:PAM	pr	34.70	29.62	26.54	18.61	14.92
r:norol;		Fm	30.25	27.70	25.96	20.26	16.91
t:ditop;	C:SAHN	pr	33.05	27.80	25.00	17.41	14.13
		Fm	28.78	25.96	24.45	18.93	15.99
D:GL;	C:PAM	pr	34.07	29.15	25.60	17.24	14.40
r:wirol;		Fm	29.80	27.29	25.09	18.74	16.31
t:ditop;	C:SAHN	pr	30.51	26.89	24.55	17.15	13.88
		Fm	26.27	24.99	23.96	18.66	15.72

Table 5.15: Precision (pr) and F-measure (Fm) values obtained suggesting 3 to 15 links (nrec) for ((**D**)).((**r**)).((**t**))+((**C**))+SPADE1+kNNwED system.

who received me in my international Phd stay in the University of Patras [58], and combines clustering and MSA.Wseq for user profiling. This system was evaluated based on two different distances: (1) edit distance (ED) and (2) combination of global and local alignment (GL) for its different variations; and for two different clustering methods: (1) SAHN and (2) PAM. The results are shown in Tables 5.16 and will later be compared to the Dist+Clust+SPADE-+kNNwED system analysed in the previous section. The initial system GL-.wirol.notop+SAHN+MSA.Wseq+ST, appears in bold, the best improved performance values to propose many links (10-15) are coloured for each clustering option where the best option is darker.

Taking the initial system, GL.wirol+SAHN+MSA.Wseq+ST and making changes to it, certain improvements occurred. On the one hand the introduction of topics works better without roles, and in this way we obtain a good system for proposing many links. The same trend was detected in the system SPADE1+kNNwED. The topic abstraction makes the URL sequences more similar because if two URLs are not equal their degree of similarity is used to compute the score. However, the introduction of roles makes two sequences more different because the alphabet is tripled and so, the match of URLs becomes more difficult. Therefore, these two strategies are pulling the alignment score in opposite directions, and the results verify that their combination is not beneficial for this application.

One noteworthy conclusion is that the initial system can be improved quite easily. Another conclusion is that the GST is not improved in an obvious way from an overall point of view. However if we focus on model complexity, GST has 60,821 nodes with 3.9 outcome edges on average, whilst GL.wirol+SAHN-+MSA.Wseq+ST has 24,297 nodes with 3.7 outcome edges on average, i.e., the latter has 2.5 times fewer nodes and 2.6 times fewer edges. As GST and the cluster based ST predictability abilities are very similar, the complexity makes the latter more desirable. Finally, the chosen variation behaves better when many links are proposed.

Another conclusion worth mentioning is that the introduction of topics reduces the model complexity. In the alignment phase, we can get a positive similarity score when the URLs are not identical (since in topic abstraction they are similar), getting topically aligned sequences. In each position of the aligned sequences there is more variety of URLs for the same topic, thus, with more variety of URLs it will be more difficult to satisfy a minimum number of URLs (minimum support). Therefore, STs created using topic abstraction, are near 1150 nodes and 3.3 edges per node. The complexity reduction is enormous and the obtained results are only slightly worse.

If we compare the system proposed in [58], GL.wirol+SAHN+MSA.Wseq-+ST, to the system proposed in [11], ED+PAM+SPADE1+kNNwED, the former outperforms when many links are proposed, however, the SPADE1 based system, ED+PAM+SPADE1+kNNwED, outperforms the ST based systems when few URLs are proposed. As a consequence, we could conclude that none of the initial systems is better than the other; each of them will be more appropriate in some environments. This behaviour leads us to a combination of both

Options		nrec	3	4	5	10	15
D:GL;	D:PAM	pr	27.20	25.62	23.70	20.01	17.90
r:norol;		Fm	23.06	23.26	22.42	20.52	18.66
t:notop;	D:SAHN	pr	28.70	26.07	25.41	18.64	15.67
		Fm	24.20	23.81	24.56	20.00	17.51
D:GL;	D:PAM	pr	28.02	24.85	24.06	19.47	17.72
r:wirol;		Fm	23.76	22.52	22.75	19.89	18.43
t:notop;	**D:SAHN**	pr	28.77	26.27	25.69	18.60	15.49
		Fm	24.34	24.03	24.83	19.99	17.33
D:ED;	D:PAM	pr	24.82	22.74	21.23	17.70	16.31
r:norol;		Fm	20.77	20.45	20.06	18.43	17.52
t:notop;	D:SAHN	pr	28.89	26.35	25.51	18.78	15.58
		Fm	24.43	24.06	24.66	20.22	17.47
D:GL;	D:PAM	pr	29.72	27.55	25.75	22.16	20.63
r:norol;		Fm	24.66	24.21	23.31	20.99	19.62
t:cotop;	D:SAHN	pr	32.39	28.27	25.73	19.95	17.47
		Fm	27.85	25.93	24.62	20.79	18.56
D:GL;	D:PAM	pr	30.15	27.13	24.95	20.79	19.52
r:wirol;		Fm	25.16	23.98	22.70	19.77	18.72
t:cotop;	D:SAHN	pr	31.07	27.75	25.19	18.40	15.38
		Fm	26.75	25.66	24.42	19.84	17.28
D:GL;	D:PAM	pr	30.42	27.69	25.41	21.25	19.45
r:norol;		Fm	25.53	24.69	23.43	20.84	19.29
t:ditop;	D:SAHN	pr	30.63	27.72	25.25	18.63	15.42
		Fm	26.22	25.51	24.41	20.07	17.31
D:GL;	D:PAM	pr	29.94	27.14	24.88	20.24	19.25
r:wirol;		Fm	25.06	24.22	23.07	19.89	19.17
t:ditop;	D:SAHN	pr	28.21	26.17	25.57	18.62	15.65
		Fm	23.82	23.92	24.71	20.02	17.53

Table 5.16: Precision and F-measure (Fm) values obtained suggesting 3 to 15 links (nrec) for ((**D**)).((**r**)).((**t**))+((**C**))+MSA.Wseq+ST systems.

systems in function of the number of proposed links.

Combination of SPADE1 based & ST based systems (Combined). The analysis of the results shows that each of the two initial systems, ED+PAM-+SPADE1+kNNwED and GL.wirol+SAHN+MSA.Wseq+ST, is specialized in different situations. Therefore a solution to improve the systems' performance was to build a combined system (called SPADE1+kNNwED and MSA.Wseq-+ST) which uses each of the systems in the area where it performs best, i.e., it uses the SPADE1 based system, ED+PAM+SPADE1+kNNwED, when proposing few links and, the ST based system, GL+SAHN+MSA.Wseq+ST, when proposing many links. Therefore, the combined system always outperforms the best of the systems (Table 5.18).

Systems based on suffix trees built using SPADE sequences (Hybrid). Both, SPADE and MSA.Wseq based algorithms provide representative sequences to build suffix trees and this allows a hybridization; the use of the SPADE output to generate suffix trees (SPADE+ST). Table 5.17 summarizes the results obtained with different variations of this strategy. All of them are the best obtained so far and the configuration we selected is the ED+SAHN-+SPADE+ST because it uses the simplest distance and the hierarchical clustering is computationally lighter than PAM partitioning clustering.

We present in Table 5.18 F-measure values of obtained with the two initial systems and the best hybrid system. It is evident that the overall behaviour of the ED+Clust+SPADE+ST is better.

According to the values obtained for the calculated metrics, in general, out of 5 URLs proposed by the ED+SAHN+SPADE+ST system, 1 or 2 will be used by the new user, and, in the same way the system will correctly predict 30% of the clicks a new user will make.

Options		nrec	3	4	5	10	15
D:GL;	D:PAM	pr	33.29	31.45	29.75	26.77	25.67
r:norol;		Fm	30.32	29.79	28.71	26.41	25.32
t:notop;	D:SAHN	pr	37.04	34.53	32.56	26.18	24.00
		Fm	31.37	30.95	30.23	25.49	23.44
D:GL;	D:PAM	pr	33.43	31.28	29.52	26.07	24.84
r:wirol;		Fm	30.52	29.63	28.51	25.66	24.43
t:notop;	D:SAHN	pr	37.12	34.22	31.83	25.94	23.68
		Fm	31.59	30.81	29.61	25.27	23.09
D:ED;	D:PAM	pr	32.10	29.28	27.76	23.40	22.22
r:norol;		Fm	29.62	28.31	27.60	24.15	23.01
t:notop;	D:SAHN	pr	36.59	33.86	31.62	24.04	21.08
		Fm	31.58	31.17	30.36	24.79	22.04
D:GL;	D:PAM	pr	31.46	28.97	27.10	23.27	22.06
r:norol;		Fm	28.59	27.42	26.16	23.03	21.82
t:cotop;	D:SAHN	pr	36.57	33.81	31.58	25.82	23.91
		Fm	31.20	30.42	29.26	24.84	23.06
D:GL;	D:PAM	pr	32.34	29.44	28.12	24.83	23.60
r:wirol;		Fm	29.19	27.52	26.74	23.99	22.76
t:cotop;	D:SAHN	pr	35.27	32.07	30.42	25.09	23.33
		Fm	29.68	28.12	27.33	23.12	21.42
D:GL;	D:PAM	pr	34.01	31.38	29.83	26.14	24.86
r:norol;		Fm	30.89	29.54	28.63	25.67	24.40
t:ditop;	D:SAHN	pr	36.25	33.39	31.22	25.14	22.62
		Fm	30.91	30.21	29.28	25.02	22.64
D:GL;	D:PAM	pr	34.54	32.16	30.33	26.60	25.48
r:wirol;		Fm	30.86	29.77	28.59	25.58	24.47
t:ditop;	D:SAHN	pr	36.93	34.01	32.03	26.10	23.73
		Fm	31.38	30.56	29.83	25.51	23.27

Table 5.17: Precision (pr) and F-measure (Fm) values obtained proposing 3 to 15 links (nrec) for ((**D**)).((**r**)).((**t**))+((**C**))+SPADE+ST system. The proposed best systems' prediction values appear coloured.

Options	nrec	3	4	5	10	15
REF1	pr	28.77	26.27	25.69	18.60	15.49
	Fm	24.34	24.03	24.83	19.99	17.33
REF2	pr	34.92	29.51	26.36	17.91	14.21
	Fm	30.48	27.54	25.77	19.45	16.07
Combined	pr	34.92	29.51	26.36	19.95	17.47
	Fm	30.48	27.54	25.77	20.79	18.56
Hybrid	pr	36.59	33.86	31.62	24.04	21.08
	Fm	31.58	31.17	30.36	24.79	22.04

Table 5.18: Precision (pr) and F-measure (Fm) values obtained proposing 3 to 15 links (nrec) for the two reference systems: (1) REF1:GL.wirol.notop-+SAHN+MSA.Wseq+ST and (2) REF2:ED+PAM+SPADE1+kNNwED; and the combined and hybrid (ED+SAHN+SPADE+ST) systems. The system's best prediction values appear coloured.

5.5 Summary

Web usage mining for web personalization is motivated by the increase of the information on the web during the last decades, because this often makes the amount of information intractable for users. As a consequence, there is a need of easier access to the required information and adaptation to their preferences or needs. That is, web personalization becomes essential.

In this chapter, generic and non-invasive systems to extract knowledge for web personalization were proposed just using the information stored in web-server log files. This knowledge extraction has been done in two senses: problem detection and link suggestion.

Regarding to contributions in problem detection, first, we performed server log files preprocessing, and we extracted a list of meaningful attributes to describe each session. After that, we clustered the sessions with the K-means clustering algorithm. The analysis of these clusters clearly showed that, for the three databases used in this work (SDSC [175, 176], EPA [62, 63] and NASA [144, 145] from The Internet Traffic Archive [51]), some user profiles could be detected using just the information in the webserver log files. With these results we validated the list of attributes extracted and the generated partitions. More precisely, different types of profiles were detected, for instance, we could find in all of them users that access the web slowly. This could happen because these users have difficulties to navigate or it could also be due to the high interest they have in the pages they are visiting. In this case, the addition of very simple web structure and content information such as the type of accessed pages (auxiliary or content page for example) could help to disambiguate both situations; show accesses to auxiliary pages would clearly show problems when accessing to the web.

Since these experiments have helped us to identify the type of information that will be useful in a data mining process to identify different navigation profiles contribute to go a step further towards web personalization.

In the context of link suggestion, we were definitely able to build general and non-invasive systems that, without any effort from the users, was able to automatically generate link proposals that will be helpful to make more comfortable and efficient the navigation of new users in the web. And as a consequence, we can claim that the machine learning techniques we used and their combination seems to be appropriate.

The preliminary system was generated for NASA [144, 145] and then, it was improved for Bidasoa Turismo website[4] (BTw). However, this system could be extended to any other environment since it uses the minimum information stored in any webserver (Common Log Format (CLF) [193]).

The designed system, after preprocessing the log files, identified different groups of users, built the corresponding profiles and automatically generated

[4] `http://www.bidasoaturismo.com`

useful link proposals for new users, to adapt the user navigation scheme, so that their browsing experience was improved. To evaluate the models, a link prediction environment was simulated.

We obtained that 53.5% (in NASA) and 36.6% (in BTw) of the suggestions were used, with a little drop in precision in the early stages of the navigations. Besides, our experiments showed that diversification of profiles help in link generation for suggestion. As the exact prediction of the URL was too strict we proposed to relax that condition by matching two URLs if they were in the same web zone or they were similar from the content point of view. In this case, we achieved up to 92.4% of precision.

Besides, the web usage based system proposed so far [11] was compared with a proposal of a research group of the University of Patras (Greece) [58]. Therefore, our system which uses a Partition Around Medoids (PAM) clustering algorithm with edit distance to group similar users, Sequential Pattern Discovery using Equivalence classes (SPADE) to generate user profiles and kNN (k-nearest neighbours algorithm) for link prediction, was compared to a second one which uses Sequential Agglomerative Hierarchical Non-overlapping (SAHN) clustering algorithm with a combination of a global and a local sequence alignment method to group similar users, multiple sequence alignment (MSA) to generate user profiles and suffix trees for link prediction.

We first introduced on small variations in the two previous approaches to improve the system as much as possible. The analysis of the comparisons showed that each of the initial systems performed differently being each of them specialized in different applications (depending on the number of links required). Based on these results we proposed two new systems: the combination and the hybridization of them. The second option, where suffix trees were built using SPADE sequences, outperformed every other approach for every number of links proposed with an average improvement of 17.25% compared to the combined method which also performed better than any of the original ones.

Chapter 6

Contributions to cluster validation

6.1 Introduction

The applied research carried out in the previous contributions has led us to a more theoretical research. Specifically, this research area arises from the necessity to validate partitions of clusters obtained by different clustering algorithms, that is, from the necessity to select the best partition. In this direction, an extensive analysis of principal Cluster Validation Indexes (CVI) was carried out. This analysis can be seen as a guide to help selecting the most appropriate CVIs in each crisp clustering problem.

Clustering is an unsupervised pattern classification method that partitions the input space into clusters. The goal of a clustering algorithm is to perform a partition where objects within a cluster are similar and objects in different clusters are dissimilar. Therefore, the purpose of clustering is to identify natural structures in a dataset [82, 104, 139, 183] and it is widely used in many fields such as psychology [93], biology [183], pattern recognition [139], image processing [45] and computer security [27].

Once a clustering algorithm has processed a dataset and obtained a partition of the input data, a relevant question arises: How well does the proposed partition fit the input data? This question is relevant for two main reasons. First, an optimal clustering algorithm does not exist, in other words, different algorithms –or even different configurations of the same algorithm– produce different partitions and none of the algorithms have proved to be the best in all situations [149]. Thus, in an effective clustering process we should apply different algorithms to obtain different partitions and select the one that best fits the data. Second, many clustering algorithms are not able to determine the number of natural clusters in the data, and therefore they must initially be supplied with this information –frequently known as the K parameter. Since this information is rarely previously known, the usual approach is to run the algorithm

several times with a different K value on each run. Then, all the partitions are evaluated and the partition that best fits the data is selected. The process of estimating how well a partition fits the structure underlying the data is known as cluster validation [82].

Cluster validation is a difficult task and lacks the theoretical background of other areas, such as supervised learning and a recent work argues the suitability of context-dependent evaluation methods [192]. Nevertheless, the authors also state that the analysis of cluster validation techniques is a valid research question in some contexts, such as clustering algorithms' optimization. In our opinion, cluster validation tools analysed in context-independent evaluations will greatly contribute to context-dependent evaluation strategies. Therefore, our general context-independent cluster evaluation process could contribute to future context dependent evaluations.

It is usual to classify the cluster validation techniques into 3 groups –internal, external and relative validation–, but the classification criteria are not always clear [38, 82, 104, 162]. In any case, there is a clear distinction between validation techniques if we focus on the information available in the validation process. Some techniques –related to external validation– validate a partition by comparing it with the correct partition. Other techniques –related to internal validation– validate a partition by examining just the partitioned data. Obviously, the former can only make sense in a controlled testing environment, since in a real application the underlying structure of the data is unknown, and therefore, the correct partition is not available.

When the correct partition is available the usual approach is to compare it with the partition proposed by the clustering algorithm based on one of the many indices that compare data partitions; e.g. Rand, Adjusted Rand, Jaccard, Fowlkes-Mallows, Variation of Information [28] (some of them have been explained in Section 3.2.3.2).

On the other hand, when the correct partition is not available there are several approaches to validate a partition. One of them is to focus on the partitioned data and to measure the compactness and separation of the clusters. In this case another type of index is used; e.g. Dunn [60], Davies-Bouldin [52], Calinski-Harabasz [40]. Another more recent approach is the stability based validation [30, 105] which is not model dependant and does not require any assumption of compactness. This approach does not directly validate a partition, but it relies on the stability of the clustering algorithm over different samples of the input dataset.

The differences of the mentioned validation approaches make it difficult to compare all of them in the same framework. Our contributions focus on the approach when the correct partition is not available, which directly estimates the quality of a partition by measuring the compactness and separation of the clusters. Although there is no standard terminology, in the remainder of this chapter we will call Cluster Validity Indices (CVI) to these type of indices. For the indices that compare two partitions we will use the term Partition Similarity Measure (PSM).

Previous works have shown that there is no single CVI that outperforms the

rest [57, 133, 137]. This is not surprising since the same occurs in many other areas and this is why we usually deal with multiple clustering algorithms, partition similarity measures, classification algorithms, validation techniques, etc. This makes it obvious that researchers and practitioners need some guidelines about the tool they should use in each environment.

Focusing on internal cluster validation, we can find some works that compare a set of CVIs and, therefore, these could be used as guidelines to select the most suitable CVI in each environment. However, most of these comparisons are related to the proposal of a new CVI [45, 84, 86, 120, 203] or variants of known CVIs [113, 149, 172] and, unfortunately, the experiments are usually performed in restricted environments –few CVIs compared on few datasets, just one clustering algorithm implied, etc. There are few works that do not propose a new CVI but compare some of them in order to draw some general conclusions [38, 57, 59, 137]. Surprisingly, the 25 year-old paper of Milligan and Cooper [137] is the work most cited as a CVI comparison reference. Certainly, to the best of our knowledge, nobody has since published such an extensive and systematic comparative study, before we carried out this work.

In this contribution we present the results of an extensive CVI comparison along the same lines as Milligan and Cooper [137], which is the last work that compares a set of 30 CVIs based on the results obtained in hundreds of environments. We claim that we have improved the referenced work in three main areas. First, we can compare many new indices that did not exist back in 1985 and discard those that have rarely been used since. Second, we can take advantage of the increase in computational power achieved in recent decades to carry out a wider experiment.

Moreover, our work is based on a corrected CVI comparison methodology presented by our research group [80] based on partition similarity measure (see Section 3.2.3.2). Therefore, we present three main contributions in this chapter. First, we present the main results of the most extensive CVI comparison ever carried out as far as we know. Second, this comparison is the first extensive CVI comparison carried out with the methodological correction proposed by our research group [80]. Finally, we analysed the behaviour of the CVIs in a real web mining application and with elevate number of clusters.

6.1.1 State of the art

Most of the works that compare CVIs use the same approach: a set of CVIs is used to estimate the number of clusters in a set of datasets partitioned by several algorithms. The number of successes of each CVI in the experiment can be called its score and is considered an estimator of its "quality".

Despite this widely used approach, most of the works are not comparable since they differ in the CVIs compared, datasets used, results analysis, etc. In this section we overview some of the works that compare a set of CVIs, focusing on the experiment characteristics.

The paper published by Milligan and Cooper [137] in 1985 is still the work of reference on internal cluster validation. That work compared 30 CVIs. The

authors called them "Stopping criteria" because they were used to stop the agglomerative process of a hierarchical clustering algorithm [104, 183] and this is why the experiments were done with hierarchical clustering algorithms (single-linkage, complete-linkage, average-linkage and Ward). They used 108 synthetic datasets with a varying number of non-overlapped clusters (2, 3, 4 or 5), dimensionality (4, 6 or 8) and cluster sizes. They presented the results in a tabular format, showing the number of times that each CVI predicted the correct number of clusters. Moreover, the tables also included the number of times that the prediction of each CVI overestimated or underestimated the real number of clusters by 1 or 2.

The same tabular format was used by Dubes [59] two years later. The novelty of this work is that the author used some tables where the score of each CVI was shown according to the values of each experimental factor – clustering algorithm, dataset dimensionality, number of clusters, etc. Moreover, they used the χ^2 statistic to test the effect of each factor on the behaviour of the compared CVIs. Certainly, the use of statistical tests to validate the experimental results is not common practice in clustering, as opposed to other areas such as supervised learning. The main drawback of this work is that it compares just 2 CVIs (Davies-Bouldin and the modified Hubert statistic). The experiment is performed in 2 parallel works of 32 and 64 synthetic datasets, 3 clustering algorithms (single-linkage, complete-linkage and CLUSTER) and 100 runs. The datasets' characteristics were controlled in the generation process and they used different sizes (50 or 100 objects), dimensionality (2, 3, 4 or 5), number of clusters (2, 4, 6 or 8), sampling window (cubic or spherical) and cluster overlap.

In 1997, Bezdek et al. [31] published a paper comparing 23 CVIs based on 3 runs of the EM algorithm and 12 synthetic datasets. The datasets were formed by 3 or 6 Gaussian clusters and the results were presented in tables that showed the successes of every CVI on each dataset. Another work that compared 15 CVIs was performed by Dimitriadou et al. [57] based on 100 runs of K-means and hard competitive learning algorithms. The 162 datasets used in this work were composed of binary attributes which made the experiment and the results presentation somewhat different to the previously mentioned ones.

