2012

A Work-Pattern Centric Approach to Building a Personal Knowledge Advantage Machine

Daniel Sloan
West Virginia University

A Work-Pattern Centric Approach to Building a Personal Knowledge Advantage Machine

Daniel Sloan

Thesis submitted to the
College of Engineering and Mineral Resources
at West Virginia University
in partial fulfillment of the requirements
for the degree of

Master of Science
in
Computer Science

Yenumula V. Reddy, Ph.D., Chair
Bojan Cukic, Ph.D.
Cynthia D. Tanner, MS.

Lane Department of Computer Science and Electrical Engineering

Morgantown, West Virginia
2012

Keywords: Work-patterns, file usage, semantic desktop, machine learning

Abstract

A Work-Pattern Centric Approach to Building a Personal Knowledge Advantage Machine

Daniel Sloan

A work pattern, also known as a usage pattern, can be broadly defined as the methods by which a user typically utilizes a particular system. Data mining has been applied to web usage patterns for a variety of purposes. This thesis presents a framework by which data mining techniques could be used to extract patterns from an individual's work flow data in order facilitate a new type of architecture known as a knowledge advantage machine. This knowledge advantage machine is a type of semantic desktop and semantic web application that would assist people in constructing their own personal knowledge networks, as well as sharing that information in an efficient manner with colleagues using the same system. A knowledge advantage machine would be capable of automatically discovering new knowledge which is relevant to the user's personal ontology.

Through experimentation, we demonstrate that a user's file usage patterns can be utilized by software in order to automatically and seamlessly learn what is "important" as defined by the user. Further research is necessary to apply this principle to a more realized knowledge advantage machine such that decisions can be fueled by work patterns as well as semantic or contextual information.

Dedication

I dedicate this work to my grandfather.

Spoczywaj w pokoju.

Acknowledgments

I would like to thank my colleagues and friends who have inspired me, motivated me, and overall have been there for me even when things seemed bleak. To those who have inspired my best and weathered my worst: Bryan Lemon, Jonathan Lynch, Brian Cain, Alex Dauphin, Lisa Soros.

I would also like to thank Dr. Reddy, without whom this thesis would not have been possible. He has taught me the importance of lateral thinking, especially when dealing with such complex and pervasive problems as the ones described in this work.

Thanks are given to all of my wonderful volunteers, who gave up their valuable time and hard drive space to help me obtain the empirical data that drives this thesis.

I give thanks to those instructors that I've worked under in the capacity of teaching assistantship: Mrs. Tanner, Mrs. Hayhurst, and, again, Dr. Reddy. I have learned as much in the process of teaching computer science as I did in my time as an undergraduate.

I must thank all of the truly excellent professors I have had the pleasure of learning from throughout my years at WVU. The professionalism, character, and strength of personality within the Lane Department of Computer Science and Electrical Engineering cannot possibly be overstated.

Finally, I thank Dr. Katerina Goseva-Popstojanova, whose Empirical Methods class provided me with the extra kick I needed to get this thing done.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Knowledge workers' jobs continue to increase in both their complexity and their plenitude of associated information; this is sometimes referred to as "information overload" [3], and is a well known problem, especially in technical fields. Therefore, a need for a mechanism capable of semantically organizing that information is apparent. In order to combat this problem we must develop a technology which we will refer to as a Knowledge Advantage Machine.

The "knowledge advantage" of a KAM comes from the similar but better-known idea of a "mechanical advantage," [9] wherein the force one uses to accomplish some physical task is reduced through the clever use of some mechanism or tool. A KAM would utilize a semantic link network [45] in order to better organize and utilize information for a given knowledge worker. Such a system would probably require some initial work on the user's part [40] to set up, but the goal would be that the resulting "knowledge advantage" would counteract this effort.

A KAM would consist of several functional parts, referred to as "agents." Agents are simply modules which act to perform some task; for a KAM, the most important agents are called Selection, Organization, Context Map, Display, and Collaboration. An envisioning of a KAM consisting of these five major agents is detailed in depth in §1.2.

The necessary building blocks of a Knowledge Advantage Machine exist in an emerging field;

for example, the semantic link networking model which works to establish semantic links across a variety of resources and is meant to be used as a semantic representation model for the Semantic Web, Semantic Grid, and the Knowledge Grid [45]. Long before this, however, the concept of an ontology itself was examined, such as in Gruber's 1995 work wherein methods of representing an ontology were explored [13].

A semantic link network, in its semantic organization of information for knowledge workers, would allow knowledge workers to perform their jobs with added ease and efficiency [40]. It follows logically that any properly designed Knowledge Advantage Machine would, in theory, facilitate the semantic link network in a manner which is more comfortable for knowledge workers to use than the traditional methods of information gathering and application.

The idea of monitoring usage patterns has long been used for websites, especially commerce sites, in order to present a more user-tailored experience [38]. File usage has also been recorded and statistically analyzed [39], but little has been done in the vein of examining or exploiting these file usage patterns for the sake of a knowledge advantage.

To this end, the purpose of this thesis is to present a framework for a KAM system which focuses upon desktop usage patterns in order to intuit ontological importance. This framework is constructed in order to demonstrate one possible basis for a knowledge advantage machine's ability to accelerate and enrich the daily tasks of any general knowledge worker, while providing an overview of the current state-of-the-art research into the various "building blocks" of the KAM.

Most specifically, this thesis offers a work pattern recognition tool which is capable of monitoring and recording file usage data, and make predictions about future file usage. I will empirically show the amount of accuracy that these classification learning based predictions offer.

1.1 Problem Statement

The problem addressed by this thesis can be expressed as follows: ”How can work-pattern centric agents augment the KAM’s ability to provide its eponymous knowledge advantage?” If accurate, these predictions could be used to assist in the identification of context. Because the KAM is predicated upon being able to determine needed JANs[1] for a given context, alternative methods of context identification could prove very useful.

The problem that this thesis addresses is not that of creating a generic knowledge advantage machine, though the concept will be explored and a case will be made for its importance. Instead, the problem requires us to specifically explore the data collection and pattern discovery agents of a KAM which is capable of functioning partially or wholly based on work patterns. This thesis therefore addresses the problem of implementing and testing what we refer to as a ”Work-Pattern Centric Approach to Building a Personal Knowledge Advantage Machine” with the ultimate main goal of improving context prediction.

1.1.1 Sample Scenario

It is perhaps easiest to explan the concept of a knowledge advantage machine by presenting a scenario in which it assists an average knowledge worker. Let us take the example of a professor, ”Dr. Doe.” Like most professors, Dr. Doe does research, but he also teaches. These categories - teaching and researching - would constitute two high-level ”contexts” of his overall knowledge base. Within the context of research, Dr. Doe might be examining several different papers of varying topics such as nanotechnology or microprocessors; these would also constitute contexts, or, alternately, sub-contexts.

A fully idealized knowledge advantage machine would be aware of these contexts at any given time. This awareness could be achieved in a variety of ways; for the purposes of this example, we

[1] See §1.2

assume it exists and is more or less accurate. Therefore, if Dr. Doe was currently using his personal computer to perform research into nanotechnology, he would be considered to be currently in the ”research” context. While these contexts represent states, they are not mutually exclusive; for example, a person can be doing research and writing a syllabus at the same time. Ideally, contexts could be tracked on-the-go, perhaps via mobile-devices, as well; Dr. Doe's personal KAM would be aware of his presence at a nanotechnology research meeting, therefore identifying him as currently residing in both the research and nanotechnology contexts. This idea is called ”context aware computing” [5] and while it is true that several context-aware applications have been been built as demonstrations, mass-availability of this sort of technology is currently rare. Some exceptions do exist, such as mobile phone applications which alter phone behavior based on location [23].

Because Dr. Doe's KAM is constantly aware of his context, and the context of documents within his personal ontology, it is able to to perform several useful tasks. When Dr. Doe sits down to write a paper concerning nanotechnology, his KAM is constantly performing semantic analysis on his emerging document and concurrently searching for papers most related to that specific content. When Dr. Doe receives an email from a student requesting a syllabus for a particular class, a fully realized KAM would be ”smart” enough to send the appropriate reply. This is because the KAM would, after examining the email, be privy to several facts; most importantly, the sender's email address identifies the sender as a student in a particular class, and the semantic content of the email points to the user requesting a syllabus document.

By combining context aware computing with a semantically understood personal ontology, a KAM would be able to perform many such time-saving knowledge-intensive tasks. However, this is a complicated and vast problem, so the purpose of this thesis is limited to examining one possible way in which context could be determined. Under the system detailed in this research, Dr. Doe's context would be identified by his patterns of file access. With accurate file usage prediction, Dr. Doe might also be offered a pane of files that he is likely to access in the near future.

1.2 Knowledge Advantage Machine

The concept of a Knowledge Advantage Machine is a combination of several other discrete ideas. In 2011 Luyi Wang et al. [40] published a context-centric approach to creating a knowledge advantage machine. This paper provides a basis for the problem; a KAM can be succinctly defined as any device which increases the efficiency at which a knowledge worker performs daily tasks.

Likewise, we aim to reduce the force exerted by a knowledge worker to accomplish tasks in their fields. Applications of a KAM are broad: a chemical engineer's KAM might automatically discover exciting new research papers in the field of chemistry and insert them into his ontology for perusal, while an instructor's KAM might automatically prepare draft email replies to students who have sent requests for syllabi or assignments. A fully realized KAM would allow for increased collaboration; for example, two computer scientists might share their ontological knowledge bases and end up padding their ontologies with extra knowledge in the context of algorithmic analysis.

Figure 1.1 depicts an objective map wherein we might divide a generic idealized KAM into five basic modules, or "agents": selection, organization, context map, display, and collaboration. Here, we also introduce the concept of a "JAN," or atom of knowledge, described by Wang et al. [40] as an "abstract object for all the general resources". The idea of a JAN is not entirely new, and, indeed, has existed under different names in the past; one might liken a JAN to a "knol" from Google's eponymous Knol project [2]. This project was intended to provide user-written scholarly articles, and "knols" were literally defined by Google as "a unit of knowledge." The term was also used to refer simply to an article within the Knol project. However, as of May, 2012, the Knol project has been discontinued [19] in favor of an upcoming project called Annotum.

The Selection Agent would be primarily concerned with how the user selects and views portions of the personal ontology, including JANs. This agent represents the "user interface" by which one directly interacts with the KAM system.

The Organization Agent would be concerned with intelligently organizing these JANs into

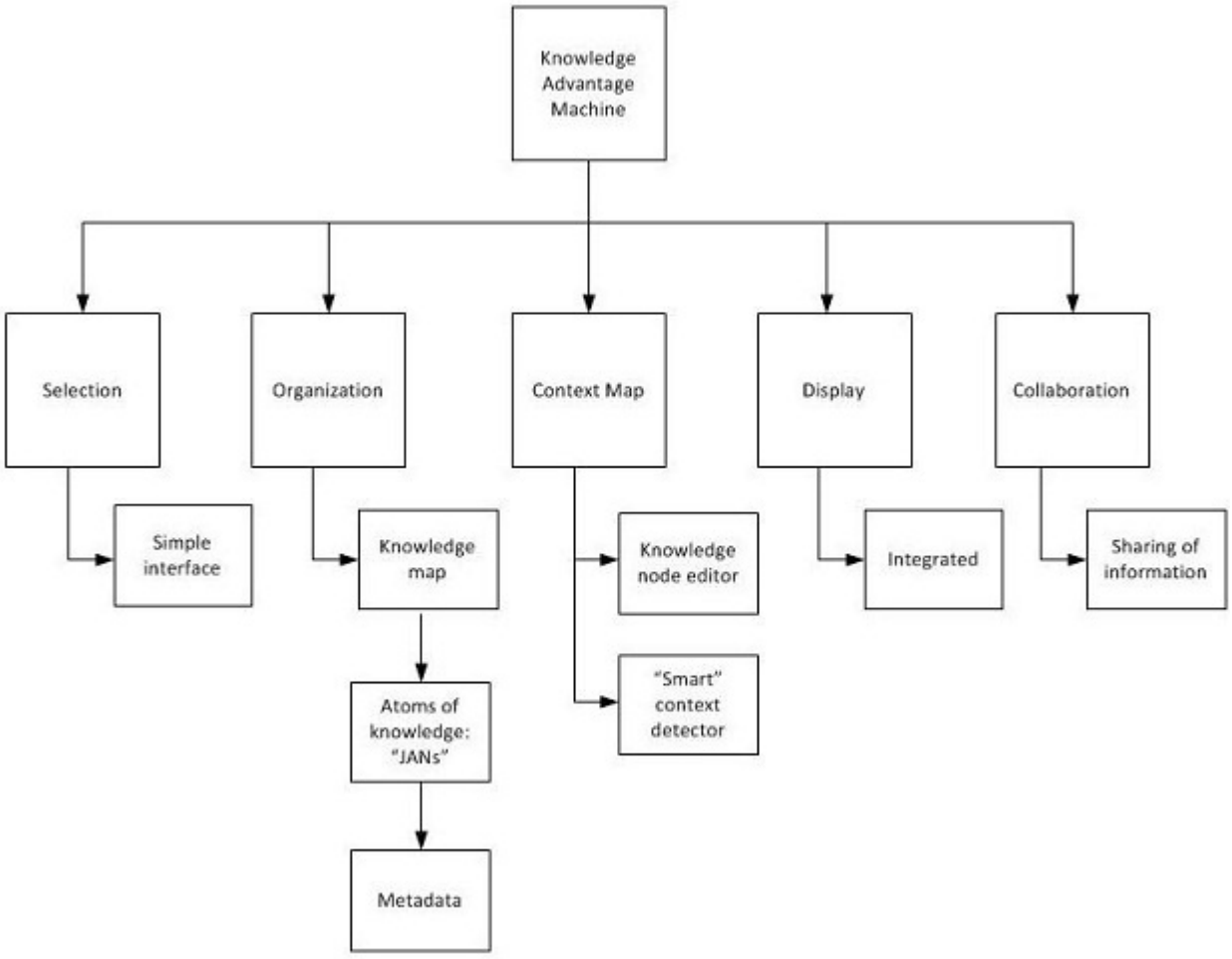

Figure 1.1: KAM Objective Map

semantically meaningful relationships within the personal ontology. This organization, in an idealized KAM, would be entirely (or mostly) automated, with manual organization being possible but generally unnecessary.

The Context Map Agent would be concerned with mapping information so as to assist in the intelligent identification of contexts. In an idealized KAM this would also be fully automated; for example, if a scientist has several research papers concerning nanotechnology within their personal ontology, the KAM should be able to automatically identify that these JANs are related and in the same context. The means of this identification vary; we might compare the semantic meaning of a

set of documents, or we may (as in the focus of this thesis) examine file usage patterns, or another method entirely. An idealized KAM's Context Map Agent would also be capable of identifying which contexts a user is currently engaged in; for example, a researcher taking part in a meeting about nanotechnology would be identified as being in the "nanotechnology" context.

The Display Agent would be concerned with how the various aspects of the KAM are displayed to the user. This agent represents the other half of the user interface, and is primarily concerned with the method of displaying the user's personal ontology. One KAM prototype called "GKAM" [21] used a radial graph to implement the Display Agent.

Finally, the Collaboration Agent would be concerned with providing ways of sharing personal ontology information between users. This is perhaps the most important portion of a fully realized KAM, as "knowledge" must come from somewhere. A subsection of this agent, called the Discovery Agent, would be concerned with the automated discovery of new JANs which are associated with a user's personal ontology.

This is just one brief overview of how a KAM could operate; countless other "agents" could be envisioned for a functional KAM. Furthermore, each of these agents has complex implementation problems associated with it, and so for the purposes of this work we must focus on very specific aspects. This thesis therefore focuses on developing a portion of a functional Context Map Agent which, instead of examining the semantics of JANs in a user's personal ontology, attempts to identify a user's work-patterns by examining his or her file usage. This data can then be used to identify contexts and predict future file usage. The difference between these two approaches is simple, but crucial: both are attempting to gain a knowledge advantage, but the former concentrates on using domain information [40] to do so, and the second focuses on user-specific work-patterns.

Vijjana-X	{J, T, R, dA, oA, cA, vA, sA, rA}
X	the domain name
J	the collection of Jans in the Vijjana-X
T	the Taxonomy used for classification of Jans
R	the domain specific relations
dA	the discovery agent which find relevant Jans
oA	the organizing agent which interlinks the Jans based on R
cA	the consistency/completeness agent
vA	the visualization agent
sA	the search agent
rA	the rating agent

Figure 1.2: Vijjana Architecture [30]

KAM-X	{J, T, R, sA, dA, oA, cA, dsA, coA}
X	the domain name
J	the collection of Jans in the KAM-X
T	the Taxonomy used for classification of Jans
R	the domain specific relations
sA	the selection agent
dA	the discovery agent which finds relevant Jans
oA	the organization agent which interlinks the Jans based on R and other factors
cA	the context map agent which classifies Jans within contexts based on semantic data
dsA	the display agent
coA	the collaboration agent

Figure 1.3: KAM Architecture

1.3 KAM Architectural View

The architectural model for the KAM could be described in many ways. A good basis for this model is found within Vijjana [30]. Vijjana, presented by Reddy et. al, is a model which constructs a semantic-based knowledge network out of URLS[2]. Described in Figure 1.2, the Vijjana model introduces the concept of a JAN, and might be considered a predecessor to the KAM since it seeks to provide a knowledge advantage through a self-organizing knowledge network.

The architectural view of a generic idealized KAM model might similarly be described as seen in Figure 1.3. This model presents a view of how the agents described in §1.2 interact to facilitate the KAM. In §4.1 this will be expanded to include work-pattern centric concepts.

1.4 Contributions of Thesis

The most salient contributions of this thesis follow:

- An overview of the KAM problem, its previous incarnations (such as the Context-Centric Knowledge Advantage Machine [40]), and its constituent parts
- An alternative model to the KAM's context predictor which instead functions through file access prediction

1.5 Outline of Thesis

The organization of the proceeding sections of this thesis is as follows:

- Chapter 2 offers the concrete objectives of this thesis. The method of inquiry and the actual research questions are described.

