SEMANTIC DOCUMENT CLUSTERING FOR CRIME INVESTIGATION

KABI G. DAGHIR

A THESIS

IN

THE CONCORDIA INSTITUTE FOR INFORMATION SYSTEMS ENGINEERING

PRESENTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE DEGREE OF MASTER OF APPLIED SCIENCE IN INFORMATION SYSTEMS SECURITY

CONCORDIA UNIVERSITY

MONTRÉAL, QUÉBEC, CANADA

SEPTEMBER 2011

CONCORDIA UNIVERSITY

School of Graduate Studies

This is to certify that the thesis prepared

By: **Kabi G. Daghir**

Entitled: **Semantic Document Clustering for Crime Investigation**

and submitted in partial fulfillment of the requirements for the degree of

Master of Applied Science in Information Systems Security

complies with the regulations of this University and meets the accepted standards with respect to originality and quality.

Signed by the final examining committee:

Dr. Lingyu Wang	Chair
Dr. Nizar Bouguila	Examiner
Dr. Govind Gopakumar	Examiner
	Examiner
Dr. Benjamin Fung	Supervisor

Approved by ______________________________

Chair of Department or Graduate Program Director

__________ September 12, 2011 ______ ____________________

Dr. Robin A. L. Drew, Dean

Faculty of Engineering and Computer Science

Abstract

Semantic Document Clustering for Crime Investigation

Kabi G. Daghir

Computers are increasingly used as tools to commit crimes such as unauthorized access (hacking), drug trafficking, and child pornography. The proliferation of crimes involving computers has created a demand for special forensic tools that allow investigators to look for evidence on a suspect's computer by analyzing communications and data on the computer's storage devices. Motivated by the forensic process at *Sûreté du Québec* (*SQ*), the Québec provincial police, we propose a new subject-based semantic document-clustering model that allows an investigator to cluster documents stored on a suspect's computer by grouping them into a set of overlapping clusters, each corresponding to a subject of interest initially defined by the investigator.

Acknowledgments

I would like to express my sincere gratitude to my supervisor, Dr. Benjamin C. M. Fung, for his resolute recommendations, guidance and assistance from the earliest to the final stages of my research. He has been a great source of encouragement throughout my study, and I am profoundly grateful to him.

My gratitude also goes to *Sûreté du Québec* (*SQ*) for providing us with real-life materials for experimentation.

Last, but definitely not least, I am endlessly grateful to all my family members for their unwavering support and motivation throughout this entire process.

“The best way to predict the future is to invent it.” - Alan Kay

To my wife *Bahaa* and my children *George* and *Gabriel*.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

The process of investigating digital devices for the purpose of generating digital evidence related to an incident under investigation is commonly referred to as *Digital Forensic Investigation* (*DFI*). According to Carrier et al. [CS04], digital evidence of an incident is any digital data that supports or refutes a hypothesis about the incident. The task of analyzing persistent documents found on a storage device of a suspect's computer is an essential part of the DFI process to gather credible and convincing evidence. However, this task is daunting due to the large number of documents usually stored on a hard disk. The continuously increasing size of storage devices makes the task even more difficult.

Existing digital forensic tools for analyzing a set of documents provide multiple levels of search techniques to answer questions and generate digital evidence related to the investigation. However, these techniques stop short of allowing the investigator to search for documents that belong to a certain subject he is interested in, or to group the document set based on a given subject.

In this thesis, we propose a new document clustering model that allows an investigator

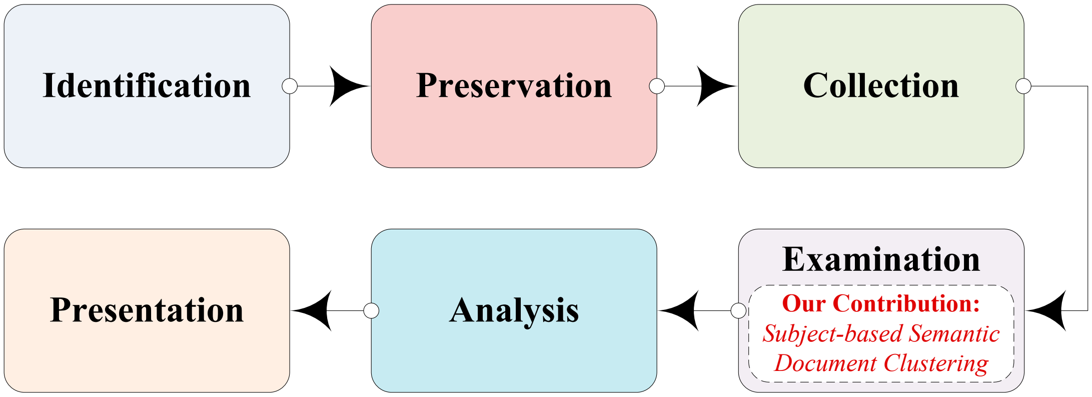

Figure 1: Digital Forensic Investigation (DFI) process as defined by DFRWS

to cluster all documents on a suspect's computer according to certain subjects he is interested in (e.g. hacking, child pornography). Once the documents are clustered in groups, each corresponding to a subject, the investigator can search for documents that belong to a certain subject.

There have been several attempts to define a digital forensic model that abstracts the forensic process from any specific technology, such as the DFRWS's model for digital forensic analysis [PC01], Lee's model of scientific crime scene investigation [LPM01], Casey's model for processing and examining digital evidence [Cas00], and Reith's model for digital forensic analysis [CRCG02]. Digital Forensics Research Workshop (DFRWS) is a pioneer in developing the forensic process. DFRWS defined Digital Forensic Science as a linear process:

> *The use of scientifically derived and proven methods toward the preservation, collection, validation, identification, analysis, interpretation, documentation and presentation of digital evidence derived from digital sources for the purpose of facilitating or furthering the reconstruction of events found to be criminal, or helping to anticipate unauthorized actions shown to be disruptive to planned operations.* [PC01]

Figure 1 illustrates the *Digital Forensic Investigation* (*DFI*) process as defined by DFRWS. After determining items, components, and data associated with the incident (Identification phase), the next step is to preserve the crime scene by stopping or preventing any activities that can damage digital information being collected (Preservation phase). Following that, the next step is collecting digital information that might be related to the incident, such as copying files or recording network traffic (Collection phase). Next, the investigator conducts an in-depth systematic search of evidence related to the incident being investigated, such as filtering, validation, and pattern matching techniques (Examination phase) [CRCG02]. The investigator then puts the evidence together and tries to develop theories regarding events that occurred on the suspect's computer (Analysis phase). Finally, the investigator summarizes the findings by explaining the reasons for each hypothesis that was formulated during the investigation (Presentation phase). In the examination phase, investigators often utilize certain forensic tools to help examine the collected files and perform an in-depth systematic search for pertinent evidence. However, there are three problems with today's computer forensic tools:

High-level Search. Since manual browsing is time consuming, investigators often rely on the automated search capability provided by either the operating system or existing DFI tools to conduct a search on the documents stored on the suspect's computer in order to identify related evidence. The main automated search techniques provided by current DFI tools include *keyword* search, *regular expression* search, *approximate matching* search, and *last modification date* search. Unfortunately, such techniques are applied directly against all of the stored documents without any advance knowledge about the topics discussed in each document. Hence, the results based on these search techniques generally suffer from a large number of false positives and false negatives.