Afterwards, Brun et al. [38] compared 8 CVIs using several clustering algorithms: K-means, fuzzy c-means, SOM, single-linkage, complete-linkage and EM. They used 600 synthetic datasets based on 6 models with varying dimensionality (2 or 10), cluster shape (spherical or Gaussian) and number of clusters (2 or 4). The novelty in this work can be found in the comparison methodology. The authors compared the partitions obtained by the clustering algorithms with the correct partitions and computed an error value for each partition. Then, the "quality" of the CVI was measured as its correlation with the measured error values. In this work, not just internal but also external and relative indices are examined. The results show that the Rand index is highly correlated with the error measure.

The mentioned correlation between the error measure and the Rand index makes one think about the adequacy of the error as a definitive measure. In

the work of our research group [80] it was accepted that there is no single way of establishing the quality of a partition and it was proposed using one of the available external indices –or even better, several of them. This is the first contribution that clearly confronted a methodological drawback ignored by many authors, but noticed by others [38, 86, 120, 137]. Since the main goal of this work was to present a modification of the traditional methodology, the comparison was carried out using just 7 CVIs based on 7 synthetic and 3 real datasets and 10 runs of the K-means algorithm.

Other CVI comparisons can be found where new CVIs are proposed, but in this case the experiment is usually limited. It is common to find works comparing 5 or 10 CVIs on a similar number of datasets [45, 84, 86, 113, 149, 172, 203].

6.2 Compared cluster validity indices

In this section we describe the 30 CVIs compared in this dissertation. We focused on the most used indices in the context of crisp clustering algorithms. First, to simplify and reduce the CVI description section we define the general notation used in this section.

6.2.1 Notation

Let us define a dataset X as a set of N objects represented as vectors in an F-dimensional space: $X = \{x_1, x_2, \ldots, x_N\} \subseteq \Re^F$. A partition or clustering in X is a set of disjoint clusters that partitions X into K groups: $C = \{c_1, c_2, \ldots, c_K\}$ where $\bigcup_{c_k \in C} c_k = X$, $c_k, c_l \in C$ $c_k \cap c_l = \emptyset$ $\forall k \neq l$. The centroid of a cluster c_k is its mean vector,

$$\bar{c_k} = \frac{1}{|c_k|} \sum_{x_i \in C_k} x_i \tag{6.1}$$

and, similarly, the centroid of the dataset is the mean vector of the whole dataset,

$$\bar{X} = \frac{1}{N} \sum_{x_i \in X} x_i. \tag{6.2}$$

We will denote the euclidean distance as $\mathrm{d_e}(x_i, x_j)$ between objects x_i and x_j. We define the Point Symmetry-Distance [25] between the object x_i and the cluster c_k as

$$\mathrm{d}^*_{\mathrm{ps}}(x_i, c_k) = \frac{1}{2} \sum \min(2)_{x_j inc_k} \{\mathrm{d_e}(2\bar{c_k} - x_i, x_j)\}. \tag{6.3}$$

The point $2\bar{c_k} - x_i$ is called the symmetric point of x_i with respect to the centroid of c_k. The function '$\sum$ min' can be seen as a variation of the 'min' function where '$\sum$ min(n)' computes the sum of the n lowest values of its argument. Similarly, we can define the '$\sum$ max' function as an analogue variation of the 'max' function.

Finally, let us define n_w since it is used by several indices. n_w is the number of object pairs in a partition that are in the same cluster,

$$n_w = \sum_{c_k \in C} \binom{|c_k|}{2}. \tag{6.4}$$

6.2.2 Index definitions

Next, we describe the 30 CVIs compared in the experiment. We focused on CVIs that can be easily evaluated by the usual methodologies and avoided those that could lead to confusion due to the need for a subjective decision by the experimenter. Therefore, we have discarded some indices that needed to determine a "knee" in a plot –such as the Modified Hubert index [95]–, need to tune a parameter or need some type of normalization –such as the v_{SV} index [112] or the Jump index [189]. We have also avoided fuzzy indices, since our goal was to focus on crisp clustering. In brief, we focused on crisp CVIs that allow selection of the best partition based on its lowest or highest value.

Most of the indices estimate the cluster cohesion (within or intra-variance) and the cluster separation (between or inter-variance) and combine them to compute a quality measure. The combination is performed by a division (ratio-type indices) or a sum (summation-type indices) [113].

For each index we define an abbreviation that will be helpful in the results section. Moreover, we accompanied each abbreviation with an upward or downward arrow. The downward arrow (↓) denotes that a lower value of that index means a "better" partition. The upward arrow (↑) means that higher values are "better".

- **Dunn index** (D↑) [60]: This index has many variants and some of them will be described next. It is a ratio-type index where the cohesion is estimated by the nearest neighbour distance and the separation by the maximum cluster diameter. The original index is defined as

$$\mathrm{D}(C) = \frac{\min\limits_{c_k \in C}\{\min\limits_{c_l \in C \setminus c_k}\{\delta(c_k, c_l)\}\}}{\max\limits_{c_k \in C}\{\Delta(c_k)\}} \tag{6.5}$$

  where

$$\delta(c_k, c_l) = \min_{x_i \in c_k} \min_{x_j \in c_l} \{\mathrm{d_e}(x_i, x_j)\}, \tag{6.6}$$

$$\Delta(c_k) = \max_{x_i, x_j \in c_k} \{\mathrm{d_e}(x_i, x_j)\}. \tag{6.7}$$

- **Calinski-Harabasz** (CH↑) [40]: This index obtained the best results in the work of Milligan and Cooper [137]. It is a ratio-type index where the cohesion is estimated based on the distances from the points in a cluster

to its centroid. The separation is based on the distance from the centroids to the global centroid, as defined in Section 6.2.1. It can be defined as

$$\mathrm{CH}(C) = \frac{N-K}{K-1} \frac{\sum\limits_{c_k \in C} |c_k| \, \mathrm{d_e}(\bar{c_k}, \bar{X})}{\sum\limits_{c_k \in C} \sum\limits_{x_i \in c_k} \mathrm{d_e}(x_i, \bar{c_k})}. \tag{6.8}$$

- **Gamma index** (G↓) [24]: The Gamma index is an adaptation of Goodman and Kruskal's Gamma index and can be described as

$$\mathrm{G}(C) = \frac{\sum\limits_{c_k \in C} \sum\limits_{x_i, x_j \in c_k} \mathrm{dl}(x_i, x_j)}{n_w \left(\binom{N}{2} - n_w \right)} \tag{6.9}$$

  where $\mathrm{dl}(x_i, x_j)$ denotes the number of all object pairs in X, namely x_k and x_l, that fulfil two conditions: (a) x_k and x_l are in different clusters, and (b) $\mathrm{d_e}(x_k, x_l) < \mathrm{d_e}(x_i, x_j)$. In this case the denominator is just a normalization factor.

- **C-Index** (CI↓) [96]: This index is a type of normalized cohesion estimator and is defined as

$$\mathrm{CI}(C) = \frac{\mathrm{S}(C) - \mathrm{S}_{min}(C)}{\mathrm{S}_{max}(C) - \mathrm{S}_{min}(C)} \tag{6.10}$$

  where

$$\mathrm{S}(C) = \sum_{c_k \in C} \sum_{x_i, x_j \in c_k} \mathrm{d_e}(x_i, x_j), \tag{6.11}$$

$$\mathrm{S}_{min}(C) = \sum \min_{x_i, x_j \in X} (n_w) \{ \mathrm{d_e}(x_i, x_j) \}, \tag{6.12}$$

$$\mathrm{S}_{max}(C) = \sum \max_{x_i, x_j \in X} (n_w) \{ \mathrm{d_e}(x_i, x_j) \}. \tag{6.13}$$

- **Davies-Bouldin** index (DB↓) [52]: This is probably one of the most used indices in CVI comparison studies. It estimates the cohesion based on the distance from the points in a cluster to its centroid and the separation based on the distance between centroids. It is defined as

$$\mathrm{DB}(C) = \frac{1}{K} \sum_{c_k \in C} \max_{c_l \in C \setminus c_k} \left\{ \frac{\mathrm{S}(c_k) + \mathrm{S}(c_l)}{\mathrm{d_e}(\bar{c_k}, \bar{c_l})} \right\} \tag{6.14}$$

  where

$$\mathrm{S}(c_k) = \frac{1}{|c_k|} \sum_{x_i \in c_k} \mathrm{d_e}(x_i, \bar{c_k}). \tag{6.15}$$

- **Silhouette index** (Sil↑) [170]: This index is a normalized summation-type index. The cohesion is measured based on the distance between all the points in the same cluster and the separation is based on the nearest neighbour distance. It is defined as

$$\mathrm{Sil}(C) = \frac{1}{N} \sum_{c_k \in C} \sum_{x_i \in c_k} \frac{\mathrm{b}(x_i, c_k) - \mathrm{a}(x_i, c_k)}{\max\{\mathrm{a}(x_i, c_k), \mathrm{b}(x_i, c_k)\}} \tag{6.16}$$

  where

$$\mathrm{a}(x_i, c_k) = \frac{1}{|c_k|} \sum_{x_j \in c_k} \mathrm{d_e}(x_i, x_j), \tag{6.17}$$

$$\mathrm{b}(x_i, c_k) = \min_{c_l \in C \setminus c_k} \{\frac{1}{|c_l|} \sum_{x_j \in c_l} \mathrm{d_e}(x_i, x_j)\}. \tag{6.18}$$

- **Graph theory based Dunn and Davies-Bouldin variations** ($\mathrm{D^{MST}}$-↑, $\mathrm{D^{RNG}}$↑, $\mathrm{D^{GG}}$↑, $\mathrm{DB^{MST}}$↓, $\mathrm{DB^{RNG}}$↓, $\mathrm{DB^{GG}}$↓) [149]: These indices are variations of Dunn and Davies-Bouldin. The variation affects how the cohesion estimators are computed –$\Delta(c_k)$ for the Dunn index and $\mathrm{S}(c_k)$ for the Davies-Bouldin index.

  For each of the 3 versions –MST, RNG and GG– these 2 functions are computed in the same way. First, a particular type of graph is computed for c_k, taking the objects in the cluster as vertices and the distance between objects as the weight of each edge. Then the largest weight is taken as the value for $\Delta(c_k)$ and $\mathrm{S}(c_k)$. The difference between the 3 variants comes from the selected graph type. For MST a Minimum Spanning Tree is built, for RNG a Relative Neighbourhood Graph and for GG a Gabriel Graph.

- **Generalized Dunn indices** (gD31↑, gD41↑, gD51↑, gD33↑, gD43↑, gD53↑) [32]: All the variations are a combination of 3 variants of δ –separation estimator– and 2 variations of Δ –cohesion estimator. Actually, Bezdek and Pal [32] proposed 6×3 variants –including the original index–, but we selected those proposals that showed the best results. Therefore we analysed the variants 3, 4 and 5 for δ and 1 and 3 for Δ.

$$\delta^3(c_k, c_l) = \frac{1}{|c_k||c_l|} \sum_{x_i \in c_k} \sum_{x_j \in c_l} \mathrm{d_e}\, x_i, x_j, \tag{6.19}$$

$$\delta^4(c_k, c_l) = \mathrm{d_e}(\bar{c_k}, \bar{c_l}), \tag{6.20}$$

$$\delta^5(c_k, c_l) = \frac{1}{|c_k| + |c_l|} \left(\sum_{x_i \in c_k} \mathrm{d_e}(x_i, \bar{c_k}) + \sum_{x_j \in c_l} \mathrm{d_e}(x_j, \bar{c_l}) \right) \tag{6.21}$$

and

$$\Delta^1(c_k) = \Delta(c_k), \tag{6.22}$$

$$\Delta^3(c_k) = \frac{2}{|c_k|} \sum_{x_i \in c_k} \mathrm{d_e}(x_i, \bar{c_k}). \tag{6.23}$$

- **SDbw index** (SDbw↓) [83]: This is a ratio-type index that has a more complex formulation based on the euclidean norm $||x|| = (x^T x)^{1/2}$, the standard deviation of a set of objects,

$$\sigma(X) = \frac{1}{|X|} \sum_{x_i \in X} (x_i - \bar{x})^2 \tag{6.24}$$

and the standard deviation of a partition,

$$\mathrm{stdev}(C) = \frac{1}{K} \sqrt{\sum_{c_k \in C} ||\sigma(c_k)||}. \tag{6.25}$$

Its definition is

$$\begin{aligned} \mathrm{SDbw}(C) = \frac{1}{K} \sum_{c_k \in C} \frac{||\sigma(c_k)||}{||\sigma(X)||} \\ + \frac{1}{K(K-1)} \sum_{c_k \in C} \sum_{c_l \in C \setminus c_k} \frac{\mathrm{den}(c_k, c_l)}{\max\{\mathrm{den}(c_k), \mathrm{den}(c_l)\}} \end{aligned} \tag{6.26}$$

where

$$\mathrm{den}(c_k) = \sum_{x_i \in c_k} \mathrm{f}(x_i, \bar{c_k}), \tag{6.27}$$

$$\mathrm{den}(c_k, c_l) = \sum_{x_i \in c_k \cup c_l} \mathrm{f}(x_i, \frac{\bar{c_k} + \bar{c_l}}{2}) \tag{6.28}$$

and

$$\mathrm{f}(x_i, c_k) = \begin{cases} 0 & \text{if } \mathrm{d_e}(x_i, \bar{c_k}) > \mathrm{stdev}(C) \\ 1 & \text{otherwise.} \end{cases} \tag{6.29}$$

- **CS index** (CS↓) [45]: This index was proposed in the image compression environment, but can be extended to any other environment. It is a ratio-type index that estimates the cohesion by the cluster diameters and the separation by the nearest neighbour distance. Its definition is

$$\mathrm{CS}(C) = \frac{\sum_{c_k \in C} \left\{ \frac{1}{|c_k|} \sum_{x_i \in c_k} \max_{x_j \in c_k} \{\mathrm{d_e}(x_i, x_j)\} \right\}}{\sum_{c_k \in C} \min_{c_l \in C \setminus c_k} \{\mathrm{d_e}(\bar{c_k}, \bar{c_l})\}}. \tag{6.30}$$

- **Davies-Bouldin*** (DB* ↓) [113]: This variation of the Davies-Bouldin index was proposed together with an interesting discussion about different types of CVIs. Its definition is

$$\mathrm{DB}^*(C) = \frac{1}{K} \sum_{c_k \in C} \frac{\max\limits_{c_l \in C \setminus c_k} \{\mathrm{S}(c_k) + \mathrm{S}(c_l)\}}{\min\limits_{c_l \in C \setminus c_k} \{\mathrm{d_e}(\bar{c_k}, \bar{c_l})\}}. \tag{6.31}$$

- **Score Function** (SF↑) [173]: This is a summation-type index where the separation is measured based on the distance from the cluster centroids to the global centroid and the cohesion is based on the distance from the points in a cluster to its centroid. It is defined as

$$\mathrm{SF}(C) = 1 - \frac{1}{\mathrm{e}^{\mathrm{bcd}(C) - \mathrm{wcd}(C)}} \tag{6.32}$$

where

$$\mathrm{bcd}(C) = \frac{\sum\limits_{c_k \in C} |c_k| \, \mathrm{d_e}(\bar{c_k}, \bar{X})}{N \times K}, \tag{6.33}$$

$$\mathrm{wcd}(C) = \sum_{c_k \in C} \left(\frac{1}{|c_k|} \sum_{x_i \in c_k} \mathrm{d_e}(x_i, \bar{c_k}) \right). \tag{6.34}$$

- **Sym-index** (Sym↑) [25]: This index is known as symmetry based cluster validity index and it is an adaptation of the I index [133] based on the Point Symmetry-Distance. It is defined as

$$\mathrm{Sym}(C) = \frac{\max\limits_{c_k, c_l \in C} \{\mathrm{d_e}(\bar{c_k}, \bar{c_l})\}}{K \sum\limits_{c_k \in C} \sum\limits_{x_i \in c_k} \mathrm{d^*_{ps}}(x_i, c_k)}. \tag{6.35}$$

- **Point Symmetry-Distance based indices** (SymDB↓, SymD↑, Sym33-↑) [172]: These 3 indices are also based on the Point Symmetry-Distance and modify the cohesion estimator of the Davies-Bouldin, Dunn and generalized-Dunn (version 33) indices.

The SymDB index is computed as DB, but the computation of S is redefined as follows:

$$\mathrm{S}(c_k) = \frac{1}{|c_k|} \sum_{x_i \in c_k} \mathrm{d}^*_{\mathrm{ps}}(x_i, c_k). \tag{6.36}$$

The symD index is like D, but the Δ function is defined as

$$\Delta(c_k) = \max_{x_i \in c_k} \{\mathrm{d}^*_{\mathrm{ps}}(x_i, c_k)\}. \tag{6.37}$$

And finally, the Sym33 index is a modification of gD33 where Δ is defined as

$$\Delta(c_k) = \frac{2}{|c_k|} \sum_{x_i \in c_k} \mathrm{d}^*_{\mathrm{ps}}(x_i, c_k). \tag{6.38}$$

- **COP index** (COP↓) [77]: Although this index was first proposed to be used in conjunction with a cluster hierarchy post-processing algorithm, it can also be used as an ordinary CVI. It is a ratio-type index where the cohesion is estimated by the distance from the points in a cluster to its centroid and the separation is based on the furthest neighbour distance. Its definition is

$$\mathrm{COP}(C) = \frac{1}{N} \sum_{c_k \in C} |c_k| \frac{\frac{1}{|c_k|} \sum\limits_{x_i \in c_k} \mathrm{d_e}(x_i, \bar{c_k})}{\min\limits_{x_i \notin c_k} \max\limits_{x_j \in c_k} \mathrm{d_e}(x_i, x_j)}. \tag{6.39}$$

- **Negentropy increment** (NI↓) [120]: This is an index based on cluster normality estimation and, therefore, is not based on cohesion and separation estimations. It is defined as

$$\mathrm{NI}(C) = \frac{1}{2} \sum_{c_k \in C} \mathrm{p}(c_k) \log |\Sigma_{c_k}| - \frac{1}{2} \log |\Sigma_X| - \sum_{c_k \in C} \mathrm{p}(c_k) \log \mathrm{p}(c_k). \tag{6.40}$$

where $\mathrm{p}(c_k) = |c_k|/N$, Σ_{c_k} denotes the covariance matrix of cluster c_k, Σ_X denotes the covariance matrix of the whole dataset and $|\Sigma|$ denotes the determinant of a covariance matrix. Although the authors proposed the index as defined above, they later proposed a correction due to the poor results obtained. Nevertheless, we will use the index in its original form since the correction does not meet the CVI selection criterion used for this work.

- **SV-Index** (SV↑) [203]: This ratio-type index is one of the most recent CVIs compared in this work. It estimates the separation by the nearest neighbour distance and the cohesion is based on the distance from the border points in a cluster to its centroid. It is defined as

$$\mathrm{SV}(C) = \frac{\sum_{c_k \in C} \min_{c_l \in C \setminus c_k} \{\mathrm{d_e}(\bar{c_k}, \bar{c_l})\}}{\sum_{c_k \in C} \frac{10}{|c_k|} \sum \max_{x_i \in c_k} (0.1|c_k|)\{\mathrm{d_e}(x_i, \bar{c_k})\}}. \tag{6.41}$$

- **OS-Index** (OS↑) [203]: This is another recent ratio-type index proposed by K. R. Žalik and B. Žalik [203] where a more complex separation estimator is used. It is defined as

$$\mathrm{OS}(C) = \frac{\sum_{c_k \in C} \sum_{x_i \in c_k} \mathrm{ov}(x_i, c_k)}{\sum_{c_k \in C} \frac{10}{|c_k|} \sum \max_{x_i \in c_k} (0.1|c_k|)\{\mathrm{d_e}(x_i, \bar{c_k})\}} \tag{6.42}$$

where

$$\mathrm{ov}(x_i, c_k) = \begin{cases} \frac{\mathrm{a}(x_i, c_k)}{\mathrm{b}(x_i, c_k)} & \text{if } \frac{\mathrm{b}(x_i, c_k) - \mathrm{a}(x_i, c_k)}{\mathrm{b}(x_i, c_k) + \mathrm{a}(x_i, c_k)} < 0.4 \\ 0 & \text{otherwise} \end{cases} \tag{6.43}$$

and

$$\mathrm{a}(x_i, c_k) = \frac{1}{|c_k|} \sum_{x_j \in c_k} \mathrm{d_e}(x_i, x_j), \tag{6.44}$$

$$\mathrm{b}(x_i, c_k) = \frac{1}{|c_k|} \sum_{x_j \notin c_k} \min(|c_k|)\{\mathrm{d_e}(x_i, x_j)\}. \tag{6.45}$$

6.3 Comparing cluster validity indices

As stated before the motivation of this work is the lack of extensive CVI comparisons and the existing comparisons were done using the methodology based on partition similarity measures.

6.3.1 Experimental setup

In this section we describe the experiment performed to compare the CVIs listed in the previous section (Section 6.2). There are many possible experimental designs for such a comparison (Section 6.1.1). Since we want to compare the CVIs in a wide variety of configurations we designed an experiment with several factors. Unfortunately, due to combinatorial explosion we had to limit each factor to just a few levels and this finally led us to an experiment with 6,480 configurations.

Partition Similarity Measures (PSM). As mentioned before, the comparative methodology that we used is a variation of the traditional methodology used to estimate the number of clusters of a dataset but using the partition similarity measures (Section 3.2.3). Therefore, CVIs are used to predict which is the "best" partition in a set of partitions. Being the "best" partition the one defined as the one that is the most similar to the correct one –measured by a partition similarity measure– which is not always the one with the correct number of clusters. In order to avoid the possible bias introduced by the selection of a particular partition similarity measure, we replicated all the experiments using 3 partition similarity measures: Adjusted Rand [95], Jaccard [100] and Variation of Information (VI) [136].