[2]Uniform Resource Locators

- Chapter 3 examines the existing related literature. This includes literature concerning the semantic desktop, semantic web, web usage patterns, and ontological research. A description of a fully realized KAM is given. Finally, the main objectives of this research are revealed.

- Chapter 4 presents a detailed overview of the inner workings of the WPCKAM (Work Pattern Centric Knowledge Advantage Machine) file usage agent. This chapter describes how collected workflow patterns could be used to produce a knowledge advantage. Technologies used to create this agent are also described.

- Chapter 5 presents statistical data collected from running the passive monitoring agent on volunteers' computers for a period of one week. An analysis of the data is given, and an assessment of prediction metrics is performed.

- Chapter 6 details conclusions gathered from the analysis of the data. A commentary on the effectiveness of predicting work patterns is given. Afterwards, there is a discussion of the various avenues that could be pursued in future KAM-related research, as well as descriptions of several technologies which will likely prove useful for continued research.

- Appendix A explains how to replicate the experiments performed in this thesis. The means to acquire the actual program code and scripts are provided.

- Appendix B consists of various tables and charting of raw data.

Chapter 2

Research Objectives

There are many avenues of research that could be pursued in regards to Knowledge Advantage Machines. This thesis focuses on work-patterns because they might act as an interesting alternative model to the KAM's context predictor [40] and they have yet to be utilized in that fashion. Thus, the purpose of this thesis is to create the basis for the WPCKAM: a type a KAM which is created with respect to work patterns. This thesis also aims to develop a roadmap towards scientifically examining this sort of system's impact on knowledge workers' efficiency in their everyday tasks.

The sections of this chapter are divided as follows:

- §2.1 details the method of inquiry used to examine the efficacy of the WPCKAM learning and prediction tasks.
- §2.2 explores the actual research questions that this thesis offers. Null and alternative hypotheses are offered which relate to these research questions.

2.1 Method of Inquiry

It is paramount to empirically demonstrate the results of our methods of examining and exploiting work patterns. We refer to this as the method of inquiry. Having collected a great deal of file usage

data, we apply several different learning classifiers[1] to it. We first select the training set, which we define as the previous ”k” file accesses. The classifier is then trained on this set, and attempts to learn (i.e. predict) the next member of the set. In this way each file access ”n” is trained on a set consisting of n-1, n-2, and so forth, up to n-k.

Next, the number of successful predictions versus failed predictions is tallied, and the process is repeated for various values of ”k”. Therefore, put simply, this research's core testing metric is how successful it is at predicting file usage. Our method of inquiry will therefore allow us to determine two important factors: which learning classifier is best at predicting file usage, and which value of ”k” is ideal for predicting file usage.

Next, the number of successful filterings versus failed filterings is tallied. The top five predictions for a given classifier are examined, and if the correct prediction is within them, the prediction is considered a success. This secondary method of inquiry allows us to determine if the context can be ”narrowed down” even if it cannot be predicted with good accuracy.

More specific information on the experimental design will be given in Chapter 5.

2.2 Research Questions

The questions this research addresses follow:

- How accurate are file usage predictions based on work patterns using various classifiers?
- How accurate is filtering predictions for file usage based on work patterns using various classifiers?

As such, we identify our hypotheses as follows:

- Null Hypothesis 1: There is no difference in accuracy between predictions of file usage among various classifiers.

[1] Naive Bayes, J48, LWL, Hyperpipes, and OneR; see §3.6

- Alternative Hypothesis 1: There is a difference in accuracy between predictions of file usage among various classifiers.
- Null Hypothesis 2: There is no difference in accuracy between predictions of file usage among various k-values.
- Alternative Hypothesis 2: There is a difference in accuracy between predictions of file usage among various k-values.
- Null Hypothesis 3: There is no difference in accuracy between filtering predictions of file usage among various classifiers.
- Alternative Hypothesis 3: There is a difference in accuracy between filtering predictions of file usage among various classifiers.
- Null Hypothesis 4: There is no difference in accuracy between filtering predictions of file usage among various k-values.
- Alternative Hypothesis 4: There is a difference in accuracy between filtering predictions of file usage among various k-values.

2.3 Summary

This chapter has examined the objectives of the this thesis. The data mining techniques which serve as the method of inquiry were identified as the following classifiers:

- Naive Bayes
- J48
- LWL

- Hyperpipes
- OneR

The means by which these classifiers are used to perform file prediction were explained. Finally, the research questions and their associated hypotheses were identified in this chapter.

Chapter 3

Background and Related Work

The background of the ”Knowledge Advantage Machine” concept resides mainly in the existing research behind the Semantic Web [4] and the Semantic Desktop [35]. The ”jumping-off point” of KAM research is an attempt to create a middle-man between those two existing areas; we are concerned with semantically understanding both the web-content that knowledge workers might encounter, and the data-rich contents of their personal computers.

However, the main goal of this thesis is to be able to identify and exploit a given knowledge worker’s ”work patterns.” We do this in order to facilitate one possible knowledge advantage machine. Other research has gone in different directions [40] [26]; this will be explored fully in the proceeding chapter.

In addition to examining existing Semantic Web and Semantic Desktop research, we must also establish a background for the ”work-patterns” which we intend to study and harness for efficiency; in this way, a robust definition of a ”work-pattern” can be defined. We will do this by making use of various data mining techniques; we will also attempt to gain insight into work pattern discovery by drawing comparisons to existing research into ”web usage patterns” [38].

This chapter is therefore divided as follows:

- §3.1 explains the background of the Semantic Web and how it relates to the Knowledge

Advantage Machine.

- §3.2 explains the background of the Semantic Desktop and how it relates to the Knowledge Advantage Machine. The ”Semantic Desktop 2.0” is also described.

- §3.3 offers a description of web usage mining [38] and how its constituent components are analogous to the sort of work pattern mining that the WPCKAM aims to achieve. Its subsections explore the three main steps of web usage mining: preprocessing, pattern discovery, and pattern analysis.

- §3.4 offers exposition on Graph Based Induction, a learning model [26] which was used for various learning tasks including file prefetching.

- §3.5 gives a background for ontological research, and how the concept of a personal ontology is core to the Knowledge Advantage Machine.

- §3.6 explains what is meant by ”classification learning” and how it pertains to this thesis. Each of the subsections explores an individual classifier used for this research.

3.1 Semantic Web

The Semantic Web, which introduces services which allow for machine-understanding of semantics on the Web, can be defined as something which ”provides the infrastructure for the semantic interoperability of Web Services.” Thus, with a Semantic Web in place, various Semantic Web Services can be developed to aid in automated discovery of information [4]. More recently, in 2008, the idea of using mobile devices in conjunction with the Semantic Web was explored [43]; as we have already shown, this is also a concept that is of interest in KAM research with respect to context aware computing.

3.2 Semantic Desktop

An overview of the idea of the ”Semantic Desktop” was published in 2005. This paper defined a semantic desktop as an attempt to ”transfer the Semantic Web to desktop computers” - not just its actual technology, but the philosophy as well. The Semantic Desktop, therefore, attempts to accomplish the same goals as the Semantic Web via a desktop computer; specifically, the purpose is to give machines a semantic understanding of the contents of ones computer or files [35].

In 2006, the Semantic Desktop 2.0 was discussed, as was the idea of using metadata to organize data resources. This proposed ”second edition” of the Semantic Desktop attempted to improve on the original design by using PIMO, the Personal Information model, as an approach for ontology organization. This model made it possible for tagging services to be created on the desktop. [34]

3.3 Web Usage Mining

In 1995, Letizia was implemented as an agent which worked with web browsers, specifically Netscape. It attempted to remember a user's patterns with concern to web browsing, taking note of things that user is interested in and using this information to recommend certain online documents that pertain to previous searches [22].

A great deal of research has been done in the area of applying data mining techniques in order to establish ”web-usage patterns” - this is sometimes referred to as ”web-usage mining” [38]. Accordingly, within this section I will explore the feasibility of using a similar plan in order to perform what we will refer to as ”work-pattern mining.”

In an idealized KAM, work patterns could be discovered in a manner based on the web usage mining techniques used by Srivastava et al [38]. In their paper, usage patterns are shown to be ”mined” in three phases: preprocessing, pattern discovery, and pattern analysis.

3.3.1 Preprocessing

Preprocessing is described [38] as consisting of taking content, structure, and usage information and abstracting them into metadata which allows for pattern discovery. Therefore, preprocessing happens in three stages, which I will first describe, then explore how these concepts might be used as analogues that are viable in the WPCKAM framework.

In web usage mining, usage preprocessing generally consists of identifying the user and dividing the click-stream into sessions. In an idealized KAM framework, the user might not need to be identified, but information analogous to a ”click-stream” could be useful. In a WPCKAM system, the analogue to a clickstream could be described as the timestamps, filenames, and other metadata associated with the files a user is accessing and editing.

Content preprocessing consists of converting the actual content (text, images, multimedia, etc) into other forms which are more useful for web usage mining. Likewise, the various file content which an idealized KAM user interacts with on a regular basis must be converted into forms which are useful for work pattern mining. This content might not just consist of the actual contents of the files which have been altered, but also the differences between the newly modified file and the original. Specifically, [16] made use of vector space model to place the content in a quantifiable format, and one might likewise use a semantic space and Latent Semantic Analysis [7] to examine the actual content of files.

Structure preprocessing consists of examining the structure (i.e. the ”hypertext links between page views”) of a site, then preprocessing it in a similar manner as content preprocessing above. Since the WPCKAM framework is primarily concerned with filesystem access and content rather than web content, there is no real analogue. Thus, this preprocessing step is omitted in our system.

3.3.2 Pattern Discovery

The second step of web usage mining, ”pattern discovery,” is described as drawing upon methodologies from many fields, including pattern recognition, data mining, and statistics [38]. We will likewise draw from the same base of methods (especially data mining) in order to perform our ”work-pattern mining.” This step consists of several sub-steps: statistical analysis, association rule generation, clustering, classification, sequential pattern discovery, and dependency modeling.

In the WPCKAM framework, statistical analysis of the work patterns is performed via classification learning techniques. [38] refers to basic statistical techniques used to find the mean, median, etc for values such as the most frequently accessed page views, viewing time, etc. Likewise, when searching for work patterns we might also record most frequently accessed files, access times, and so forth.

Association rule generation is intended to relate pages which are referenced together in one server session. The analogue here is that we will instead relate files which are accessed together in one ”session” or ”K-set”[1]. Specifically, [38] attempts to identify sets of pages which are commonly accessed together at rates above some specified value. Thus, using statistical analysis or data mining techniques we might also specify some threshold which will allow us to determine whether we can consider two files to be typically accessed ”together” by a particular user.

Clustering is an establish data mining technique which simply groups together items which have similar characteristics. For web usage, [38] was primarily concerned with establishing two clusters: usage clusters and page clusters. Usage clusters were clusters of users which apparently had similar browsing habits, and page clusters are clusters of pages with related content. Both clusters are directly analogous to our KAM framework; in an idealized KAM we might find clusters of users with similar work patterns, and clusters of files with related content. However, for the WPCKAM, we are already performing a form of clustering by virtue of using the Locally Weighted

[1]The previous ”K” file accesses which act as the current training set for the learning classifier.

Learning[2] classifier which makes use of the k-means clusterer.

Classification learning is another data mining technique which will be detailed extensively in §3.6. For web usage, [38] sought to classify users. In the WPCKAM framework we instead attempt to perform classification on files in order to predict future file usage. An idealized KAM might also classify users into various categories.

Sequential pattern discovery is defined as an attempt to find ”inter-session” patterns. In web usage mining, this might consist of some specific set of items which is proceeded by some other specific item within an overall set of sessions. Likewise, we might find sequential patterns in work-pattern mining; this is essentially the core concept insofar as we are attempting to predict which files within the user's personal ontology are most likely to be accessed in the near future.

Dependency modeling is the sixth and final step of pattern discovery. This consists of attempting to build models which are ”capable of representing significant dependencies among the various variables in the Web domain. [38]” An example given is the stages a potential customer might go through whilst shopping online; these might be used to identify a casual visitor or an actual buyer. Since this is primarily concerned with navigation, there may be no good analogue in the WPCKAM framework. On the other hand, in an idealized KAM (with full knowledge of the user's activities) we might liken this to the actual steps a user takes while working; for example, the user first searches and researches a subject, then a document is accessed, then the same document is edited.

3.3.3 Pattern Analysis

The third and final step of web usage mining, ”pattern analysis,” focuses on the filtering of patterns or rules which are ”uninteresting.” Anything that was found within the pattern discovery phase is examined and discarded if it is deemed to be useless. This is our exact goal in work-pattern

[2] See §5.4.4

mining. [38] describes various methods of accomplishing this, such as the use of data cubes[3], which allow for On-Line Analytical Processing (OLAP) operations.

OLAP is known to be an effective method of analyzing very large amounts of data [27]. However, [38] ultimately asserts that the analysis method should be governed by the application one is performing web mining on. Thus, in our analysis of work-patterns, we will design our own methodology wherein analysis is performed through data mining classification techniques.

3.4 Graph Based Induction

In 1998 Graph Based Induction [26], or GBI, was proposed as a learning model in order to perform various learning tasks, including file prefetching via a user-obscured ”Prefetch daemon” tool. File prefetching is a potent technique which works by predicting soon-to-be-used disk blocks so that they can be ”prefetched” into memory, increasing efficiency. Shriver et al. [36] showed that application throughput can be improved by up to fifty percent via file prefetching.

GBI is able to ”extract regularities” from a graph of user-dependent data using top-down induction. It was originally designed to examine inference patterns and extract any patterns which appear frequently within the inference trace. GBI is, therefore, looking for ”regularities” in input traces, allowing the algorithm to hasten tasks such as learning and classification.

The algorithm works finding patterns, then contracting the graph after replacing the found pattern with single new node. However, the graph cannot ever contract so far as to become a single node, because the graph size is defined by both the sizes of extract patterns and the size of the contracted graph. This core ”contraction” technique used in GBI is called pairwise chunking, and is shown in Figure 3.1. Pairwise chunking works within the overall GBI algorithm to ”chunk” appropriated linked pairs of nodes at each step of any search.

GBI’s ”extracted regularities” come in the form of subpatterns. The algorithm only returns sub-

[3]Sometimes known as OLAP cubes.

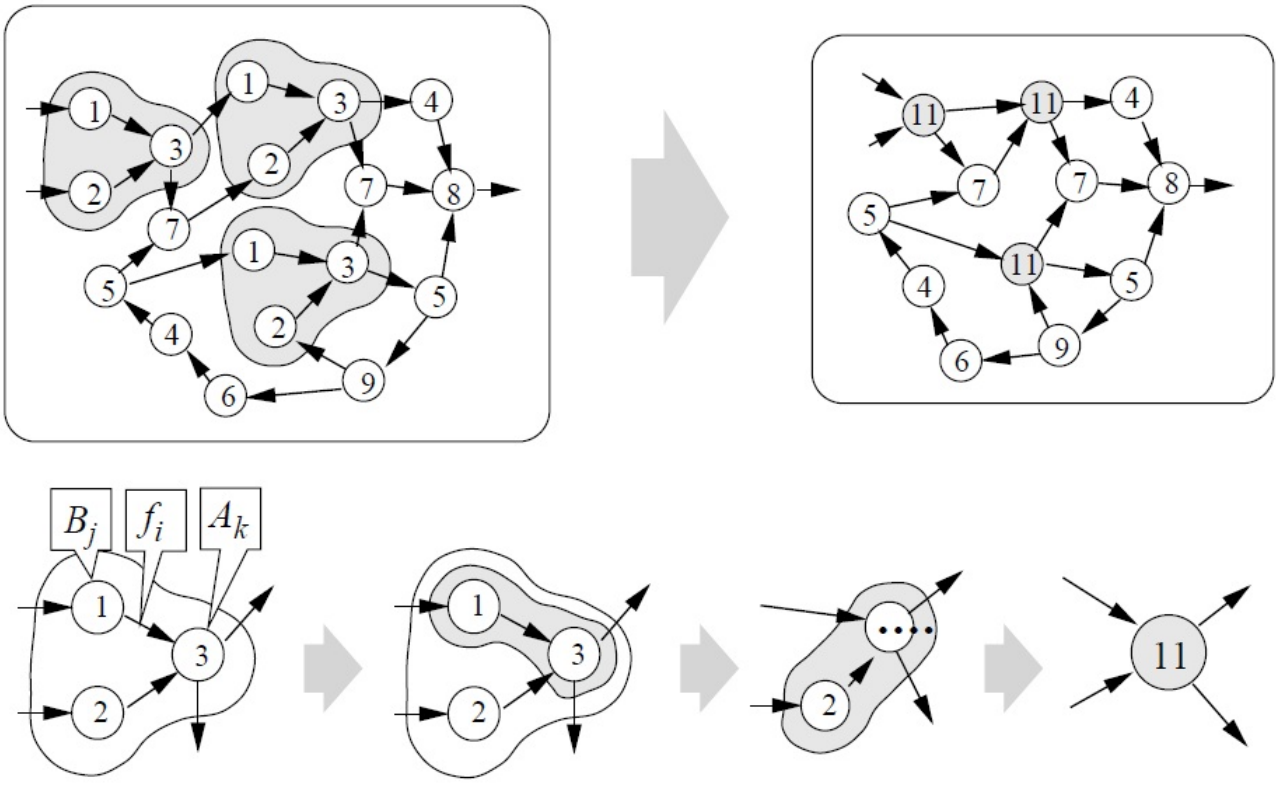

Figure 3.1: Pairwise Chunking [26]

patterns where the graph size was ultimately minimum; these are considered to be the ”interesting” patterns.

3.5 Ontological Research

Ontology is defined as the study of the nature of existence. A ”formal ontology” can be described as a designed structure or graph whose purpose is supporting knowledge sharing activities; this concept could be used in such knowledge-intensive work as engineering and mathematics [13].