Evidence-oriented Design. Existing DFI tools are designed for solving crimes committed against people, in which the evidence exists on a computer; they were not created to address cases where crimes took place on computers or against computers. In general, DFI techniques are designed to find evidence where the possession of evidence is the crime itself; it is easier to solve child pornography cases than computer hacking cases. [Gar10]

Limited Level of Integration. Most existing forensic tools are designed to work as stand-alone applications and provide limited capability for integration with each other or other custom tools or resources the digital forensic team might have already developed.

Our solution attempts to address these problems by answering the question of whether or not evidence for events defined by the investigator, such as hacking or child pornography, is present in the documents collected from the suspect's computer. The investigator initially defines the subjects (events) he is interested in investigating by providing a set of terms to describe each subject, such as vocabularies that are commonly used in the subject. We introduce a novel subject-based semantic document clustering algorithm that groups (clusters) all documents into a set of overlapping clusters, each corresponding to one unique subject.

The general intuition of our clustering approach is to generate a set of expansion vectors for each given subject using its initial subject definition. Each expansion vector consists of a set of weighted terms related to the subject, where each term is generated using WordNet [Mil95]. Our clustering algorithm also supports integration with the *Extended Synonym List* (*ESL*), a list of forensic-specific synonyms and related terms provided by the digital forensic investigation team at *Sûreté du Québec* (*SQ*) for the purpose of generating terms that are related to the subject. Once the expansion vectors for a subject are generated, they will be used with the initial subject definition to construct a vector of weighted term frequencies called *subject vector* such that the problem of measuring the similarity between a document and a subject is reduced to measuring the similarity between the document and the subject vector.

The generated clusters overlap for two reasons: First, the subjects are not necessarily independent of each other; for example, "hacking" and "cyberterrorism" subjects can be related as the cyberterrorist might be a hacker who broke into a government's website. Second, a document might discuss more than one subject (topic), and consequently it belongs to more than one cluster. In order to define a subject, an investigator has to provide a set of associated terms that are commonly used in the same context of the subject.

Example 1. Let us assume that the investigator is interested in investigating whether hacking events occurred on the suspect's computer. He might provide the following set of terms (along with their PoS tag) to define the subject:

⟨Hacking⟩hack:Verb,security breach:Noun,login:Noun,
Nmap:Noun,permission:Noun,exploit:Verb⟨/Hacking⟩ ■

During the clustering process, the files that do not belong to any of the subjects are grouped together in one generic cluster. The investigator can then browse the documents in this cluster manually or apply a standard clustering algorithm, such as bisecting k-means algorithm, to conduct further analysis on them.

Contribution. As illustrated in Figure 1, our contributions fall under the *examination* phase of the digital forensic investigation process. We summarize the major contributions of the thesis as follows:

- Subject Vector Space Models: We model our clustering solution by proposing *Subject Vector Space Model* (*SVSM*), a new model based on *Vector Space Model* (*VSM*) [Sal89] and *Topic-based Vector Space Model* (*TVSM*) [BK03]. In SVSM, each dimension represents a subject, where terms and documents are represented in the space according to their relations to all subjects. This allows a more realistic representation of the terms because terms inherently are not are orthogonal to each other. This representation also allows us to reduce the clustering problem of a document to the determination of the coordinate of this document on each subject vector. This is reflected in our proposed similarity function $Sim(d, s_i)$ that measures the similarity between any document $d \in D$ and subject $s_i \in S$, where D is a collection of documents and S is a set of subjects.

- Novel subject-based semantic document clustering algorithm: We introduce an efficient and scalable subject-based semantic document clustering algorithm that expands the term vector representing each subject using *WordNet* [Mil95]. *Word Sense Disambiguation* (*WSD*) algorithm [DS10] is integrated in the process to determine the appropriate sense (and accordingly, the synonym set) of a term in WordNet in the context of the initial terms that define a subject. The integration of WSD improves the precision of our clustering algorithm by reducing the polysemy affect [DDF$^+$90] [SJ88].

- Dynamically capturing suspects' terminologies: We make use of the document set to incrementally expand the subject vector by adding top frequent terms from the most similar documents to the subject. WSD is also integrated in this phase to determine the dominant sense of the term, which is the sense used the most in the several contexts in which the term appears.

- Experiments on real-life data: We conduct an extensive experimental study over two real-life data sets and examine the effectiveness of the algorithm according to subject vector length and document-score thresholds. We also demonstrate that our approach is highly scalable for large data sets.

1.2 Thesis Organization

The rest of the thesis is organized as follows. Chapter 2 reviews background knowledge and related work. Chapter 3 provides the formal definition of our clustering problem. Chapter 4 demonstrates the modeling of our solution. Chapter 5 introduces a three-stage semantic clustering algorithm. Comprehensive experimental results are presented in Chapter 6. Finally, we conclude the thesis in Chapter 7.

Chapter 2

Related Work and Background Knowledge

In this chapter, we summarize the various types of text mining techniques. We discuss major algorithms related to classification, semi-supervised clustering, and unsupervised clustering. We also discuss the data mining techniques developed/integrated with forensic investigations. We then discuss the concept of topic-based vector space model, and finally we introduce WordNet at the end of this chapter.

2.1 Text Classification

Text classification is the task of assigning a boolean value to each pair $(d_j, c_i) \in D \times \mathcal{C}$, where D is a set of documents and $\mathcal{C} = \{c_1, c_2, \ldots, c_{|\mathcal{C}|}\}$ is a set of predefined classes [Seb02]. If $(d_j, c_i) = true$ then document d_j is assigned to class c_i; otherwise, document d_j is not assigned to class c_i.

The assignment of documents to classes is based on their scores and can be achieved in two ways: *mono classification* and *multi classification*. Mono classification occurs when each document is assigned to one and only one class with the best score. On the other hand, multi classification occurs when each document is assigned to zero or multiple classes with scores above certain threshold.

Recent advancements in information retrieval and artificial intelligence fields makes text classification a hot topic. Text classification may be applied in many applications such as e-mail spam filtering, the categorization of newspaper articles into topics, mail routing, and the organization of web pages into hierarchical categories.

To classify a set of documents into a given set of classes, a profile for each class must be constructed. In general, there are two common approaches to achieve that: *manual rule-based approach* and *supervised machine learning-based approach* [CMSW92].

Classification systems based on manual constructions of class rules can be very accurate when the rules are written by experts, and classification criteria can be easily controlled when the number of rules are small. However, the higher the number of rules, the more difficult the maintenance of the rules becomes. Besides, the rules must be reconstructed every time a target domain changes.

On the other hand, supervised machine learning-based approach provides automatic learning techniques that are efficient and more suitable for rapidly evolving classifications. The basic process of supervised machine learning classification is as follows. First, prepare a set of training data, then create a classifier by applying a machine learning tool to the training data, and finally classify the new documents using the classifier.

Many algorithms have been developed for building classifiers. According to Kotsiantis [Kot07], the supervised machine learning classification techniques can be divided into four categories: *Logic-based Classification*, *Perceptron-based Classification*, *Statistical Learning Classification*, and *Support Vector Machines Classification*.

2.1.1 Logic-based Classification

In this section we will discuss the two most common groups of logical learning methods: *decision trees* and *rule-based* classifiers.