Clustering algorithms. We used 3 clustering algorithms to compute partitions from the datasets: K-means, hierarchical clustering with Ward linkage criteria and hierarchical clustering with average-linkage criteria [104]. These are well known and it is easy to obtain different partitions by modifying the parameter that controls the number of clusters of the output partition. Each algorithm was used to compute a set of partitions with the number of clusters ranging from 2 to $\sqrt{N}$ (rule of thumb exploration limit), where N is the number of objects in the dataset. In the case of the real datasets, the number of clusters in a partition was limited to 25 to avoid computational problems with large datasets.

Databases. As usual, we used several synthetically generated datasets for the CVI evaluation. Furthermore, we also compared them using 20 real datasets drawn from the UCI repository [70]. In any case, it is important to note that results based on real datasets should be analysed with caution since these datasets are usually intended to be used with supervised learning and, therefore, they are not always well adapted to the clustering problem [192]. On the contrary, the synthetic datasets avoid many problems found with real datasets. For instance, in synthetic datasets categories exist independent of human experience and their characteristics can be easily controlled by the experiment designer.

- **Synthetic datasets.** The synthetic datasets were created to cover all the possible combinations of 5 factors: number of clusters (K), dimensionality (dim), cluster overlap (ov), cluster density (den) and noise level (nl). We defined 2 types of overlap: *strict*, meaning that the ov overlap level must be exactly satisfied, and *bounded*, meaning that ov is the maximum allowed overlap.

  A fixed hypercubic sampling window was defined to create all the synthetic datasets. The window was defined by the $(0, 0, \ldots, 0)$ and $(50, 50, \ldots, 50)$ coordinates. In a similar way, a reduced sampling window is defined by the $(3, 3, \ldots, 3)$ and $(47, 47, \ldots, 47)$ coordinates. Then, the centre for the first cluster, c_0, was randomly drawn in the reduced sampling window based on a uniform distribution. The first cluster was created by randomly drawing

Param.	**Value**
n_{min}	100
K	2, 4, 8
dim	2, 4, 8
ov	1.5 (strict), 5 (bounded)
den	1, 4
nl	0, 0.1

Table 6.1: Values of the parameters used in the synthetic dataset generation step.

$n_{min} \times den$ points, where n_{min} defines the minimum number of points in each cluster and den refers to the cluster point density which multiplies the n_{min} to increase it. The cluster points follow a multivariate normal distribution of dim dimensions with mean c_0 and the identity as covariance matrix. All the points located outside were sampling window are removed and new points were drawn to replace them.

The remaining clusters had n_{min} points and this produces a density asymmetry when $den \neq 1$. This occurs because a different number of points will be located in the same approximate volume.

In particular, we built the remaining $K-1$ clusters as follows: if the overlap was *bounded*, the centre of the cluster, c_i, was drawn uniformly from the reduced sampling window. Otherwise, a previously created cluster centre, c_k, was randomly selected and the new cluster centre, c_i, was set to a random point located at a distance of $2 \times ov$ from c_k. In any case, if $\mathrm{d_e}(c_i, c_l) < 2 \times ov \quad \forall c_l \neq c_i$ the cluster centre was discarded and a new one was selected. Once the cluster centre was defined the cluster was built by drawing n_{min} points in the same way as we did for the first cluster.

Finally, when all the clusters were built, $nl \times N'$ points were randomly created following an uniform distribution in the sampling window, where N' was the number of non-noise points in the dataset, $N' = n_{min} \times (den + K - 1)$.

The values of the parameters used to create the synthetic datasets are shown in Table 6.1, making 72 different configurations. As we created 10 datasets from each configuration we used 720 synthetic datasets. Multiplying this value by 3 partition similarity measures and 3 clustering algorithms we obtain the 6,480 configurations previously mentioned. Notice that the n_{min} parameter ensures that every cluster is composed of at least 100 objects.

Figure 6.1 shows an example of 4 two-dimensional datasets we used. In the figure we can see how the different values of the generation parameters affect the point distribution in the datasets. Figure 6.1a shows a dataset with 4 clusters, with no cluster overlap, no noise and no density asym-

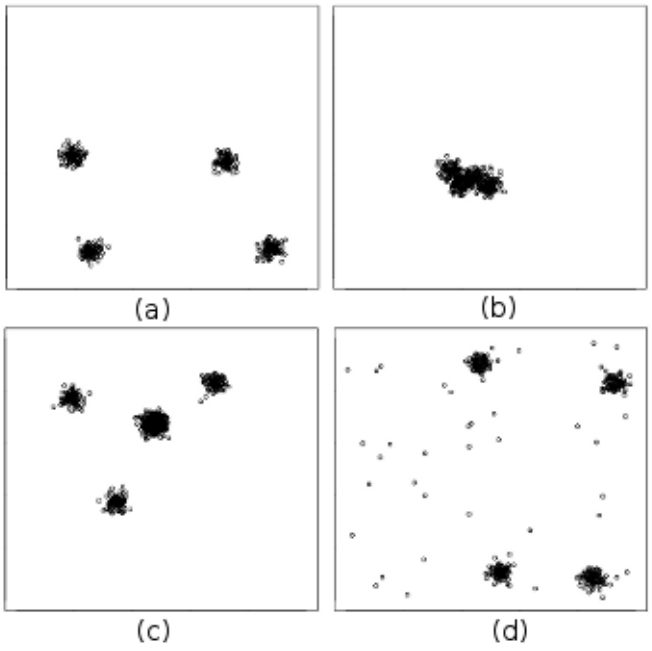

Figure 6.1: 2-dimensional plots of 4 synthetic datasets used in the experiment. (a) shows a "neutral" dataset with no cluster overlap, no density asymmetry and no noise. (b) shows a similar dataset with high cluster overlap. (c) shows a dataset with cluster density asymmetry. (d) shows a dataset with noise.

metry. The other three plots show datasets with similar characteristics except for overlap, density and noise parameters.

- **Real datasets.** The 20 real datasets and their main characteristics are shown in Table 6.2. In this case the experiment is based on 180 configurations –20 datasets, 3 algorithms and 3 partition similarity measures.

Including synthetic and real datasets, and taking into account the different numbers of partitions computed for each dataset, each of the 30 CVIs was computed for 156,069 partitions.

6.3.2 Results and Analysis

One of the goals of this work was to present the results in such a way that other researchers can focus on the particular configurations they are interested in. However, the vast amount of results obtained can not been shown in this section. Therefore, here we focus on the overall results; drawing some important

Dataset	No. of objects	Features	Classes
Breast tissue	106	9	6
Breast Wisconsin	569	30	2
Ecoli	336	7	8
Glass	214	9	7
Haberman	306	3	2
Ionosphere	351	34	2
Iris	150	4	3
Movement libras	360	90	15
Musk	476	166	2
Parkinsons	195	22	2
Segmentation	2 310	19	7
Sonar all	208	60	2
Spectf	267	44	2
Transfusion	748	4	2
Vehicle	846	18	4
Vertebral column	310	6	3
Vowel context	990	10	11
Wine	178	13	3
Winequality red	1 599	11	6
Yeast	1 484	8	10

Table 6.2: The characteristics of the real datasets drawn from the UCI repository.

conclusions. However, all the detailed results are available in the web[1].

In this section we first describe the results obtained for the synthetic datasets and then the results for the real datasets are described. Finally, we present a brief discussion on the use of statistical analysis in clustering and we show the conclusions we drew by applying some statistical tests to the results.

6.3.2.1 Synthetic datasets

The overall results for the synthetic datasets are shown in Figure 6.2. The figure shows the percentage of correct guesses (successes) achieved by each CVI, which are sorted by the number of successes. Notice that this percentage refers to the 6,480 configurations. The graph shows that Silhouette achieves the best overall results and is the only one that exceeds the 50% score. DB* and CH also show a good result, with a success rate beyond 45%.

It is also noticeable that in most cases variations of a CVI behave quite similarly; they appear in contiguous positions in the figure. The clearest cases are the generalized Dunn indices that use Δ^3 as cohesion estimator –gD33, gD43 and gD53– and the graph theory based Dunn indices –D^{MST}, D^{RNG} and D^{GG}.

[1] `http://www.aldapa.eus/res/cvi/`

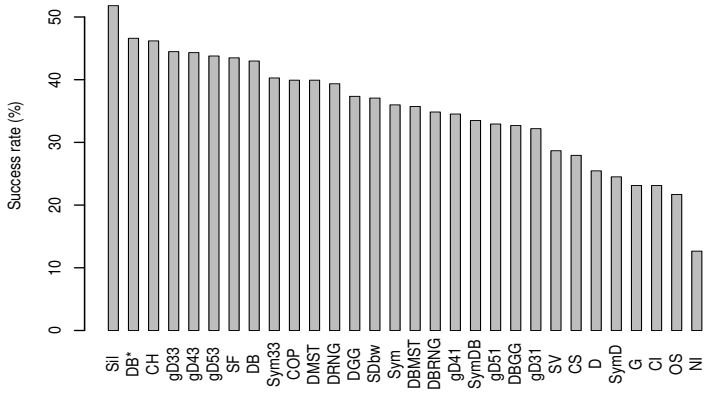

Figure 6.2: Overall results for the experiment with synthetic datasets.

Next we will show a similar graph for each experimental factor. In this case the value of each CVI is shown for each value of the analysed factor. We will keep the CVI order shown in Figure 6.2, so a decreasing graph will denote that the analysed factor does not change the overall ranking.

First of all, let us focus on the graph corresponding to the partition similarity measure. As a reminder, this is a parameter of the validation methodology we used (see Section 6.3.1). In Figure 6.3 we can see that the selected partition similarity measure does not affect the results. This result suggests that the CVI comparison is not affected by the particular selection of a parameter of the evaluation methodology and, therefore, we can be confident of the results. Also notice that although Adjusted Rand and Jaccard show very similar results the use of the VI partition similarity measure produces slightly higher success rates.

In the following figures (Figures 6.4, 6.5, 6.6, 6.7 and 6.9) a similar breakdown can be found with regard to the characteristics of the datasets.

Number of clusters. In Figure 6.4 we can see how the number of clusters of the datasets affects the results. As expected, all the CVIs obtain better results with fewer clusters –average result for $k = 2$ drops from 50.2% to 30.7% ($k = 4$) and 24.8% ($k = 8$). We can also see that for high values of this parameter the differences between the CVIs are reduced. Furthermore, some indices, such as COP, show little sensitivity to this parameter making it the best CVI for $k = 8$.

Dimensionality. With respect to dimensionality (see Figure 6.5), the results show that the difficulty imposed by an increment in the number of dimensions does not severely affect the behaviour of the CVIs –except for NI. Moreover, some indices, such as Sym, show a better behaviour for datasets with higher

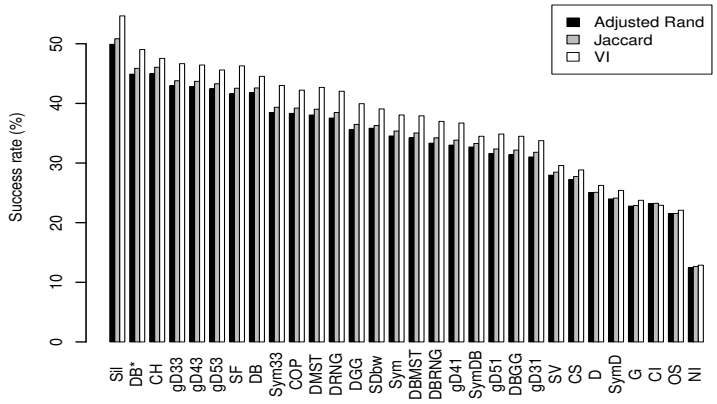

Figure 6.3: Results for synthetic datasets broken down by partition similarity measure.

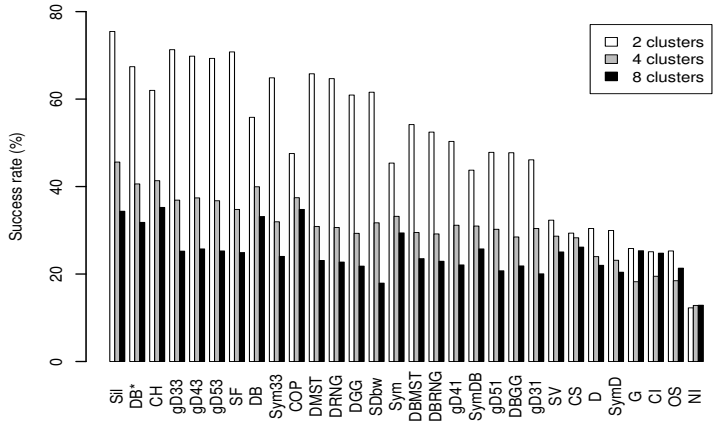

Figure 6.4: Results for synthetic datasets broken down by number of clusters.

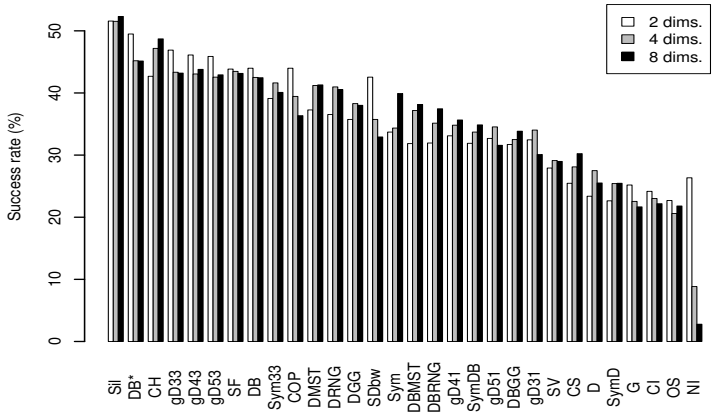

Figure 6.5: Results for synthetic datasets broken down by dimensionality.

dimensionality. Silhouette also achieves the best results for every value analysed for this parameter.

Overlapping. Let us now focus on the results shown in Figure 6.6. This graph shows that, as expected, datasets with no overlapping clusters lead to better CVI success rates. The average result decreases from 52.9% to 17.6% when well separated clusters are replaced by overlapped clusters. The graph also shows that although this parameter does not severely affect the overall trend, some CVIs are more hardly affected by cluster overlap, e.g. DB, COP and SymDB. Some others, such as G, CI and OS, seem not to work at all when clusters overlap.

Density. With respect to the density of the clusters, Figure 6.7 shows that having a cluster four times denser than the others, does not severely affect the CVIs. It seems that the best behaving indices are quite insensitive to this parameter while the rest show a better result when density heterogeneity is present. Silhouette is again, clearly, the CVI showing the best results.

Noise level. Was the last dataset characteristic analysed in this work, it had a major impact on the scores of the CVIs (Figure 6.8). In fact, the scores in noisy environments are on average 3 times lower than they are when no noise is present. Silhouette, and mostly SDbw, are the main exception to this rule since they show similar score values for noisy and noiseless environments. Besides, the overall trend is not always followed and CH is the CVI that achieves the best results when no noise is present.

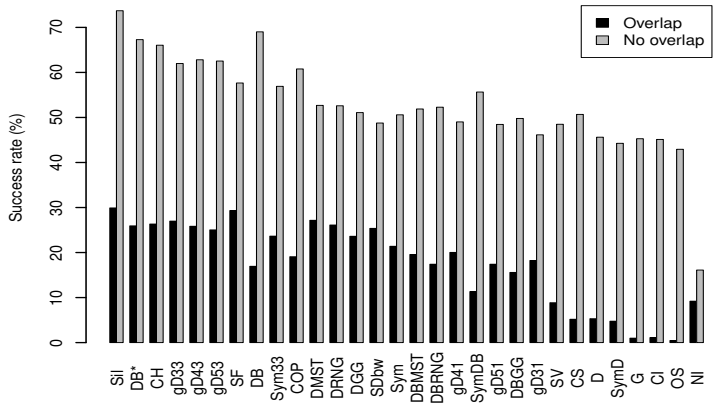

Figure 6.6: Results for synthetic datasets broken down by cluster overlap.

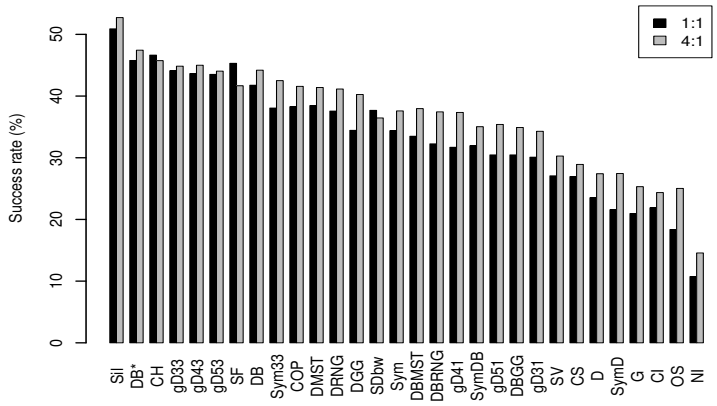

Figure 6.7: Results for synthetic datasets broken down by density.

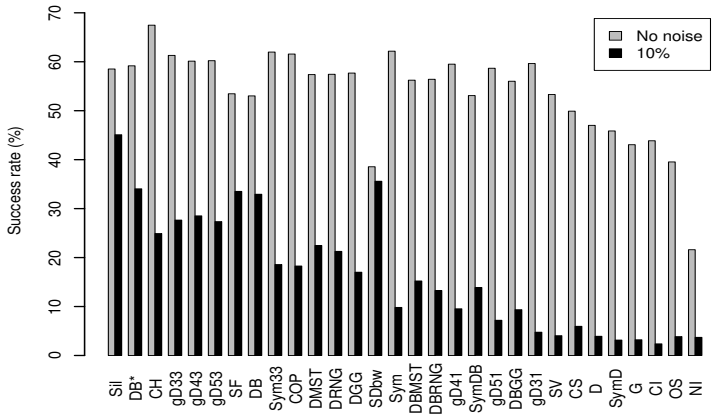

Figure 6.8: Results for synthetic datasets broken down by noise.

Algorithm. Finally, Figure 6.9 shows how the clustering algorithm used in the experiment affects the scores of the evaluated indices. Although we cannot find a clear pattern, it seems that the overall comparative results are not severely affected since the decreasing pattern of the graph is somehow maintained. Most of the CVIs obtain their worst results for the K-means algorithm, but there are some exceptions where the opposite holds –COP, G, CI and OS are the most remarkable examples. Silhouette is again the one achieving the best results for hierarchical algorithms, but CH is the best CVI when K-means is used as clustering algorithm.

6.3.2.2 Real datasets

In this section we show the results obtained for 20 real datasets following a similar style to the one we used for synthetic datasets. Obviously, since we do not have control over the dataset design the number of experimental factors is reduced to two: partition similarity measure and clustering algorithms.

First, in Figure 6.10 we show the overall results for real datasets. A quick comparison to the overall results for the synthetic datasets (Figure 6.2) shows that the results are qualitatively similar. Most of the CVIs that obtained worst results with synthetic datasets are also in the tail of the ranking in the figure for real datasets. Focusing on the head of the ranking we can see that the generalized Dunn indices –gD33, gD43 and gD53– remain in a similar position; SF, graph theory based Dunn and COP improve their position; and Silhouette, DB* and CH go down the ranking. Considering these results we can say that the mentioned generalizations of the Dunn index show the steadiest results.

Returning to the 2 experimental factors involved in the experiments with real datasets, in Figure 6.11 we show the results broken down by partition similarity

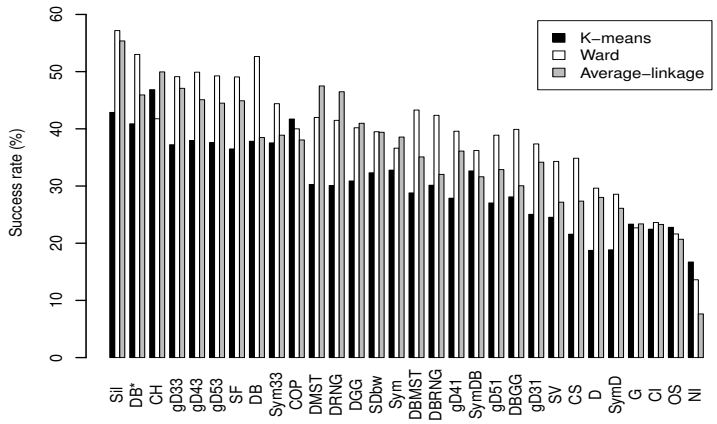

Figure 6.9: Results for synthetic datasets broken down by clustering algorithm.

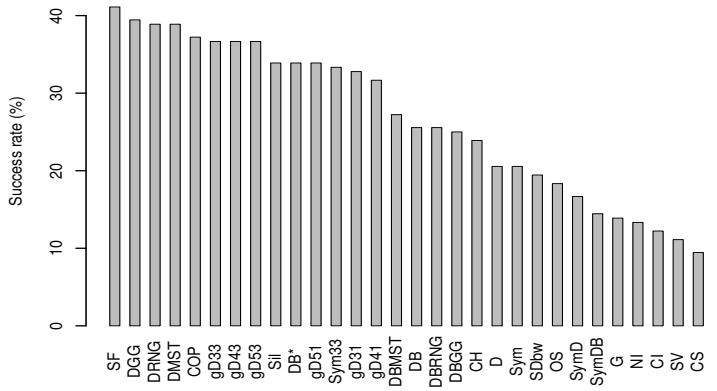

Figure 6.10: Overall results for real datasets.

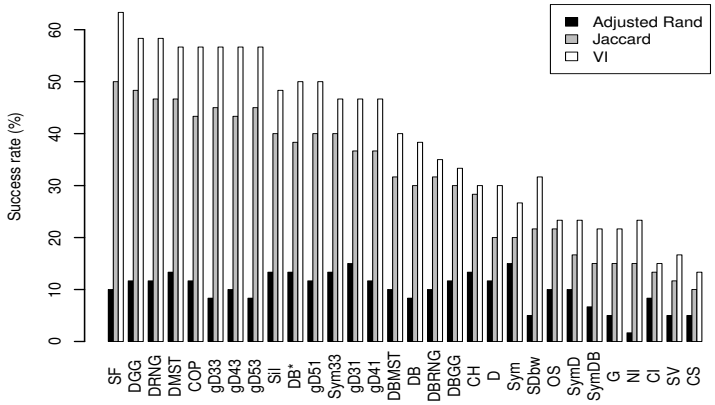

Figure 6.11: Results for real datasets broken down by partition similarity measure.

measure. We can see that in this case it seems that the partition similarity measure selected can affect the results. Although Jaccard and VI follow the overall pattern the Adjusted Rand index does not. Furthermore, it is clear that in every case the average scores are much lower when Adjusted Rand is used, dropping from 39.1% (VI) or 31.1% (Jaccard) to 10.0%.