Chen et al. [6] described ”COBRA-ONT[4],” a type of ontology specifically designed to support systems which are context-aware. [6] describes the movement of computing towards these context-aware systems and the need for an ontology that supports them. ”Intelligent agents” are

[4]Context Broker Architecture Ontology

described, such as the ones described in §1.2, which are capable of intelligently understanding context information and using this information to support various applications.

When referring to KAM research, it is important to be familiar with the concept of a ”personal ontology,” which is an overall description of an individual via his or her interests. This thesis uses the phrase ”knowledge base” interchangeably with ”personal ontology”; put as simply as possible, we view a personal ontology as the sum total of an individual's collected knowledge, which can be discretized into clusters called contexts and tagged with metadata to describe the semantic contents.

3.6 Classifiers

Several classifiers were tested in order to perform learning tasks for this research. Statistical classification is defined as the process in which a training set of data is examined, and a new observation is placed into one of a set of several existing ”categories.” Likewise, classification learning in data mining examines a set of classified examples (generally still referred to as the training set), with the intent of learning how to classify unknown examples. Throughout this thesis we will refer to the ”K” value, which is the number of previous examples in the training set that will be examined to prepare for classification. The experimental design significance surrounding this ”K” value will be described in Chapter 5.

Classification learning in data mining is considered a kind of ”supervised” learning [42]; this simply means that, unlike unsupervised learning methods such as clustering, classification training sets always provide the true outcome (or ”class”) of each training example.

In this research, we are concerned with predicting file usage, and so our categories are defined as the files which are being accessed, and the new observation is/are the file(s) which have yet to be accessed. While ”multilabeled instances” do exist for some scenarios - that is, classification examples which belong to multiple classes - for the purposes of this thesis every example must

exist in only one class because each item may only have one file location.

Classification algorithms are ubiquitous in data mining research. As shown by Giudici [12], classification learning and other data mining techniques have been applied real world scenarios, such as business and industry problems. The ”Bayes' Theorem” that Naive Bayes classification employs has been used in myriad fields [25], including codebreaking, DNA de-coding, and Homeland Security tasks. Bayesian approaches have even been used in order to filter through junk mail [33].

The efficiency of these classifiers in extracting relevant file usage patterns will be examined in Chapter 5; here, I provide a brief overview of how each of the studied classifiers functions. The following classifiers were not implemented directly; instead, implementations from WEKA [14][5] were used.

3.6.1 Naive Bayes

Naive Bayes is a type of probabilistic classifier which works around Bayes's theorem. Bayes' theorem, also known as ”Bayes' rule” [17], specifies a relationship between the probability of some conditional hypothesis on some set of data with the inverse probability of some conditional data on the hypothesis. The theorem can be expressed in several ways, such as with the symmetry principle for probability ratios seen in Figure 3.3.

Naive Bayes itself is a simple and efficient [41] classifier which, despite its power, operates on an ostensibly ”naive” principle: observed events occur entirely independently of each other. This assumption is considered naive because it is generally false. The Naive Bayes classifier calculates probabilities of examples being in a particular class [44]; the classifier can ultimately be depicted as seen in Figure 3.4.

Naive Bayes which has been empirically shown by Rish [31] to typically be extremely effective in comparison to other more advanced classifiers, despite its generally flawed assumption of in-

[5]Described in §4.3.2.

```
function Training(data)
  counts = array()
  classes = array()

  for(row in data)
    for(column in row)
      counts[column.index][column.value][row.class]++
      classes[row.class]++
  return {classes, counts}

function Testing(data, model)
  classes = model.classes
  counts = model.counts

  for(row in data)
    scores = array()
    for(class in classes)
      score = classes[class] / training.length

      for(column in row)
        score *= counts[column.index][row[column.index]][row.class] / classes[class]
      scores[class] = score

    classWithMaxScore = maximum(scores)
    results += (classWithMaxScore == row.class)

  return results

function NaiveBayes(data)
  model = Training(data)
  return Testing(data, model)
```

Figure 3.2: Pseudo code for Naive Bayes [20]

```
PR(H, E) = PR(E, H)
```

Figure 3.3: Symmetry Principle for Probability Ratios [17]

$$f_{nb}(E) = \frac{p(C=+)}{p(C=-)} \prod_{i=1}^{n} \frac{p(x_i|C=+)}{p(x_i|C=-)}.$$

Figure 3.4: Naive Bayes Classifier [44]

dependence. This empirical study showed that, in spite of inaccurate probability estimates, Naive Bayes's classification decisions were often correct.

It was also demonstrated that the best performance of Naive Bayes occurs when features are completely independent (i.e. the "naive" assumption of independence is actually true) and, interestingly, when features are functionally dependent. When features are only partially dependent, Naive Bayes has poorer performance. Therefore, we might expect Naive Bayes to perform well for file usage prediction if each atom of file usage is totally unrelated to the others, or if they are totally dependent on each other.

3.6.2 J48

J48 is WEKA's open source Java implementation [14] of Ross Quinlan's decision tree generating algorithm called C4.5 [29]. As with most classifiers, a training set of previously classified samples is provided to the algorithm. C4.5 then builds decision trees based on this training data. One attribute of data is chosen for each node of the tree, based on which attribute best separates the sample set into subsets of classes. Information gain is measured for the purposes of this separation; that is, the attribute that has the highest information gain is used to create the separation, and the algorithm proceeds recursively until it reaches one of three base cases, shown in Figure 3.5.

3.6.3 LWL

LWL stands for Locally Weighted Learning, and functions as a locally weighted version of Naive Bayes [10]. This local weighting serves to "relax" the typically naive assumption of attribute

```
function Training(data)
   if allSameClass(data)
      return Node(data.row.class)

   for(attribute in data)
      informationGain = InfoGain(attribute)

   bestAttribute = attribute with maximum informationGain

   if bestAttribute is continuous
      threshold = value which, if bestAttribute is split on will have the highest informationGain
                  across the two subsets of the data
      nodes.push(Training(data given bestAttribute value > threshold))
      nodes.push(Training(data given bestAttribute value ≤ threshold))
   else
      for(value in bestAttribute)
         nodes.push(Training(data given bestAttribute == value))

   nodes = Prune(nodes, data)

   return nodes

function Prune(nodes)
   errorOfChilderen = 0
   for each node in nodes
      errorOfChilderen += ClassifyByMajorityClass(data given attribute == value)

   errorOfParent = ClassifyByMajorityClass(data)

   if errorOfParent < errorOfChilderen
      return {}
   else
      return nodes

function Testing(data)
   for(row in data)
      results += Classify(row, tree)
   return results

function Classify(row, tree)
   if(IsLeaf(tree))
      return tree.class

   return Classify(row, tree.nodeForValue(row[tree.attribute]))
```

Figure 3.5: Pseudo code for C4.5 [20]

```
function K-Means(data)
  Centroids = A random subset of K instances from data

  for each row in data
    m(row) = the cluster closest to row

  while m has changed
    for each centroid in Centroids
      centroid = the centroid of the current instances assigned to that centroid

    for each row in data
      m(row) = the cluster closes to row

  return Centroids
```

Figure 3.6: Pseudo code for K-Means [18]

independence, as local models are learned at prediction time. This is sometimes called a ”lazy” learning approach because of how the learning effort is deferred. LWL's performance is typically not impacted by the k[6] value of the inherent k-means clustering algorithm used to perform the local weighting. The ”k” in k-means clustering represents the number of clusters that the observations will be divided into. [24].

3.6.4 Hyperpipes

Hyperpipes is a learning classifier created by Lucio de Souza Coelho and Len Trigg [42]. This is a rule-based learner, and is based on having only one simple rule per class. For each category attribute in the training data, a range of values is observed. The ”rule” is then to decide which ranges have the attribute values of a particular test instance; the category that has the most correct ranges is then chosen.

Because of its simplicity, Hyperpipes performs classification quickly when there are large numbers of attributes [8]. It was shown in [8] that Hyperpipes outperformed several other classifiers.

[6]Not to be confused with the K value described at the beginning of this section.

```
function Training(data)
  training = data

function Testing(data)
  for(row in data)
    knn = k nearest neighbors from training to row
    knn = ApplyWeighting(knn, row)
    results += Classify(knn, row)
  return results

function ApplyWeighting(data, testRow)
  for(row in data)
    row.weight = row.distanceFrom(testRow) / maxDistanceFromTest

function Classify(training, testingRow)
  counts = array()
  classes = array()

  for(row in training)
    for(column in row)
      counts[column.index][column.value][row.class]++
      classes[row.class]++

  for(class in classes)
    score = classes[class] / training.length

    for(column in testingRow)
      score *= counts[column.index][testingRow[column.index]][testingRow.class] /
               classes[class]

  return classWithMaxScore == testingRow.class
```

Figure 3.7: Pseudo code for Locally Weighted Naive Bayes [20]

```
function train() {
   klass = $Klass   # get the class attribute
   Klasses[k] = 1     # remember we have one more class
   for(i=1;i<=Attr;i++)
     value=$i;
     if (i != Klass)
      if (value !~ /\?/)
         if (numericp(i) {
               if (value > Max[klass,i]) Max[klass,i]= value
               if (value < Min[klass,i]) Min[klass,i]= value
        } else  Seen[klass,i,value]= 1 }
}

function contains(klass,i,value) {
     if (numericp(i) {
         if ((Max[klass,i] >= value) && (Min[klass,i] <= value))
              return 1
      } else { if (Seen[klass,i,value])
                 return 1 }
     return 0
}

function mostContained() {
   best = -1;
   for(klass in Klasses) {
      count=0;
      for(i=1;i<=Attr;i++) {
          value=$i
         if (i != Klass)
            if ( value !~ /\?/ )
               count += contains(klass,i,value)
      count = count / (Attr - 1)
      if ( count >= best ) {
            best = count; what=klass
   }
   return what
}
```

Figure 3.8: Pseudo code for Hyperpipes [11]

```
function Training(data)
  for(attribute in data)
    for(value in attribute)
      rule.push(attribute, value, majorityClass(data given attribute and value)
    rules.push(rule)

  for(rule in rules)
    subset = data given rule.attribute == rule.value
    rule.score = frequencyOf(rule.class in subset) / subset.length

  return rules for the attribute with the

function Testing(data)
  for(row in data)
    results += row.class == rule.class for row.attributeValue
```

Figure 3.9: Pseudo code of OneR [20]

3.6.5 OneR

OneR, or ”One Rule” classification has been shown to work well for most commonly used datasets [15]. This classifier works by creating only one rule for each attribute. This is a rule based learner which states that the majority class is C for any attribute A and value V. After training, the accuracies associated with the created rules are then examined with respect to hypothesis H. Accuracies below the majority class C are eliminated.

The algorithm is such that a singular poor choice of rules can drastically reduce classification accuracy. This learner is therefore not expected to perform well; it is included for comparison purposes only. Pseudo code for OneR can be found in Figure 3.9.

3.7 Summary

This chapter has reviewed the literature pertaining to the Knowledge Advantage Machine and the building blocks that have made it possible. The Semantic Web and Semantic Desktop were

reviewed, as well as the background of ontological research. Web usage mining was examined piece-by-piece and its constituent steps were compared to the KAM framework.

Various learning techniques were explored, including:

- Graph Based Induction
- Naive Bayes
- J48
- LWL
- Hyperpipes
- OneR

Chapter 4

Work Pattern Centric Knowledge Advantage Machine

In this chapter, we describe the WPCKAM framework, its goals, and its associated technologies. Thoughts on how the WPCKAM's contributions could be used to help implement an idealized KAM are also offered.

This chapter is therefore divided as follows:

- §4.1 presents the architectural model for the WPCKAM.
- §4.2 offers a brief description of file usage prediction techniques.
- §4.3 and its subsections describe the technologies used to create and analyze various aspects of a prototype WPCKAM, including WEKA [14], the data mining suite.
- §4.4 shows examples of how a knowledge advantage can be more easily gained through work-pattern centric elements. This section demonstrates how WPCKAM concepts relate back to the greater problem of creating an idealized KAM.

WPCKAM-X	{J, T, R, sA, dA, oA, wpdA, cA, dsA, coA}
X	the domain name
J	the collection of Jans in the WPCKAM-X
T	the Taxonomy used for classification of Jans
R	the domain specific relations
sA	the selection agent
dA	the discovery agent which finds relevant Jans
oA	the organization agent which interlinks the Jans based on R and other factors, including workflow data
wpdA	the work pattern discovery agent
cA	the context map agent which classifies Jans within contexts based on semantic and workflow data
dsA	the display agent
coA	the collaboration agent

Figure 4.1: WPCKAM Architecture

4.1 WPCKAM Architectural View

The architectural view for the WPCKAM necessarily differs from the KAM's model, as we must now account for file prediction. This model therefore demonstrates how workflow pattern prediction could be used within a KAM to produce a knowledge advantage. We see in Figure 4.1 that the model is similar to the one presented in §1.3, with the utilization of workflow knowledge added.

A new agent, the "Work Pattern Discovery Agent," is introduced; this module would be responsible for passively monitoring the filesystem (or the user's personal ontology) as described throughout this thesis, and storing statistical data or training sets pertaining to the collected data. The context and organization agents would now be responsible for properly husbanding this data in various ways.

4.2 File Usage Prediction

In this thesis, "file usage prediction" is centered around the idea that knowing what has been used before can predict what will be needed. As a simple example, we might examine the case of a person getting ready for work: on a daily basis, they wake up, shower, eat breakfast, and brush

their teeth in that order. While this pattern might exhibit some fluctuation, such as if a person is running late and skips breakfast, the overall pattern will remain the same. We could therefore expect similar patterns to arise from workflow data.

In this thesis, file usage prediction is meant to primarily act as an alternative context predictor. It could therefore be described as an alternative context map agent[1] Because file usage prediction is one of the main goals of this research, it is imperative to examine appropriate methods of implementing it.

A decision making process such as MDP[2] [28] would, at first, seem to be the ideal solution for these sorts of predictions. However, our system fails to obey the ”Markov Property” - that is, a property that demands our system be ”memoryless.” This stochastic process makes decisions based solely on the current state and on how far we intend to look ahead; for the WPCKAM, our goal is to instead base the decision process on past file usage choices.

GBI [26] has also been used, as described in §3.4, but was interwoven with interface and command data, and was designed for alternate purposes.

We therefore instead choose to use a variety of classifiers, described in §3.6, to perform file usage prediction via classification learning. In Chapter 5 the complete experimental setup of these classifiers is described, as well as an analysis of the resulting data.

The motivation for using a variety of classifiers is simple: each represents a different approach to learning from the data. We survey various classifiers in order to discover which will be ”best” for workflow prediction and the WPCKAM overall. As described in §3.6, classifiers tend to work differently on different types of sets. Furthermore, some classifiers are known for their speed, such as Hyperpipes [14], and others are shown to be superior for their general prediction accuracy [31]. It is therefore important to take a variety of criteria into account, and using multiple classifiers allows a more robust survey of prediction metrics.

[1] See §1.2.
[2] Markov Decision Process

4.3 Technologies

The proceeding subsections will explore various pertinent technologies that were used to assist in workflow prediction activities. These technologies could also prove useful in an idealized WPCKAM system.

4.3.1 Windows Auditing

Various versions of Windows contain an auditing system [37]. This system can be used for a variety of purposes to track various activites on a computer. For this research, file auditing was used to keep track of file usage data, therefore monitoring workflow. This workflow data was saved in XML format. Finally, before analysis, a variety of preprocessing steps are applied to the XML; this process will be described in depth in §5.2. The end result of these preprocessing steps, however, is a simple list of file accesses.

4.3.2 WEKA

WEKA [14] is a Java-based open source software package designed to provide various data mining utilities. This package offers implementations of various classifiers, clusterers, assocation learning utilities, and other learning tools. It also contains implementations of various experimentation and statistical tools such as paired t-tests. For this thesis, WEKA's implementations of the classifiers J48, Naive Bayes, Hyperpipes, and LWL were used to perform classification learning on the file usage data sets in order to learn work patterns.

4.4 How the Advantage is Gained

The core concept of the KAM is producing the aforementioned "knowledge advantage." Therefore, when describing a WPCKAM, we are primarily concerned with identifying how the "work-pattern

centric" elements are producing that advantage. In a fully realized KAM, work-patterns discovered within the WPCKAM framework could be utilized in several ways. This section will examine several areas where this strategy could be employed to produce a knowledge advantage.

The subsections are therefore divided as follows:

- §4.4.1 shows how workflow prediction could be utilized.
- §4.4.2 examines the possibility of presenting predictions directly to the user as an alternative route to file access.
- §4.4.3 discusses how file usage data can be used to establish a metric for the importance of JANs within a personal ontology.

4.4.1 Context Awareness

As demonstrated in the sample scenario in §1.1.1, context awareness is integral for a KAM to produce a knowledge advantage. File usage prediction offers an interesting alternative to a typical KAM's context prediction. Typically a KAM identifies similarity between JANs by semantically analyzing a user's entire knowledge-base and comparing the content in order to establish a metric of comparison; for example, one project called GKAM [21][3] used TFIDF[4] calculations to accomplish this.

Instead, the WPCKAM could identify a JAN's "contextually similar" JANs as those that are most frequently accessed together. Alternately, these contextually similar JANs could be identified as the "N" JANs that are most likely to be predicted after a particular access.

It would also be possible to use the probablility as a modifier for relatedness detection within the context map or organization agent. Access probability could be coupled with a semantic-centric

[3]Graphical Knowledge Advantage Machine
[4]Term Frequency Times Inverse Document Frequency

metric (such as TFIDF as in GKAM [21]) in order to weight probable related documents without eliminating related but under-accessed documents.

These alternative methods for identification of context could prove more effective than context awareness accomplished solely through semantic analysis, and therefore bear further investigation. This thesis establishes a basis for that investigation by providing empirical results from file prediction via classification.