Decision Tree Classifiers. A decision tree classifier [Mit97] [Mur98] is a tree in which each internal node is labeled by a term, each branch going out of the node represents the weight of the term in a training document, and each leaf represents a class.

The classification of a document occurs by recursively checking the weight of the terms labeling the internal nodes and comparing that with the weight of the terms in the document vector, until a leaf node is reached. The label of this node (class) is then assigned to the document. Decision tree classifiers are typically built by a top-down "divide-and-conquer" approaches. Decision tree classifiers can be significantly complex due to the replication at node level. Markovitch et al. [MR02] presented the FICUS algorithm that implements complex features at nodes in order to avoid replications.

There are several standard packages for decision tree learning, such as ID3 [FB91], C4.5 [CS99] [Joa98] [LC94], and C5 [LJ98]. Text classification efforts based on experimental decision tree packages include Dumais et al. [DPHS98], and Weiss et al. [WAD+99].

Decision Rule Classifiers. It is possible to convert a decision tree into a set of rules by creating a separate rule for each path from the root to a leaf (class node) in the tree [Qui93]. Rules, however, can also be induced from the training data directly using a variety of rule-based algorithms [F9̈9]. Each class is represented by *disjunctive normal form* (DNF). Unlike decision trees, DNF decision rules are typically built by a bottom-up approaches. The goal is to generate a minimum set of rules while maintaining consistency with the training data. Some fundamental learning classifiers based on decision rules include RIPPER [Coh95] and CN2 [CN89].

2.1.2 Incremental Classification

Incremental (also called On-Line) classification methods start to build classifiers as soon as the first training document is processed, and then incrementally refine the classifier as more training documents are processed. This is useful if the training data is not completely available in its entirety from the start, or if the "meaning" of the classification classes might change over time. The two most popular incremental classifiers are *Perceptron* [Ros62] and *Winnow* [LW94].

Perceptron Classifier. An additive weight updating algorithm that was first applied in text classification by Schütze et al. [SHP95] and Wiener et al. [WPW95], and then by Dagan et al. [DKR97] and Ng et al. [NGL97]. A weight vector is kept for all terms such that the weights of active terms are updated whenever a mistake is made. The weights are updated in an additive fashion (additive learning); that is, a weight is promoted by adding a learning rate value α to it and demoted by subtracting α from it.

When the processing of all training data is completed, if the weight of a term is very low, that means the term has negatively contributed to the classification process so far, and may thus be discarded from the representation. Hence, the perceptron classifier can be perceived as a runtime dimensionality reduction technique [DKR97].

Positive Winnow Classifier. A multiplicative weight updating algorithm that is a variant of the perceptron classifier [DKR97] [KSB03]. It differs from perceptron in two ways. First, constant $\alpha_1 > 1$ used for promoting weights is different from constant $0 < \alpha_2 < 1$ which is used for weight demotion. Second, promotions and demotions are reached by multiplying, instead of adding, by α_1 and α_2, respectively.

Balanced Winnow. A variation of the positive winnow classifier algorithm where negative and positive weights are maintained for each term. This makes balanced winnow classifier more robust against variations in document length in the data set.

2.1.3 Statistical Learning Classification

Instead of classifying a document into one or more classes, classifiers based on statistical learning algorithms provide the probability for each document to belong in each class [Lew98]. In this section, we take a look at *naïve bayes* [Jen96] and *instance-based* classification algorithms, the most well known representatives of statistical learning algorithms.

Naïve Bayes Classifier. It is a simple probabilistic classifier based on applying Bayes' theorem [McG11] with strong (naïve) independence assumptions. It is composed of directed acyclic graphs with only one parent and several children. The parent represents the unobserved node, whereas the children represent the observed nodes. It assumes a strong assumption of independence among child nodes in the context of their parent [Kot07]. The major advantage of the naïve Bayes classifier is its short computational time for training. For this algorithm to work, it assumes non-zero probability for each feature. Since the feature space in text classification is very large, some unseen terms must therefore start with a reasonable value for their probability to exist in each class. Kohavi et al. [KBS97] proposed to assign 0.001 as a starting probability for such terms. There has been several attempts to overcome the independence assumption. Friedman et al. [FGG97] added extra edges to include some of the dependencies between the features, whereas Kononenko [Kon91] proposed a semi-naïve bayesian classifier in which attributes are partitioned into groups, and two features are considered conditionally independent if and only if they belong to different groups.

Instance-based Learning Classifiers. Such classifiers are also called *lazy learners*, since "they defer the decision on how to generalize beyond the training data until each new query instance is encountered" [Mit97]. The first application of instance-based classifiers to text categorization is due to Creecy et al. [CMSW92] and Masand et al. [MLW92]. Lopez de Mantaras and Armengol [LdMA98] presented a historical survey of logic and instance based learning classifiers. One of the most straightforward instance-based learning classifiers is the *k-nearest neighbor* [YC94], which determines whether a document belongs to a class by verifying if the k training documents most similar to the document are also in that class. Cohen and Hirsh [CH98] implemented an instance-based learning classifier by proposing a new WHIRL classification system which uses different similarity function from the one used in *k-nearest neighbor* classifier.

In addition to naïve bayes and instance-based learning classifiers, other statistical learning algorithms include *linear discriminant analysis* (*LDA*) [Fri89] and its variations [GHT07] [XBP09], commonly used in machine learning algorithms for the purpose of finding a linear combination of features which best characterize or separate two or more classes, and *maximum entropy* [Csi96], a statistical method for estimating probability distributions from data.

2.1.4 Support Vector Machine Classification

The *support vector machine* (SVM) method has been introduced in text classification by Joachims [Joa98] and subsequently extended by Drucker et al. [DWV99], Dumais et al. [DPHS98], and Klinkenberg and Joachims [KJ00]. It attempts to find, among all the hyperplanes of the feature space that separate the training documents into two classes, the optimum hyperplane that separates the documents into two classes by the widest possible margin. Documents that lie on the margin of the optimum separating hyperplane are known as support vectors. In the case of linearly separable data, the solution then is represented as a linear combination of only these support vectors.

SVM classifiers typically involves two classes. However, if more than two classes exist, then the SVM classifier uses a *one*-on-*one* approach. In a *one*-on-*one* approach, the algorithm creates a binary classification model for every possible combination of classes, and it uses a voting mechanism to identify the best class. If the voting mechanism identifies more than one class, the algorithm then selects the class that is the closest to the training document. Joachims [Joa98] argues that support vector machine classifiers do not require term selection, as they tend to be fairly robust to overfitting and can scale up to considerable dimensionalities. Besides, non-traditional data types like strings and trees can also be used as input, instead of feature vectors. Dumais et al. [DPHS98] introduced a novel algorithm for efficiently training support vector machine classifiers.

In summary, text classification approaches all depend on the availability of training data for the purpose of training their classifiers. Therefore, text classification fails to address our subject-based clustering problem because of the lack of training data in digital forensic investigations.

2.2 Semi-supervised Text Clustering

Unlike text classification algorithms which depend on labeled documents for training, semi-supervised text clustering algorithms provide a limited form of supervision by utilizing a small amount of available domain knowledge for the purpose of guiding or adjusting the clustering process.

Next, we discuss three common techniques in semi-supervised text clustering: *seed-based clustering*, *constraint-based clustering*, and *feedback-based clustering*.