6.3.2.3 Statistical tests

Although the assessment of the experimental results using statistical tests is a widely studied technique in machine learning, it is rarely used in the clustering area. Among the works cited in Section 6.1.1 just Dubes [59] used a statistical test to assess the influence of each experimental factor on the results obtained. However, in our case we focused on checking whether the differences observed between CVIs were statistically significant or not.

We argue that an effort should be made by the clustering and statistics communities to adapt these tools to clustering and effectively introduce them in the area. These types of tests would be more important in extensive comparative works such as the one described in this section. Therefore, we propose a possible direct adaptation of a comparison method used in supervised learning. This method was chosen due to the proximity of the supervised learning area to clustering and because the use of statistical tests in this area has been widely studied [53, 56, 73].

We based our statistical method on a common scenario in supervised learning where classification algorithms are compared. In this case it is usual to run the algorithms on several datasets and to compute a "quality" estimate, such as the accuracy or the AUC value, for each algorithm and database pair. A usual

approach is to test the quality values achieved by all the algorithms for each dataset independently [56]. However, Demšar [53] argued that a single test based on all the algorithms and all the datasets is a better choice. One of the advantages of this method is that the different values compared in the statistical test are independent, since they come from different datasets.

We adapted the method proposed by Demšar [53] and subsequently extended by García and Herrera [73] to CVI comparisons. In brief, we simply replaced the classification algorithms by CVIs. However, this is not enough, since in our experiments we obtained a Boolean value for each CVI-configuration pair instead of a "quality" estimate. Moreover, the configurations we obtained by varying the clustering algorithm and partition similarity measure are based on the same dataset, so it can be argued that they are not sufficiently independent.

Our solution was to add for each dataset the number of successes each CVI obtained for each clustering algorithm-partition similarity measure pair. Moreover, in order to obtain a more precise estimate, we also added the number of successes obtained in every run –remember that we created 10 datasets for each combination of dataset characteristics. We thus obtained 72 values ranging from 0 to 90 for each CVI, that gave us a "quality" estimate for independent datasets. Finally, we applied the statistical tests with no further modifications.

The tests we used were designed for comparisons of multiple classifiers (CVIs) in an all-to-all way. We used the Friedman test to check if any statistical difference existed and the Nemenyi test for pair-wise CVI comparison [53]. Furthermore, we performed additional pair-wise CVI comparisons with the Shaffer test as suggested by García and Herrera [73]. In both cases we performed the tests with 5% and 10% confidence level.

The main conclusion obtained by applying the above tests is that there are undoubtedly statistically significant differences between the 30 CVIs, as the Friedman test categorically shows with a p-value on the order of 10^{-80}. All the performed pair-wise comparisons show a very similar result, so in Figure 6.12 we only show the results for the most powerful test that we performed –Shaffer with a confidence level of 10%.

Since the used statistical tests are based on average rank values, Figure 6.12 shows all the CVIs sorted by average rank. The results are very similar to those based on average scores (Figure 6.2), but there are a couple of differences that should be underlined. First of all, the CVI order slightly changed, but most of the movements occurred in the central part of the ranking. Second, the CVIs formed quite well separated groups. In the first group there are 10 indices with an average rank between 9 and 13. Taking into account variations of a CVI as a single one, the group contains 6 indices: Silhouette, Davies-Bouldin, Calinski-Harabasz, generalized Dunn, COP and SDbw. There is also a crowded central group with 14 CVIs and average rank between 14 and 17; and finally, a group of 6 indices with average rank between 19 and 23.

The bars in Figure 6.12 group the indices that do not show statistically significant differences. The highly overlapped bars difficult the task of drawing categorical conclusions, but on the following we resume the information in the graph and remark the most interesting points:

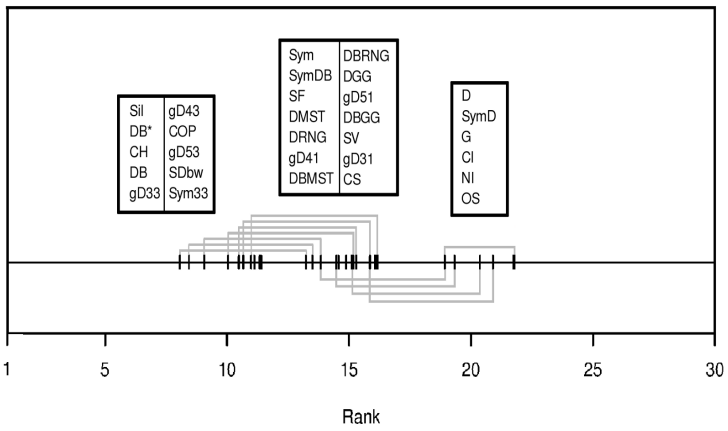

Figure 6.12: Results for Shaffer test with a significance level of 10%.

- No significant difference exists between CVIs in the same group.
- All the CVIs in the first group perform significantly better than the CVIs in the third group.
- The best behaving CVI, Sil, obtains significantly better results than all the CVIs in the second group, except Sym.
- All the CVIs in the second group, except Sym and SymDB, have no statistically significant differences with at least one CVI in the third group.

In conclusion, the data do not show sufficiently strong evidence to distinguish a small set of CVIs as being significantly better than the rest. Nevertheless, there is a group of about 10 indices that seems to be recommendable and Silhouette, Davies-Bouldin* and Calinski-Harabasz are in the top of this group. We also performed statistical test for real datasets, but no CVI can be considered significantly better than the others in any case.

6.4 Application of CVI-s in elevated number of clusters

In this section we are going to analyse the behaviour of the CVIs in a context where the number of clusters is high. To carry out this analysis we applied the CVIs to one of the web mining environments described in the Chapter 5 where web usage mining was performed using the information of webserver log files to predict the navigation of the new users. Specifically, we used logs from the tourism website called Bidasoa Turismo.

As clustering algorithms are appropriate to group into the same segment users with similar navigation characteristics they are very used in web usage mining. However, the diversity of the types of users in these contexts is huge and as a consequence, the partitions generated have many clusters.

Previous works found in literature compare some CVIs with the aim of obtaining some general conclusions about their behaviour [137, 13]. Nevertheless, all of them have performed the evaluation in databases with few clusters (up to 6 or 8 in general); being 15 the greatest number of clusters explored [13]. To our knowledge there is no study on how CVIs deal with large databases where the data require to be partitioned in more clusters. In this context, they seem to have some non analysed problems.

In this section we present a preliminary analysis of the behaviour of a set of CVIs in such context, we propose a procedure based on some of those CVIs to select a good quality partition.

6.4.1 Application context of CVIs

The usage mining application used in this section combines clustering techniques (PAM [111] explained in Section 3.2.2.2) with sequential pattern mining techniques (SPADE1, see Section 3.3.1.1) to build navigation profiles from preprocessed logs in a tourism website. And then, based on these profiles, the system proposes links (links in this case to compute prediction ability) to new users navigating in the website.

The core part is the clustering process where we group into the same segment users that show similar navigation patterns. We used PAM clustering algorithm (see Section 3.2.2.2), and as many other clustering algorithms, it requires the K parameter to be estimated.

Before starting with the pattern discovery and analysis phase we performed a preliminary analysis to decide whether or not any structure existed in the data. For that, we analysed the distance to the nearest sequence for each example and ordered them in increasing order. We concluded that the database was disperse; many of the sequences were not similar to any other sequence since their distance was very big. This suggest that it would be very difficult and not natural to find structure in the complete database because clustering algorithms generally partition all the data in a concrete number of clusters what makes very probable to be obliging to group together patterns that are very far away from each other.

The results of the described analysis led us to divide the database in two parts: (1) the one used for profiling contains the closer sessions (12,238 sequences), the sequences which have the nearest one at a distance smaller or equal than 0.5 and (2) the one with the most distant examples, containing the remaining sessions.

Index		Proposed in	Proposed by
DunnMST (D)	↑	1997	Pal and Biswas [149].
Gamma (G)	↓	1975	Baker and Hubert [24].
C-Index (CI)	↓	1976	Hubert and Levin [96].
DaviesMST	↓	1997	Pal and Biswas [149].
Davies*	↓	2005	Kim and Ramakrishna [113].
Silhouette	↑	1987	Rousseeuw [170].
CS	↓	2004	Chou, Su and Lai [45].
Score function	↑	2007	Saitta et al. [173].
COP	↓	2010	Gurrutxaga et al. [77].
SV	↑	2011	Žalik and Žalik [203].
OS	↑	2011	Žalik and Žalik [203].

Table 6.3: List of explored CVIs with a large number of clusters.

6.4.1.1 Selected CVIs

As no previous study of CVIs in this context was found, from the CVIs evaluated in Section 6.3 we selected the ones we found implementations that could be directly applied to distance matrices (summarized in Table 6.3). We adapted some of them replacing the centroids by medoids (since centroid cannot be calculated in sequential representation). Note that some of the indices require data points for their calculations and having the web navigation sessions represented as URL sequences do not allow this possibility, thus, we focused on those indices that work only with distance matrices.

6.4.2 Experimental setup

In this subsection, the selected evaluation methodology and the system's parameters are summarized:

- **Log preprocessing.** The initial preprocess was described in Section 5.2.2 and we obtained a database with 21,916 sequences with in average 7.7 clicks length. After discarding distant examples we ended with 12,238 sequences of 6.0 clicks in average.

- **Validation strategy.** To evaluate the system we split the database and simulated the link prediction environment. More precisely, we applied a hold-out strategy and divided the database according to temporal criteria (to simulate a real situation) into three parts: training data (8,567 examples, 70%), validation data (2,447 examples, 20%) and test data (1,224 examples, 10%). We used the training data to generate the data partitions and corresponding user profiles, the validation-set to tune the parameter K of the clustering algorithm and the test data to estimate the prediction capacity.

- **Evaluated navigation stage.** The first 25% of the new sequences, that is, more or less 1.5 clicks.
- **Validation metrics:** We evaluated the quality of the partitions according to the precision (pr), recall (re) and F-measure (Fm) metrics.
- **System parameters:**
  - **Explored range of K.** We experimented with nine values: 10, 15, 20, 25, 50, 75, 100, 150 and 200. We explored beyond the usual exploration limit $\sqrt{n}$ (being n the number of training examples), to explore the CVIs behaviour in a wider range.
  - **Profiling the clusters.** Based on the previous experience we fixed the minimum support in sequential pattern mining to 0.2.
  - **kNN linkage criteria.** Edit distance between the test sequence and the cluster by medoid-linkage criteria.
  - **Number of selected profiles.** Based on the previous experience we used the two nearest medoids (2-NN) to select the nearest clusters.
  - **Number of proposed links.** 3 (a trade-off between not too many links and not too few).

6.4.3 Results and Analysis: selection of the best partition

The first aim of our study was to determine the best partition and to analyse the behaviour of the CVIs in this context. We applied PAM algorithm to the training data, we generated nine partitions varying K from 10 to 200 clusters, and finally, we calculated the values of the 11 CVIs for each of the partitions.

The CVI evaluations we found in bibliography were for partitions with few clusters (below 15) and the increase of the number of groups could affect or even invalidate the effect of one or both components, intra-cluster distance or inter-cluster distance, taken into account in the indices; i.e. the increase in the number of clusters might affect as much as the quality of the partition to the values obtained for the CVIs.

Based on the trends of the values obtained by the CVIs in this experiment we could divide them into two main groups:

- A group of 6 CVIs that fluctuated and, as a consequence, did not show any coherent behaviour or did not give any insight about the characteristics of the best partition.
- A group of 5 CVIs that showed a trend and, as a consequence, made possible the analysis of the quality of the partition based on CVIs. The CVIs in this group were DunnMST, C-Index, OS, COP and Gamma.

The CVIs showing a trend, improved their value as K increases, what would suggest us to generate new partitions with larger number of clusters. This could

K	D↑	CI↓	OS↑	COP↓	G↓
15	0.000000	0.008569	-0.007116	-39.469200	0.004325
20	0.000000	-0.005739	-0.002989	-25.098500	-0.007791
25	0.000000	-0.005433	-0.003357	-21.061840	-0.006066
50	0.002103	-0.003017	-0.001871	-6.977660	-0.003889
75	0.000667	-0.000280	-0.001148	-2.978823	-0.000534
100	0.000000	0.000365	-0.000586	-1.666815	-0.000170
150	0.000000	-0.000326	-0.000447	-0.850714	-0.000424
200	-0.000462	-0.000054	-0.000303	-0.442405	-0.000212

Table 6.4: Derivatives of the values obtained for the CVIs showing a trend. The best value for each CVI is shaded.

be considered a bias of the indices but a further analysis showed that there is a point where the improvement of the index value (increment or decrement depending on the type of index), suffers a change in slope which could be considered to be the point obtaining a trade-off between the quality of the partition and K. We therefore hypothesised that this trade-off point could help detecting if the improvement in the value of the CVI was due to the quality of the partition or due to the bias of the index. This bias could be generated because the inner components' weights (the inter-cluster distance and intra-cluster distance) are unbalanced.

In order to find the trade-off point we proposed to analyse the derivative of the values obtained for each of the CVIs in the second group and to use as stop criteria the change of magnitude order of the decrement or increment on the obtained index value. When the change happens, the previous K will be considered the one obtaining the best partition. Table 6.4 shows the values of the derivatives for the 5 indices, where the values selected according to the stop criteria are shaded.

Based on the proposed rule, 3 of the CVIs, DunnMST, C-Index and Gamma, proposed as best partition the one with 50 clusters whereas OS proposed the one with 75 clusters and COP the one with 25 clusters. We decided to use the most voted strategy which would suggest that the best number of clusters is near 50 (mode value).

6.4.4 Results and Analysis: validating the best partition

The aim of this subsection is to evaluate the generated system and to confirm that the selected partition was the best one according to the obtained results. With this aim, we evaluated profiles generated for each of the partitions generated with the training sample (10, 15, 20, 25, 50, 75, 100, 150 and 200 clusters), according to the test sample which would be equivalent to new users navigating in the website.

Table 6.5 shows average results for precision (pr), recall (re) and F-measure (Fm) obtained with the test-set for different values of K. The second column,

Clusters (K)	#URLs	PrMo	ReMo	FmMo	Pr	Re	Fm
10	2.9	0.341	0.277	0.326	0.232	0.237	0.233
15	3.0	0.434	0.378	0.421	0.300	0.323	0.305
20	2.8	0.460	0.380	0.441	0.321	0.325	0.322
25	2.8	0.460	0.381	0.441	0.317	0.321	0.318
50	3.0	0.551	0.496	0.539	0.383	0.416	0.389
75	2.9	0.523	0.460	0.509	0.340	0.359	0.343
100	2.9	0.545	0.487	0.532	0.353	0.376	0.357
150	2.9	0.573	0.511	0.559	0.363	0.385	0.367
200	2.9	0.554	0.494	0.541	0.346	0.366	0.349

Table 6.5: Precision, recall and F-measure values obtained when evaluating profile concordance (left) and link prediction capacity (right) with test examples. Coloured K =50 the value selected by the CVIs.

#URLs, shows the average number of links proposed in each profile and it shows that, although the number of proposed URLs was fixed to 3, for some of the cases fewer links were generated. The remaining columns are divided into two parts, the ones that show values for the evaluation of the concordance of the profiles (prMo, reMo and FmMo) and the ones related to the evaluation of the system for link prediction (pr, re and Fm). The solutions obtained with the partitions selected according to the CVIs appear shaded whereas the best solution according to the values of the performance metrics appear underlined in the table.

The results in Table 6.5 show that the profiles generated by the system have a considerable concordance with the profile of new users. Moreover, the values of the performance metrics are greater than in other state of the art publications [177]. The solution selected using the CVI based procedure (K =50) is the break even point in this experiment; more than 50% of the proposed links are hits, and, moreover, nearly 50% of the places navigated by the new users are guessed by the system. Although with slightly lower values, the system shows similar behaviour when it is used for link prediction, so, it will help new users to achieve their objectives faster. We can therefore state that the proposed CVI based rule seems appropriate because the best results for link prediction are obtained with the selected partitions and in the case of the concordance of profiles, the difference of those partitions with the ones obtaining the best values is not more than 0.02.

6.5 Summary

With regard this chapter shows, as far as we are concerned, the most extensive CVI comparison using the partition similarity measure based evaluation methodology proposed by our research group [80]. This methodology validates the clustered partition comparing it with the correct one by using partition sim-

ilarity measures instead of just comparing it with the real number of clusters. Moreover, the results were adjusted to apply Friedman test with Nemenyi and Shaffer post-hoc procedures to find significant differences between CVIs. For this comparison 30 CVIs were selected and redefined using the same notation.

Since we wanted to compare the CVIs in a wide variety of configurations we designed an experiment with several factors. Three partition similarity measures: Adjusted Rand, Jaccard and Variation of Information. Three clustering algorithms: K-means, SAHN with Ward linkage criteria and SAHN with average-linkage criteria. 20 real databases and for synthetic databases all combinations of the following 5 factors were created: number of clusters, dimensionality, cluster overlap, cluster density and noise level. This finally led us to an experiment with 6,480 configurations.

We observed that some CVIs appear to be more suitable for certain configurations, although the results were not conclusive. Furthermore, the overall trend never changed dramatically when we focused on a particular factor. With regard to the experimental factors, noise and cluster overlap had the greatest impact on CVI performance. A statistical significance analysis of the results showed that there are 3 main groups of indices. We also analysed the performance of the 30 CVIs in 20 real word databases, aiming to similar conclusions.

Moreover, we analysed the behaviour of CVIs in a real web mining application with elevated number of clusters in which they tend to be unstable.

Due to the application context, where sequences were used, no all the CVIs evaluated in the previous phase were valid. We adapted 11 usable indices and we selected the five most appropriate ones because not all of them showed a coherent behaviour. We proposed a procedure which selects the best partition analysing the slope of the CVI values to select the best partition indicated by a CVI and the implemented a voting strategy to select the best partition. A performance evaluation of the system showed that the best partition selected according to this procedure was one of the best performing ones.

Part IV

Conclusions

Chapter 7

Conclusions

This thesis makes contributions in **modelling behaviours found in different types of data acquired from the Internet and in the field of clustering evaluation**. Two different types of Internet data have been processed, on the one hand, internet traffic with the objective of attacks detection and on the other hand, web surfing activity with the objective of web personalization, both being data of sequential nature. To model these data and achieve the objectives, machine learning techniques have been applied, mostly unsupervised ones. Within unsupervised learning techniques, contributions are made in cluster evaluation, in order to make the selection of the best partition in clustering problems easier.

7.1 Conclusions in network security

The internet traffic was modelled for building a **network Intrusion Detection System (nIDS)** that works by scanning the network traffic and is able to automatically detect intrusions. In this context, two types of intrusions can be found, flood attacks which need to send a large number of packets in a short time, thus, they are easy to detect and **non-flood attacks** in which the intruder, who sends very few packets, takes the complete control of the attacked system. Flood attacks are easy to detect by simple systems that scan network traffic analysing the header part of the network packages. However, non-flood attacks are not easy to detect, it is necessary to analyse **the content part of the network packages, called payload**. The features of the payload vary depending on the type of network connection and service. As a consequence, most payload based IDSs we can find in bibliography are service-specific. Therefore, since flood attacks can be detected easily and non-flood attacks have the most catastrophic consequences, in this dissertation we focused on non-flood attacks. We proposed a **service independent nIDS system** that is able to work in any environment by using unsupervised anomaly detection techniques and the payload or information transferred in network connections. Moreover,

since most of the previous works were based on packet header analysis and service and context specific payload analysis, our experiments showed that general payload processing and analysis can also be effective to detect non-flood attacks.

In order to obtain the best nIDS system, a combination of some of the approaches is usually the best option. For example, a flood detecting firewall could first filter most flood attacks; a signature-based IDS could then be used to remove the known attacks and unsupervised anomaly detection could finally focus on detecting the unknown attacks.

The first step in this process was to **build and publish the gureKDDCup database** with normal and attack connections. This database includes the 41 attributes of the KDDCup99 plus the connections' payloads. KDDCup99's attributes were composed by header-based variables and some content-based specific variables extracted applying ad-hoc signature finding procedures in the payload. In this context there was a **lack of databases to analyse the payload in a generic way, context-independently**. As a consequence, we generated the **gureKDDCup database which included payloads** to facilitate the research in general payload processing for intrusion detection.

Once we prepared the suitable database to work with, we proposed an **automatic and general system that processes payload for non-flood attack detection**. We proposed different generic payload processing approaches, on the one hand, **three ways to represent the payload** (sequence, hash and histogram), and on the other hand, **two payload modelling approaches for anomaly detection** (based on fixed-width clustering algorithm and on Probabilistic Suffix Tree (PST)). We tested them and compared them with several systems such as, the system built using header-based variables and the system built using experts generated and context dependant content-based variables. Up to a point, it was able to differentiate between normal traffic and 27 types of intrusions. The most promising techniques showed to be the histogram (or 1-gram) based ones and the PST based one. Another noteworthy conclusion is that we **overcame the results** of the intrusion detection rates **of the system based on ad-hoc content-based variables**. Therefore we claim that payload analysis can be used in a general manner, with no service specific modelling, to detect non-flood attacks in network traffic.

During the generation of this system, other mentionable contributions have been done. The **fixed-width clustering algorithm** was able to detect attacks by scoring connections as anomalous. Also, we proposed two **new distances** for payload comparison and different simple payload vector representations. Likewise, the use of the **PST tree structure** to model payloads by carrying out anomaly detection was tested with satisfactory results.

These techniques showed to be efficient discriminating normal connections from attacks and also discriminating between different types of attacks. Hence, since **each technique was more appropriate than others to detect specific types of attacks, we decided to combine them** to take advantage of this specialization feature. The results obtained showed that it is possible to integrate the knowledge of **different payload-based techniques** and the **packet-header-based technique** and improve the previous results. All the

tested combinations contributed to increase the overall AUC values. Above all, the **minimum AUC value increased in all cases** and this means that the system using this nIDS system will be better protected against any type of attack.