4.4.2 Presenting Predicted Files

With an agent capable of predicting file accesses with a high degree of success, some new methods of producing a knowledge advantage become possible for the KAM. One simple way to harness file prediction would be providing the WPCKAM's user with a small application which continuously shows likely-to-be-accessed files within a panel. This potential application offers a centralized location where files can be accessed more conveniently and quickly, providing a small but relevant advantage.

An alternative, less intrusive method of presenting these predictions would be to populate a folder with links to files which are predicted to be accessed in the near future.

4.4.3 Measuring JAN Importance

With access to file usage data, we might proceed with the reasonable assumption that the most frequently accessed JANs are more "important" to the user. A KAM with access to this data could therefore automatically know the elements of a user's personal ontology which are most important to them. These "important" JANs could be tagged as such, and semantic analysis agents could later be employed to glean the semantic significance of those particular JANs. Some form of discovery agent[5] could then be used to discover related JANs from outside the personal ontology, offering

[5]See §1.2.

them to the user and potentially producing a knowledge advantage.

4.5 Summary

This chapter has described the Work Pattern Centric Knowledge Advantage Machine and its architectural model. The motivations behind the choice of file usage prediction metrics were discussed. Technologies used to create this thesis's experimental contribution were also examined. Finally, this chapter explored how work-pattern centric elements could be used to leverage a knowledge advantage within a fully realized KAM.

In the following chapters, the discussion will move to research objectives, the experimental setup, and the findings for the various classifiers when used for the purposes that have been described in the previous chapters.

Chapter 5

Experimental Setup and Analysis of Results

This chapter offers an overview of the experimental methods, some raw results across data sets, and how these results were analyzed.

This chapter is therefore divided as follows:

- §5.1 gives a brief overview of the experimental setup,
- §5.2 displays an in-depth view of the preprocessing steps that the collected data was subjected to before classification.
- §5.3 demonstrates the actual results of the data collection. Several tables of data are provided, as well as box plot charting
- §5.4 offers thoughts on the performance of the individual classifiers based on the data across all sets. This section also gives some implementation-specific details for these classifiers.
- §5.5 consists of more in-depth statistical analysis performed on the data.

5.1 Overview of Setup

The experimental setup for this research consists of three main parts: data collection and preprocessing, classification/prediction, and data analysis. Data collection is accomplished using Windows object access auditing [37]. The means for this have been described in §4.3. The data was collected from a group of volunteers gathered from various science and information fields. These volunteers were all running Windows 7 Professional.

In order to attempt to find interesting results, various values of ”k” (§5.2) are examined in various increments from k=1 to k=17. These values of ”k” were generated using an xlog(x) scale, starting at x=2 and incrementing. The ”k” values used in this, extend as seen in Table B.1. This table also offers a view at which data sets have been run for each classifier and value of K. The significance of k-values has been described §2.1.

In order to determine the efficacy of these techniques for filtering, we define a number ”T” which represents the number of predictions we are narrowing the selection to. As described in §2.1, the ”T” value used in this thesis is T=5. Therefore, a successful filtering is considered to be a case where the correct prediction was within the top five predictions offered by the classifier.

5.2 Preprocessing

Preprocessing of the raw XML data is done in several steps. A series of scripts perform the following tasks:

- Strip out any record which only occurs once in the data set. This step is performed in order to eliminate any chance of predicting access for a JAN which will only ever occur once within the workflow. This is done post-hoc for this data, but could easily be done on-the-fly in an actual implementation of a WPCKAM via manipulation of the training sets.

- Strip out any record which is not part of the whitelist seen in Table B.4. This eliminates any-

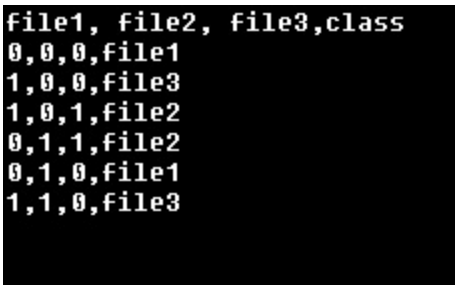

Figure 5.1: Example Comma Separated Values File

thing that is not considered a document or a programming source code file. These extensions were gathered from [1]. This action is performed so that we are specifically searching for knowledge-work-related patterns.

- Convert each record into an MD5 hash. The purpose of this step is simply to avoid any sensitive user data from being displayed in plaintext.

The final step of the preprocessing is to construct a CSV[1] from the collected data. These CSV files consist of records (lines) and fields (each value, separated by commas). A small sample CSV file is shown in Figure 5.1. The first line of this file is a listing of the attribute names. Every other record is a series of digits, then the attribute value. This attribute value represents the filename. The digits represent whether the various files were seen in the last ”k” records of the training set; a ”1” indicates that the corresponding attribute value was seen within the last k units, and a ”0” indicates that it was not. Several values of ”k” were used for this experimental setup, shown in Table B.1. Note that the example CSV file in Figure 5.1 is constructed using a k value of ”2”.

5.3 Results

This section offers description of charting for global results.

[1]Comma-separated values

- Table 5.1 shows various statistical data for predictions across all datasets for values of k, as well as a box plot of this data.
- Table 5.2 shows various statistical data for predictions across all datasets for classifiers, as well as a box plot of this data.
- Table 5.3 shows various statistical data for predictions across all datasets for each dataset, as well as a box plot of this data.
- Table 5.4 shows various statistical data for filtering across all datasets for values of k, as well as a box plot of this data.
- Table 5.5 shows various statistical data for filtering across all datasets for classifiers, as well as a box plot of this data.
- Table 5.6 shows various statistical data for filtering across all datasets for each dataset, as well as a box plot of this data.
- Table 5.7 demonstrates prediction counts across all values of k, as well as prediction means.
- Table 5.8 demonstrates prediction counts across all classifiers, as well as prediction means.
- Table 5.9 demonstrates prediction counts across all datasets, as well as prediction means.
- Table 5.10 demonstrates filter counts across all values of k, as well as filtering means.
- Table 5.11 demonstrates filter counts across all classifiers, as well as filtering means.
- Table 5.12 demonstrates filter counts across all datasets, as well as filtering means.

Results						
k-value	min	q1	median	q3	max	Quartile
1	0	16	52	67	78	
3	0	29	44	62	80	
6	0	25	52.5	63	79	
8	0	25	49.5	59	79	
11	0	20	42.5	58	77	
14	0	19	36.5	52	69	

Table 5.1: Prediction success rates by k-value for all datasets

Results						
Classifier	min	q1	median	q3	max	Quartile
Hyperpipes	8	23	33.5	56.5	70	
Naive Bayes	6	46	58	67	79	
OneR	0	11	14.5	33	63	
J48	6	37	56.5	69.5	80	
LWL	5	25	48	59.5	78	

Table 5.2: Prediction success rates by classifier for all datasets

Results						
Dataset	min	q1	median	q3	max	Quartile
Set 1	10	21	33	41	53	
Set 2	38	56	59	63	67	
Set 3	27	48	54	63	68	
Set 4	14	30	62.5	75	77	
Set 5	1	5.5	8	11.5	18	
Set 6	0	6	20	35.5	55	
Set 7	12	52	56.5	62	68	
Set 8	10	24	39	51	60	
Set 9	21	33	48.5	70	75	
Set 10	14	34	58.5	77	80	

Table 5.3: Prediction success rates by dataset for all datasets

Results						
k-value	min	q1	median	q3	max	Quartile
1	24	74.5	93	99	100	
3	27	73.5	94	100	100	
6	23	82	94	99	100	
8	25	85.5	94.5	99	100	
11	25	83.5	94.5	99.5	100	
14	22	75.5	91	99	100	

Table 5.4: Filtering success rates by k-value for all datasets

Results						
Classifier	min	q1	median	q3	max	Quartile
Hyperpipes	24	79	94	98	100	
Naive Bayes	27	86	94	98	100	
J48	27	86.5	99	100	100	
LWL	22	66	89	95	100	

Table 5.5: Filtering success rates by classifier for all datasets

Results						
Dataset	min	q1	median	q3	max	Quartile
Set 1	64	72.5	81.5	86	100	
Set 2	100	100	100	100	100	
Set 3	87	92	94	99	100	
Set 4	66	88	93.5	96	100	
Set 5	22	26.5	29.5	74	96	
Set 6	27	33	63	81.5	90	
Set 7	98	98	99	100	100	
Set 8	69	86.5	91.5	94	100	
Set 9	91	93.5	98	100	100	
Set 10	70	89	96	97	100	

Table 5.6: Filtering success rates by dataset for all datasets

k-value	Successes	Failures	Average of Success Rates
1	15850	23115	44.6%
3	15471	23494	42.74%
6	16352	22613	44.78%
8	15728	23237	43.02%
11	14130	24835	39.46%
14	12558	26407	35.32%

Table 5.7: Prediction data with number of predictions by k-value for all datasets

Classifier	Successes	Failures	Average of Success Rates
Hyperpipes	15028	31682	36.45%
Naive Bayes	24105	22665	54.02%
OneR	8939	37831	23.05%
J48	23455	23315	51.78%
LWL	18562	28208	42.97%

Table 5.8: Prediction data with number of predictions by classifier for all datasets

Dataset	Successes	Failures	Average of Success Rates
Set 1	5251	11759	31%
Set 2	3208	2522	55.9%
Set 3	1938	1782	52.1%
Set 4	8471	7369	53.53%
Set 5	2747	29443	8.47%
Set 6	7246	23984	23.17%
Set 7	15055	14135	51.53%
Set 8	12493	22097	36.13%
Set 9	15851	15529	50.47%
Set 10	17829	15081	54.23%

Table 5.9: Prediction data with number of predictions by dataset for all datasets

k-value	Successes	Failures	Average of Success Rates
1	24323	6847	82.02%
3	26014	5156	86.37%
6	26108	5062	86.57%
8	25925	5245	86.1%
11	26352	4818	87%
14	24746	6424	82.52%

Table 5.10: Filtering data with number of predictions by k-value for all datasets

Classifier	Successes	Failures	Average of Success Rates
Hyperpipes	37602	9108	83.95%
Naive Bayes	40941	5829	88.85%
J48	41839	4931	91.82%
LWL	33086	13684	75.78%

Table 5.11: Filtering data with number of predictions by classifier for all datasets

Dataset	Successes	Failures	Average of Success Rates
Set 1	10965	2643	80.62%
Set 2	4572	0	100%
Set 3	2820	156	94.75%
Set 4	11595	1077	91.46%
Set 5	11860	13892	46.04%
Set 6	14897	10087	59.63%
Set 7	23174	178	99.04%
Set 8	24601	3071	88.96%
Set 9	24379	725	97%
Set 10	24605	1723	93.5%

Table 5.12: Filtering data with number of predictions by dataset for all datasets

5.4 Classification

For all of the tested classifiers, training sets are required. We begin by acquiring a basis of ”N” records which will be used as the initial training set. The first ”N” records will therefore not be a part of the predictive algorithm. The algorithm then attempts to classify record ”R”, the first record after the basis ”N”. The classifier's prediction is then compared to ”R”, and the number of correct and incorrect classifications is tallied up. At this point, the algorithm also tallies up the number of ”correctly filtered” predictions; we consider a record ”correctly filtered” if the actual record ”R” was within the classifier's top five predictions.

After the results are recorded, the algorithm adds R to the training set, and moves on to classify R+1. This process continues until the entire data set has been classified.

5.4.1 Hyperpipes

Hyperpipes's performance as a predictor was poor, outperforming only OneR with respect to the other classifiers across all data sets. While its performance as a filterer could be considered adequate, it was still outperformed by all of the other classifiers across all data sets.

5.4.2 Naive Bayes Classification

Naive Bayes's performance as a predictor was superb, outperforming all over the other tested classifiers. Its performance as a filterer was adequate, as it was outperformed only by J48.

5.4.3 J48 Classification

This research's implementation of J48 had the following attributes:

- Confidence threshold for pruning: 0.25
- Minimum number of instances per leaf: 2

- Number of folds for reduced error pruning: 3

J48's performance as a predictor was adequate, outperforming OneR, Hyperpipes, and LWL. Its performance as a filterer was superb, as it outperformed all other tested classifiers.

5.4.4 LWL Classification

This research's implementation of LWL had the following attributes:

- Inputs are normalized.
- All neighbors are used to set kernel bandwidth.
- Weighting kernel is linear.

LWL's performance as a predictor was adequate, outperforming OneR and Hyperpipes, but being outperformed by J48 and Naive Bayes. As a filterer, LWL performed well, but was outperformed by every other tested classifier except Hyperpipes.

5.4.5 OneR Classification

This research's implementation of OneR had the following attributes:

- Minimum bucket size is 6.

As stated in §3.6.5, OneR is a classifier that is not typically expected to perform well. It is therefore no surprise that OneR was outperformed by all of the other tested classifiers when used as a predictor. Due to the nature of OneR's implementation, data could not be collected for its efficacy as a predictor.

5.5 Analysis

Statistical analysis and graphing was done using ezANOVA [32]'s implementation of ANOVA. ANOVA stands for Analysis of Variance, and, in this case, consists of a "repeated measures" design in that it uses the same subjects repeatedly. ANOVA's F-Test seeks to compare the ratio of explained variance to unexplained variance.

For prediction, the following p-values were calculated:

- k-value: p=0.159987
- Classifier: p=0.000001

For filtering, the following p-values were calculated:

- k-value: p=0.079020
- Classifier: p=0.001818

In terms of the hypotheses[2], these results indicate:

- We fail to reject null hypothesis 1 at the 0.05 level of significance. The k-value is not a significant factor for predictions.
- We reject null hypothesis 2 at the 0.05 level of significance. The classifier is a significant factor for predictions.
- We fail to reject null hypothesis 3 at the 0.05 level of significance. The k-value is not a significant factor for filtering.
- We reject null hypothesis 4 at the 0.05 level of significance. The classifier is a significant factor for filtering.

[2] See §2.2

5.6 Summary

This chapter has consisted of an in-depth look at the experimental portion of this thesis. Information was given about how data was collected, as well as how the data was preprocessed and analyzed. The importance of values of ”k” was explained, as well as how these k-values were generated.

This chapter has also directly presented the most interesting and pertinent data for this research. A statistical analysis of this data has been given.

Chapter 6

Conclusions

This chapter provides conclusions garnered from the resulting data found in Chapter 5.

This chapter is therefore divided as follows:

- §6.1 reviews the hypotheses laid out in §2.2 and offers conclusions based on the ANOVA tests performed on §5.3 and shown in §5.5.
- §6.2 explores how the specific results of this research could be applied to producing a knowledge advantage, with respect to the concepts detailed in §4.4.
- A future work section is provided in§6.3, which offers an overview of the possible ”next steps” that this research could take in the pursuit of the implementation of a fully realized KAM.

6.1 Hypotheses

As seen in §5.5, the following conclusions were reached through ANOVA testing:

- The k-value is not a significant factor for predictions.
- The classifier is a significant factor for predictions.

- The k-value is not a significant factor for filtering.
- The classifier is a significant factor for filtering.

One interesting result is that the choice of k-value does not appear to be a factor in both prediction and filtering. However, upon examining the charting in §5.3, it becomes obvious that this is for two separate reasons. In predictions, the data indicates that the k-value is not a factor because, across classifiers, they universally perform poorly. However, in filtering, the data indicates that the k-value is not a factor because they universally perform very well.

The statistical analysis indicates that classifiers are a significant factor for both prediction and filtering. We see in Table 5.8 that, on average, Naive Bayes and J48 were the only classifiers to perform with higher than 50 percent accuracy. Likewise in Table 5.11, we find that Naive Bayes and J48 are the highest performing classifiers, though all classifiers (aside from OneR) performed with above 75 percent accuracy. Regardless, if this workflow prediction concept were to be used for a KAM, it follows that more testing would be useful to determine if there are classifiers which prove superior to the ones tested in this research.

Finally, we see in Table 5.12 that a few sets of data appear significantly less efficient for filtering. This suggests that certain patterns of file access exist which are less copacetic with the type of prediction-filtering-via-classification-learning which is presented within this research. Put simply, some workflow patterns do not lend themselves to workflow analysis. For sets with such patterns, it might become more prudent to place more importance on alternate forms of context identification, such as semantic analysis.

6.2 How the Advantage is Gained

As seen in §4.4 and its subsections, we have explored many areas in which an advantage might be gained through the use of this technology. Now that we have empirically shown the efficacy of

filtering over prediction, each of these concepts can be reexamined with the promising results of filtering in mind.

6.2.1 Context Awareness

As discussed in §4.4.1, context awareness is necessary for any KAM and workflow prediction's main goal is to assist in this awareness of context. With filtering, a WPCKAM may be able to identify contextually similar JANs. Since this thesis has empirically shown that filtering works, one might start by tallying up occurences of JANs appearing with other JANs in the sets of "top T" predictions.

Contextually similar JANs might be defined as JANs who appear concurrently within the "top T" predictions beyond a certain threshold of times. Once an awareness of conextual similarity is established, we might then proceed to use this information for any task which requires context awareness. Furthermore, we might semantically analyze the JANs which have been identified as contextually similar through workflow prediction; this combination of methods might lead us to a more robust and accurate context prediction mechanic.

6.2.2 Presenting Predicted Files

Because filtering has been shown to work, prediction presentation could be accomplished without any further research. This thesis has provided an agent capable of filtering file accesses with a high degree of success; therefore, a tool could be created which simply pushes these "top T" predictions to a pane or folder as described in §4.4.2. This is a simple example of a knowledge advantage which could be quickly and easily be created using only the methods described in this research.

6.2.3 Measuring JAN Importance

Using filtering, JAN importance could be assessed by tallying up how frequently various JANs appear in the ”top T” predicted results. Various thresholds of importance could be arbitrarily set, and once a JAN crosses one of these thresholds, it might be selected for semantic analysis. As described in §4.4.3, we might then have the discovery agent search for similar JANs and place them into the user's personal ontology.