2.2.1 Seed-based Clustering

Seed-based clustering approaches use supervised (labeled) data in order to help initialize cluster centroids. Proper initialization of clusters reduces the chances of the search space getting stuck in poor local optima, while simultaneously produces a clustering similar to the user-specified labels [BM03].

Basu et al. [BBM02] used seed-based clustering to initialize the k-means algorithm. They defined the seed set as a subset of the document set such that there exists k partitions (disjoint and non-empty) in the seed set, each of which corresponds to one and only one cluster. In k-means, instead of initializing the algorithm by randomly selecting k means, the seed set is used for this purpose by initializing the mean of each cluster with the mean of the corresponding partition. The seeds are only used for initialization, and do not get used in the following steps of the clustering algorithm. Basu et al. experimentally demonstrated that k-means benefits from the application of seed-based clustering.

2.2.2 Constraint-based Clustering

Constraint-based clustering in the field of information retrieval addresses the problem of partitioning document points into a specified number of clusters when limited supervision is provided in the form of pairwise constraints. Pairwise supervision is typically available as *must-link* and *cannot-link* constraints on document points: a *must-link* constraint indicates that both document points in the pair should be placed in the same cluster, while a *cannot-link* constraint indicates that two document points in the pair should belong to different clusters. If the constraints are *soft*, it means that the algorithm can violate them, but that would be undesirable.

Constraint-based methods use the provided supervision to guide the algorithm towards a data partitioning that avoids violating the constraints [DBE99] [WCRS01]. Pairwise relations naturally occur in various domains and applications. In information retrieval, for example, if an expert mention that two documents should not be in the same cluster, this is equivalent to *cannot-link* pairwise constraint. Basu et al. [BBM02] designed a model to store the pairwise constraints and used them to formulate the objective function in k-means algorithm. They also compared seeded approaches and constrained approaches based on the spherical k-means [DM01] and observed that the constrained version fares at least as well as the seeded one.

2.2.3 Feedback-based Clustering

Feedback-based clustering approaches execute a regular clustering process and then adjust the resulting clusters based on feedbacks from the user. There are many different types of feedback the users might provide to a semi-supervised clustering system. For example, the user might specify that the resulting clusters are too small or too big. The user might also provide certain constraints on individual document points.

Cohn et al. [CCM03] presented a new approach of feedback-based clustering by allowing the user to iteratively provide feedback to a clustering algorithm. The feedback is incorporated in the form of constraints which the clustering algorithm attempts to satisfy on future iterations. These constraints allow the user to guide the clustering process toward what he finds more useful. Zhong et al. [ZG03] [Zho06] presented a deterministic annealing extensions of the three semi-supervised clustering methods: seed-based clustering, constraint-based clustering and feedback-based clustering, and compares their performance on real text data sets. Experimental results show that feedback-based clustering is superior when the labeled data contain only a partial set of text categories.

Having reviewed the main approaches in semi-supervised text clustering, and since the initial subject definition vector is not in the format of labeled data or pairwise constraint and the subject information is very limited, semi-supervised text clustering cannot address the subject-based clustering problem presented in this thesis.

2.3 Unsupervised Text Clustering

Text (document) clustering is the process of partitioning unlabeled data set into a set of clusters, such that documents within a cluster are very similar, and documents in different clusters are very different. Clustering is considered part of the unsupervised learning approach because clustering is usually performed when no information is available concerning the membership of documents to predefined clusters.

Several taxonomies of clustering methods were suggested in [KR90], [JMF99] and [ELL09]. In the following sections, we describe the major categories of clustering methods that are best represented in the literature.

2.3.1 Partitional Clustering

Partitional clustering algorithms generate a set of non-hierarchical clusters. Because only one set of clusters is the output of the algorithm, the user is required to input the desired number of clusters (usually called k). The following are some of the well known partitional clustering algorithms.

k-means algorithm is a clustering algorithm that aims to partition a data set into k clusters in which each data point belongs to the cluster with the nearest mean. The mean is the average of all the points in the cluster; that is, its coordinates are the arithmetic mean for each dimension separately over all the points in the cluster [For65] [Mac67]. The algorithm of k-means can be presented as follows:

1. Initialize k clusters randomly.

2. Until converged: assign each data point to the cluster with the nearest mean, and then recompute the cluster means.

k-means algorithm discovers hard clusters since each data point belongs to one and only one cluster, and it tends to produce clusters of widely different sizes. Several methods were developed for cases where a mean (centroid) cannot be defined, namely the k-medoid [KR90] method and the k-modes [Hua97] method. k-means algorithm has some limitations if clusters are empty or if they differ in sizes and densities.

Fuzzy-C-Means algorithm is a variation of k-means algorithm that discovers soft clusters where a particular document point can belong to more than one cluster with certain probability. This method was developed by Dunn [Dun73] and improved by Bezdek [Bez81]. The algorithm of Fuzzy-C-Means is as follows:

1. Initialize k clusters randomly.

2. Until converged: compute the probability of a data point belong to a cluster for every (point,cluster) pair, and then recompute the cluster means using above probability membership values of points to clusters.

2.3.2 Hierarchical Clustering

Hierarchical clustering aims to obtain a hierarchy of clusters, called *dendrogram*, that shows how the clusters are related to each other. These methods proceed either by iteratively merging small clusters into larger ones (agglomerative) or by splitting large clusters (divisive). A partition of the data items can be obtained by cutting the *dendrogram* at a desired level.

Agglomerative clustering starts by treating each document as a singleton cluster and then successively merges (or agglomerates) pairs of clusters until all clusters have been merged into a single cluster that contains all documents. Bottom-up hierarchical clustering is therefore called hierarchical agglomerative clustering (HAC). The criteria for the merging of pairs of clusters are variants of the classical single-link [SS73], complete-link [Kin67] or minimum-variance [War63] criteria.

CURE [GRS98] is an agglomerative clustering algorithm which employs multiple representatives per cluster in order to obtain clusters of arbitrary shape while avoiding the problems of the single-link criterion. OPTICS [ABKS99] is another agglomerative clustering algorithm which does not build an explicit clustering of the collection of items, but rather an ordered representation of the data that reflects its clustering structure.

Divisive clustering is a variant of hierarchical clustering also called top-down clustering . We start at the top with all documents in one cluster. The cluster is split using a flat clustering algorithm, such as k-means algorithm. This procedure is applied recursively until each document is in its own singleton cluster. Divisive clustering is conceptually more complex than agglomerative clustering since we need a second, flat clustering algorithm as a “subroutine”; however, it is more efficient than the agglomerative approach if we do not generate a complete hierarchy all the way down to individual document leaves.

Bisecting k-means is a divisive clustering algorithm that generates a hierarchy of clusters by utilizing k-means algorithm to split and produce clusters with the highest similarity. The bisecting k-means algorithm starts with a single cluster of all the documents in the data set and works in the following order:

1. Select a cluster to split.

2. Find 2 sub-clusters using the basic k-means algorithm. (bisecting step)

3. Repeat step 2, the bisecting step, for a fixed number of times and take the split that produces the clustering with the highest overall similarity. (The similarity for each cluster is the average pairwise document similarity)

4. Repeat steps 1, 2 and 3 until the desired number of clusters is reached.

Unlike k-means algorithm, bisecting k-means algorithm tends to produce clusters of relatively uniform size.