7.2 Conclusions in web mining

The next application is the **analysis of the web surfing activity** for **web personalization**. In this context, generic and non-invasive systems to extract knowledge for web personalization were proposed just using the information stored in **webserver log files**. Contributions were done in two senses: problem detection and link suggestion.

7.2.1 Problem detection

After preprocessing server log files, we extracted a **list of attributes that describes each session**. After that, we clustered the sessions with the K-**means clustering algorithm**. The analysis of these clusters clearly shows that, for the three databases used in this work (SDSC [175, 176], EPA [62, 63] and NASA [144, 145] from The Internet Traffic Archive [51]), some user profiles can be detected using just the information in the webserver log files. With these results we validated the list of extracted attributes and the generated partition of clusters. More precisely, **different types of profiles were detected**. For instance, we could find users that access the web slowly in all of them. This could happen because these users have difficulties to navigate or it could also be due to the high interest they have in the pages they are visiting. In this case, the addition of very simple web structure and content information such as the type of accessed pages (auxiliary or content page for example) could help disambiguate both situations; showing accesses to auxiliary pages would clearly show problems when accessing to the web.

Therefore, we concluded that the web usage mining process could clearly be enriched using some more information such as web structure and web content information, unfortunately this was impossible because the information was not available in The Internet Archive. However, these experiments helped us identify the type of information that will be useful in a data mining process to identify different navigation profiles and as a consequence, contributes going a step further towards web personalization.

7.2.2 Link suggestion

In the context of link suggestion, we were definitely able to **build a general and non-invasive system** that, without any effort from the users, it was able to **automatically generate link proposals** that will be helpful to make the navigation of new users in the web more comfortable and efficient. And as a

consequence, we can claim that the machine learning techniques we used and their combination seems to be appropriate.

The proposed system was first generated for NASA log files [144, 145] and then, was improved using the log files of the Bidasoa Turismo website[1] (BTw). However, **this system could be extended to any other environment** since it uses the minimum information stored in any webserver (Common Log Format (CLF) [193]).

The designed system, after preprocessing the log files, identified different groups of users, built the corresponding profiles and automatically generated useful link proposals for new users, to adapt the user navigation scheme, so that their browsing experience was improved. In next paragraph the exact techniques used in this web personalization process are summarized.

We preprocessed the data to identify users and sessions on the one hand, and prepared it so that it could be used with machine learning algorithms. Afterwards, we applied a clustering algorithm to the training data to discover groups of users with similar interests or navigation patterns. More precisely, we used **two clustering algorithms**: hierarchical agglomerative clustering (SAHN, Sequential Agglomerative Hierarchical Non-overlapping) with different techniques to extract a partition from the hierarchy of clusters and Partition Around Medoids (PAM) clustering algorithms, both able to work with sequences. Once the groups of users were identified, we used **sequential mining technique** (SPADE, Sequential Pattern Discovery using Equivalence classes) to discover the profiles associated to each of the clusters, that is, the links that will be proposed to new users. **To evaluate the models, a link prediction environment was simulated**. We evaluated different configurations of the system at different stages of the user navigation, including the cold-start or 0-day problem (very early stages of the navigation).

We obtained that 53.5% (in NASA) and 36.6% (in BTw) of the suggestions were used, with a little drop in precision in the early stages of the navigations. Besides, our experiments showed that **diversification of profiles helps in link generation for suggestion**. As the exact prediction of the URL was too strict we proposed relaxing that condition by matching two URLs if they were in the same web zone or they were similar from the content point of view. In this case, we achieved up to 92.4% of precision.

During the generation of this system, other mentionable contributions were done. We added thematic information to each URL based on the website contents (by using topic modelling techniques) or based on the URL string (if the URL string encodes thematic information) to be used in the sequence alignment used in the clustering process. Likewise, we proposed a method to enrich the suggested links with semantically similar ones. And we applied a SEP/COP technique proposed by our research group [77] which is a flexible and parameter free partition extractor technique from hierarchies of clusters.

Besides, the **web usage based system proposed** so far [11] was compared

[1] `http://www.bidasoaturismo.com`

with a **proposal of a research group of the University of Patras (Greece)** [58]. Our system uses a **PAM clustering algorithm with edit distance** to group similar users, **SPADE to generate user profiles** and kNN (k-nearest neighbours algorithm) for link prediction. Whereas the greek system uses **SAHN clustering algorithm with a combination of a global and a local sequence alignment method** to group similar users, **multiple sequence alignment (MSA) to generate user profiles** and **suffix trees** for link prediction.

We first focused on small variations of the two previous approaches analysing the effects of changing the sequence alignment strategy, the clustering algorithms, the way to weigh and exploit the suffix trees, and analyse the influence of including semantic information in the representation of user sessions. The inclusion of semantics generated simpler models almost without loss of prediction ability. Moreover, we evaluated different proposals for failure functions of suffix trees, the best one being to go to the next longest suffix combined with going to the 1-length suffix in the tree structure. Therefore, we selected the best variations for the two systems to be compared.

First, we compared these systems to a **link suggestion system based on Markov Chain (baseline system)**, which is based on the transition matrix between URLs. This matrix is computed counting transitions in the training sequences. The results show that the two compared systems improved the Markov Chain based system is prediction ability.

The analysis of the obtained results showed that each of the initial systems performs differently and **each of them is specialized in different applications depending on the number of links required**. The SPADE based system seems more appropriate when a small number of links needs to be proposed, whereas the MSA based system seems to perform better for larger number of links proposed. Based on these results we proposed two new systems: **combination and hybridization of them**. The second option, where **suffix trees were built using SPADE sequences**, outperforms every other approach for every number of links proposed with an average improvement of 17.25% compared to the combined method.

7.3 Conclusions in cluster validation

With regard to the analysis of Cluster Validity Indices (CVI), on the one hand, **the most extensive CVI comparison found up to a moment was carried out using partition similarity measure based evaluation methodology** proposed by our research group [80]. This methodology validates the clustered partition comparing it with the correct one by using partition similarity measures instead of just comparing it with the real number of clusters. Moreover, the results were adjusted to apply Friedman's test with the Nemenyi and Shaffer post-hoc procedures to find significant differences between CVIs. For this comparison 30 CVIs were selected and some of them redefined to use the same notation.

Since we wanted to compare the CVIs in a wide variety of configurations we designed an experiment with several factors. **Three partition similarity measures**: Adjusted Rand, Jaccard and Variation of Information. **Three clustering algorithms**: K-means, SAHN with Ward linkage criteria and SAHN with average-linkage criteria. Synthetic and **20 real databases**. **For synthetic databases all combinations of the following 5 factors** were created: number of clusters, dimensionality, cluster overlap, cluster density and noise level. This finally led us to an experiment with **6,480 configurations**.

Now we summarize the main conclusions we drew from the Cluster Validity Indices (CVI) comparison. First, we observed that some CVIs appear to be more suitable for certain configurations, although the results were not conclusive. Furthermore, **the overall trend never changed dramatically** when we focused on a particular factor.

With regard to the experimental factors, **noise and cluster overlap had the greatest impact** on CVI performance. The number of successes is dramatically reduced when noise is present or clusters overlap. In particular, the inclusion of 10% random noise reduces the average score to a third part. A very similar score reduction was found when the clusters were moved closer so they highly overlapped. Another remarkable and surprising fact is that some indices showed better results in (a priori) more complex configurations. For example, some indices improved their results when the dimensionality of the datasets increased or when the homogeneity of the cluster densities disappeared.

Another fact worth noting is that **the results for real and synthetic datasets are qualitatively similar**, although they show disagreements for some particular indices.

Finally, we confirmed that the selection of a **partition similarity measure** to determine the correctness of the partition **is not a critical factor**. Nevertheless, it is clear that it can produce some variations in the results, so our suggestion is to use several of them to obtain more robust results. This result show that CVIs appear to be better adapted to the Variation of Information (VI) and Jaccard partition similarity measures than to Adjusted Rand.

A **statistical significance analysis** of the results showed that there are **3 main groups of indices** and the indices in the first group –Silhouette, Davies-Bouldin, Calinski-Harabasz, generalized Dunn, COP and SDbw– behave better than indices in the last group –Dunn and its Point Symmetry-Distance based variation, Gamma, C-Index, Negentropy increment and OS-Index–, being the differences statistically significant.

Although no all the analyses have been shown in this work, all the possible combinations are accessible in the Internet[2] and the scientific community can focus on the results for the configurations they are interested in. We therefore provide a tool that can be used to select the most suitable CVIs for their particular application. This procedure is very recommendable since there is not a single CVI that showed clear advantage over the rest in every context, although **Silhouette index obtained the best results** in many of them.

[2] `http://www.aldapa.eus/res/cvi/`

7.3.1 CVIs for elevated number of clusters

Moreover, we analysed the **behaviour of CVIs** in a real web mining application and **with elevated numbers of clusters** in which they tend to be unstable. The fact is that when CVIs are analysed and compared it is done with few clusters (up to 6 or 8 in general and 15 being the greatest number of clusters explored), however, in real applications higher numbers of clusters are used. In this contribution, the most stable CVIs for big Ks were identified and a method to select the best partition using many CVIs proposed.

In the first step, **we selected the five most appropriate indices** because not all of them show a coherent behaviour. For example, the Score Function (SF) index decreases the value very fast to zero when the K is increased. This fact makes the CVI unusable for big Ks. Therefore, we found a trade-off point between complexity (number of clusters) and CVI values. This procedure automatically **selects the best partition analysing the slope of the CVI values**. This procedure to select the best partition can be seen as relative partition validation method. To select the final partition, we did an analysis of the behaviour of a set of CVIs with big Ks and **proposed a procedure to select the best partition based on some of these CVIs**. **We selected the partition that the most CVIs had selected**, that is, we used the majority vote strategy (the mode value) to select the best partition from many CVIs' suggestions. After this, we evaluated the generated partitions in a link prediction context and the results given by CVIs were confirmed. Therefore, this methodology to select the best partition of clusters from many CVIs and for elevated number of clusters seems to be appropriate.

7.4 Further work

There are several lines of work that remain open in relation to the outcomes presented in this dissertation. In general, in the systems proposed in this thesis, the re-learning can be used to modify the created models as new data is stored. Although being a batch process this is not very critical, a further work could be to use incremental learning to avoid the complete modelling process when the input data changes. Using incremental learning the system will save on computational time.

In the context of the nIDS system, this proposed system still needs to be tested in a real environments. In this way, an experiment with newer data would be interesting. The way in which the classifiers can be combined is another area where a deeper analysis can be carried out and more sophisticated approaches tried. The possibility of using other clustering algorithms and the optimization of their parameters is also an area where more work can be done.

As the Probabilistic Suffix Tree (PST) representation works well with 5-length suffixes, in the histogram representation we should try with longer n-grams (more memory), that is, instead of working with 1-grams we should work

with n-grams where $n > 1$.

The work done in the context of web mining for the problem detection system was an starting point and we still have a lot of work to do. First, we could work on refining the data preprocessing step and analyse other possible variables that can be extracted from the Common Log Format files. In the pattern discovery step, we would need to deeper analyse the effect of the number of clusters and we could also try with other machine learning paradigms. Furthermore, in this context of problem detection, we could add web structure and content information to the webpages used in the experiments and disambiguate some of the doubts about the characteristics of the different groups of users we have. In this direction, in further experiments we proposed a method to automatically or semi-automatically detect navigation problems the users are having in the website discapnet[3] (where it is known that many of the users are visually impaired people) using usage, content and structure data [18] (was a collaboration, not included in this dissertation). In that case, we needed to extract some more features from the server's database that will give us information about physical or cognitive characteristics of the users.

With regard to the link suggestion systems we plan to evolve the research presented in the dissertation in multiple directions.

First, the outcome of the link suggestion systems could be applied to different application environments to get more general results. Regarding the evaluation, an user study should be performed in an on-line web environment where links can be proposed to new users and the effect of the proposed links in their navigation can be assessed.

Moreover, in this context of link suggestion, although web content information was included further improvements could be done in this direction and also including web structure information.

Furthermore, when the database of navigation sessions is too big to load in the main memory, we took the first steps and we proposed a modular approach making use of the topic division of the website [17] (this was a collaboration, not included in this dissertation).

The extension of the system to operate with logged users is quite interesting. Tracking logged users is easier and data regarding to their behaviour (navigational patters and topic preferences) are captured thoroughly. This can lead to a better classification of users in groups. For this case, the embedding of content topics and time based roles in web access sequences can be interpreted alternatively as users might behave differently on different topics.

Finally, we need to think about the ways to introduce or add the information derived from short navigation sequences (1-length and 2-length sequences) in the system because they appear considerably in log files but as they do not have enough activity for sequence analysis we did not take them into account.

[3] http://www.discapnet.es

The extensive comparison of the CVIs also raises some questions and, therefore, suggests some future work. It is obvious that this type of work can always be improved. Although we consider that we performed an extensive comparison there is room for extending it to include more CVIs, datasets, clustering algorithms and so on. In this context noise and overlap would appear to be the most interesting factors to analyse in greater depth. We also limited this work to crisp clustering, so a fuzzy CVI comparison would be a natural continuation. The analysis of some other and newer types of indices, such as stability based ones, would also be of great interest.

Finally, we argued that statistical tests are a very valuable tool in data mining and an effort should be made to use them more widely in clustering. We adapted a method widely accepted in the supervised learning area for our work, but this is just a first approach to the problem and there is a vast field of theoretical research to be addressed.

With regard to the CVI application in number of clusters larger than usual, this work opens the door to further analysis in many directions. Some indices are discarded because they show saturated values and fluctuations, thus, an analysis of the effect of the number of clusters in the inter-cluster and intra-cluster distance components used to calculate the different CVIs would also help to refine the procedure proposed to select the best partition.

7.5 Related publications

The majority of the work presented in this dissertation has already been published. Their references appear below grouped by type of publication. As in this dissertation the contributions are done in three main research areas, we indicated each of them using the corresponding initial letters of the area of research: network Intrusion Detection System (nIDS), Web Mining (WM) and Cluster Validity Indices (CVI).

- International journals:
  - CVI [13]: Olatz Arbelaitz, Ibai Gurrutxaga, Javier Muguerza, Jesús María Pérez, and Iñigo Perona. An extensive comparative study of cluster validity indices. *Pattern Recognition*, 46(1):243–256, 2013.
  - WM [11], part related to link suggestion: Olatz Arbelaitz, Ibai Gurrutxaga, Aizea Lojo, Javier Muguerza, Jesús María Pérez, and Iñigo Perona. Web usage and content mining to extract knowledge for modelling the users of the bidasoa turismo website and to adapt it. *Expert Systems with Applications*, 40(18):7478–7491, 2013.
  - WM [160], submitted on 29/04/2015: Iñigo Perona, Olatz Arbelaitz, Christos Makris, Javier Muguerza, and Evangelos Theodoridis. Improving link recommendation systems: clustering, sequential pattern mining and suffix trees. *Knowledge and Information Systems*.

- International Conferences:
  - nIDS [161]: Iñigo Perona, Ibai Gurrutxaga, Olatz Arbelaitz, José Ignacio Martín, Javier Muguerza, and Jesús María Pérez. Service-independent payload analysis to improve intrusion detection in network traffic. In John F. Roddick, Jiuyong Li, Peter Christen, and Paul J. Kennedy, editors, *Seventh Australasian Data Mining Conference (AusDM 2008)*, volume 87 of *CRPIT*, pages 171–178, Darlinghurst, South Australia, November 2008. ACS.
  - nIDS [155]: Iñigo Perona, Iñaki Albisua, Olatz Arbelaitz, Ibai Gurrutxaga, José Ignacio Martín, Javier Muguerza, and Jesús María Pérez. Histogram based payload processing for unsupervised anomaly detection systems in network intrusion. In Lus Seabra Lopes, Nuno Lau, Pedro Mariano, and Luís M. Rocha, editors, *New Trends in Artificial Intelligence. 14th Portuguese Conference on Artificial Intelligence. EPIA 2009. Aveiro, October 12-15, 2009. Proceedings*, pages 329–340, Aveiro, Portugal, October 2009. Universidade de Aveiro.
  - nIDS [158]: Iñigo Perona, Olatz Arbelaitz, Ibai Gurrutxaga, José Ignacio Martín, Javier Muguer za, and Jesús María Pérez. Unsupervised anomaly detection system for nidss based on payload and probabilistic suffix trees. In Hans Weghorn and Pedro Isaías, editors, *Proceedings of the IADIS International Conference on Applied Computing*, Rome, Italy, November 2009.
  - WM [1]: Julio Abascal, Olatz Arbelaitz, Javier Muguerza, and Iñigo Perona. Data mining based user modelling systems for web personalization applied to people with disabilities. Mini Symposia in Assistive Machine Learning for People with Disabilities hold in 23th Neural Information Processing Systems (NIPS) Conference, December 2009.
  - WM [19]: Olatz Arbelaitz, Javier Muguerza, Iñgo Perona, and Julio Abascal. Automatic data processing for web accessibility personalization. Oral communication in the 24th European conference on Operational Research (EURO 2010), Lisboa, Portugal, July 2010.
  - WM [12]: Olatz Arbelaitz, Ibai Gurrutxaga, Aizea Lojo, Javier Muguerza, and Iñigo Perona. SAHN with SEP/COP and SPADE, to build a general web navigation adaptation system using server log information. In *Proceedings of the 14th International Conference on Advances in Artificial Intelligence: Spanish Association for Artificial Intelligence (CAEPIA)*, CAEPIA'11, pages 413–422, Berlin, Heidelberg, 2011. Springer-Verlag.
  - WM, [8]: Olatz Arbelaitz, Ibai Gurrutxaga, Aizea Lojo, Javier Muguerza, Jesús María Pérez, and Iñigo Perona. Web usage mining for automatic link generation. In C. Badica, G. Eleftherakis, G. J. Nalepa, M. I. Capel-Tuñón, K. Ghdira, and V. Monfort, editors, *Proceedings of the 10th International Workshop on Modelling, Simulation, Verification and Validation of Enterprise Information Systems*

(MSVVEIS 2012) & 1st International Workshop on Web Intelligence (WEBI 2012), pages 71–80, Wrocaw, Poland, June 2012. SciTePress.

- WM, [6]: Olatz Arbelaitz, Ibai Gurrutxaga, Aizea Lojo, Javier Muguerza, Jesús María Pérez, and Iñigo Perona. Adaptation of the user navigation scheme using clustering and frequent pattern mining techniques for profiling. In *Proceedings of the 4th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management (IC3K 2012) & 4th International Conference on Knowledge Discovery and Information Retrieval (KDIR 2012)*, pages 187–192, Barcelona, 2012.

- National Conferences:

  - WM [159]: Iñigo Perona, Olatz Arbelaitz, Ibai Gurrutxaga, and Javier Muguerza. Towards web personalization for users with special needs using web navigation logs. In Alicia Troncoso and José C. Riquelme, editors, *Actas del V Simposio de Teoría y Aplicaciones de Minería de Datos (TAMIDA 2010) - in English: Proceedings of the V Workshop on theory and applications of data mining*, Valencia, Spain, September 2010. Red Española de Minería de Datos y Aprendizaje.
  - CVI [9]: Olatz Arbelaitz, Ibai Gurrutxaga, Aizea Lojo, Javier Muguerza, Jesús María Pérez, and Iñigo Perona. Clustering based navigation profiling in a tourism website using cluster validity indices to select the best partition. In *Actas de la XV Conferencia de la Asociacin Espaola para la Inteligencia Artificial (CAEPIA 2013) - in English: Procedings of the XV Conference of the Spanish Association for Artificial Intelligence*, pages 29–38, Madrid, Spain, September 2013.
  - CVI [14]: Olatz Arbelaitz, Ibai Gurrutxaga, Javier Muguerza, Jesús María Pérez, and Iñigo Perona. Un amplio estudio comparativo de índices de validación de agrupamiento. Actas de la XV Conferencia de la Asociacin Espaola para la Inteligencia Artificial (CAEPIA 2013) sección Key Works - in English: Procedings of the XV Conference of the Spanish Association for Artificial Intelligence section Key Works, September 2013.

- Technical reports:

  - nIDS [156]: Iñigo Perona, Olatz Arbelaitz, Ibai Gurrutxaga, José Ignacio Martín, Javier Muguerza, and Jesús María Pérez. gureKDDcup99 datu-basearen sorrera (in basque). Technical Report EHU-KAT-IK-08-08, University of the Basque Country (UPV/EHU), May 2008.
  - nIDS, [157]: Iñigo Perona, Olatz Arbelaitz, Ibai Gurrutxaga, José Ignacio Martín, Javier Muguerza, and Jesús María Pérez. Application

of probabilistic suffix trees: payload analysis for nIDSs. Technical Report EHU-KAT-IK-08-09, University of the Basque Country (UPV/EHU), July 2009.

- WM [15]: Olatz Arbelaitz, Aizea Lojo, Javier Muguerza, and Iñigo Perona. Aplicación de técnicas de minería de datos para la extracción de conocimiento de la web de bidasoa turismo (in spanish). Technical Report EHU-KAT-IK-13-12, University of the Basque Country (UPV/EHU), December 2012.
- WM [16]: Olatz Arbelaitz, Aizea Lojo, Javier Muguerza, and Iñigo Perona. Applying data mining techniques to extract knowledge from discapnet website. Technical Report EHU-KAT-IK-01-14, University of the Basque Country (UPV/EHU), February 2014.

- Collaborations:
  - nIDS [79], international conference: Ibai Gurrutxaga, Olatz Arbelaitz, Jesús María Pérez, Javier Muguerza, José Ignacio Martín, and Iñigo Perona. Evaluation of malware clustering based on its dynamic behaviour. In John F. Roddick, Jiuyong Li, Peter Christen, and Paul J. Kennedy, editors, *Seventh Australasian Data Mining Conference (AusDM 2008)*, volume 87 of *CRPIT*, pages 163–170, Glenelg, South Australia, November 2008. ACS.
  - CVI [78], international conference: Ibai Gurrutxaga, Olatz Arbelaitz, José Ignacio Martín, Javier Muguerza, Jesús María Pérez, and Iñigo Perona. SIHC: A stable incremental hierarchical clustering algorithm. In *ICEIS (2)'09*, pages 300–304, Milan, Italy, 2009.
  - CVI [77], international journal: Ibai Gurrutxaga, Iñaki Albisua, Olatz Arbelaitz, José Ignacio Martín, Javier Muguerza, Jesús María Pérez, and Iñigo Perona. SEP/COP: An efficient method to find the best partition in hierarchical clustering based on a new cluster validity index. *Pattern Recognition*, 43(10):3364–3373, 2010.
  - WM [7], international conference: Olatz Arbelaitz, Ibai Gurrutxaga, Aizea Lojo, Javier Muguerza, Jesús María Pérez, and Iñigo Perona. Enhancing a web usage mining based tourism website adaptation with content information. In *Proceedings of the 4th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management (IC3K 2012) & 4th International Conference on Knowledge Discovery and Information Retrieval (KDIR 2012)*, pages 287–292, Barcelona, 2012.
  - WM [10], international conference: Olatz Arbelaitz, Ibai Gurrutxaga, Aizea Lojo, Javier Muguerza, Jesús María Pérez, and Iñigo Perona. A navigation-log based web mining application to profile the interests of users accessing the web of bidasoa turismo. In *e-Review of Tourism Research (eRTR). Conference on Information and Communication Technologies in Tourism (ENTER 2013)*, Innsbruck, Austria, 2013.