6.3 Future Work

A great abundance of future work waits to be explored in the area of KAM research. For future work pertaining specifically to work-patterns, however, there are some interesting research avenues that the results of this thesis point towards. These include:

- Test more learning techniques. Five classifiers were tested in this thesis; many other classification techniques exist. Furthermore, learning techniques such as clustering could be employed and examined for efficacy.

- Examine other types of workflow patterns aside from file usage data, using techniques similar to those used in this thesis. One example would be applying classification techniques to web usage data.

- Combine the work-pattern centric approach to context prediction with other context identification techniques. Algorithms which perform semantic analysis, for example, could be augmented by examining the filtered predictions provided by workflow prediction.

- Examine these concepts as they pertain to predicting application use in addition to file usage, such as seen in [26].

Appendix A

Reproducing the Experiments

This appendix provides information on how to replicate the described experiments via the same methods used in this thesis. Various scripts and instructions are provided. Sample data is not provided, due to privacy concerns; however, specific instructions on how to collect data are shown.

A.1 Acquiring the Software

The scripts used to preprocess the data are available at the WVU'S CERC server. It is suggested that these tools be used on a Unix system, as some of the preprocessing is done using Unix command line utilities. A zip file of all the software is available at `cerc.wvu.edu/kam/wpckam.zip`. This contains:

- Two pdf files, "auditingbegin.pdf" and "auditingend.pdf." These are instructions on how to start and stop file usage auditing via Windows 7 Pro.
- One PHP script, "auditing.php". This will take in an XML file and print out filenames in sequence. Output should be sent to a text file.
- One extensionless script, "preprocess". This performs some preliminary whitelisting on the

output file generated with auditing.php.

- Two Java sourcecode files, FileUsageDataAnalyzer.java and Main.java. These should be jarred up and run. Using the preprocessed file, they apply more preprocessing, generate MD5 hashed CSV files, and perform classifications in sequence.

- Two jar files, "commons-lang3-3.1jar" and "weka.jar". These are included for convenience, and are necessary for running the Java files listed above.

A.2 Obtaining Data

After running the provided Java code, a file will be generated based on the name of the dataset file, the value of k used, and the classifier used. The first line of this file will consist of the number of successful predictions using that classifier, then a comma, then the number of unsuccessful predictions using that classifier. The second line of this file will consist of the number of successful filterings using a "T" value of 5, then a comma, then the number of unsuccessful filterings using that classifier.

From the resultant data, whatever statistical analysis you deem appropriate can be performed.

A.3 Running your Experiments

After XML datasets have been acquired using the instructions found in the two hosted pdf files described in §A.1, classification can begin. The provided suite of tools can be used on a Unix system as follows:

- Run "auditing.php" from the command line, using the syntax "php auditing.php filename.XML CSV". Send the output of this command to an extensionless text file, "output".

- Run ”preprocess” from the command line. This will generate a preprocessed file, outputFinal.txt.
- After jarring the sourcecode files, run the jar with the outputFinal file in the same folder.
  - The first command line argument is the name of the dataset file.
  - The second command line argument is the number of records to use as the basis training set.
  - The third command line argument is the ”x” in the xlogx scale used to generate the k-value.
  - The fourth command line argument is the classifier: options are ”PIPES”, ”BAYES”, ”LWL”, ”J48”, and ”ONER”. It is also important to provide the Java virtual machine with enough memory. Therefore, a sample command line run would be: ”java -Xms256m -Xmx1500m -jar Analyzer.jar outputFinal 100 2 PIPES”.

Appendix B

Raw Data and Charts

B.1 Various Charts

Table B.2: Prediction success rates by all factors for all datasets

	Successes	Failures	Average of Success Rates
Set 1 k1 Hyperpipes	173	394	31%
Set 1 k1 Naive Bayes	272	295	48%
Set 1 k1 OneR	62	505	11%
Set 1 k1 J48	210	357	37%

Continuing next page

	k-values						
Classifiers	1	3	6	8	11	14	17
Hyperpipes	1-5	1-5	1-5	1-5	1-5	1-5	1-5
Naive Bayes	1-5	1-5	1-5	1-5	1-5	1-5	1-5
OneR	1-5	1-5	1-5	1-5	1-5	1-5	1-5
J48	1-5	1-5	1-5	1-5	1-5	1-5	1-5
LWL	1-5	1-5	1-5	1-5	1-5	1-5	1-5

Table B.1: Values of k

	Successes	Failures	Average of Success Rates
Set 1 k1 LWL	231	336	41%
Set 1 k3 Hyperpipes	175	392	31%
Set 1 k3 Naive Bayes	236	331	42%
Set 1 k3 OneR	61	506	11%
Set 1 k3 J48	210	357	37%
Set 1 k3 LWL	168	399	30%
Set 1 k6 Hyperpipes	219	348	39%
Set 1 k6 Naive Bayes	300	267	53%
Set 1 k6 OneR	62	505	11%
Set 1 k6 J48	241	326	43%
Set 1 k6 LWL	196	371	35%
Set 1 k8 Hyperpipes	202	365	36%
Set 1 k8 Naive Bayes	290	277	51%
Set 1 k8 OneR	60	507	11%
Set 1 k8 J48	233	334	41%
Set 1 k8 LWL	186	381	33%
Set 1 k11 Hyperpipes	151	416	27%
Set 1 k11 Naive Bayes	185	382	33%
Set 1 k11 OneR	61	506	11%
Set 1 k11 J48	166	401	29%
Set 1 k11 LWL	154	413	27%
Set 1 k14 Hyperpipes	133	434	23%

Continuing next page

	Successes	Failures	Average of Success Rates
Set 1 k14 Naive Bayes	251	316	44%
Set 1 k14 OneR	59	508	10%
Set 1 k14 J48	197	370	35%
Set 1 k14 LWL	107	460	19%
Set 2 k1 Hyperpipes	70	113	38%
Set 2 k1 Naive Bayes	129	64	67%
Set 2 k1 OneR	121	72	63%
Set 2 k1 J48	126	67	65%
Set 2 k1 LWL	129	64	67%
Set 2 k3 Hyperpipes	69	114	38%
Set 2 k3 Naive Bayes	115	78	60%
Set 2 k3 OneR	112	81	58%
Set 2 k3 J48	110	83	57%
Set 2 k3 LWL	116	77	60%
Set 2 k6 Hyperpipes	69	114	38%
Set 2 k6 Naive Bayes	127	66	66%
Set 2 k6 OneR	118	75	61%
Set 2 k6 J48	122	71	63%
Set 2 k6 LWL	122	71	63%
Set 2 k8 Hyperpipes	69	114	38%
Set 2 k8 Naive Bayes	115	78	60%
Set 2 k8 OneR	112	81	58%

Continuing next page

	Successes	Failures	Average of Success Rates
Set 2 k8 J48	114	79	59%
Set 2 k8 LWL	113	80	59%
Set 2 k11 Hyperpipes	69	114	38%
Set 2 k11 Naive Bayes	113	80	59%
Set 2 k11 OneR	109	84	56%
Set 2 k11 J48	112	81	58%
Set 2 k11 LWL	113	80	59%
Set 2 k14 Hyperpipes	70	113	38%
Set 2 k14 Naive Bayes	111	82	58%
Set 2 k14 OneR	109	84	56%
Set 2 k14 J48	111	82	58%
Set 2 k14 LWL	113	80	59%
Set 3 k1 Hyperpipes	75	49	60%
Set 3 k1 Naive Bayes	84	40	68%
Set 3 k1 OneR	37	87	30%
Set 3 k1 J48	78	46	63%
Set 3 k1 LWL	82	42	66%
Set 3 k3 Hyperpipes	78	46	63%
Set 3 k3 Naive Bayes	82	42	66%
Set 3 k3 OneR	38	86	31%
Set 3 k3 J48	66	58	53%
Set 3 k3 LWL	65	59	52%

Continuing next page

	Successes	Failures	Average of Success Rates
Set 3 k6 Hyperpipes	77	47	62%
Set 3 k6 Naive Bayes	80	44	65%
Set 3 k6 OneR	39	85	31%
Set 3 k6 J48	66	58	53%
Set 3 k6 LWL	60	64	48%
Set 3 k8 Hyperpipes	76	48	61%
Set 3 k8 Naive Bayes	78	46	63%
Set 3 k8 OneR	37	87	30%
Set 3 k8 J48	68	56	55%
Set 3 k8 LWL	59	65	48%
Set 3 k11 Hyperpipes	81	43	65%
Set 3 k11 Naive Bayes	74	50	60%
Set 3 k11 OneR	35	89	28%
Set 3 k11 J48	63	61	51%
Set 3 k11 LWL	59	65	48%
Set 3 k14 Hyperpipes	78	46	63%
Set 3 k14 Naive Bayes	68	56	55%
Set 3 k14 OneR	34	90	27%
Set 3 k14 J48	64	60	52%
Set 3 k14 LWL	57	67	46%
Set 4 k1 Hyperpipes	367	161	70%
Set 4 k1 Naive Bayes	394	134	75%

Continuing next page

	Successes	Failures	Average of Success Rates
Set 4 k1 OneR	83	445	16%
Set 4 k1 J48	398	130	75%
Set 4 k1 LWL	404	124	77%
Set 4 k3 Hyperpipes	344	184	65%
Set 4 k3 Naive Bayes	404	124	77%
Set 4 k3 OneR	78	450	15%
Set 4 k3 J48	398	130	75%
Set 4 k3 LWL	333	195	63%
Set 4 k6 Hyperpipes	331	197	63%
Set 4 k6 Naive Bayes	399	129	76%
Set 4 k6 OneR	77	451	15%
Set 4 k6 J48	396	132	75%
Set 4 k6 LWL	301	227	57%
Set 4 k8 Hyperpipes	330	198	62%
Set 4 k8 Naive Bayes	385	143	73%
Set 4 k8 OneR	76	452	14%
Set 4 k8 J48	394	134	75%
Set 4 k8 LWL	292	236	55%
Set 4 k11 Hyperpipes	326	202	62%
Set 4 k11 Naive Bayes	374	154	71%
Set 4 k11 OneR	74	454	14%
Set 4 k11 J48	385	143	73%

Continuing next page

	Successes	Failures	Average of Success Rates
Set 4 k11 LWL	263	265	50%
Set 4 k14 Hyperpipes	160	368	30%
Set 4 k14 Naive Bayes	219	309	41%
Set 4 k14 OneR	78	450	15%
Set 4 k14 J48	274	254	52%
Set 4 k14 LWL	134	394	25%
Set 5 k1 Hyperpipes	86	987	8%
Set 5 k1 Naive Bayes	159	914	15%
Set 5 k1 OneR	15	1058	1%
Set 5 k1 J48	66	1007	6%
Set 5 k1 LWL	72	1001	7%
Set 5 k3 Hyperpipes	106	967	10%
Set 5 k3 Naive Bayes	115	958	11%
Set 5 k3 OneR	15	1058	1%
Set 5 k3 J48	82	991	8%
Set 5 k3 LWL	57	1016	5%
Set 5 k6 Hyperpipes	175	898	16%
Set 5 k6 Naive Bayes	192	881	18%
Set 5 k6 OneR	15	1058	1%
Set 5 k6 J48	111	962	10%
Set 5 k6 LWL	65	1008	6%
Set 5 k8 Hyperpipes	165	908	15%

Continuing next page

	Successes	Failures	Average of Success Rates
Set 5 k8 Naive Bayes	186	887	17%
Set 5 k8 OneR	16	1057	1%
Set 5 k8 J48	120	953	11%
Set 5 k8 LWL	68	1005	6%
Set 5 k11 Hyperpipes	103	970	10%
Set 5 k11 Naive Bayes	117	956	11%
Set 5 k11 OneR	18	1055	2%
Set 5 k11 J48	76	997	7%
Set 5 k11 LWL	74	999	7%
Set 5 k14 Hyperpipes	130	943	12%
Set 5 k14 Naive Bayes	160	913	15%
Set 5 k14 OneR	17	1056	2%
Set 5 k14 J48	101	972	9%
Set 5 k14 LWL	65	1008	6%
Set 6 k1 Hyperpipes	181	860	17%
Set 6 k1 Naive Bayes	66	975	6%
Set 6 k1 OneR	0	1041	0%
Set 6 k1 J48	62	979	6%
Set 6 k1 LWL	68	973	7%
Set 6 k3 Hyperpipes	316	725	30%
Set 6 k3 Naive Bayes	294	747	28%
Set 6 k3 OneR	0	1041	0%

Continuing next page

	Successes	Failures	Average of Success Rates
Set 6 k3 J48	322	719	31%
Set 6 k3 LWL	198	843	19%
Set 6 k6 Hyperpipes	577	464	55%
Set 6 k6 Naive Bayes	539	502	52%
Set 6 k6 OneR	0	1041	0%
Set 6 k6 J48	312	729	30%
Set 6 k6 LWL	203	838	20%
Set 6 k8 Hyperpipes	514	527	49%
Set 6 k8 Naive Bayes	525	516	50%
Set 6 k8 OneR	0	1041	0%
Set 6 k8 J48	317	724	30%
Set 6 k8 LWL	196	845	19%
Set 6 k11 Hyperpipes	206	835	20%
Set 6 k11 Naive Bayes	504	537	48%
Set 6 k11 OneR	0	1041	0%
Set 6 k11 J48	321	720	31%
Set 6 k11 LWL	174	867	17%
Set 6 k14 Hyperpipes	415	626	40%
Set 6 k14 Naive Bayes	475	566	46%
Set 6 k14 OneR	0	1041	0%
Set 6 k14 J48	315	726	30%
Set 6 k14 LWL	146	895	14%

Continuing next page

	Successes	Failures	Average of Success Rates
Set 7 k1 Hyperpipes	362	611	37%
Set 7 k1 Naive Bayes	635	338	65%
Set 7 k1 OneR	596	377	61%
Set 7 k1 J48	637	336	65%
Set 7 k1 LWL	644	329	66%
Set 7 k3 Hyperpipes	313	660	32%
Set 7 k3 Naive Bayes	601	372	62%
Set 7 k3 OneR	546	427	56%
Set 7 k3 J48	624	349	64%
Set 7 k3 LWL	580	393	60%
Set 7 k6 Hyperpipes	128	845	13%
Set 7 k6 Naive Bayes	649	324	67%
Set 7 k6 OneR	589	384	61%
Set 7 k6 J48	663	310	68%
Set 7 k6 LWL	605	368	62%
Set 7 k8 Hyperpipes	128	845	13%
Set 7 k8 Naive Bayes	557	416	57%
Set 7 k8 OneR	548	425	56%
Set 7 k8 J48	601	372	62%
Set 7 k8 LWL	565	408	58%
Set 7 k11 Hyperpipes	120	853	12%
Set 7 k11 Naive Bayes	502	471	52%

Continuing next page

	Successes	Failures	Average of Success Rates
Set 7 k11 OneR	531	442	55%
Set 7 k11 J48	566	407	58%
Set 7 k11 LWL	548	425	56%
Set 7 k14 Hyperpipes	137	836	14%
Set 7 k14 Naive Bayes	485	488	50%
Set 7 k14 OneR	516	457	53%
Set 7 k14 J48	542	431	56%
Set 7 k14 LWL	537	436	55%
Set 8 k1 Hyperpipes	322	831	28%
Set 8 k1 Naive Bayes	666	487	58%
Set 8 k1 OneR	126	1027	11%
Set 8 k1 J48	593	560	51%
Set 8 k1 LWL	612	541	53%
Set 8 k3 Hyperpipes	356	797	31%
Set 8 k3 Naive Bayes	534	619	46%
Set 8 k3 OneR	126	1027	11%
Set 8 k3 J48	601	552	52%
Set 8 k3 LWL	375	778	33%
Set 8 k6 Hyperpipes	434	719	38%
Set 8 k6 Naive Bayes	659	494	57%
Set 8 k6 OneR	126	1027	11%
Set 8 k6 J48	689	464	60%

Continuing next page

	Successes	Failures	Average of Success Rates
Set 8 k6 LWL	475	678	41%
Set 8 k8 Hyperpipes	402	751	35%
Set 8 k8 Naive Bayes	643	510	56%
Set 8 k8 OneR	125	1028	11%
Set 8 k8 J48	657	496	57%
Set 8 k8 LWL	457	696	40%
Set 8 k11 Hyperpipes	280	873	24%
Set 8 k11 Naive Bayes	458	695	40%
Set 8 k11 OneR	124	1029	11%
Set 8 k11 J48	522	631	45%
Set 8 k11 LWL	337	816	29%
Set 8 k14 Hyperpipes	238	915	21%
Set 8 k14 Naive Bayes	558	595	48%
Set 8 k14 OneR	121	1032	10%
Set 8 k14 J48	584	569	51%
Set 8 k14 LWL	293	860	25%
Set 9 k1 Hyperpipes	280	766	27%
Set 9 k1 Naive Bayes	776	270	74%
Set 9 k1 OneR	354	692	34%
Set 9 k1 J48	772	274	74%
Set 9 k1 LWL	786	260	75%
Set 9 k3 Hyperpipes	307	739	29%

Continuing next page

	Successes	Failures	Average of Success Rates
Set 9 k3 Naive Bayes	750	296	72%
Set 9 k3 OneR	349	697	33%
Set 9 k3 J48	742	304	71%
Set 9 k3 LWL	632	414	60%
Set 9 k6 Hyperpipes	264	782	25%
Set 9 k6 Naive Bayes	721	325	69%
Set 9 k6 OneR	350	696	33%
Set 9 k6 J48	728	318	70%
Set 9 k6 LWL	520	526	50%
Set 9 k8 Hyperpipes	263	783	25%
Set 9 k8 Naive Bayes	700	346	67%
Set 9 k8 OneR	350	696	33%
Set 9 k8 J48	746	300	71%
Set 9 k8 LWL	490	556	47%
Set 9 k11 Hyperpipes	266	780	25%
Set 9 k11 Naive Bayes	692	354	66%
Set 9 k11 OneR	349	697	33%
Set 9 k11 J48	763	283	73%
Set 9 k11 LWL	480	566	46%
Set 9 k14 Hyperpipes	216	830	21%
Set 9 k14 Naive Bayes	645	401	62%
Set 9 k14 OneR	349	697	33%