Since all approaches in unsupervised text clustering do not accept any type of user input, none of them will be able to utilize and take advantage of the information included in the initial subject definition vectors in order to guide the clustering process. Therefore, unsupervised text clustering is not suitable to address the subject-based clustering problem presented in this thesis.

2.4 Forensic Data Mining

Data mining is a powerful tool that allows forensic investigators to explore large volume of data quickly and efficiently [FU02]. Traditional data mining techniques such as association analysis, classification and prediction, clustering, and outlier analysis are already being used by law enforcement to identify patterns in structured data [HK06].

Classification techniques find common properties among different crime entities and organize them into predefined classes. This technique has been used to identify the source of e-mail spamming based on the sender's linguistic patterns and structural features [dVACM01]. Classification is used to predict crime trends; however, it requires a predefined classification scheme, as well as reasonably complete training and testing data set because a high degree of missing data would limit prediction accuracy [CCX$^+$04].

Clustering techniques group data items into classes with similar characteristics to maximize or minimize intra-class similarity, for example, to identify suspects who conduct crimes in similar ways or distinguish among groups belonging to different criminal groups. *Link analysis* is a technique used to evaluate relationships between nodes identified in terms of network theory, where relationships may be identified among various types of nodes including organizations, people and transactions. Link analysis has been used for investigation of criminal activity, namely fraud detection, counterterrorism, and money laundering and other financial crimes [SGW$^+$95].

Chen et al. [CCX$^+$04] introduced a crime data mining framework that identifies relationships between data mining techniques applied in criminal and intelligence analysis.

The objective is to help investigators to use more effectively those techniques in order to identify trends and patterns, address problem areas, and even predict crimes.

As a conclusion, researchers have developed/applied various automated data mining techniques in the forensic investigation field; however, the focus has been more on analyzing and extracting information from existing databases of previous cases rather than analyzing new document sets retrieved from suspects' computers in order to extract evidences. We believe that none of the existing forensic data mining techniques are capable of solving our clustering problem by taking advantage of the information the investigator initially provides about each subject while taking in consideration that no training data is available.

2.5 Topic-based Vector Space Model

Kuropka et al. [BK03] proposed an algebraic model called *topic-based vector space model* (*TVSM*) for information retrieval. TVSM is a d-dimensional space where each dimension represents one fundamental topic, and all fundamental topics are independent from each other (orthogonal). Terms, documents, and queries are represented as vectors, each represented by its coordinates on all dimensions (topics). TVSM does not define the term-topic relation (length of each term vector and the inter-term vector angles). This is important in order to determine the similarity between a document and topic.

Enhanced topic-based vector space model (*eTVSM*) [PK07] attempted to define the term-topic relation by providing an algorithm that uses WordNet ontology as source of semantics. Our proposed *subject vector space model* (*SVSM*) on the other hand is also based on TVSM, where each dimension corresponds to an interested subject originally provided by the investigator. However, we propose a novel approach to define the subject vector for each dimension in SVSM, consequently allowing us to determine the term-subject relation of any term and subject in SVSM. In addition, we propose a novel similarity function that calculates the similarity (angle) between any document vector and subject vector in SVSM.

2.6 WordNet

Wordnet [Mil95], a lexical database for the English language, is utilized to establish semantic relations between terms during several phases of our clustering algorithm. WordNet's lexicon is divided into four major categories: nouns, verbs, adjectives, and adverbs. The basic unit of a WordNet lexicon is *synonym set* (*synset*), in which each synset includes a word, its synonyms, definition, and sometimes example. Any word is assumed to have a finite number of discrete meanings, where each meaning under one type of part of speech (PoS) is called a sense. Each sense of a word in WordNet is represented in a separate synset. WordNet supports two type of relations: semantic and lexical.

- *Semantic relation* defines a relationship between two synsets by relating all of the words in one of the synsets to all of the words in the other synset. *Hypernymy*, *hyponymy*, *meronymy*, and *troponymy* are some examples of sematic relations.

- *Lexical relation* defines a relationship between two particular words within two synsets of WordNet synsets. *Antonym* and *Synonymy* are two examples of lexical relations.

According to Leibniz[1], synonymy can be defined as follows: *"two expressions are synonymous if the substitution of one for the other never changes the truth value of a sentence in which the substitution is made."* However, since such global synonymy is rare, we use synonymy relative to a context: *"two expressions are synonymous in a linguistic context C if the substitution of one for the other in C does not alter the truth value."*

[1] http://plato.stanford.edu/entries/leibniz/

Some words are *monosemous* with only one meaning or sense. However, many words have multiple senses, so words can be either *homonyms* [2] or *polysemous* [3].

[2] Two senses of a word are considered *homonyms* if they have the same spelling but completely different meanings.

[3] A word is considered polysemous if all of its senses are various shades of the same basic meaning.

Chapter 3

Problem Definition

In this chapter, we formally define the research problem. First, we illustrate the user input to define each clustering subject in Section 3.1. Then, we explain the preprocessing steps of the document set in Section 3.3 and provide the formal definition of our clustering system in Section 3.4, followed by a problem statement.

3.1 Initial Subject Definition

Suppose an investigator of a digital forensic case is interested in clustering a collection of documents by interested subjects, denoted by S. For each subject $s_i \in S$, the investigator has to provide a set of terms, denoted by $s_i^0 = \{t_1, \ldots, t_{|s_i^0|}\}$, that describes the subject. The input terms, for example, can be the vocabularies that are commonly used in the subject.

For each input term $t \in s_i^0$, the investigator also needs to provide its part of speech PoS_t. This helps increase the accuracy of our clustering algorithm by avoiding homonymous word problems where more than one distinct meaning exists for the same word. Moreover, PoS_t should be either "verb" or "noun." We limited ourselves to only verbs and nouns because they are the most content-bearing types of words [LCH08].

3.2 Extended Synonym List (ESL)

The digital forensic investigation team at *Súreté du Québec* (*SQ*) maintains the *Extended Synonym List* (*ESL*), a list of forensic-related synonyms and related terms such as acronyms and slang terms that are commonly used by criminals and that do not usually exist in standard dictionaries. ESL is not an incident-specific list, but rather a shared resource that is accessible by all investigators and usable in any incident. Our clustering algorithm utilizes this tool in order to generate terms related to clustering subjects.

3.3 Preprocessing

We apply three common used preprocessing procedures of text mining, namely *stopwords removal*, *stemming*, and *tokenization*, on each document with the goal of reducing its dimensionality, noise, and computational complexity while avoiding a significant loss of information.

Stopwords removal: Words that do not convey any meaning as well as common words, such as 'a' and "the", are removed. We compiled a static list of stopwords to be used for this purpose.

Stemming: Words that have morphological forms are normalized to their canonical form. A stemming algorithm, forexample, reduces the words "playing", "played", "play", and "player" to the root word, "play". We employed the *Porter* stemming algorithm [Por97].

Tokenization: Given a sequence of characters in a document, the tokenization process separates the characters into tokens by using white spaces and punctuation marks as separators. For example, the string "George, Sam and Mike" will yield three tokens: "George", "Sam", "Mike".

After the preprocessing, the set of distinct terms in a given document set is denoted by $\mathbb{T}$. Next, we index all documents by their terms and compute the weight of the terms in each document:

Indexing. For each term $t \in \mathbb{T}$, we maintain a list of documents that contain t so this list can be efficiently retrieved.