- WM, [17], international conference: Olatz Arbelaitz, Aizea Lojo, Javier Muguerza, and Iñigo Perona. Global versus modular link prediction approach for discapnet: website focused to visually impaired people. In Maria Ganzha, Leszek Maciaszek, and Marcin Paprzycki, editors, *Proceedings of the 2014 Federated Conference on Computer Science and Information Systems (FedCSIS 2014)*, volume 2, Warsaw, Poland, September 2014.
- WM [18], international journal: Olatz Arbelaitz, Aizea Lojo, Javier Muguerza, and Iñigo Perona. Web mining for navigation problem detection and diagnosis in discapnet: A website aimed at disabled people. *Journal of the Association for Information Science and Technology (JAIST)*, pages n/a–n/a, 2015.
- WM [179], national conference: Lierni Sestorain, Iñigo Perona, Ainhoa Yera, Javier Muguerza and Olatz Arbelaitz. Modification of SEP/COP, an efficient method to find the best partition in hierarchical clustering, to improve a link prediction system. In *Procedings of the XVII Conference of the Spanish Association for Artificial Intelligence*, Albacete, Spain, November 2015.

Likewise, the gureKDDCup database is publicly available on the Internet[4] which includes KDDCup99 attributes plus raw payloads to facilitate the research in general payload processing for intrusion detection. It has been accessed by 86 entities from all over the world (checked on 01/12/2015).

Furthermore, all the possible combinations tried in the extensive comparison of Cluster Validity Indices (CVI) are accessible on the Internet[5] and the scientific community can focus on the results for the configurations they are interested in.

[4] `http://www.aldapa.eus/res/gureKddcup/`
[5] `http://www.aldapa.eus/res/cvi/`

Bibliography

[1] J. Abascal, O. Arbelaitz, J. Muguerza, and I. Perona. Data mining based user modeling systems for web personalization applied to people with disabilities. Mini Symposia in Assistive Machine Learning for People with Disabilities hold in 23th Neural Information Processing Systems (NIPS) Conference, December 2009.

[2] M. Abou-Shouk, W. M. Lim, and P. Megicks. Internet adoption by travel agents: a case of egypt. *International Journal of Tourism Research*, 15(3):298–312, 2013.

[3] R. Agrawal and R. Srikant. Fast algorithms for mining association rules in large databases. In *Proceedings of the 20th International Conference on Very Large Data Bases*, VLDB '94, pages 487–499, San Francisco, CA, USA, 1994. Morgan Kaufmann Publishers Inc.

[4] A. Anitha. Article:a new web usage mining approach for next page access prediction. *International Journal of Computer Applications*, 8(11):7–10, October 2010. Published By Foundation of Computer Science.

[5] C. Antunes and A. L. Oliveira. Generalization of pattern-growth methods for sequential pattern mining with gap constraints. In P. Perner and A. Rosenfeld, editors, *Machine Learning and Data Mining in Pattern Recognition*, volume 2734 of *Lecture Notes in Computer Science*, pages 239–251. Springer Berlin Heidelberg, 2003.

[6] O. Arbelaitz, I. Gurrutxaga, A. Lojo, J. Muguerza, J. M. Pérez, and I. Perona. Adaptation of the user navigation scheme using clustering and frequent pattern mining techniques for profiling. In *Proceedings of the 4th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management (IC3K 2012) & 4th International Conference on Knowledge Discovery and Information Retrieval (KDIR 2012)*, pages 187–192, Barcelona, 2012.

[7] O. Arbelaitz, I. Gurrutxaga, A. Lojo, J. Muguerza, J. M. Pérez, and I. Perona. Enhancing a web usage mining based tourism website adaptation with content information. In *Proceedings of the 4th International Joint Conference on Knowledge Discovery, Knowledge Engineering and*

Knowledge Management (IC3K 2012) & 4th International Conference on Knowledge Discovery and Information Retrieval (KDIR 2012), pages 287–292, Barcelona, 2012.

[8] O. Arbelaitz, I. Gurrutxaga, A. Lojo, J. Muguerza, J. M. Pérez, and I. Perona. Web usage mining for automatic link generation. In C. Badica, G. Eleftherakis, G. J. Nalepa, M. I. Capel-Tuñón, K. Ghdira, and V. Monfort, editors, *Proceedings of the 10th International Workshop on Modelling, Simulation, Verification and Validation of Enterprise Information Systems (MSVVEIS 2012) & 1st International Workshop on Web Intelligence (WEBI 2012)*, pages 71–80, Wrocaw, Poland, June 2012. SciTePress.

[9] O. Arbelaitz, I. Gurrutxaga, A. Lojo, J. Muguerza, J. M. Pérez, and I. Perona. Clustering based navigation profiling in a tourism website using cluster validity indices to select the best partition. In *Actas de la XV Conferencia de la Asociacin Espaola para la Inteligencia Artificial (CAEPIA 2013) - in English: Procedings of the XV Conference of the Spanish Association for Artificial Intelligence*, pages 29–38, Madrid, Spain, September 2013.

[10] O. Arbelaitz, I. Gurrutxaga, A. Lojo, J. Muguerza, J. M. Pérez, and I. Perona. A navigation-log based web mining application to profile the interests of users accessing the web of bidasoa turismo. In *e-Review of Tourism Research (eRTR). Conference on Information and Communication Technologies in Tourism (ENTER 2013)*, Innsbruck, Austria, 2013.

[11] O. Arbelaitz, I. Gurrutxaga, A. Lojo, J. Muguerza, J. M. Pérez, and I. Perona. Web usage and content mining to extract knowledge for modelling the users of the bidasoa turismo website and to adapt it. *Expert Systems with Applications*, 40(18):7478 – 7491, 2013.

[12] O. Arbelaitz, I. Gurrutxaga, A. Lojo, J. Muguerza, and I. Perona. SAHN with SEP/COP and SPADE, to build a general web navigation adaptation system using server log information. In *Proceedings of the 14th International Conference on Advances in Artificial Intelligence: Spanish Association for Artificial Intelligence (CAEPIA)*, CAEPIA'11, pages 413–422, Berlin, Heidelberg, 2011. Springer-Verlag.

[13] O. Arbelaitz, I. Gurrutxaga, J. Muguerza, J. M. Pérez, and I. Perona. An extensive comparative study of cluster validity indices. *Pattern Recognition*, 46(1):243–256, 2013.

[14] O. Arbelaitz, I. Gurrutxaga, J. Muguerza, J. M. Pérez, and I. Perona. Un amplio estudio comparativo de índices de validación de agrupamiento. Actas de la XV Conferencia de la Asociación Española para la Inteligencia Artificial (CAEPIA 2013) sección Key Works - in English: Procedings of the XV Conference of the Spanish Association for Artificial Intelligence section Key Works, September 2013.

[15] O. Arbelaitz, A. Lojo, J. Muguerza, and I. Perona. Aplicación de técnicas de minería de datos para la extracción de conocimiento de la web de bidasoa turismo (in spanish). Technical Report EHU-KAT-IK-13-12, University of the Basque Country (UPV/EHU), December 2012.

[16] O. Arbelaitz, A. Lojo, J. Muguerza, and I. Perona. Applying data mining techniques to extract knowledge from discapnet website. Technical Report EHU-KAT-IK-01-14, University of the Basque Country (UPV/EHU), February 2014.

[17] O. Arbelaitz, A. Lojo, J. Muguerza, and I. Perona. Global versus modular link prediction approach for discapnet: website focused to visually impaired people. In M. Ganzha, L. Maciaszek, and M. Paprzycki, editors, *Proceedings of the 2014 Federated Conference on Computer Science and Information Systems (FedCSIS 2014)*, volume 2, Warsaw, Poland, September 2014.

[18] O. Arbelaitz, A. Lojo, J. Muguerza, and I. n. Perona. Web mining for navigation problem detection and diagnosis in discapnet: A website aimed at disabled people. *Journal of the Association for Information Science and Technology (JAIST)*, pages n/a–n/a, 2015.

[19] O. Arbelaitz, J. Muguerza, I. Perona, and J. Abascal. Automatic data processing for web accessibility personalization. Oral communication in the 24th European conference on Operational Research (EURO 2010), Lisboa, Portugal, July 2010.

[20] A. B. Ashfaq, M. J. Robert, A. Mumtaz, M. Q. Ali, A. Sajjad, and S. A. Khayam. A comparative evaluation of anomaly detectors under portscan attacks. In *Proceedings of the 11th International Symposium on Recent Advances in Intrusion Detection*, RAID '08, pages 351–371, Berlin, Heidelberg, 2008. Springer-Verlag.

[21] M. L. G. at the University of Waikato. Weka (data mining software in java). `http://www.cs.waikato.ac.nz/ml/weka/` (accesed on 2015/04/13).

[22] J. Ayres, J. Flannick, J. Gehrke, and T. Yiu. Sequential pattern mining using a bitmap representation. In *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '02, pages 429–435, New York, NY, USA, 2002. ACM.

[23] R. Bace and P. Mell. Nist special publication on intrusion detection systems (NIST sp800-31). *National Institute of Standards and Technology (NIST)*, 2003.

[24] F. B. Baker and L. J. Hubert. Measuring the power of hierarchical cluster analysis. *Journal of the American Statistical Association*, 70(349):pp. 31–38, 1975.

[25] S. Bandyopadhyay and S. Saha. A point symmetry-based clustering technique for automatic evolution of clusters. *IEEE Transactions on Knowledge and Data Engineering*, 20 (Issue: 11):1441 – 1457, April 2008.

[26] A. Banerjee and J. Ghosh. Clickstream clustering using weighted longest common subsequences. In *In Proceedings of the Web Mining Workshop at the 1st SIAM Conference on Data Mining*, pages 33–40, 2001.

[27] D. Barbará and S. Jajodia, editors. *Applications of Data Mining in Computer Security*, volume 6 of *Advances in Information Security*. Springer US, 2002.

[28] V. Batagelj and M. Bren. Comparing resemblance measures. *Journal of Classification*, 12(1):73–90, 1995.

[29] G. Bejerano and G. Yona. Modeling protein families using probabilistic suffix trees. In *Proceedings of the 3rd annual international conference on research in computational molecular biology (RECOMB)*, pages 15–24, 1999.

[30] A. Ben-Hur, A. Elisseeff, and I. Guyon. A stability based method for discovering structure in clustered data. *Pacific Symposium on Biocomputing. Pacific Symposium on Biocomputing*, pages 6–17, 2002.

[31] J. C. Bezdek, W. Q. Li, Y. Attikiouzel, and M. Windham. A geometric approach to cluster validity for normal mixtures. *Soft Computing*, 1(4):166–179, 1997.

[32] J. C. Bezdek and N. R. Pal. Some new indexes of cluster validity. *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, 28(3):301–315, June 1998.

[33] S. Bhawsar, K. Pathak, and V. Patidar. New framework for web access prediction. *International Journal of Computer Technology and Electronics Engineering (IJCTEE)*, 2(1):48–53, 2012.

[34] D. M. Blei. Introduction to probabilistic topic models. *Communications of the ACM*, pages n/a–n/a, 2011.

[35] D. M. Blei. Probabilistic topic models. *Communications of the ACM*, 55(4):77–84, April 2012.

[36] D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent dirichlet allocation. *J. Mach. Learn. Res.*, 3:993–1022, March 2003.

[37] B. E. Boser, I. M. Guyon, and V. N. Vapnik. A training algorithm for optimal margin classifiers. In *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*, COLT '92, pages 144–152, New York, NY, USA, 1992.

[38] M. Brun, C. Sima, J. Hua, J. Lowey, B. Carroll, E. Suh, and E. R. Dougherty. Model-based evaluation of clustering validation measures. *Pattern Recognition*, 40(3):807 – 824, 2007.

[39] P. Brusilovsky, A. Kobsa, and W. Nejdl. *The Adaptive Web: Methods and Strategies of Web Personalization*. Springer-Verlag, Berlin, Heidelberg, 2007.

[40] T. Calinski and J. Harabasz. A dendrite method for cluster analysis. *Communications in Statistics*, 3(1):1–27, 1974.

[41] V. Chandola, A. Banerjee, and V. Kumar. Anomaly detection: A survey. *ACM Computing Surveys (CSUR)*, 41(3):15:1–15:58, July 2009.

[42] X. Chen and X. Zhang. A popularity-based prediction model for web prefetching. *Computer*, 36(3):63–70, March 2003.

[43] S. Chimphlee, N. Salim, M. S. Ngadiman, W. Chimphlee, and S. Srinoy. Rough sets clustering and markov model for web access prediction. In *Proceedings of the Postgraduate Annual Research Seminar (PARS)*, pages 470–475, Paris, May 2006.

[44] B. S. Chordia and K. P. Adhiya. Grouping web access sequences using sequence alignment methods. *Indian Journal of Computer Science and Engineering (IJCSE)*, 2(3):308–314, June 2011.

[45] C.-H. Chou, M.-C. Su, and E. Lai. A new cluster validity measure and its application to image compression. *Pattern Analysis and Applications*, 7(2):205–220, 2004.

[46] Cisco. Vni forecast highlights. `http://www.cisco.com/c/en/us/solutions/service-provider/visual-networking-index-vni/vni-forecast.html`, 2015.

[47] C. Cortes and V. Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, 1995.

[48] T. Cover and P. Hart. Nearest neighbor pattern classification. *IEEE Transactions Information Theory*, 13(1):21–27, Sept. 2006.

[49] P. Cunningham, M. Cord, and S. J. Delany. Supervised learning. In M. Cord and P. Cunningham, editors, *Machine Learning Techniques for Multimedia*, Cognitive Technologies, pages 21–49. Springer Berlin Heidelberg, 2008.

[50] N. Dagorn. WebIDS: A cooperative bayesian anomaly-based intrusion detection system for web applications (extended abstract). In R. Lippmann, E. Kirda, and A. Trachtenberg, editors, *Recent Advances in Intrusion Detection (RAID)*, volume 5230 of *Lecture Notes in Computer Science*, pages 392–393. Springer Berlin Heidelberg, 2008.

[51] P. Danzig, J. Mogul, V. Paxson, and M. Schwartz. The internet traffic archive. `http://ita.ee.lbl.gov/` (Accessed on 2015/06/12), 2008. Sponsored by ACM SIGCOMM.

[52] D. L. Davies and D. W. Bouldin. A cluster separation measure. *IEEE Trans. Pattern Anal. Mach. Intell.*, 1(2):224–227, February 1979.

[53] J. Demšar. Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research*, 7:1–30, December 2006.

[54] D. E. Denning. An intrusion-detection model. *IEEE Trans. Softw. Eng.*, 13(2):222–232, February 1987.

[55] P. Desikan, J. Srivastava, V. Kumar, and P.-N. Tan. Hyperlink analysis: Techniques and applications. Technical report, University of Minnesota, Minneapolis, MN, USA, 2002.

[56] T. G. Dietterich. Approximate statistical tests for comparing supervised classification learning algorithms. *Neural Computation*, 10(7):1895–1923, October 1998.

[57] E. Dimitriadou, S. Dolničar, and A. Weingessel. An examination of indexes for determining the number of clusters in binary data sets. *Psychometrika*, 67(1):137–159, 2002.

[58] C. Dimopoulos, C. Makris, Y. Panagis, E. Theodoridis, and A. K. Tsakalidis. A web page usage prediction scheme using sequence indexing and clustering techniques. *Data and Knowledge Engineering*, 69(4):371–382, 2010.

[59] R. C. Dubes. How many clusters are best? - an experiment. *Pattern Recognition*, 20(6):645 – 663, 1987.

[60] J. C. Dunn. A fuzzy relative of the isodata process and its use in detecting compact well-separated clusters. *Journal of Cybernetics*, 3(3):32–57, 1973.

[61] B. Efron. Estimating the error rate of a prediction rule: Improvement on Cross-Validation. *Journal of the American Statistical Association*, 78(382):316–331, 1983.

[62] EPA-HTTP_logs. HTTP requests to the EPA WWW server located at research triangle park, NC. `http://ita.ee.lbl.gov/html/contrib/EPA-HTTP.html` (Accessed on 2015/06/12), 1995.

[63] EPA_website. United states environmental protection agency. `http://www.epa.gov/` (Accessed on 2015/06/12), 2010.

[64] E. Eskin, A. Arnold, M. Prerau, L. Portnoy, and S. J. Stolfo. A geometric framework for unsupervised anomaly detection: Detecting intrusions in unlabeled data. In *Applications of Data Mining in Computer Security*. Kluwer, 2002.

[65] T. E. T. C. (ETC). New media trend watch - online travel market. http://www.newmediatrendwatch.com/worldoverview/91-online-travel-market?showall=1 (accessed 2015/07/20), 2012.

[66] F. M. Facca and P. L. Lanzi. Mining interesting knowledge from weblogs: A survey. *Data Knowl. Eng.*, 53(3):225–241, June 2005.

[67] T. Fawcett. ROC graphs: Notes and practical considerations for researchers. Technical report, HP Laboratories, Palo Alto, CA, USA, 2004.

[68] U. Fayyad, G. Piatetsky-shapiro, and P. Smyth. From data mining to knowledge discovery in databases. *AI Magazine*, 17:37–54, 1996.

[69] E. Fix and J. L. Hodges. Discriminatory Analysis: Nonparametric Discrimination: Consistency Properties. Technical Report Project 21-49-004, Report Number 4, USAF School of Aviation Medicine, Randolf Field, Texas, 1951.

[70] A. Frank and A. Asuncion. UCI machine learning repository. http://archive.ics.uci.edu/ml/, 2010.

[71] M. Friedman. The use of ranks to avoid the assumption of normality implicit in the analysis of variance. *Journal of the American Statistical Association*, 32 (200)(675–701), 1937.

[72] E. García, C. Romero, S. Ventura, and C. De Castro. An architecture for making recommendations to courseware authors using association rule mining and collaborative filtering. *User Modeling and User-Adapted Interaction*, 19(1-2):99–132, 2009.

[73] S. García and F. Herrera. An extension on "statistical comparisons of classifiers over multiple data sets" for all pairwise comparisons. *Journal of Machine Learning Research*, 9:2677–2694, 2008.

[74] U. Gretzel. Intelligent systems in tourism: A social science perspective. *Annals of Tourism Research*, 38(3):757 – 779, 2011.

[75] U. Gretzel, Y.-L. Yuan, and D. R. Fesenmaier. Preparing for the new economy: Advertising strategies and change in destination marketing organizations. *Journal of Travel Research*, 39(2):146–156, 2000.

[76] J. Guo, V. Kešelj, and Q. Gao. Integrating web content clustering into web log association rule mining. In *Proceedings of the 18th Canadian Society Conference on Advances in Artificial Intelligence*, AI'05, pages 182–193, Berlin, Heidelberg, 2005. Springer-Verlag.

[77] I. Gurrutxaga, I. Albisua, O. Arbelaitz, J. I. Martín, J. Muguerza, J. M. Pérez, and I. Perona. SEP/COP: An efficient method to find the best partition in hierarchical clustering based on a new cluster validity index. *Pattern Recognition*, 43(10):3364 – 3373, 2010.

[78] I. Gurrutxaga, O. Arbelaitz, J. I. Martín, J. Muguerza, J. M. Pérez, and I. Perona. SIHC: A stable incremental hierarchical clustering algorithm. In *ICEIS (2)'09*, pages 300–304, Milan, Italy, 2009.

[79] I. Gurrutxaga, O. Arbelaitz, J. M. Pérez, J. Muguerza, J. I. Martín, and I. Perona. Evaluation of malware clustering based on its dynamic behaviour. In J. F. Roddick, J. Li, P. Christen, and P. J. Kennedy, editors, *Seventh Australasian Data Mining Conference (AusDM 2008)*, volume 87 of *CRPIT*, pages 163–170, Glenelg, South Australia, November 2008. ACS.

[80] I. Gurrutxaga, J. Muguerza, O. Arbelaitz, J. M. Pérez, and J. I. Martín. Towards a standard methodology to evaluate internal cluster validity indices. *Pattern Recognition Letters*, 32(3):505 – 515, 2011.

[81] D. Gusfield. *Algorithms on Strings, Trees, and Sequences: Computer Science and Computational Biology.* Cambridge University Press, New York, NY, USA, 1997.

[82] M. Halkidi, Y. Batistakis, and M. Vazirgiannis. On clustering validation techniques. *Journal of Intelligent Information Systems*, 17(2-3):107–145, December 2001.

[83] M. Halkidi and M. Vazirgiannis. Clustering validity assessment: finding the optimal partitioning of a data set. In *Data Mining, 2001. ICDM 2001, Proceedings IEEE International Conference on*, pages 187–194, 2001.

[84] M. Halkidi and M. Vazirgiannis. A density-based cluster validity approach using multi-representatives. *Pattern Recognition Letters*, 29(6):773 – 786, 2008.

[85] R. W. Hamming. Error Detecting and Error Correcting Codes. *Bell System Technical Journal*, 26(2):147–160, 1950.

[86] A. Hardy. On the number of clusters. *Computational Statistics & Data Analysis*, 23(1):83 – 96, 1996. Classification.

[87] B. Hay, G. Wets, and K. Vanhoof. Clustering navigation patterns on a website using a sequence alignment method. In *In Proceedings of 17th International Joint Conference on Artificial Intelligence*, pages 1–6, 2001.