Continuing next page

	Successes	Failures	Average of Success Rates
Set 9 k14 J48	718	328	69%
Set 9 k14 LWL	493	553	47%
Set 10 k1 Hyperpipes	687	410	63%
Set 10 k1 Naive Bayes	841	256	77%
Set 10 k1 OneR	161	936	15%
Set 10 k1 J48	849	248	77%
Set 10 k1 LWL	851	246	78%
Set 10 k3 Hyperpipes	636	461	58%
Set 10 k3 Naive Bayes	867	230	79%
Set 10 k3 OneR	157	940	14%
Set 10 k3 J48	873	224	80%
Set 10 k3 LWL	739	358	67%
Set 10 k6 Hyperpipes	623	474	57%
Set 10 k6 Naive Bayes	851	246	78%
Set 10 k6 OneR	153	944	14%
Set 10 k6 J48	869	228	79%
Set 10 k6 LWL	665	432	61%
Set 10 k8 Hyperpipes	626	471	57%
Set 10 k8 Naive Bayes	814	283	74%
Set 10 k8 OneR	151	946	14%
Set 10 k8 J48	862	235	79%
Set 10 k8 LWL	647	450	59%

Continuing next page

	Successes	Failures	Average of Success Rates
Set 10 k11 Hyperpipes	619	478	56%
Set 10 k11 Naive Bayes	802	295	73%
Set 10 k11 OneR	151	946	14%
Set 10 k11 J48	842	255	77%
Set 10 k11 LWL	618	479	56%
Set 10 k14 Hyperpipes	255	842	23%
Set 10 k14 Naive Bayes	482	615	44%
Set 10 k14 OneR	163	934	15%
Set 10 k14 J48	605	492	55%
Set 10 k14 LWL	370	727	34%

Table B.3: Filtering success rates by all factors for all datasets

	Successes	Failures	Average of Success Rates
Set 1 k1 Hyperpipes	453	114	80%
Set 1 k1 Naive Bayes	393	174	69%
Set 1 k1 OneR	N/A	N/A	N/A
Set 1 k1 J48	479	88	84%
Set 1 k1 LWL	463	104	82%
Set 1 k3 Hyperpipes	473	94	83%
Set 1 k3 Naive Bayes	487	80	86%
Set 1 k3 OneR	N/A	N/A	N/A
Set 1 k3 J48	491	76	87%
Set 1 k3 LWL	392	175	69%

Continuing next page

	Successes	Failures	Average of Success Rates
Set 1 k6 Hyperpipes	464	103	82%
Set 1 k6 Naive Bayes	490	77	86%
Set 1 k6 OneR	N/A	N/A	N/A
Set 1 k6 J48	530	37	93%
Set 1 k6 LWL	385	182	68%
Set 1 k8 Hyperpipes	459	108	81%
Set 1 k8 Naive Bayes	475	92	84%
Set 1 k8 OneR	N/A	N/A	N/A
Set 1 k8 J48	537	30	95%
Set 1 k8 LWL	374	193	66%
Set 1 k11 Hyperpipes	448	119	79%
Set 1 k11 Naive Bayes	451	116	80%
Set 1 k11 OneR	N/A	N/A	N/A
Set 1 k11 J48	567	0	100%
Set 1 k11 LWL	361	206	64%
Set 1 k14 Hyperpipes	435	132	77%
Set 1 k14 Naive Bayes	432	135	76%
Set 1 k14 OneR	N/A	N/A	N/A
Set 1 k14 J48	565	2	100%
Set 1 k14 LWL	361	206	64%
Set 2 k1 Hyperpipes	183	0	100%
Set 2 k1 Naive Bayes	193	0	100%

Continuing next page

	Successes	Failures	Average of Success Rates
Set 2 k1 OneR	N/A	N/A	N/A
Set 2 k1 J48	193	0	100%
Set 2 k1 LWL	193	0	100%
Set 2 k3 Hyperpipes	183	0	100%
Set 2 k3 Naive Bayes	193	0	100%
Set 2 k3 OneR	N/A	N/A	N/A
Set 2 k3 J48	193	0	100%
Set 2 k3 LWL	193	0	100%
Set 2 k6 Hyperpipes	183	0	100%
Set 2 k6 Naive Bayes	193	0	100%
Set 2 k6 OneR	N/A	N/A	N/A
Set 2 k6 J48	193	0	100%
Set 2 k6 LWL	193	0	100%
Set 2 k8 Hyperpipes	183	0	100%
Set 2 k8 Naive Bayes	193	0	100%
Set 2 k8 OneR	N/A	N/A	N/A
Set 2 k8 J48	193	0	100%
Set 2 k8 LWL	193	0	100%
Set 2 k11 Hyperpipes	183	0	100%
Set 2 k11 Naive Bayes	193	0	100%
Set 2 k11 OneR	N/A	N/A	N/A
Set 2 k11 J48	193	0	100%

Continuing next page

	Successes	Failures	Average of Success Rates
Set 2 k11 LWL	193	0	100%
Set 2 k14 Hyperpipes	183	0	100%
Set 2 k14 Naive Bayes	193	0	100%
Set 2 k14 OneR	N/A	N/A	N/A
Set 2 k14 J48	193	0	100%
Set 2 k14 LWL	193	0	100%
Set 3 k1 Hyperpipes	115	9	93%
Set 3 k1 Naive Bayes	108	16	87%
Set 3 k1 OneR	N/A	N/A	N/A
Set 3 k1 J48	124	0	100%
Set 3 k1 LWL	116	8	94%
Set 3 k3 Hyperpipes	118	6	95%
Set 3 k3 Naive Bayes	116	8	94%
Set 3 k3 OneR	N/A	N/A	N/A
Set 3 k3 J48	124	0	100%
Set 3 k3 LWL	116	8	94%
Set 3 k6 Hyperpipes	118	6	95%
Set 3 k6 Naive Bayes	117	7	94%
Set 3 k6 OneR	N/A	N/A	N/A
Set 3 k6 J48	123	1	99%
Set 3 k6 LWL	114	10	92%
Set 3 k8 Hyperpipes	119	5	96%

Continuing next page

	Successes	Failures	Average of Success Rates
Set 3 k8 Naive Bayes	117	7	94%
Set 3 k8 OneR	N/A	N/A	N/A
Set 3 k8 J48	124	0	100%
Set 3 k8 LWL	112	12	90%
Set 3 k11 Hyperpipes	117	7	94%
Set 3 k11 Naive Bayes	115	9	93%
Set 3 k11 OneR	N/A	N/A	N/A
Set 3 k11 J48	124	0	100%
Set 3 k11 LWL	113	11	91%
Set 3 k14 Hyperpipes	118	6	95%
Set 3 k14 Naive Bayes	115	9	93%
Set 3 k14 OneR	N/A	N/A	N/A
Set 3 k14 J48	123	1	99%
Set 3 k14 LWL	114	10	92%
Set 4 k1 Hyperpipes	490	38	93%
Set 4 k1 Naive Bayes	479	49	91%
Set 4 k1 OneR	N/A	N/A	N/A
Set 4 k1 J48	507	21	96%
Set 4 k1 LWL	489	39	93%
Set 4 k3 Hyperpipes	497	31	94%
Set 4 k3 Naive Bayes	498	30	94%
Set 4 k3 OneR	N/A	N/A	N/A

Continuing next page

	Successes	Failures	Average of Success Rates
Set 4 k3 J48	528	0	100%
Set 4 k3 LWL	469	59	89%
Set 4 k6 Hyperpipes	498	30	94%
Set 4 k6 Naive Bayes	506	22	96%
Set 4 k6 OneR	N/A	N/A	N/A
Set 4 k6 J48	528	0	100%
Set 4 k6 LWL	458	70	87%
Set 4 k8 Hyperpipes	498	30	94%
Set 4 k8 Naive Bayes	509	19	96%
Set 4 k8 OneR	N/A	N/A	N/A
Set 4 k8 J48	528	0	100%
Set 4 k8 LWL	459	69	87%
Set 4 k11 Hyperpipes	493	35	93%
Set 4 k11 Naive Bayes	499	29	95%
Set 4 k11 OneR	N/A	N/A	N/A
Set 4 k11 J48	528	0	100%
Set 4 k11 LWL	464	64	88%
Set 4 k14 Hyperpipes	467	61	88%
Set 4 k14 Naive Bayes	398	130	75%
Set 4 k14 OneR	N/A	N/A	N/A
Set 4 k14 J48	454	74	86%
Set 4 k14 LWL	351	177	66%

Continuing next page

	Successes	Failures	Average of Success Rates
Set 5 k1 Hyperpipes	255	818	24%
Set 5 k1 Naive Bayes	337	736	31%
Set 5 k1 OneR	N/A	N/A	N/A
Set 5 k1 J48	287	786	27%
Set 5 k1 LWL	293	780	27%
Set 5 k3 Hyperpipes	321	752	30%
Set 5 k3 Naive Bayes	992	81	92%
Set 5 k3 OneR	N/A	N/A	N/A
Set 5 k3 J48	716	357	67%
Set 5 k3 LWL	291	782	27%
Set 5 k6 Hyperpipes	319	754	30%
Set 5 k6 Naive Bayes	987	86	92%
Set 5 k6 OneR	N/A	N/A	N/A
Set 5 k6 J48	558	515	52%
Set 5 k6 LWL	244	829	23%
Set 5 k8 Hyperpipes	314	759	29%
Set 5 k8 Naive Bayes	967	106	90%
Set 5 k8 OneR	N/A	N/A	N/A
Set 5 k8 J48	417	656	39%
Set 5 k8 LWL	270	803	25%
Set 5 k11 Hyperpipes	301	772	28%
Set 5 k11 Naive Bayes	872	201	81%

Continuing next page

	Successes	Failures	Average of Success Rates
Set 5 k11 OneR	N/A	N/A	N/A
Set 5 k11 J48	1023	50	95%
Set 5 k11 LWL	264	809	25%
Set 5 k14 Hyperpipes	279	794	26%
Set 5 k14 Naive Bayes	289	784	27%
Set 5 k14 OneR	N/A	N/A	N/A
Set 5 k14 J48	1032	41	96%
Set 5 k14 LWL	232	841	22%
Set 6 k1 Hyperpipes	400	641	38%
Set 6 k1 Naive Bayes	382	659	37%
Set 6 k1 OneR	N/A	N/A	N/A
Set 6 k1 J48	539	502	52%
Set 6 k1 LWL	507	534	49%
Set 6 k3 Hyperpipes	661	380	63%
Set 6 k3 Naive Bayes	706	335	68%
Set 6 k3 OneR	N/A	N/A	N/A
Set 6 k3 J48	713	328	68%
Set 6 k3 LWL	289	752	28%
Set 6 k6 Hyperpipes	658	383	63%
Set 6 k6 Naive Bayes	939	102	90%
Set 6 k6 OneR	N/A	N/A	N/A
Set 6 k6 J48	744	297	71%

Continuing next page

	Successes	Failures	Average of Success Rates
Set 6 k6 LWL	300	741	29%
Set 6 k8 Hyperpipes	657	384	63%
Set 6 k8 Naive Bayes	922	119	89%
Set 6 k8 OneR	N/A	N/A	N/A
Set 6 k8 J48	789	252	76%
Set 6 k8 LWL	304	737	29%
Set 6 k11 Hyperpipes	655	386	63%
Set 6 k11 Naive Bayes	893	148	86%
Set 6 k11 OneR	N/A	N/A	N/A
Set 6 k11 J48	832	209	80%
Set 6 k11 LWL	296	745	28%
Set 6 k14 Hyperpipes	645	396	62%
Set 6 k14 Naive Bayes	862	179	83%
Set 6 k14 OneR	N/A	N/A	N/A
Set 6 k14 J48	926	115	89%
Set 6 k14 LWL	278	763	27%
Set 7 k1 Hyperpipes	966	7	99%
Set 7 k1 Naive Bayes	965	8	99%
Set 7 k1 OneR	N/A	N/A	N/A
Set 7 k1 J48	970	3	100%
Set 7 k1 LWL	967	6	99%
Set 7 k3 Hyperpipes	967	6	99%

Continuing next page

	Successes	Failures	Average of Success Rates
Set 7 k3 Naive Bayes	969	4	100%
Set 7 k3 OneR	N/A	N/A	N/A
Set 7 k3 J48	967	6	99%
Set 7 k3 LWL	958	15	98%
Set 7 k6 Hyperpipes	967	6	99%
Set 7 k6 Naive Bayes	968	5	99%
Set 7 k6 OneR	N/A	N/A	N/A
Set 7 k6 J48	971	2	100%
Set 7 k6 LWL	966	7	99%
Set 7 k8 Hyperpipes	967	6	99%
Set 7 k8 Naive Bayes	967	6	99%
Set 7 k8 OneR	N/A	N/A	N/A
Set 7 k8 J48	966	7	99%
Set 7 k8 LWL	956	17	98%
Set 7 k11 Hyperpipes	967	6	99%
Set 7 k11 Naive Bayes	968	5	99%
Set 7 k11 OneR	N/A	N/A	N/A
Set 7 k11 J48	965	8	99%
Set 7 k11 LWL	961	12	99%
Set 7 k14 Hyperpipes	967	6	99%
Set 7 k14 Naive Bayes	968	5	99%
Set 7 k14 OneR	N/A	N/A	N/A

Continuing next page

	Successes	Failures	Average of Success Rates
Set 7 k14 J48	962	11	99%
Set 7 k14 LWL	959	14	99%
Set 8 k1 Hyperpipes	1046	107	91%
Set 8 k1 Naive Bayes	992	161	86%
Set 8 k1 OneR	N/A	N/A	N/A
Set 8 k1 J48	1065	88	92%
Set 8 k1 LWL	1053	100	91%
Set 8 k3 Hyperpipes	1083	70	94%
Set 8 k3 Naive Bayes	1083	70	94%
Set 8 k3 OneR	N/A	N/A	N/A
Set 8 k3 J48	1079	74	94%
Set 8 k3 LWL	895	258	78%
Set 8 k6 Hyperpipes	1079	74	94%
Set 8 k6 Naive Bayes	1083	70	94%
Set 8 k6 OneR	N/A	N/A	N/A
Set 8 k6 J48	1105	48	96%
Set 8 k6 LWL	864	289	75%
Set 8 k8 Hyperpipes	1067	86	93%
Set 8 k8 Naive Bayes	1065	88	92%
Set 8 k8 OneR	N/A	N/A	N/A
Set 8 k8 J48	1117	36	97%
Set 8 k8 LWL	853	300	74%

Continuing next page

	Successes	Failures	Average of Success Rates
Set 8 k11 Hyperpipes	1052	101	91%
Set 8 k11 Naive Bayes	1046	107	91%
Set 8 k11 OneR	N/A	N/A	N/A
Set 8 k11 J48	1153	0	100%
Set 8 k11 LWL	833	320	72%
Set 8 k14 Hyperpipes	1035	118	90%
Set 8 k14 Naive Bayes	1008	145	87%
Set 8 k14 OneR	N/A	N/A	N/A
Set 8 k14 J48	1152	1	100%
Set 8 k14 LWL	793	360	69%
Set 9 k1 Hyperpipes	1029	17	98%
Set 9 k1 Naive Bayes	1017	29	97%
Set 9 k1 OneR	N/A	N/A	N/A
Set 9 k1 J48	1036	10	99%
Set 9 k1 LWL	1026	20	98%
Set 9 k3 Hyperpipes	1029	17	98%
Set 9 k3 Naive Bayes	1022	24	98%
Set 9 k3 OneR	N/A	N/A	N/A
Set 9 k3 J48	1042	4	100%
Set 9 k3 LWL	957	89	91%
Set 9 k6 Hyperpipes	1029	17	98%
Set 9 k6 Naive Bayes	1028	18	98%

Continuing next page

	Successes	Failures	Average of Success Rates
Set 9 k6 OneR	N/A	N/A	N/A
Set 9 k6 J48	1041	5	100%
Set 9 k6 LWL	971	75	93%
Set 9 k8 Hyperpipes	1029	17	98%
Set 9 k8 Naive Bayes	1021	25	98%
Set 9 k8 OneR	N/A	N/A	N/A
Set 9 k8 J48	1046	0	100%
Set 9 k8 LWL	956	90	91%
Set 9 k11 Hyperpipes	1028	18	98%
Set 9 k11 Naive Bayes	1009	37	96%
Set 9 k11 OneR	N/A	N/A	N/A
Set 9 k11 J48	1045	1	100%
Set 9 k11 LWL	957	89	91%
Set 9 k14 Hyperpipes	1027	19	98%
Set 9 k14 Naive Bayes	1002	44	96%
Set 9 k14 OneR	N/A	N/A	N/A
Set 9 k14 J48	1044	2	100%
Set 9 k14 LWL	988	58	94%
Set 10 k1 Hyperpipes	1053	44	96%
Set 10 k1 Naive Bayes	1037	60	95%
Set 10 k1 OneR	N/A	N/A	N/A
Set 10 k1 J48	1072	25	98%

Continuing next page

	Successes	Failures	Average of Success Rates
Set 10 k1 LWL	1051	46	96%
Set 10 k3 Hyperpipes	1062	35	97%
Set 10 k3 Naive Bayes	1056	41	96%
Set 10 k3 OneR	N/A	N/A	N/A
Set 10 k3 J48	1092	5	100%
Set 10 k3 LWL	993	104	91%
Set 10 k6 Hyperpipes	1064	33	97%
Set 10 k6 Naive Bayes	1066	31	97%
Set 10 k6 OneR	N/A	N/A	N/A
Set 10 k6 J48	1097	0	100%
Set 10 k6 LWL	967	130	88%
Set 10 k8 Hyperpipes	1064	33	97%
Set 10 k8 Naive Bayes	1067	30	97%
Set 10 k8 OneR	N/A	N/A	N/A
Set 10 k8 J48	1096	1	100%
Set 10 k8 LWL	975	122	89%
Set 10 k11 Hyperpipes	1059	38	97%
Set 10 k11 Naive Bayes	1055	42	96%
Set 10 k11 OneR	N/A	N/A	N/A
Set 10 k11 J48	1097	0	100%
Set 10 k11 LWL	979	118	89%
Set 10 k14 Hyperpipes	1023	74	93%