Weight assignment. As we will see in Section 4.2, each document is represented by a vector whose coordinates are the weights of the document in each subject dimension. To achieve such representation, we first must determine the weight of each term in each document. Specifically, we must determine each weighted term frequency $\mho_{t,d} = tf(t,d) \times idf(t,D)$, where $tf(t,d)$ is the frequency of term t in document d and $idf(t,D)$ is the *inverse document frequency* of term t in the document set D:

$$idf(t,D) = 1 + log(\frac{|D|}{1 + Freq_{t,D}}) \tag{1}$$

where $|D|$ is the number of documents in the document set D, and $Freq_{t,D}$ is the number of documents in D that contain the term t.

3.4 Clustering System Definition

We can now define our clustering system as a tuple:

$$CL = [D, S, SVSM, Sim(d, s_i), \mathcal{C}]$$

where:

- $D = \{d_1, d_2, \ldots, d_{|D|}\}$ is a set of input documents to be clustered.
- $S = \{s_1, s_2, \ldots, s_n\}$ represents a set of n subjects, where each subject s_i is a set of weighted terms that are semantically generated based on the initial definition vector s_i^0.

- *SVSM* is a framework for representing terms, documents, and subjects as vectors, as well as defining the relationship between them.

- $Sim(d, s_i)$ is a similarity function that measures the similarity between a document $d \in D$ and a subject $s_i \in S$, and returns a real value that ranges between 0 and 1.

- $\mathcal{C} = \{\mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_n\}$ is a set of n overlapping output clusters, each of which is associated with one subject. In other words, each cluster $\mathcal{C}_i \in \mathcal{C}$ contains a set of retrieved documents for subject $s_i \in S$ given a similarity threshold τ:

$$\mathcal{C}_i = \{d \mid d \in D, Sim(d, s_i) \geq \tau\} : \tau \in [0, 1] \tag{2}$$

Problem Statement. Given a set of subjects S, the initial definition of each subject $s_i \in S$, a collection of documents D, and a similarity threshold τ, the problem of subject-based semantic document clustering is to group the documents in D into a set of overlapping clusters $\mathcal{C} = \{\mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_{|\mathcal{C}|}\}$ such that each cluster $\mathcal{C}_i$ is associated with one and only one subject $s_i \in S$ and $\mathcal{C}_i = \{d \mid d \in D, Sim(d, s_i) \geq \tau\}$.

Chapter 4

Solution Modeling

One major contribution of our proposed clustering method is to incrementally expand the subject vectors from the initial subject definitions. In this section, we first describe the concept of *expansion vector*, followed by the *Subject Vector Space Model* (*SVSM*) and the similarity function that measures the similarity between a document and subject vector in SVSM.

4.1 Subject Vector Generation

Our goal is to represent each given subject by a set of weighted terms that are most related to the subject. Therefore, for each subject $s_i \in S$, our clustering algorithm generates a set of expansion vectors $\{s_i^1, s_i^2, \ldots\ s_i^{p_i}\}$ such that each expansion vector s_i^r is a set of weighted terms that are semantically related to the terms in the previous expansion vector s_i^{r-1}, and the first expansion vector s_i^1 is semantically generated from the initial subject definition vector s_i^0 provided by the investigator for subject s_i. The number of generated expansion vectors varies from one subject to another. Note that all terms that belong to the same expansion vector are assigned the same weight value. However, the more expansion vectors we generate for a subject, the higher the chance of introducing noise from unrelated terms that degrades the accuracy of our clustering algorithm. Therefore, the weight of the generated terms decreases as we generate more expansion vectors.

Subject vector s_i can then be constructed by taking the union of the terms in the initial definition vector s_i^0 with all the terms in the expansion vectors of s_i:

$$s_i = s_i^0 \bigcup_{r=1}^{r=p_i} s_i^r \tag{3}$$

4.2 Subject Vector Space Model (SVSM)

We introduce the *Subject Vector Space Model* (*SVSM*), an algebraic model generated based on the *Vector Space Model* (*VSM*) [Sal89] and the *Topic-based Vector Space Model* (*TVSM*) [BK03]. SVSM is an n-dimensional vector space in which each dimension (axis) represents a subject $s_i \in S$. Similar to TVSM, all axis coordinates in SVSM are positive, and all axes are orthogonal to each other:

$$\forall\, \overrightarrow{s_i}, \overrightarrow{s_j} \in SVSM^n_{\geq 0} : \overrightarrow{s_i} \cdot \overrightarrow{s_j} = 0$$

where $(\overrightarrow{s_i} \cdot \overrightarrow{s_j})$ denotes the dot product between $\overrightarrow{s_i}$ and $\overrightarrow{s_j}$.

Figure 2 illustrates an example of the Subject Vector Space Model.

The representation of terms and documents in SVSM is analogous to their representation in TVSM.

Term Representation. Terms are represented in SVSM with respect to subject dimensions in SVSM, i.e., each term $t \in \mathbb{T}$ is represented as an n-dimensional vector $\overrightarrow{t}$ whose coordinates are the weights of the term in each dimension:

$$\overrightarrow{t} = \{\mho_{t,s_i} \mid i = 1, 2, \ldots, n\}$$

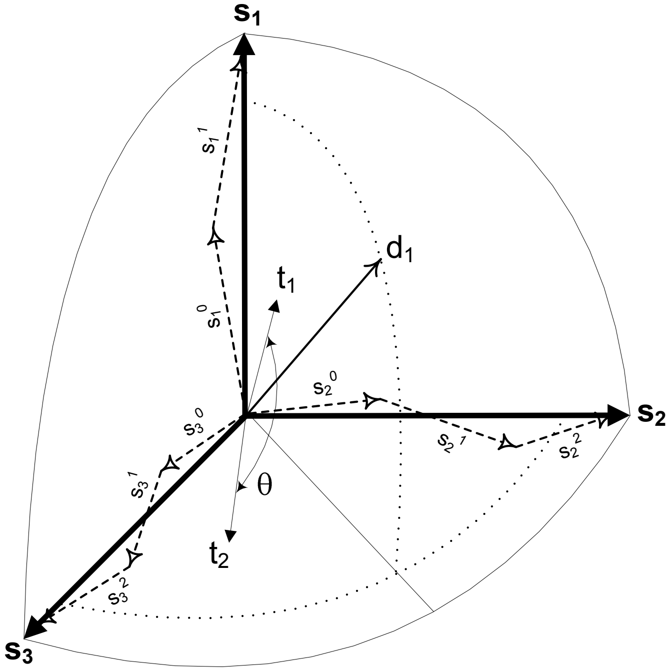

Figure 2: Subject Vector Space Model (SVSM)

where $\mho_{t,s_i}$ is the weight of term t in subject dimension s_i. If a term t exists in the set of weighted terms of subject s_i, then there will be a weight value assigned to t and $\mho_{t,s_i}$ will be equal to this value; otherwise, $\mho_{t,s_i} = 0$.

The norm (positive length) of a term vector $||\overrightarrow{t}||$ represents the global weight of that term:

$$||\overrightarrow{t}|| = \sqrt{\sum_{i=1}^{n} (\mho_{t,s_i})^2}$$

Since all axis coordinates are positive, it allows the similarity values between term vectors to be between zero and 1:

$$0 \le \theta(\overrightarrow{t_i}, \overrightarrow{t_j}) \le \pi/2 \;\Rightarrow\; \cos(\theta) \in [0, 1]$$

where θ is the angle between term vectors $\overrightarrow{t_i}$ and $\overrightarrow{t_j}$ in SVSM.