[88] D. He and A. Göker. Detecting session boundaries from web user logs. In *In Proceedings of of the BCS-IRSG 22nd Annual Colloquium on Information Retrieval Research*, pages 57–66, 2000.

[89] E. Hellinger. Neue begründung der theorie quadratischer formen von unendlichvielen veränderlichen. *Journal für die reine und angewandte Mathematik*, 136:210–271, 1909.

[90] S. Hettich and S. D. Bay. The uci kdd archive. `http://kdd.ics.uci.edu`, 1999. Irvine, CA: University of California, Department of Information and Computer Science.

[91] M. Hills. Allocation rules and their error rates. *Journal of the Royal Statistical Society. Series B (Methodological)*, (131), 1966.

[92] V. Hodge and J. Austin. A survey of outlier detection methodologies. *Artificial Intelligence Review*, 22(2):85–126, October 2004.

[93] K. J. Holzinger and H. H. Harman. *Factor Analysis*. University of Chicago Press, 1941.

[94] C.-I. Hsu, M.-L. Shih, B.-W. Huang, B.-Y. Lin, and C.-N. Lin. Predicting tourism loyalty using an integrated bayesian network mechanism. *Expert Systems with Applications*, 36(9):11760 – 11763, 2009.

[95] L. Hubert and P. Arabie. Comparing partitions. *Journal of Classification*, 2(1):193–218, 1985.

[96] L. J. Hubert and J. R. Levin. A general statistical framework for assessing categorical clustering in free recall. *Psychological Bulletin*, 83:1072–1080, 1976.

[97] C. S. Iliopoulos, C. Makris, Y. Panagis, K. Perdikuri, E. Theodoridis, and A. Tsakalidis. The weighted suffix tree: An efficient data structure for handling molecular weighted sequences and its applications. *Fundam. Inf.*, 71(2,3):259–277, February 2006.

[98] R. L. Iman and J. M. Davenport. Approximations of the critical region of the fbietkan statistic. *Communications in Statistics - Theory and Methods*, 9(6):571–595, January 1980.

[99] P. Jaccard. Étude comparative de la distribution florale dans une portion des Alpes et des Jura. *Bulletin del la Société Vaudoise des Sciences Naturelles*, 37:547–579, 1901.

[100] P. Jaccard. Nouvelles recherches sur la distribution florale. *Bulletin de la Sociète Vaudense des Sciences Naturelles*, 44:223–270, 1908.

[101] P. Jaccard. The distribution of the flora in the alpine zone. *New Phytologist*, 11(2):37–50, February 1912.

[102] A. K. Jain. Data clustering: 50 years beyond k-means. *Pattern Recognition Letters*, 31(8):651 – 666, 2010. Award winning papers from the 19th International Conference on Pattern Recognition (ICPR) 19th International Conference in Pattern Recognition (ICPR).

[103] A. K. Jain. Data clustering: 50 years beyond k-means. *Pattern Recognition Letter*, 31(8):651–666, June 2010.

[104] A. K. Jain and R. C. Dubes. *Algorithms for Clustering Data*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1988.

[105] A. K. Jain and J. V. Moreau. Bootstrap technique in cluster analysis. *Pattern Recognition*, 20(5):547 – 568, 1987.

[106] S. E. Jespersen, J. Thorhauge, and T. B. Pedersen. A hybrid approach to web usage mining. In *Proceedings of the 4th International Conference on Data Warehousing and Knowledge Discovery*, DaWaK 2000, pages 73–82, London, UK, UK, 2002. Springer-Verlag.

[107] K. P. Joshi, A. Joshi, and Y. Yesha. On using a warehouse to analyze web logs. *Distrib. Parallel Databases*, 13(2):161–180, March 2003.

[108] J. Just. A short survey of web data mining. In *Proceedings of the 22nd Annual Conference of Doctoral Students WDS 2013*, Part I - Mathematics and Computer Sciences, pages 59–62, 2013.

[109] B. Kahle. Internet archive. `htpps://archive.org` (accessed on 2015/09/23), 1996. San Francisco-based nonprofit digital library with the stated mission of "universal access to all knowledge".

[110] J.-K. Kamarainen, V. Kyrki, J. Ilonen, and H. Kälviäinen. Improving similarity measures of histograms using smoothing projections. *Pattern Recognition Letters*, 24(12):2009–2019, 2003.

[111] L. Kaufman and P. J. Rousseeuw. *Finding Groups in Data: An Introduction to Cluster Analysis*. Wiley-Interscience, 9th edition, March 1990.

[112] D.-J. Kim, Y.-W. Park, and D.-J. Park. A novel validity index for determination of the optimal number of clusters. *IEICE Transactions on Information and Systems*, E84-D:281–285, 2001.

[113] M. Kim and R. Ramakrishna. New indices for cluster validity assessment. *Pattern Recognition Letters*, 26(15):2353 – 2363, 2005.

[114] R. Kohavi. A study of cross-validation and bootstrap for accuracy estimation and model selection. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence - Volume 2*, IJCAI'95, pages 1137–1143, San Francisco, CA, USA, 1995. Morgan Kaufmann Publishers Inc.

[115] R. Kohavi and F. Provost. Glossary of terms. *Machine Learning - Special issue on applications of machine learning and the knowledge discovery process*, 30(2-3):271–274, February 1998.

[116] R. Kosala and H. Blockeel. Web mining research: A survey. *ACM SIGKDD Explorations Newsletter*, 2(1):1–15, June 2000.

[117] T. M. Kroeger, D. D. E. Long, and J. C. Mogul. Exploring the bounds of web latency reduction from caching and prefetching. In *Proceedings of the USENIX Symposium on Internet Technologies and Systems on USENIX Symposium on Internet Technologies and Systems*, USITS'97, pages 2–2, Berkeley, CA, USA, 1997. USENIX Association.

[118] C. Krügel, T. Toth, and E. Kirda. Service specific anomaly detection for network intrusion detection. In *Proceedings of the 2002 ACM Symposium on Applied Computing*, SAC '02, pages 201–208, New York, NY, USA, 2002. ACM.

[119] P. Kumar, P. R. Krishna, R. S. Bapi, and S. K. De. Rough clustering of sequential data. *Data and Knowledge Engineering*, 63(2):183–199, 2007.

[120] L. F. Lago-Fernández and F. Corbacho. Normality-based validation for crisp clustering. *Pattern Recognition*, 43(3):782 – 795, 2010.

[121] S. C. Larson. The shrinkage of the coefficient of multiple correlation. *Journal of Educational Psychology*, 22(1):4555, 1931.

[122] W. Lee. *A Data Mining Framework for Constructing Features and Models for Intrusion Detection Systems*. PhD thesis, Columbia University, 1999.

[123] W. Lee, S. J. Stolfo, and K. W. Mok. Mining in a data-flow environment: Experience in network intrusion detection. In *Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '99, pages 114–124, New York, NY, USA, 1999. ACM.

[124] K. Leung and C. Leckie. Unsupervised anomaly detection in network intrusion detection using clusters. In *Proceedings of the Twenty-eighth Australasian Conference on Computer Science - Volume 38*, volume 38 of *ACSC '05*, pages 333–342, Darlinghurst, Australia, Australia, 2005. Australian Computer Society, Inc.

[125] V. I. Levenshtein. Binary Codes Capable of Correcting Deletions, Insertions and Reversals. *Soviet Physics Doklady*, 10:707, 1966.

[126] M. Li, X. Chen, X. Li, B. Ma, and P. Vitanyi. The similarity metric. *IEEE Transactions on Information Theory*, 50(12):3250–3264, December 2004.

[127] B. Liu. *Web Data Mining: Exploring Hyperlinks, Contents, and Usage Data (Data-Centric Systems and Applications)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.

[128] H. Liu and V. Kešelj. Combined mining of web server logs and web contents for classifying user navigation patterns and predicting users' future requests. *Data and Knowledge Engineering*, 61(2):304–330, May 2007.

[129] Q. Liu and Y. Wu. *Encyclopedia of the Sciences of Learning*, chapter Supervised learning, pages 3243–3245. Springer US, 2012.

[130] P. Makkar, P. Gulati, and A. K. Sharma. A novel approach for predicting user behavior for improving web performance. *nternational Journal on Computer Science and Engineering (IJCSE)*, 2(4):1233–1236, 2010.

[131] S. Makker and R. Rathy. Web server performance optimization using prediction prefetching engine. *International Journal of Computer Applications*, 23(9):19–24, June 2011. Published by Foundation of Computer Science.

[132] C. Makris, S. Stamou, E. Theodoridis, and V. Tzekou. A semantically enhanced web recommendation scheme. In *In Proceedings of LWA KDML 2011*, pages 1–7, 2011.

[133] U. Maulik and S. Bandyopadhyay. Performance evaluation of some clustering algorithms and validity indices. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(12):1650–1654, December 2002.

[134] G. Mazeroff, V. D. Cerqueira, J. Gregor, and M. G. Thomason. Probabilistic trees and automata for application behavior modeling. In *41st ACM Southeast Regional Conference Proceedings*, pages 435–440, 2003.

[135] W. S. Mc Culloch and W. Pitts. A logical calculus of the ideas immanent in nervous activity. In J. A. Anderson and E. Rosenfeld, editors, *Neurocomputing: Foundations of Research*, chapter A Logical Calculus of the Ideas Immanent in Nervous Activity, pages 15–27. MIT Press, Cambridge, MA, USA, 1988.

[136] M. Meilă. Comparing clusterings by the variation of information. In B. Schölkopf and M. K. Warmuth, editors, *Learning Theory and Kernel Machines*, volume 2777 of *Lecture Notes in Computer Science*, pages 173–187. Springer Berlin Heidelberg, 2003.

[137] G. W. Milligan and M. C. Cooper. An examination of procedures for determining the number of clusters in a data set. *Psychometrika*, 50(2):159–179, 1985.

[138] M. Minsky. Steps toward artificial intelligence. *Proceedings of the IRE*, 49(1):8–30, January 1961.

[139] B. Mirkin. *Clustering For Data Mining: A Data Recovery Approach (Chapman & Hall/Crc Computer Science)*. Chapman & Hall/CRC, 2005.

[140] MIT_Lincoln_Laboratory. DARPA intrusion detection evaluation dataset. `http://www.ll.mit.edu/mission/communications/cyber/CSTcorpora/ideval/data/` (Accessed on 2015/06/05), 1998.

[141] B. Mobasher and O. Nasraoui. *Web Data Mining: Exploring Hyperlinks, Contents, and Usage Data (Data-Centric Systems and Applications)*, chapter Web Usage Mining, pages 527–604. Springer-Verlag New York, Inc., 2006.

[142] V. S. Motegaonkar and M. V. Vaidya. A survey on sequential pattern mining algorithms. *International Journal of Computer Science & Information Technologies (IJCSIT)*, 5 (2):2486–2492, March 2014.

[143] A. Nanopoulos, D. Katsaros, and Y. Manolopoulos. Exploiting web log mining for web cache enhancement. In *Revised Papers from the Third International Workshop on Mining Web Log Data Across All Customers Touch Points*, WEBKDD '01, pages 68–87, London, UK, UK, 2002. Springer-Verlag.

[144] NASA-HTTP_logs. HTTP requests to the NASA kennedy space center WWW server in florida. `http://ita.ee.lbl.gov/html/contrib/NASA-HTTP.html` (Accessed on 2015/06/12), 1995.

[145] NASA_website. National aeronautics and space administration. `http://www.nasa.gov/` (Accessed on 2015/06/12), 2010.

[146] S. B. Needleman and C. D. Wunsch. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of Molecular Biology*, 48(3):443 – 453, 1970.

[147] P. B. Nemenyi. *Distribution-free Multiple Comparisons*. PhD thesis, Princeton University, 1963.

[148] S. Noh, G. Jung, K. Choi, and C. Lee. Compiling network traffic into rules using soft computing methods for the detection of flooding attacks. *Applied Soft Computing*, 8(3):1200–1210, 2008. Forging the Frontiers - - Soft Computing.

[149] N. Pal and J. Biswas. Cluster validation using graph theoretic concepts. *Pattern Recognition*, 30(6):847 – 857, 1997.

[150] B. Pan and D. R. Fesenmaier. Travel information search on the internet: A preliminary analysis. In *e-Review of Tourism Research (eRTR). Conference on Information and Communication Technologies in Tourism (ENTER 2013)*, Innsbruck, Austria, 2013.

[151] V. Paxson. Bro: A system for detecting network intruders in real-time. *Computer Networks*, 31(23-24):2435–2463, December 1999.

[152] J. Pearl. Bayesian networks: A model of self-activated memory for evidential reasoning. In *Proceedings of the 7th Conference of the Cognitive Science Society*, page 329334, Irvine, August 1985. University of California.

[153] J. Pei, J. Han, B. Mortazavi-Asl, J. Wang, H. Pinto, Q. Chen, U. Dayal, and M.-C. Hsu. Mining sequential patterns by pattern-growth: The prefixspan approach. *IEEE Trans. on Knowl. and Data Eng.*, 16(11):1424–1440, November 2004.

[154] D. Perera, J. Kay, I. Koprinska, K. Yacef, and O. R. Zaïane. Clustering and sequential pattern mining of online collaborative learning data. *IEEE Transactions on Knowledge and Data Engineering*, 21(6):759–772, June 2009.

[155] I. Perona, I. Albisua, O. Arbelaitz, I. Gurrutxaga, J. I. Martín, J. Muguerza, and J. M. Pérez. Histogram based payload processing for unsupervised anomaly detection systems in network intrusion. In L. S. Lopes, N. Lau, P. Mariano, and L. M. Rocha, editors, *New Trends in Artificial Intelligence. 14th Portuguese Conference on Artificial Intelligence. EPIA 2009. Aveiro, October 12-15, 2009. Proceedings*, pages 329–340, Aveiro, Portugal, October 2009. Universidade de Aveiro.

[156] I. Perona, O. Arbelaitz, I. Gurrutxaga, J. I. Martín, J. Muguerza, and J. M. Pérez. gureKDDcup99 datu-basearen sorrera (in basque). Technical Report EHU-KAT-IK-08-08, University of the Basque Country (UPV/EHU), May 2008.

[157] I. Perona, O. Arbelaitz, I. Gurrutxaga, J. I. Martín, J. Muguerza, and J. M. Pérez. Application of probabilistic suffix trees: payload analysis for nIDSs. Technical Report EHU-KAT-IK-08-09, University of the Basque Country (UPV/EHU), July 2009.

[158] I. Perona, O. Arbelaitz, I. Gurrutxaga, J. I. Martín, J. Muguerza, and J. M. Pérez. Unsupervised anomaly detection system for nidss based on payload and probabilistic suffix trees. In H. Weghorn and P. Isaías, editors, *Proceedings of the IADIS International Conference on Applied Computing*, Rome, Italy, November 2009.

[159] I. Perona, O. Arbelaitz, I. Gurrutxaga, and J. Muguerza. Towards web personalization for users with special needs using web navigation logs. In A. Troncoso and J. C. Riquelme, editors, *Actas del V Simposio de Teoría y Aplicaciones de Minería de Datos (TAMIDA 2010) - in English: Proceedings of the V Workshop on theory and applications of data mining*, Valencia, Spain, September 2010. Red Española de Minería de Datos y Aprendizaje.

[160] I. Perona, O. Arbelaitz, C. Makris, J. Muguerza, and E. Theodoridis. Improving link recommendation systems: clustering, sequential pattern mining and suffix trees. *Knowledge and Information Systems*, 2015.

[161] I. Perona, I. Gurrutxaga, O. Arbelaitz, J. I. Martín, J. Muguerza, and J. M. Pérez. Service-independent payload analysis to improve intrusion detection in network traffic. In J. F. Roddick, J. Li, P. Christen, and P. J. Kennedy, editors, *Seventh Australasian Data Mining Conference (AusDM 2008)*, volume 87 of *CRPIT*, pages 171–178, Darlinghurst, South Australia, November 2008. ACS.

[162] D. Pfitzner, R. Leibbrandt, and D. Powers. Characterization and evaluation of similarity measures for pairs of clusterings. *Knowledge and Information Systems*, 19(3):361–394, 2009.

[163] D. Pierrakos, G. Paliouras, C. Papatheodorou, and C. D. Spyropoulos. Web usage mining as a tool for personalization: A survey. *User Modeling and User-Adapted Interaction*, 13(4):311–372, November 2003.

[164] R. Popa and T. Levendovszky. Marcov models for web access prediction. *Magyar Kutatók 8. Nemzetközi Szimpóziuma 8th International Symposium of Hungarian Researchers on Computational Intelligence and Informatics*, pages 539–550, 2007.

[165] L. Portnoy, E. Eskin, and S. J. Stolfo. Intrusion detection with unlabeled data using clustering. In *In Proceedings of ACM CSS Workshop on Data Mining Applied to Security (DMSA 2001)*, pages 5–8, 2001.

[166] J. R. Quinlan. Induction of decision trees. *Machine Learning*, 1(1):81–106, 1986.

[167] W. M. Rand. Objective criteria for the evaluation of clustering methods. *Journal of American Statistical Association*, 66:846–850, 1971.

[168] M. Rehák, M. Pěchouček, K. Bartoš, M. Grill, P. Čeleda, and V. Krmíček. Improving anomaly detection error rate by collective trust modeling. In R. Lippmann, E. Kirda, and A. Trachtenberg, editors, *Recent Advances in Intrusion Detection (RAID)*, volume 5230 of *Lecture Notes in Computer Science*, pages 398–399. Springer Berlin Heidelberg, 2008.

[169] D. Ron, Y. Singer, and N. Tishby. The power of amnesia: Learning probabilistic automata with variable memory length. *Machine Learning*, 25(2-3):117–149, 1996.

[170] P. J. Rousseeuw. Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. *Journal of Computational and Applied Mathematics*, 20:53 – 65, 1987.

[171] R. A. Ruddle. Using string-matching to analyze hypertext navigation. In U. K. Wiil, P. J. Nrnberg, and J. Rubart, editors, *Hypertext*, pages 49–52. ACM, 2006.

[172] S. Saha and S. Bandyopadhyay. Performance evaluation of some symmetry-based cluster validity indexes. *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, 39 (Issue: 4):420 – 425, March 2009.

[173] S. Saitta, B. Raphael, and I. C. Smith. A bounded index for cluster validity. In P. Perner, editor, *Machine Learning and Data Mining in Pattern Recognition*, volume 4571 of *Lecture Notes in Computer Science*, pages 174–187. Springer Berlin Heidelberg, 2007.

[174] S. N. Schiaffino and A. Amandi. *Intelligent User Profiling.*, volume 5640 of *Lecture Notes in Computer Science*. Springer, 2009.

[175] SDSC-HTTP_logs. HTTP requests to the SDSC WWW server located at the san diego supercomputer center in san diego, california. `http://ita.ee.lbl.gov/html/contrib/SDSC-HTTP.html` (Accessed on 2015/06/12), 1995.

[176] SDSC_website. San diego supercomputer center. `http://www.sdsc.edu/` (Accessed on 2015/06/12), 2010.

[177] P. Senkul and S. Salin. Improving pattern quality in web usage mining by using semantic information. *Knowl. Inf. Syst.*, 30(3):527–541, 2012.

[178] F. Serratosa and A. Sanfeliu. Signatures versus histograms: Definitions, distances and algorithms. *Pattern Recognition*, 39(5):921–934, 2006.

[179] L. Sestorain, I. Perona, A. Yera, O. Arbelaitz, and J. Muguerza. Modification of SEP/COP, an efficient method to find the best partition in hierarchical clustering, to improve a link prediction system. In *Procedings of the XVII Conference of the Spanish Association for Artificial Intelligence*, Albacete, Spain, November 2015.

[180] J. P. Shaffer. Modified sequentially rejective multiple test procedures. *Journal of the American Statistical Association*, 81(395):826–831, 1986.

[181] T. Slimani and A. Lazzez. Efficient analysis of pattern and association rule mining approaches. *Computing Research Repository (CoRR)*, abs/1402.2892, 2014.

[182] T. F. Smith and M. S. Waterman. Identification of common molecular subsequences. *Journal of Molecular Biology*, 147(1):195 – 197, 1981.

[183] P. H. A. Sneath and R. R. Sokal. *Numerical Taxonomy. The Principles and Practice of Numerical Classification.* Freeman, 1973.

[184] J. Souza, S. Matwin, and N. Japkowicz. Evaluating data mining models: A pattern language. In *9th Conference on Pattern Language of Programs (PLoP 2002)*, Monticello, Illinois, September 2002.

[185] H. Späth. *Cluster analysis algorithms for data reduction and classification of objects.* Computers and their applications. Halsted Press, Chichester, Eng. E. Horwood New York, 1980. Translation of Cluster-Analyse-Algorithmen zur Objektklassifizierung und Datenreduktion.

[186] R. Srikant and R. Agrawal. Mining quantitative association rules in large relational tables. *SIGMOD Record*, 25(2):1–12, June 1996.

[187] J. Srivastava, P. Desikan, and V. Kumar. Web mining concepts, applications and research directions. In W. Chu and T. Young Lin, editors, *Foundations and Advances in Data Mining*, volume 180 of *Studies in Fuzziness and Soft Computing*, pages 275–307. Springer Berlin Heidelberg, 2005.

[188] A. Steinbauer and H. Werthner. *Information and Communication Technologies in Tourism 2007 - Proceedings of the International Conference in Ljubljana, Slovenia, 2007*, chapter Consumer Behaviour in e-Tourism, pages 65–76. Springer Vienna, 2007.

[189] C. A. Sugar and G. M. James. Finding the number of clusters in a data set: An information theoretic approach. *Journal of the American Statistical Association*, 98:750–763, 2003.

[190] P. Sun, S. Chawla, and B. Arunasalam. Mining for outliers in sequential databases. In *Proceedings of the Sixth SIAM International Conference on Data Mining, April 20-22, 2006, Bethesda, MD, USA*, pages 94–105, 2006.

[191] S. Vijayarani and S. Deepa. Article: Sequential pattern mining - a study. *IJCA Proceedings on International Conference on Research Trends in Computer Technologies 2013*, ICRTCT(1):14–18, February 2013.

[192] U. Von Luxburg, R. C. Williamson, and I. Guyon. Clustering: Science or art? In I. Guyon, G. Dror, V. Lemaire, G. W. Taylor, and D. L. Silver, editors, *ICML Unsupervised and Transfer Learning*, volume 27 of *JMLR Proceedings*, pages 65–80, Vancouver, Canada, 2012. JMLR.org.