Continuing next page

	Successes	Failures	Average of Success Rates
Set 10 k14 Naive Bayes	868	229	79%
Set 10 k14 OneR	N/A	N/A	N/A
Set 10 k14 J48	941	156	86%
Set 10 k14 LWL	771	326	70%

Table B.4: List of document file extensions allowed through preprocessing whitelist

$$p	0	0	1	1sp	2	2
212	3	3	301	3d	3d	3d6
3df	3dg	3dz	4	4w7	4wt	602
a5r	a5w	a7p	a7r	aa	aad	ab65
abicollab	abs	abw	aca	acc	accdp	acp
acp	acp	acrypt	ada	adb	adc	ade
adn	ados	adp	adt	adv	adx	aep
aepx	af2	af3	afd	aff	afp	aft
agldei	aglsl	agp	agr	ahf	alb3	alb4
alb5	ald	ald5	alg	ali	ali	all
alt3	alt5	alt6	amsm	amst	amx	and
anl	ans	ans	ansr	anx	apa	apf
apf	apo	applocalize	apr	apr	apt	apw
asd	asn	asp	asp	asp	ast	asv
at2	at65	ath	aup	av	aw	aw
awp	aws	awt	aww	awwp	axg	axr

Continuing next page

b26	b27	b4s	b4u	bbl	bbprojectd	bc5
bcp	bdsproj	bdt2	bdt3	bean	bfx	bibtex
bil	bina	biz	biz	bizdocument	bk	bkg
bkr	bks	bld	blg	blg	blg	blt
bmm	bobo	boc	bok	boo	book	book
bookexport	?booktemplate	brh	bro	bsb	btd	btf
btw	btx	burn	burntheme	bwp	bxx	bzabw
c00	c2e	cap	cap	cap	cap	cap
cap	cap	cap	cap	cap	cbf	cbl
cbs	cbt	ccc	cch	cd2	cdc	cdc
cdc	cdd	cdd	cdf	cdk	cdl	cdmz
cdp	cds	cdt	cdt6	cdx	cdz	cer
cer	cfd	cfl	cfl	cfm	cfr	cgdc
ch4	che	chi	chm	cho	chp	chp
chs	cht	cht	cht	cht	cht	cht
cht	cht	cif	cipo	cit	cl4	clb
clbx	cld	clg	cml	cmp	cmp	cmr
cms	cmx	cnq	cns	cnt	cod	comicdoc
converterx	cov	cp	cpf	cpf	cpi	cpi
cpl	cpp	cpr	cpt	cpt	cptx	cpy
crf	crp	crwl	cs	csa	csd	csd
cse	csf	csk	csp	cst	ctd	ctk

Continuing next page

ctp	ctx	cty	cvj	cvl	cvr	cvt
cw3	cwk	cwks	cwwp	cxl	cxp	da
da11	daf	dal	dbc	dbi	dbi	dbm
dbp	dcf	dcf	dcs	dd	ddc	ddif
ddt	dfl	dft	dfv	dgpd	dgr	dgr
dgrh	dgs	dhe	dia	dic	dict	disco
dj	dk@p	dl	dnt	do	d?4d	doc
doc	doc	doc	doc	doc	doc	dochtml
docm	docmhtml	docx	docxml	dor	dot	dothtml
dotm	dotx	dox	dox	dox	dox	dp
dpd	dpe	dpg	dpgraph	dpo	dproj	dps
dpt	drd	drf	drg	drm	drmx	drt
dsf	dsn	dtf	dtp	dtp	dtp	dtr
dtr	dvb	dvi	dwz	dx	dxd	dxn
dxstudio	dzm	easmx	eb	ebh	ebkproj	ebs
ec4	ecg	edd	edm	edml	edn	edn
edoc	edrwx	edt	efp	efx	egt	ehp
emd	eml	emlx	emr	enex	enm	env
enx	enyd	epdf	epp	eprtx	ept	epub
es	esd	esp	ess	et	ete	eth
evo	evt	evt	evy	ewb	ewl	exc
exc	exp	ez	f96	fan	faq	far

Continuing next page

fax	fbd	fbl	fbok	fcs	fd2	fdb
fdf	fdm	fdr	fds	fdt	fdx	fee
ffdata	fff	ffs	fft	fft	fhz	fig
fil	fin	fire	flb	flg	flm	flo
flo	flp	flp	fls	flw	flw	fly
fly	fm	fmap	fmd	fmp	fmp3	fmt
fmt	fmt	fmt	fodp	fodt	fp	fpage
fpc	fpj	frg	frm	fsd	fsif	ftil
ftl	ftp	ftpl	ftr	ftr	fts	fts
fwk	fwrt	fx	fx2	fxd	fxr	gam
gca3	gca4	gca4base	gca4party	gcf	gcx	gda
gdc	gdf	gdoc	gen	gexf	gfc	gform
gif2	gks	gmk	gmp	gmx	gna	gnd
gno	gp1	gp3	gp4	gp5	gph	gpn
gpx	gra	grade	grf	grf	grf	grk
grv	grx	gs	gsa	gsc	gsp	gsp
gsw	gtable	gtd	gtp	gui	gwb	gwb
h2o	hcr	hcx	hda	hdc	hdt	hed
help	hfd	hft	hhp	hhp	hht	his
his	hlf	hlp	hlp	hlp	hlp	hm2
hm3	hmk	hmp	hmx	hmxp	hmxz	hnc
hnd	hot	hpd	hpd	hpj	hpo	hpt

Continuing next page

hqz	hsp	hst	hw3	hw3	hwp	hxc
hxs	hxv	hyp	hype	i3d	i3f	iaf
ibatemplate	ibcd	ic	icalevent	icaltodo	icodeproj	icst
idc	idml	idx	if	iff	ifo	igx
ila	ildoc	imf	imm	imp	imp	imr
ims	imsp	imv	inct	ind	ind	indb
indn	indt	infopathxml	ini	ink	inp	inrs
insx	inter	inx	iof	ipf	ipr	ipr
iqp	ish1	it	itp	its	iv-vrml	ivt
ivt	iw	iwp	iwprj	iwzip	ix	ix2
ixf	ixi	ixv	jnl	jnt	jpx	jrf
jsd	jsd	jtd	jtp	jtt	jtx	jw
jw	jwl	jwrp	kbd	kcl	kdc	kdd
key	keynote	kfl	kfm	kht	kid	kjv
kmp	knt	kpr	kpt	kwd	la	lab
latex	lax	lb	lbl	lch	ldf	ldf
let	lgc	lgf	lgf	lgpl	lic	lic
lix	ll	ll3	lma	lnt	loc	loc
lof	lof	logonxp	lp2	lpc	lpd	lrp
lsd	lsl	lsp	lst	lst	lst	lth
ltx	lwp	lwp	lyr	lyx	m!93	m11
m13	maca	mag	manu	map	markdn	mars

Continuing next page

maw	max	mbbk	mbd	mbox	mbp	mbx
mc	mcbn	mcc	mcd	mcr	mcs	mcsp
mcw	mdb	mdbhtml	mdf	mdhtml	mdk	mdr
me	me	meb	med	mell	mellel	met
mfa	mfg	mfo	mfp	mfp	mft	mhe
mhp	mht	mif	mindnode	mio	mjdoc	mla
mlj	mlp	mls	mm	mm	mmap	mmas
mmat	mmd	mmf	mml	mmo	mmp	mmpr
mmsw	mol	mon	mp	mp2	mpc	mpj
mpls	mpp	mpp	mpp_	mpr	mpt	mpv
mpw	mpwd	mpwr	mpx	mrf	ms	ms
msd	msdvd	mse	msf	msg	mso	mst
mst	msw	mswd	mswmm	mtp	mtp	mtx
mtx	mtx	mug	mvb	mvt	mvw	mw
mw	mwd	mwpd	mwpp	mwpr	mwt	mx2
mx3	nb	nb	nb	nbp	ncb	ncd
ncf	nct	ncw	ne3	nfo	ng	njx
?nmbtemplate	nml	not	not	not	note	note
np	npd	npf	npi	npl	npl	npp
nrp	ns	nst	nte	nvd	nxd	nx^d
nx__	oa2	oa3	oas	obd	obd	obr
obx	ocdc	oda	odc	odccubefile	odf	odif

Continuing next page

odm	odo	odp	ods	odt	odt#	ofl
ofm	ofm	ofn	oft	ogc	ohw	ole
ole2	olv	oml	omp	omp	one	oos
op2	opd	opd	opj	opj	opn	opt
opw	opx	opx	or3	osc	otc	otf
otg	oth	otl	otp	otp	ott	out
ova	ovd	ovs	owm	ows	oxps	oxt
p2bp	p2s	p3	p65	pac	pad	pag
pag	pages	pat	pb1	pbd	pbk	pbproj
pc	pc	pcb	pcr	pcr	pd	pdf
pdfxml	pdf_	pdf_tsid	pdi	pdl	pdp	pdp
pdt	pez	pfd	pfd	pfl	pfp	pgs
ph	phb	pj4	pj5	pjt	pkg	pkp
pl	plb	plf	plg	plp	pls	plx
ply	pm	pm3	pm4	pm5	pm6	pm?
pmd	pml	pmp	pmt	pmw	pmx	pod
pol	pot	pothtml	potx	ppd	ppf	ppg
ppj	ppl	ppnt	ppot	pps	ppsm	ppsx
ppt	ppt3	ppthtml	pptm	pptmhtml	pptv	pptx
pptxml	ppv	ppv	pr2	pr3	pr4	prc
prd	pre	pre	prel	prf	prn	prn
prnx	pro4	project	prproj	prs	prs	prs1

Continuing next page

prs2	prt	prt1	prt2	prv	prx	ps
ps2	psf	psf	psg	psmd	psn	psr
pss	psw	pt	pt3	pt4	pt5	pt6
ptg	ptm	pto	ptx	pub	pub	pub
pubf	pubhtml	pubmhtml	pve	pw	pwd	pwd
pwi	pwp	pwt	pwt	pwt	px	pxp
pxt	pxt	pzfx	pzt	qbl	qcd	qct
qdf	qhcp	qhp	qht	qhtm	qprj	qpt
qrc	qrf	qrt	qu2	qw	qwd	qwt
qxb	qxt	r0c	r0f	r0h	r0z	r3t
ra	rap	rav	rcl	rcp	rdf	rdf
rdf	rdl	rdlx	rec	ref	rels	rep
rep	rep	rep	rep	ret	rev	rf
rft	rgn	rit	rmd	rmd	rmr	ro
roff	rosa	rpc	rpl	rpmsg	rpn	rpt
rptr	rrd	rrpa	rs	rs	rsf	rt
rtf	rtfd	rvc	rvf	rw3	rxf	rzb
s6bn	s85	s8bn	sa5	sam	sam	sbk
sbp	sbz	sc	sc	scb	scd	scr
scriv	scrivx	sct	scw	scw	sd	sdbn
sdbz	sdc	sdd	sdd	sdf	sdg	sdi
sdl	sdm	sdp	sdp	sdv	sdw	se

Continuing next page

se	seek	sem	seo	sff	sff	sff
sfs	sgf	sgl	sgm	sgm	sgm	sgml
sgml	sgp	shb	shb	shf	shr	shs
shw	si	sid	sig	sig	sil	sim
sla	sld3	sld8	slds	sle	slf	slf
slf	slp	slt	sm	smf	smf	smf
smh	smm	smm	smp	smp	sms	snf
snf	snp	sod	soi	sox	sp4	spam
spd	spdf	spf	spf	spj	spk	spl
spo	spp	spp	spr	sps	sps	sql
ss4	ssc	ssiw	sskd	ssx	st	sta
stc	stc	std	std	stg	sti	sti
stl	stl	stm	stm	stp	stp	stw
stw	stx	stx	stx	stx	sty	sty
sty	su	sub	sum	svs	swd	swe
swe	swp	sws	sxg	sxg	sxi	sxi
sxm	sxm	sxml	sxw	sy3	t	t2k
t3001	t65	tab	tabula-doc	tah	tal	tbf
tcd	tch	tdoc	tds	tef	tex	texi
tg1	thr	tip	tip	tk	tld	tlt
tlx	tmb	tmb	tmd	tmd	tmd	tml
tmv	tns	top	topc	tp	tp	tp

Continuing next page

tp3	tpl	tpl	tpl	tpl	tpl	tpl
tpo	tpt	tpx	tr5	tre	tsm	tst
tst	tud	tun	tut	tv4	tvc	twbx
tww	txk	txm	txn	txt	txt	txt
txt	u98	udf	udt	ulys	uml	uof
uop	uot	updf	uxf	vac	vai	vap
vbd	vbp	vbproj	vcal	vcard	vce	vcg
vcp	vcproj	vcxproj	vdi	vdoc	vdproj	vdx
vfc	vh	vhd	vip	vm	vmc	vmm
vmr	vmx	vor	vsd	vsp	vst	vsw
vsx	vthought	vts	vtx	vup	vxml	w
w	w51	w60	w61	w6bn	w6w	w8bn
w8tn	wb	wbk	wbk	wcl	wcl	wcm
wcp	wd0	wd1	wd2	wdbn	wdcd	wdf
wdl	wdm	wdoc	?webtemplate	wgm	wht	whtt
wid	wis	wizhtml	wkb	wlf	wlp	wls
wmc	wor	word	word	wp	wp4	wp42
wp5	wp50	wp6	wp7	wp?	wpa	wpc2
wpd	wpd0	wpd1	wpd2	wpd3	wpf	wpf
wpf	wph	wpl	wpm	wpost	wpostx	wpr
wps	wps	wpt	wpw	wpw	wrd	wrf
wrg	wri	wrlk	wrt	ws	ws	ws

Continuing next page

ws	ws1	ws2	ws3	ws4	ws5	ws6
ws7	wsa	wsd	wsm	wsq	wsr	wss
wt0	wtbn	wtp	wts	wwcx	wwh	wwk
wws	wxmx	wxp	wzn	x40	x50	xa0
xap	xav	xbk	XBRL	xdoc	xdp	xdw
xe0	xej	xel	xfdf	xfdl	xft	xgmml
xhp	xlc	xlc3	xlc4	xlc_	xlr	xls
xlshtml	xlsmhtml	xlthtml	xlw	xmind	xmls	xmmas
xmmat	xms	xmt	xpf	xpf	xpr	xpr3
xprj	xps	xsc	xsf	xsn	xtg	xy4
xy4v	xy?	xzfx	yar	ybhtm	ymg	ywp
zif	zn	zoi	zpt	zrn		

Table B.5: List of programming source code file extensions allowed through preprocessing whitelist

11	19	2clk	3rf	4ge	4gl	4th
8	8xk	a	a	a2w	a2x	a51
a66	a80	a86	aas	abap	abc	abl
abs	abt	acgi	acm	acr	act	act
action	actionscript	actproj	actx	acu	ad	ad
ad2	ada	?adiumscripts	ads	adt	adx	aep
aex	agc	agi	agls	ago	ags	ahk
ahtml	aidl	akp	akt	alb	alg	alw
alx	aml	amos	amw	anm	ap	ap?