Document Representation. Each document $d \in D$ is represented as an n-dimensional vector $\overrightarrow{d}$ whose coordinate in each dimension $\overrightarrow{s_i}$ is the summation of the products of each term coordinate with the weight of a term in document d:

$$\overrightarrow{d} = \frac{1}{\psi_d} \sum_{t \in \mathbb{T}} \overrightarrow{t}.\mho_{t,d} \mid \psi_d = || \sum_{t \in \mathbb{T}} \overrightarrow{t}.\mho_{t,d} ||$$

where, $\mho_{t,d}$ is the weight of term t in document d. Also note that we divide all the coordinates of the document by its norm value ψ_d to normalize the document length to 1.

Since the similarity value between any two term vectors is between zero and one, the similarity value between any two document vectors is also between 0 and 1. This provides a more intuitive way to understand the similarity between documents, where 0 means the two document vectors are not similar at all and 1 means the two document vectors are either identical or their coordinates differ by a constant factor.

4.3 Document-Subject Similarity Function

Having determined the way to represent terms and documents in SVSM, we can now define a new similarity function that measures the similarity between a document and a subject and returns a real value that ranges between 0 and 1. We observe that the similarity between a document d and a subject s_i corresponds to the coordinate value of document vector $\overrightarrow{d}$ in dimension $\overrightarrow{s_i}$, i.e.,

$$Sim(d, s_i) = \frac{1}{\psi_d} \sum_{t \in \mathbb{T}} \mho_{t,s_i} \times \mho_{t,d} \tag{4}$$

where $\mho_{t,s_i}, \mho_{t,d}$ are the weight of term t in subject s_i and document d, respectively.

Chapter 5

Semantic Clustering Algorithm

In this chapter, we illustrate our approach for semantically clustering a document set D based on initial subject definitions $\{s_1^0, s_2^0, \ldots, s_n^0\}$. The objective is to utilize the initial subject definition vector of each subject $s_i \in S$ to semantically generate expansion vectors such that the final representation of the subject can be expressed as defined by (3).

After modeling the subjects, the algorithm clusters the documents by measuring the similarity between each document $d \in D$ and each subject $s_i \in S$ and generates a set of overlapping clusters $\mathcal{C}$ accordingly.

Algorithm 1 provides an overview of the subject vector generation process. The algorithm iterates through the following three steps to generate expansion vectors for each subject $s_i \in S$:

Step1- ESL Lookup (Lines 6-11): Generate an expansion vector by looking up synonyms/related words in the *Extended Synonym List* (*ESL*) of each input term of this step.

Step2- WordNet Synonyms (Lines 17-18): Utilize WordNet to determine the synsets of each input term of this step, and then generate an expansion vector using the synonym terms resulting from applying Lesk's [Les86] word sense disambiguation technique to determine the appropriate synonyms of each term.

Step3- Top Frequent Terms (Lines 24-30): Compute frequently used terms from the

Algorithm 1 Subject Vector Generation

```
Require: $|s_i^0| > 0$
$s_i \leftarrow s_i^0$
$r \leftarrow 1$
for each subject $s_i \in S$ do
    $\ddot{a} \leftarrow s_i^0$
    while $|s_i| \leq \delta$ do
        $s_i^r \leftarrow Lookup_{ESL}(\ddot{a})$
        if $r == 1$ then
            $\mho_{s_i^r} = 1$
        else
            $\mho_{s_i^r} = \lambda \times \mho_{s_i^{r-1}}$
        end if
        if $|s_i + s_i^r| > \delta$ then
            break
        else
            $s_i \leftarrow s_i + s_i^r$
            $r \leftarrow r + 1$
            $s_i^r = Synonym(wsd_{WordNet}(\ddot{a} + s_i^{r-1}))$
            $\mho_{s_i^r} = \lambda \times \mho_{s_i^{r-1}}$
            if $|s_i + s_i^r| > \delta$ then
                break
            else
                $s_i \leftarrow s_i + s_i^r$
                $r \leftarrow r + 1$
                $\ddot{q} \leftarrow \{\bigcup_{l=1}^{l=r-1} s_i^l\}$
                $D' \leftarrow \{d \mid d \in D, score_\epsilon(q, d)\}$
                $\hat{\mathcal{M}} \leftarrow tfidf(t \in D' \times d \in D')$
                $\mathit{tft}(\hat{P}) \leftarrow \{t \mid \sum_{d \in \hat{\mathcal{M}}} tf(t, d) \cdot idf(t, D') \geq \sigma\}$
                $wsd(t) \leftarrow \mathbb{S}_t \subset_{dominant} \bigcup_{x \in X} wsd^x(t) : t \in \mathit{tft}(\hat{P})$
                $s_i^r = \{t \mid t \in \mathit{tft}(\hat{P}), \sum_{t_c \in s_i^0} jcn(t, t_c) \geq \sigma'\}$
                $\mho_{s_i^r} = \lambda \times \mho_{s_i^{r-1}}$
                if $|s_i + s_i^r| > \delta$ then
                    break
                else
                    $s_i \leftarrow s_i + s_i^r$
                    $\ddot{a} \leftarrow s_i^r$
                    $r \leftarrow r + 1$
                end if
            end if
        end if
    end while
end for
return $S = \{s_1, s_2, \ldots, s_n\}$
```

top-ranked documents of the document set, and then generate an expansion vector by extracting top frequent terms (*tft*) using Jiang-Conrath's [JC97] relatedness distance measure. *Word sense disambiguation* (WSD) technique is utilized for part-of-speech (PoS) tagging and sense determination in a context to address the problem of homonyms and polysemous words.

Note that the weight of each term in any initial vector s_i^0 is equal to 1. However, as we start to construct each subject $s_i \in S$ by generating a set of expansion vectors, the weight of each term in any expansion vector s_i^{r+1} will be less or equal to the weight of any term in the previous expansion vector s_i^r, i.e.,

$$\mho_{t'} = \lambda \times \mho_t \; : \; t \in s_i^r \; and \, t' \in s_i^{r+1}$$

Based on an extensive number of experiments conducted, we observed that our clustering algorithm provides better results when $\lambda = 1$ in Step1- ESL Lookup. This is because introducing new terms using ESL imposes a minimal risk of adding noise because the list is provided by the investigator. We also observed that our clustering algorithm provides better results when $\lambda = 0.5$ in Step2- WordNet Synonyms and Step3- Top Frequent Terms. This is because using ontology terms or external source terms as additional features may introduce noise [HZL$^+$09].

Figure 3 illustrates the subject s_i vector generation process.

5.1 ESL Lookup (Lines 6-11)

The objective of this step is to generate an expansion vector s_i^r by looking up synonyms/related words in the *Extended Synonym List* (*ESL*) such that $r - 1$ expansion vectors have already been generated.