[193] W3C. Common log format (clf). `http://www.w3.org/Daemon/User/Config/Logging.html#common-logfile-format`, 1995. The World Wide Web Consortium (W3C).

[194] Y. Waizumi, M. Tsuji, H. Tsunoda, N. Ansari, and Y. Nemoto. Distributed early worm detection based on payload histograms. In *Communications, 2007. ICC '07. IEEE International Conference on*, pages 1404–1408, June 2007.

[195] K. Wang and S. J. Stolfo. Anomalous payload-based network intrusion detection. In E. Jonsson, A. Valdes, and M. Almgren, editors, *Recent Advances in Intrusion Detection*, volume 3224 of *Lecture Notes in Computer Science*, pages 203–222. Springer Berlin Heidelberg, 2004.

[196] J. H. Ward. Hierarchical grouping to optimize an objective function. *Journal of the American Statistical Association*, 58(301):236–244, 1963.

[197] C. Warrender, S. Forrest, and B. Pearlmutter. Detecting intrusion using system calls: alternative data models. In *In Proceedings of the IEEE Symposium on Security and Privacy*, 1999.

[198] S. Wehner. Analyzing worms and network traffic using compression. *J. Comput. Secur.*, 15(3):303–320, August 2007.

[199] I. H. Witten and E. Frank. *Data Mining: Practical Machine Learning Tools and Techniques, Second Edition (Morgan Kaufmann Series in Data Management Systems)*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2005.

[200] X. Wu, V. Kumar, J. Ross Quinlan, J. Ghosh, Q. Yang, H. Motoda, G. McLachlan, A. Ng, B. Liu, P. Yu, Z.-H. Zhou, M. Steinbach, D. Hand, and D. Steinberg. Top 10 algorithms in data mining. *Knowledge and Information Systems*, 14(1):1–37, 2008.

[201] J. Yang and W. Wang. Towards automatic clustering of protein sequences. In *IEEE Computer Society Bioinformatics Conference (CSB)*, pages 175–186. IEEE Computer Society, 2002.

[202] M. J. Zaki. Spade: An efficient algorithm for mining frequent sequences. *Mach. Learn.*, 42(1-2):31–60, Jan. 2001.

[203] K. R. Žalik and B. Žalik. Validity index for clusters of different sizes and densities. *Pattern Recognition Letters*, 32(2):221 – 234, 2011.

[204] Q. Zhao, S. S. Bhowmick, and L. Gruenwald. Wam-miner: In the search of web access motifs from historical web log data. In *Proceedings of the 14th ACM International Conference on Information and Knowledge Management*, CIKM '05, pages 421–428, New York, NY, USA, 2005. ACM.

Part V

Appendix

Appendix A

KDDCup99 database's attributes

The KDDCup99 database has 41 predictor attributes for each connection and the class attribute defining the type of the connection is attack or normal. Those attributes are divided in 3 groups: intrinsic attributes, content attributes and traffic attributes.

- Intrinsic attributes (I): There are 9 attributes and they are computed using the fields in the headers' area of the network packages. Those connection attributes are the following ones: (1) duration, (2) protocol type, (3) service name, (4) flag or status, (5) source bytes or amount of sent bytes, (6) destination bytes or amount of received bytes, (7) if there are landed packages or if we are sending packages to ourselves, (8) number of wrong fragments and lastly, (9) number of urgent packages. All previous attributes are directly available in the header fields without doing any processing or doing little processing.

- Content attributes (C): There are 13 attributes and they are computed finding signatures in the content area of the network packages. The payload or content area of the packages is service dependant, so those critical signatures are also service dependant and moreover they are proposed or created by experts' using their knowledge in network security. Those connection attributes are the following ones: (10) hot, (11) num_failed_logins, (12) logged_in, (13) num_compromised, (14) root_shell, (15) su_attempted, (16) num_root, (17) num_file_creations, (18) num_shells, (19) num_access_files, (20) num_outbound_cmds, (21) is_hot_login, and (22) is_guest_login.

- Traffic attributes (T): There are 19 attributes and they are computed using the connections preceding the current connection. There are two criteria to determine the number of previous connections that are needed to compute these attributes: (1) time traffic features (9 features) (2) machine traffic features (10 features). The difference between the former

group and the latter is the mode to select the previous connections. To calculate time traffic attributes the connections that occurred in the past 2 seconds were considered, while to calculate machine traffic attributes the previous 100 connections were taken into account. Therefore, to calculate these attributes unlike the previous ones, we need the information about a group of connections to be able to calculate them.

Time traffic attributes are the following ones: (23) count, (24) srv_count, (25) serror_rate, (26) srv_serror_rate, (27) rerror_rate, (28) srv_error_rate, (29) same_srv_rate, (30) diff_srv_rate, and (31) srv_diff_host_rate.

Machine traffic attributes are the following ones: (32) dst_host_count, (33) dst_host_srv_count, (34) dst_host_same_srv_rate, (35) dst_host_diff_srv_rate, (36) dst_host_same_src_port_rate, (37) dst_host_srv_diff_host_rate, (38) dst_host_serror_rate, (39) dst_host_srv_serror_rate, (40) dst_host_rerror_rate, and (41) dst_host_srv_error_rate.

Now all 41 attributes of KDDCup99 database will be defined and explained how they are computed. For more information about the realization of them look the internal report "Generation of the database gurekddcup" [156].

1. I.duration: Duration of the connection in seconds, therefore, the attribute is of type integer.

2. I.protocol_type: Data transmission protocol: TCP, UDP or ICMP, therefore this attribute's type is nominal.

3. I.service: Application protocol, i.e., service name, such as, http, ftp, smtp, telnet... and other (if not very used service). This attribute is nominal.

4. I.flag: The state of the connection. This attribute is nominal. The different states or situations the connection can reach are the following ones:

   - SF: The connection has started and finished correctly.
   - S0: The request to establish the connection has not received any response.
   - S1: The connection has started (or established) but it has not finished.
   - S2: The connection has started (or established) and the source-side machine has requested to close it but the destination-side machine does not respond.
   - S3: The connection has started (or established) and the destination--side machine has requested to close it but the sourceside machine does not respond.
   - OTH: There have been no steps to establish the connection but traffic packages have been detected.
   - REJ: The request to establish the connection has been rejected.

- RSTO: The connection has started (or established) correctly but the source-side machine has aborted using the RST (reset flag) field of the datagram.
- RSTR: The connection has started (or established) correctly but the destination-side machine has aborted using the RST (reset flag) field of the datagram.
- RSTOS0: The source machine has sent a SYN signal and then a RST signal. However the destination machine has not responded with SYNACK signal.
- RSTRH: The destination machine sends a SYNACK signal and then a RST signal. In this case the origin machine did not send the SYN signal.
- SHR: The destination machine sends a SYNACK signal and then a FIN signal. In this case the origin machine did not send the SYN signal.

The UDP and ICMP packages do not have an strict control of transmitted packages, so, they differ from TCP, thus, they trigger up 4 states: SF, S0, SHR and OTH.

5. I.src_bytes: the variables src_bytes and dst_bytes will be explained together in the next point.

6. I.dst_bytes: src_bytes indicates the number of bytes sent from the origin to the destination machine while dst_bytes denotes the bytes sent from the destination to the origin machine.

7. I.land: '1' if the source and destination port and IP addresses are the same in one connection, otherwise '0'. This attribute was generated precisely to detect the land attack which is a DoS type attack. This attack sends a lot of request packages to the service provider machine with the same origin and destination addresses.

8. I.wrong_fragment: number of wrong fragments in a connection. According to Lee's thesis [122] if the length of the package (number of bytes) is not multiple of 8 this package will have a wrong fragment. Therefore, a TCP connection can have many wrong fragments while UDP an ICMP connections at most 1. However, the values obtained with this criteria were much bigger than in KDDCup99, thus, we proved other interpretations and saw that to count the strings "bad_ICMP_checksum", "bad_TCP_-checksum" and "bad_UDP_checksum" in the attribute "name" of the bro event "conn_weird(name: string, c:connection)" obtained preferable range of values.

9. I.urgent: The number of urgent packages transmitted, that is to say, the packages which have the urgent bit activated.

10. C.hot: number of hot actions in a connection. According to Lee's thesis [122] is defined as a hot action entering into the system's directories, creating a program and running it.

11. C.num_failed_logins: number of failed logins in a connection.

12. C.logged_in: it takes two values, 1 if the loggings were correct in a connection, 0 otherwise.

13. C.num_compromised: Lee's thesis [122] defines this attribute as the number of "file path not found" error in a connection. To meet KDDCup99 values we relaxed the error string and found "not found" errors.

14. C.root_shell: it takes two values, 1 if the user obtained the root shell in a connection and 0 otherwise.

15. C.su_attempted: it takes two values, 1 if the user used the "su" command in a connection and 0 otherwise. However, analysing this attribute in KDDCup99 we noticed that it takes also the value 2, so we decided to redefine the attribute as the times the user tries the command "su".

16. C.num_root: the number of operations done in root mode in a connection.

17. C.num_file_creations: number of times a file is created in a connection.

18. C.num_shells: the number of prompt lines used in a connection. This was interpreted as follows: number of logins of normal users (number of no root shells the user obtains).

19. C.num_access_files: number of operations done in control files in a connection. It was not clear how to implement this definition because the control files can be all human-readable files that are not in home folder, however some files hanging from homeuser are also control files such as .bashrc. Therefore, we needed to define a list of control files appeared and check if they were accessed or not.

20. C.num_outbound_cmds: number of outbound commands in a FTP session. The outbound option of ftp is one of many methods FTP has to send files to FTP servers and is based in RCP (remote copy). This option does not encrypt the user-name and the password. And as a consequence, it is not recommended to use it. Being dangerous to work with outbound, in KDDCup99 was included via this attribute.

21. C.is_hot_login: it takes two values, 1 if the user is logged in with superuser privileges, i.e., as a root or adm; 0 otherwise.

22. C.is_guest_login: it takes two values, 1 if the user is logged in with basic permissions, with no privileges, i.e., as a guest, anonymous or visitor; and 0 otherwise.

23. T.count: number of connections to the same destination IP address as the current connection's destination IP address (in the last 2 seconds).

24. T.srv_count: number of connections to the same destination port number as the current connection (in the last 2 seconds).

25. T.serror_rate: percentage of connections that satisfied the condition of count attribute and also the "SYN" error occurred (in the last 2 seconds). Bro offers methods to detect synchronization errors, but using them we did not meet KDDCup99 values, thus, we considered the connections with the flag attribute S0, S1, S2 or S3. Because those values happen when there is a problem with the starting or ending process of the connection.

26. T.srv_serror_rate: percentage of connections that satisfied the condition of srv_count attribute and also the "SYN" error occurred (in the last 2 seconds). The SYN error was defined the same way it was done for attribute serror_rate.

27. T.rerror_rate: percentage of connections that satisfied the condition of count attribute and were also "REJ" connections (in the last 2 seconds). To be a REJ connection is to have the value REJ in the flag attribute.

28. T.srv_error_rate: percentage of connections that satisfied the condition of srv_count attribute and are also "REJ" connections (in the last 2 seconds).

29. T.same_srv_rate: percentage of connections that satisfied the condition of count attribute and were also using the same service as the current one (in the last 2 seconds).

30. T.diff_srv_rate: percentage of connections that satisfied the condition of count attribute and were NOT using the same service as the current one (in the last 2 seconds). It seems that the value of this attribute should be computed by doing 1-same_srv_rate, in this case the information would be repeated or 100% correlated in the database. However, this is not the case because we assumed that the service-name "other" corresponds to different service-names.

31. T.srv_diff_host_rate: percentage of connections that satisfied the condition of srv_count attribute added to the connections that have different destination IP addresses to the current one (in last 2 seconds).

32. T.dst_host_count: number of connections to the same destination IP address as the current connection's destination IP address (in the last 100 connections).

33. T.dst_host_srv_count: number of connections to the same destination port number as the current connection's destination port number (in the last 100 connections).

34. T.dst_host_same_srv_rate: percentage of connections that satisfied the condition of dst_host_count attribute and are also using the same service as the current one (in the last 100 connections).

35. T.dst_host_diff_srv_rate: percentage of connections that satisfied the condition of dst_host_count attribute and were NOT using the same service as the current one (in the last 100 connections).

36. T.dst_host_same_src_port_rate: percentage of connections that satisfied the condition of dst_host_count and were also connected to the same source port number as the current connection (in the last 100 connections).

37. T.dst_host_srv_diff_host_rate: percentage of connections that satisfied the condition of dst_host_srv_count attribute and also the connections that have different destination IP addresses to the current one (in the last 100 connections).

38. T.dst_host_serror_rate: percentage of connections that satisfied the condition of dst_host_count attribute and also the "SYN" error occurred (in the last 100 connections).

39. T.dst_host_srv_serror_rate: percentage of connections that satisfied the condition of dst_host_srv_count attribute and also the "SYN" error occurred (in the last 100 connections).

40. T.dst_host_rerror_rate: percentage of connections that satisfied the condition of dst_host_count attribute and were also "REJ" connections (in the last 100 connections).

41. T.dst_host_srv_error_rate: percentage of connections that satisfied the condition of dst_host_srv_count attribute and were also "REJ" connections (in the last 100 connections).

Appendix B

Impact of gureKDDCup database

According to the provided data, the entities and universities that have been interested with the gureKDDCup database (checked on 01/12/2015).

1. Aberystwyth University[1], Wales, 1 access.
2. ABV-Indian Institute of Information Technology[2], India, 1 access.
3. Amrita Vishwa Vidyapeetham (Amrita University)[3], India, 2 accesses.
4. Banasthali University[4], India, 1 access.
5. Bharat Institute of Engineering and Technology (BIET)[5], India, 1 access.
6. Bigtapp Analytics[6], Singapore, 4 accesses.
7. Biju Patnaik University of Technology (BPUT)[7], India, 1 access.
8. Birla Institute of Technology and Science (BITS Pilani)[8], India, 1 access.
9. Cairo University[9], Egypt, 4 accesses.
10. Central University of Tamil Nadu[10], India, 1 access.
11. Charotar University of Science and Technology[11], India, 2 accesses.
12. Christ University[12], India, 2 accesses.

[1] http://www.aber.ac.uk/en/
[2] http://www.iiitm.ac.in/
[3] https://www.amrita.edu/
[4] http://www.banasthali.org
[5] http://www.biet.ac.in/
[6] http://bigtappanalytics.com/
[7] http://www.bput.ac.in/
[8] http://www.bits-pilani.ac.in/
[9] http://cu.edu.eg/Home
[10] http://www.cutn.ac.in/
[11] http://www.charusat.ac.in/
[12] http://www.christuniversity.in/

13. Damghan University[13], Iran, 2 accesses.
14. Easwari Engineering College[14], India, 1 access.
15. École nationale supérieure des télécommunications de Bretagne[15], France, 1 access.
16. Edinburgh Napier University[16], United Kingdom, 1 access.
17. Electronic Engineering Polytechnic Institute of Surabaya[17], Indonesia, 3 accesses.
18. ETRI: Electronics and Telecomunications Research Institute[18], Korea, 2 accesses.
19. Federal University of Lavras[19], Brazil, 2 access.
20. Gebze University of Technology[20], Turkey, 1 access.
21. George Washington University[21], United States, 1 access.
22. Indiana University - Purdue University Indianapolis[22], United States, 1 access.
23. Indian Institute of Technology Patna (IIT Patna)[23], India, 10 accesses.
24. Islamic University of Gaza[24], Palestine, 1 access.
25. Jayoti Vidyapeeth Women's University[25], India, 1 access.
26. Jekdnfjaks[26], United States, 1 access.
27. Kalasalingam University[27], India, 2 accesses.
28. Khon Kaen University (KKU)[28], Thailand, 4 accesses.
29. King Saud University[29], Saudi Arabia, 1 access.
30. Korea Advanced Institute of Science and Technology (KAIST)[30], South Korea, 1 access.
31. Kumaun University[31], India, 1 access.

[13] http://du.ac.ir/en/
[14] http://srmeaswari.ac.in/
[15] http://www.telecom-bretagne.eu/
[16] http://www.napier.ac.uk/
[17] http://www.pens.ac.id/
[18] https://www.etri.re.kr/eng/main/main.etri
[19] http://www.ufla.br/
[20] http://www.gyte.edu.tr/
[21] http://www.gwu.edu/
[22] http://www.iupui.edu/
[23] http://www.iitp.ac.in/
[24] http://www.iugaza.edu.ps/en/
[25] http://www.jvwomensuniv.com/
[26] http://www.jenjacs.com/
[27] http://www.kalasalingam.ac.in/
[28] https://www.kku.ac.th/?l=en
[29] http://ksu.edu.sa/en/
[30] http://www.kaist.edu/
[31] http://www.kunainital.ac.in/

32. Le Quy Don Technical University[32], Vietnam, 1 access.
33. London Metropolitan University[33], England, 1 access.
34. Madhav Institute of Technology & science[34], India, 2 accesses.
35. McGill University[35], Canada, 1 accesses.
36. Military University Nueva Granada[36], Colombia, 1 access.
37. Mohamed Premier University[37], Morocco, 1 access.
38. Multimedia University[38], Malaysia, 3 accesses.
39. Mumbai University[39], India, 2 accesses.
40. Narsee Monjee Institute of Management Studies[40], India, 1 access.
41. National Institute of Technology Rourkela[41], India, 4 accesses.
42. National Institute of Technology, Tiruchirappalli[42], India, 4 accesses.
43. National Taiwan University[43], China, 4 accesses.
44. National University of Mongolia[44], Mongolia, 4 accesses.
45. National University of Science and Technology MISiS[45], Russia, 2 accesses.
46. North Maharashtra University[46], India, 2 accesses.
47. Nova Southeastern University[47], United States, 8 accesses.
48. Osmania University[48], India, 1 access.
49. Pablo de Olavide University[49], Spain, 1 access.
50. Pace University[50], United States, 1 access.
51. Pukyong National University[51], South Korea, 1 access.
52. Purdue University[52], United States, 1 access.

[32] http://mta.edu.vn/
[33] http://www.londonmet.ac.uk/
[34] http://mitsgwalior.in/
[35] https://www.mcgill.ca/
[36] http://www.umng.edu.co/
[37] http://www.ump.ma/?lang=en
[38] https://www.mmu.edu.my/
[39] http://mu.ac.in/portal/
[40] http://www.nmims.edu/
[41] www.nitrkl.ac.in
[42] http://www.nitt.edu/
[43] http://www.ntu.edu.tw/english/
[44] http://www.num.edu.mn/en/
[45] http://en.misis.ru/
[46] http://www.nmu.ac.in/
[47] http://www.nova.edu/
[48] http://www.osmania.ac.in/
[49] https://www.upo.es
[50] http://www.pace.edu/
[51] http://www.pknu.ac.kr/usrEngIndex.do
[52] http://www.purdue.edu/

53. Rajabhat University[53], Thailand, 1 access.
54. Rajagiri School of Engineering and Technology[54], India, 1 access.
55. Rensselaer Polytechnic Institute[55], United States, 1 access.
56. Rutgers University[56], United States, 1 access.
57. Sadat Academy for Management Sciences[57], Egypt, 1 access.
58. Sardar Vallabhbhai National Institute of Technology, Surat[58], India, 4 accesses.
59. Sepuluh Nopember Institute of Technology[59], Indonesia, 2 accesses.
60. Shahid Beheshti University[60], Iran, 4 accesses.
61. Shahid Chamran University of Ahvaz[61], Iran, 7 accesses.
62. Shanghai University[62], China, 1 access.
63. Sheffield Hallam University[63], United Kingdom, 1 access.
64. Sriwijaya University[64], Indonesia, 1 access.
65. St. Joseph Engineering College (SJEC)[65], India, 1 access.
66. Suez Canal University[66], Egypt, 1 access.
67. Tennessee State University[67], United States, 1 access.
68. The Quaid-i-Azam University[68], Pakistan, 1 access.
69. Universidade Federal de Goiás[69], Brazil, 4 access.
70. Universitas Gadjah Mada[70], Indonesia, 1 access.
71. Université libre de Bruxelles[71], Belgium, 1 access.
72. University of Colorado Boulder[72], United States, 1 access.
73. University of Hertfordshire[73], United Kingdom, 1 access.

[53] http://www.pkru.ac.th/eng/
[54] http://rajagiritech.ac.in
[55] http://rpi.edu/
[56] http://www.rutgers.edu
[57] http://www.sams.edu.eg/
[58] http://www.svnit.ac.in/
[59] http://www.its.ac.id/en/
[60] http://http://en.sbu.ac.ir
[61] http://www.scu.ac.ir/
[62] http://www.shu.edu.cn/
[63] http://www.shu.ac.uk/
[64] http://www.unsri.ac.id/
[65] http://www.sjec.ac.in/
[66] http://scuegypt.edu.eg/en/
[67] http://www.tnstate.edu/
[68] https://www.qau.edu.pk
[69] http://www.ufg.br/
[70] http://www.ugm.ac.id/en/
[71] http://www.ulb.ac.be/
[72] http://www.colorado.edu/
[73] http://www.herts.ac.uk/

74. University of Indonesia[74], Indonesia, 1 access.
75. University of KwaZulu-Natal[75], South Africa, 1 access.
76. University of Louisville[76], United States, 1 access.
77. University of Malaya[77], Malaysia, 1 access.
78. University of Maryland[78], United States, 2 accesses.
79. University of Nottingham[79], United Kingdom, 1 access.
80. University of Pretoria[80], South Africa, 1 access.
81. University of Saida[81], Algeria, 2 accesses.
82. University of Southern California[82], United States, 1 access.
83. University of South Florida[83], United States, 1 access.
84. University of Tsukuba[84], Japan, 1 access.
85. Visvesvaraya Technological University[85], India, 4 accesses.
86. VIT University[86], India, 4 access.

[74] http://www.ui.ac.id/
[75] http://www.ukzn.ac.za/
[76] http://louisville.edu/
[77] http://www.um.edu.my/
[78] http://www.umd.edu/
[79] https://www.nottingham.ac.uk/
[80] http://www.up.ac.za/
[81] http://www.univ-saida.dz/
[82] http://www.usc.edu/
[83] http://www.usf.edu/
[84] https://www.tsukuba.ac.jp/english/
[85] http://vtu.ac.in/
[86] http://www.vit.ac.in/