Continuing next page

apg	apl	aplt	app	applescript	aps	armx
aro	arq	art	artproj	ary	as	as3
as?	asax	asbx	asc	asc	asc	ascx
asf	ash	asi	asic	asm	asm	asmx
aso	aso	asp	asp	asp+	asproj	aspx
asr	ass	asx	asz	atl	atomsvc	atp
atp	au3	au?	aut	avs	awk	awl
axb	axd	axe	axs	b	b	b24
b2d	bas	bas	bat	bb	bbc	bbf
bcc	bcf	bcp	bdt	beam	bet	bgm
bhs	bi	bil	bin	bks	bli	bml
bml	bml	bmo	bms	boo	borland	box
bp	bpk	bpo	bpr	bps	bpt	brk
brml	brs	brt	brx	bs	bs2	bsc
bsc	bsh	bsh	bsm	bsv	?bufferedimage	bxb
bxl	bxp	bzs	c	c	c	c
c#	c++	c–	c–	c86	cal	cap
cap	car	cas	cb	cba	cbl	cbp
cbq	cbs	cc	cc	ccs	cd	cel
cfi	cfo	cfs	cg	cgi	cgi	cgvp
cgx	ch	chd	cl	cla	cla	class
clm	clp	cls	cls	clss	clu	clw

Continuing next page

clw	cma	cmake	cmd	cml	cmm	cmp
cms	cob	cod	cod	cod	cod	coffee
cola	common	con	config	configure	cos	coverage
?coveragexml	cp	cp	cp	cp?	cpb	cpp
cpr	cpy	cpy	cpz	cr	crd	cs
cs	cs	csattr	csb	csc	csc	csf
csgrad	csh	csh	csh	csh	cshtml	csm
csm	csm	csp	csp	csproj	css	csview
csx	ctl	ctp	cx	cxs	cxt	cxx
c__	d	d	d2j	d4	datasource	db2
db2tbl	db2tr	db2vw	dba	dbg	dbheader	dbml
dbo	dbp	dbpro	dbproj	dc	dcd	dcf
dcp	dct	dd	ddb	ddp	deb	def
def	def	def	def	def	def	defi
dep	depend	des	des	des	dev	devpak
dfb	dfd	dfm	dfm	dfn	dg	dgml
dht	dhtml	dia	dic	dif	dil	dkc
dlg	dlg	dmc	dml	dml	dml	dms
do	do	dob	docstates	dor	dot	dpd
dpj	dpk	dpk	dplt	dpq	dpr	dpr
dpr	dqy	drc	dro	ds	ds	dsa
dsb	dsd	dsl	dso	dsp	dsr	dsym

Continuing next page

dsym	dt	dtd	dto	dts	dtx	dvb
dwarf	dwp	dws	dwt	dwt	dxl	e
e	e	e	ebc	ebs	ebs	ebs
ebs2	ebuild	ebx	ec	ecore	ecorediag	edge
edml	egg	el	elc	enml	ens	epj
epl	epp	eps2	epsf	epsi	ept	eql
eqn	es	es	esp	ex	exc	exe
exp	exp	exu	exw	f	f	f40
f77	f90	f95	fasl	fcgi	fdml	fdt
ff	fgl	fil	flm	fmb	fmt	for
for	for	fpc	fpi	fpp	frbd	frj
frs	frt	fs	fsi	fsproj	fsproj	fsscript
fsx	ftn	fus	fwx	fxl	galaxy	gas
gbap	gbl	gc1	gc3	gch	generictest	gfe
gg	gitignore	gl	glade	gld	glf	glf
gls	gml	gml	gnt	goh	gp	gq
gs	gs	gsb	gss	gst	gsym	gus
gv	gyp	h	h++	h–	h16	h2o
h6h	h86	hal	has	hbx	hbz	hc
hcw	hh	hic	hkp	hks	hlsl	hms
hom	hp?	hpf	hpp	hrh	hs	hsc
hsdl	hsm	ht4	htc	htd	htm	htr

Continuing next page

hxa	hxml	hxp	hxx	hydra	h__	i
i	iap	iba	ic	ice	icl	icn
idb	idb	idc	ide	idl	idl	ifp
ig	ii	ijs	ik	il	il	ilk
image	iml	imp	inb	inc	inc	inc
inc	inf	ini	ino	inp	ins	ins
io	io	ipb	ipch	ipf	ipp	ipproj
ips	iqy	irc	irobo	is	isa	ism
iss	iss	isu	isym	ix	j	jacl
jad	jav	java	javajet	jbc	jcl	jcm
jdp	jks	jl	jlc	jomproj	jpage	jpd
js	js	js	js	jsa	jsb	jse
jsf	jsfl	jsh	jsm	json	jsp	jss
jsx	jsxinc	judo	kb	kcl	kdevprj	ked
kex	kix	kmdi	kml	kmt	komodo	kon
kpl	ksc	ksh	kst	kumac	l	l
l	l1i	lamp	lap	lasso	lay	lbi
lds	lds	less	lex	lex	lgt	lhs
li$	lib	lib_	licx	lisp	lit	ll
ll	lml	lmp	lmv	lng	lng	lng
lng	lng	lng	lnk	lnp	lnx	lo
loc	lol	lp	lpr	lpx	lrf	lrs

Continuing next page

ls1	lsp	lsp	lss	lst	lua	luca
lwa	lxk	lxsproj	lzco	m	m	m
m	m2	m2r	m3	m4	m4x	mac
mac	magik	mak	mak	mak	make	make
maki	mal	maml	map	mash	master	mat
max	mb	mbs	?mbtemmplate	mc	mc	mc
mcl	mcm	mcml	mcp	mcr	mcr	md
mdex	mdf	mdf	mdp	mdp	mdp	mec
mel	mem	mex	mfcribbon-ms	mfl	mg	mi
mingw	mingw32	mis	mix	mk	mke	ml
mli	mli	mln	mls	mlsxml	mlts	mm
mm	mmb	mmch	mmjs	mml	mnd	mo
moc	mod	mod	mod	moo	mp?	mpd
mpm	mpp	mpx	mqt	mrc	mrd	mrl
mrs	ms	ms	msc	mscr	msdev	msha
msil	msl	msl	msl	msm	mso	msp
mss	mss	mst	msvc	msym	mt	mtp
mtx	mtx	mv	mvc	mwp	mx	mxe
mxmf	myapp	mzp	nbin	nbk	ncb	ncx
nes	netboot	nlc	nml	nms	npi	nqc
nrs	nse	nsi	nt	nxc	o	obj
obj	obr	obs	ocb	odc	odh	odl

Continuing next page

odl	ods	ogl	ogr	ogs	ogx	oks
oplm	opt	opx	oqy	orc	osas	osax
osg	ow	owd	owl	ox	p	p
pag	pal	palm	param	pas	pas	pas
pas	pas	pb	pba	pbi	pbl	pbl
pbp	pbq	pbxbtree	pbxproj	pc	pcd	pch
pcm	pcs	pd	pdb	pdb	pdb	pde
pdl	pdl	pdl	pdo	pdp	pds	pem
perl	pf0	pf1	pf2	pf4	pf?	pfa
pfx	pgm	pgm	pgml	ph	ph	ph3
phl	php	php1	php2	php3	php4	php5
phps	phs	phtml	pjt	pjt	pjx	pkb
pkg	pkh	pl	pl	pl1	plc	plc
plex	pli	plm	pls	plx	plx	pm
pm	pmp	pnproj	pnpt	poc	policy	pom
pp	pp	ppa	ppam	ppo	prg	prg
prg	pri	pri	prl	prm	pro	pro
proto	prx	psc1	psd	psf	psl	psl
psl	psm1	psn	pspscript	psu	ptb	ptl
ptl	ptx	ptxml	pun	pvs	pwn	pxl
pxl	pxo	pxt	py	py	pyc	pyo
pyw	pyx	qcf	qdl	qlc	qml	qpr

Continuing next page

qrc	qry	qx	r	r	raf	rap
rapc	rb	rb	rb	rb	rbc	rbf
rbp	rbs	rbt	rbw	rbx	rc	rc2
rcc	rdf	rdf	rdoff	rdv	reb	res
res	resources	resx	rex	rexx	rfs	rfx
rgs	rguninst	rh	rip	rlz	rml	rng
rob	robo	robo	rpg	rpj	rptproj	rpy
rpyc	rqy	rrc	rrh	rsm	rsp	rss
rssc	rsym	rts	rul	run	rvb	rvt
rws	rxs	rxs	s	s	s2s	s43
s4e	sal	sar	sas	sas	sax	sb
sbi	sbl	sbr	sbs	sc	sc	sc
sca	scb	scb	scm	scm	scm	scp
scp	scpt	scptd	scr	scr	scr	scs
sct	sct	sct	scx	scz	sda	sdef
sdl	seman	sen	sfx	sh	si	sim
sim	simple	sit	sjava	sjc	sjs	skp
sl	sl	slf	sln	slt	sm	sm
sma	smd	sml	sml	smm	smw	smx
snippet	sno	sp?	spi	spk	spr	sps
spt	spt	spt	spx	sqb	sql	?sqldataprovider
sqljet	src	src	src	srp	srz	ss

Continuing next page

ss	ssc	ssc	ssc	ssc	ssh2	ssi
ssq	st	sti	stl	stm	sts	stx
sus	svc	svx	sw	swg	swt	sxs
sxt	sym	sym	sym	t	t	t
t	t2w	tab	tag	tal	tal	targets
tcl	tcl	tcsh	tds	tec	tem	template
texinfo	text	tgml	thtml	ti	tig	tik
til	tiprogram	tk	tla	tlc	tld	tlh
tlh	tli	tli	tmh	tokend	tpl	tpm
tps	tpt	tpx	tql	tql	tra	triple-s
trs	trt	tru	tsc	tsq	tst	ttl
tu	tur	turboc3	txc	txl	txml	txt
txx	udf	ufdl	ui	uit	uix	ulp
umlclass	unx	uvproj	v	v18	v4e	v4s
vad	vap	vb	vba	vbe	vbg	vbi
vbp	vbproj	vbs	vbw	vbx	vc1	vc15
vc2	vc4	vc5	vc6	vc7	vce	vcp
vcproj	vcwin32	vcxproj	vd	vddproj	vdp	vdproj
vgc	vi	vic	vim	vip	viw	vls
vmx	vpc	vpj	vps	vre	vrw	vsmacros
vspolicy	vssscc	vstemplate	vtm	vtml	vup	vx
vxml	w	w	waf	was	wax	wbc

Continuing next page

wbt	wch	wcm	wdl	wdx9	wfs	win
?win32manifest	wis	wix	wixout	wmc	wml	wml
wmlc	wmls	wmlsc	woa	wod	wowproj	wpj
wpk	wpm	wpm	ws	wsc	wsd	wsdd
wsdl	wsf	wsrc	wsym	wx	wxi	wxl
wxs	wxs	wzs	x	x	x	xaml
xap	xbap	xbc	xbd	xbl	xbn	xcl
xcodeproj	xcp	xdo	xds	xfm	xgl	xhtm
xib	xin	xjb	xl	xla	xlm	xlm3
xlm4	xlm_	xlv	xmap	xme	xml	xmljet
xms	xmta	xn	xoml	xpl	xr	xrc
xsc	xsc	xsc	xsd	xsl	xslt	xsql
xst	xtx	xtxt	xu	xui	xul	xwc
y	yab	yml2	yxx	z	zasm	zbi
zcls	zero	zfd	zfrm	zfs	zh_tw	zms
zpk	zpl	zsc	zsh	zsrc	zts	zws
˜1˜	˜df	˜pa				

Dataset	Number of Records
Set 1	567
Set 2	193
Set 3	124
Set 4	528
Set 5	1073
Set 6	1041
Set 7	973
Set 8	1153
Set 9	1046
Set 10	1097

Table B.6: Record counts across datasets

Bibliography

[1] The source for fire extensions information. Website, 2012. http://www.file-extensions.org/.

[2] Barman Badan. Knol: a gateway to encyclopedic article. http://badanbarmanknol.wordpress.com/article/knol-a-gateway-to-encyclopedic-article-3bexxvrdm2i7n-13/, March 2010.

[3] M. Baez, A. Birukou, F. Casati, and M. Marchese. Addressing information overload in the scientific community. *Internet Computing, IEEE*, 14(6):31 –38, nov.-dec. 2010.

[4] Liliana Cabral, John Domingue, Enrico Motta, Terry Payne, and Farshad Hakimpour. Approaches to semantic web services: An overview and comparisons. pages 225–239, 2004.

[5] Guanling Chen and David Kotz. A Survey of Context-Aware Mobile Computing Research. Technical report, Hanover, NH, USA, 2000.

[6] Harry Chen, Tim Finin, and Anupam Joshi. An ontology for context-aware pervasive computing environments. *Special Issue on Ontologies for Distributed Systems, Knowledge Engineering Review*, 18:197–207, 2003.

[7] Scott Deerwester, Susan T. Dumais, George W. Furnas, Thomas K. Landauer, and Richard Harshman. Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, 41:391–407, 1990.

[8] Jacob Eisenstein and Randall Davis. Visual and linguistic information in gesture classification. In *ACM SIGGRAPH 2007 courses*, SIGGRAPH '07, New York, NY, USA, 2007. ACM.

[9] L. Fisher. *How to Dunk a Doughnut: The Science Of Everyday Life*. Arcade Pub., 2003.

[10] Eibe Frank, Mark Hall, and Bernhard Pfahringer. Locally weighted naive bayes. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence*, pages 249–256. Morgan Kaufmann, 2003.

[11] Gregory Gay, Tim Menzies, Bojan Cukic, and Burak Turhan. How to build repeatable experiments. In *Proceedings of the 5th International Conference on Predictor Models in Software Engineering*, PROMISE '09, pages 15:1–15:9, New York, NY, USA, 2009. ACM.

[12] Paolo Giudici. *Applied Data Mining: Statistical Methods for Business and Industry (Statistics in Practice)*. Wiley, 2003.

[13] Thomas R. Gruber. Toward principles for the design of ontologies used for knowledge sharing. *Int. J. Hum.-Comput. Stud.*, 43(5-6):907–928, December 1995.

[14] Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H. Witten. The weka data mining software: an update. *SIGKDD Explor. Newsl.*, 11(1):10–18, November 2009.

[15] Robert C. Holte. Very simple classification rules perform well on most commonly used datasets. In *Machine Learning*, pages 63–91, 1993.

[16] Xin Jin, Yanzan Zhou, and Bamshad Mobasher. Web usage mining based on probabilistic latent semantic analysis. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '04, pages 197–205, New York, NY, USA, 2004. ACM.

[17] James Joyce. Bayeśtheorem. In Edward N. Zalta, editor, *The Stanford Encyclopedia of Philosophy*. Fall 2008 edition, 2008.

[18] M.W. Kadous, Mohammed Waleed Kadous, and Supervisor Claude Sammut. Temporal classification: Extending the classification paradigm to multivariate time series. Technical report, University of New South Wales, 2002.

[19] Marziah Karch. Annotum definition. Website, 2012. `http://google.about.com/od/experiment_graveyard/g/Annotum-Definition.htm/`.

[20] Bryan Lemon. *The effect of locality based learning on software defect prediction* . ProQuest, UMI Dissertation Publishing, 2011.

[21] Bryan Lemon and Daniel Sloan. Graphical knowledge advantage machine. 2012.

[22] Henry Lieberman. Letizia: an agent that assists web browsing. In *Proceedings of the 14th international joint conference on Artificial intelligence - Volume 1*, IJCAI'95, pages 924–929, San Francisco, CA, USA, 1995. Morgan Kaufmann Publishers Inc.

[23] Pamela J. Ludford, Dan Frankowski, Ken Reily, Kurt Wilms, and Loren Terveen. Because i carry my cell phone anyway: functional location-based reminder applications. In *Proceedings of the SIGCHI conference on Human Factors in computing systems*, CHI '06, pages 889–898, New York, NY, USA, 2006. ACM.

[24] J. B. MacQueen. Some methods for classification and analysis of multivariate observations. In L. M. Le Cam and J. Neyman, editors, *Proc. of the fifth Berkeley Symposium on Mathematical Statistics and Probability*, volume 1, pages 281–297. University of California Press, 1967.

[25] Sharon Bertsch McGrayne. *The Theory That Would Not Die: How Bayes' Rule Cracked the Enigma Code, Hunted Down Russian Submarines, and Emerged Triumphant from Two Centuries of Controversy*. Yale University Press, 2011.

[26] Hiroshi Motoda and Kenichi Yoshida. Machine learning techniques to make computers easier to use, 1998.

[27] Tapio Niemi, Marko Niinimäki, Jyrki Nummenmaa, and Peter Thanisch. Constructing an olap cube from distributed xml data. In *Proceedings of the 5th ACM international workshop on Data Warehousing and OLAP*, DOLAP '02, pages 22–27, New York, NY, USA, 2002. ACM.

[28] Martin L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, Inc., New York, NY, USA, 1st edition, 1994.

[29] J. Ross Quinlan. *C4.5: Programs for Machine Learning (Morgan Kaufmann Series in Machine Learning)*. Morgan Kaufmann, 1992.

[30] R. Reddy, L. Wang, S. Reddy, S. Devalapalli, G. Sasanka, S. Macha, S. Teja, R. Doppalapudi, J. Yu, and J. Yu. Vijjana: A pragmatic model for collaborative, self-organizing, domain centric knowledge networks. In *IKE*, pages 116–121, 2008.

[31] Irina Rish. An empirical study of the naive Bayes classifier. In *IJCAI-01 workshop on "Empirical Methods in AI"*.

[32] Chris Rorden. ezanova free statistical software. Website, 2012. `http://www.mccauslandcenter.sc.edu/mricro/ezanova/index.html`.

[33] Mehran Sahami, Susan Dumais, David Heckerman, and Eric Horvitz. A bayesian approach to filtering junk e-mail, 1998.

[34] Leo Sauermann, Gunnar Aastr, Malte Kiesel, Heiko Maus, Dominik Heim, Danish Nadeem, Benjamin Horak, and Andreas Dengel. A.: Semantic desktop 2.0: The gnowsis experience. In *International Semantic Web Conference. Volume 4273 of Lecture Notes in Computer Science*, pages 887–900. Springer, 2006.

[35] Leo Sauermann, Ansgar Bernardi, and Andreas Dengel. Overview and outlook on the semantic desktop. In *In Proc. of Semantic Desktop Workshop at the ISWC*, 2005.

[36] Elizabeth Shriver, Christopher Small, and Keith A. Smith. Why does file system prefetching work? In *Proceedings of the annual conference on USENIX Annual Technical Conference*, ATEC '99, pages 6–6, Berkeley, CA, USA, 1999. USENIX Association.

[37] Randy Franklin Smith. *The Windows Server 2003 Security Log Revealed*. Monterey Technology Group, Incorporated, USA, 2nd edition, 2007.

[38] Jaideep Srivastava, Robert Cooley, Mukund Deshpande, and Pang-Ning Tan. Web usage mining: discovery and applications of usage patterns from web data. *SIGKDD Explor. Newsl.*, 1(2):12–23, January 2000.

[39] Werner Vogels. File system usage in windows nt 4.0. *SIGOPS Oper. Syst. Rev.*, 33(5):93–109, December 1999.

[40] Luyi Wang, Ramana Reddy, Sumitra Reddy, and Asesh Das. A context centric model for building a knowledge advantage machine based on personal ontology patterns. In *SWWS*, pages 99–105, 2011.

[41] Geoffrey I. Webb, Janice R. Boughton, and Zhihai Wang. Not so naive bayes: Aggregating one-dependence estimators. *Machine Learning*, 58:5–24, 2005. 10.1007/s10994-005-4258-6.

[42] Ian H. Witten and Eibe Frank. *Data Mining, Second Edition: Practical Machine Learning Tools and Techniques, Second Edition (The Morgan Kaufmann Series in Data Management Systems)*. Morgan Kaufmann, 2005.

[43] Wolfgang Woerndl and Georg Groh. A social item filtering approach for a mobile semantic desktop application. In *AAAI Spring Symposium: Social Semantic Web: Where Web 2.0 Meets Web 3.0*, pages 82–83, 2009.

[44] Harry Zhang. The Optimality of Naive Bayes. In Valerie Barr and Zdravko Markov, editors, *FLAIRS Conference*. AAAI Press, 2004.

[45] Hai Zhuge. Autonomous semantic link networking model for the knowledge grid. *Concurrency and Computation: Practice and Experience*, 19(7):1065–1085, 2007.