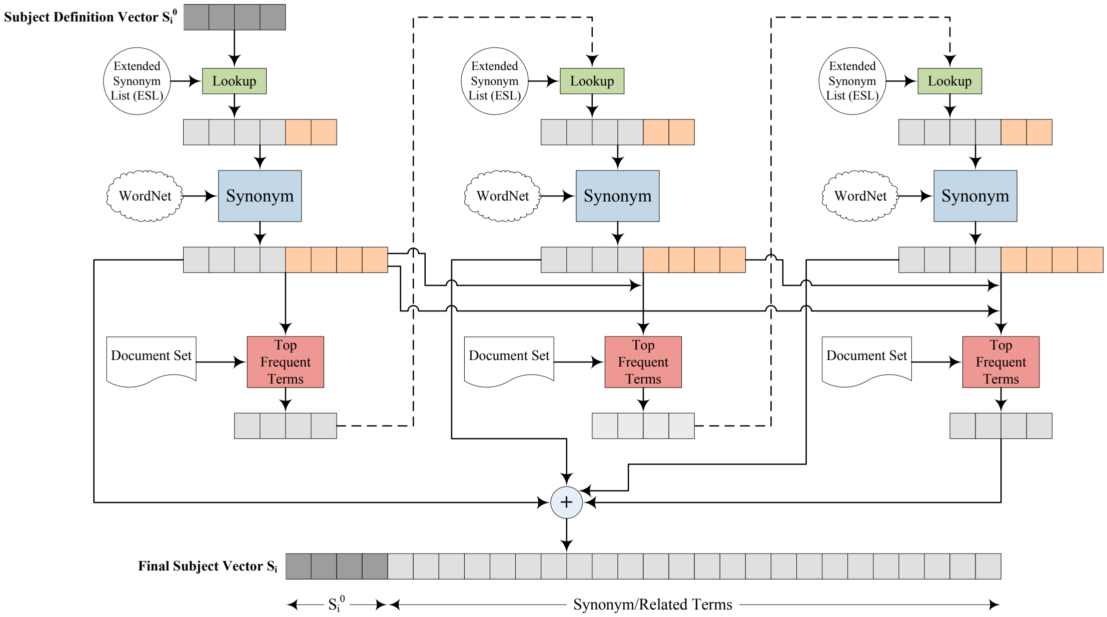

Figure 3: Subject s_i Vector Generation Process

The input in this step is a set of weighted terms denoted by $\ddot{a} = \{t_1, t_2, \ldots, t_{|\ddot{a}|}\}$. If this is the first expansion vector to be generated ($r = 1$), then $\ddot{a}$ is the initial subject definition vector s_i^0 for subject s_i; otherwise, $\ddot{a}$ is the expansion vector s_i^{r-1} for subject s_i.

For each $t \in \ddot{a}$, Line 6 scans the ESL. If $t \in ESL$, then any related term $t' \in ESL$ will be added to s_i^r such that t' will be assigned a weight equal to the weight of t.

5.2 WordNet Synonyms (Lines 17-18)

The next phase is to utilize WordNet to determine the synsets of each input term and to generate an expansion vector s_i^{r+1} using the synonym terms resulting from applying Lesk's word sense disambiguation technique [Les86] to determine the appropriate synonyms of each term.

The input for this step is a set of weighted terms denoted by $\ddot{b}$ that combines the terms from both $\ddot{a}$ and s_i^r from the previous step (ESL Lookup).

While generating s_i^{r+1}, it is important that we handle homonyms and polysemous words carefully by finding the best sense that represents the meaning of each term $t \in \ddot{b}$ in context. To achieve this goal, we utilize a *word sense disambiguation* (WSD) technique for part-of-speech (PoS) tagging and sense determination in a context.

Our lexical semantic expansion approach using WordNet involves three stages: First, we identify all the senses for each term $t \in \ddot{b}$ (Section 5.2.1), then use WSD algorithm to select the most appropriate sense for each term (Section 5.2.2), and finally generate the expansion vector s_i^{r+1} by assigning the synonym terms (Section 5.2.3). Figure 4 illustrates the lexical semantic expansion approach using WordNet.

5.2.1 Synset Repository Construction

Let $\ddot{b} = \{t_1, t_2, \dots, t_{|b|}\}$ be the set of input terms. We define a synonym function Syn() that takes a term t as input and returns the synsets of t that correspond to the senses matching the term's part of speech (PoS) from WordNet:

$$Syn(t) = \{\mathbb{S}_j | \mathbb{S}_j \in Synset_{WN}(t) \ \wedge \ PoS_{\mathbb{S}_j} = PoS_t\}$$

where $PoS_{\mathbb{S}_j}$ denotes the part of speech of all synonyms in synset $\mathbb{S}_j$. We observe that associating only the synsets of the senses with matching PoS for each term has a major impact on the WSD algorithm, as it helps improve the overall disambiguation accuracy while reducing the computational time of the algorithm due to the reduction in search space. However, if the term is not tagged (no PoS is assigned to it), then we associate the synsets of both verb and noun senses with that term:

$$Syn(t) = \{\mathbb{S}_j | \mathbb{S}_j \in Synset_{WN}(t), \ PoS_{\mathbb{S}_j} \in [Verb, Noun]\}$$

Note that $Syn(t) = \emptyset$ when term t cannot be recognized by WordNet. This case could

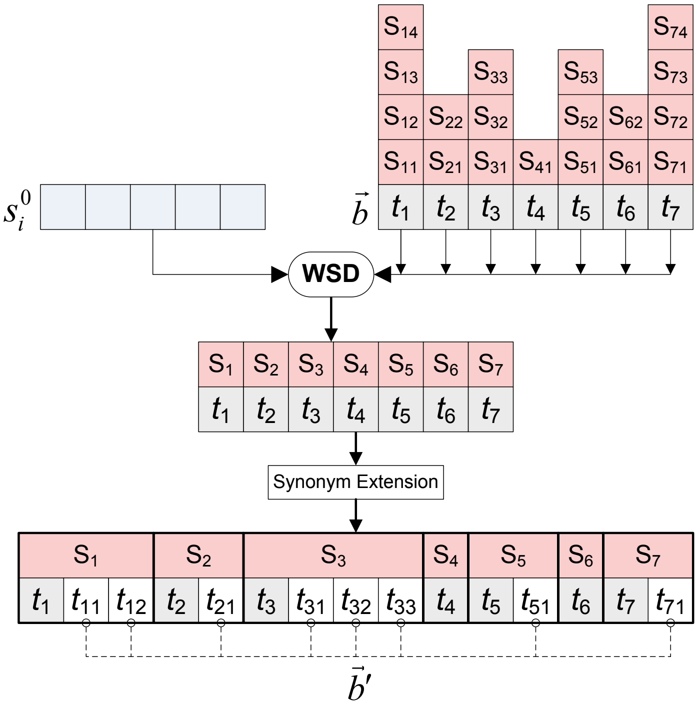

Figure 4: Lexical Semantic Expansion Using WordNet

occur since WordNet lexicon contains the majority (but not all) of the English words. This could also happen if the initial subject definition vector s_i^0 contains special terms, e.g., slang and special expressions, that are commonly used within criminal society but do not exist in standard English dictionaries.

5.2.2 Unique Synset Assignment

In the previous step, we associated a set of synsets with each term t. In this step, the objective is to identify the best fit synset in the context of subject s_i for each term $t \in \ddot{b}$. To achieve that, we use an adapted version of a word sense disambiguation method based on Lesk's algorithm [Les86]. Lesk's algorithm disambiguates a word by comparing the gloss of each of its senses to the glosses of every other word in a phrase. A word is assigned to the sense whose gloss shares the largest number of words in common with the glosses of the other words. The adapted version of Lesk's algorithm [DS10] will be applied to each term $t \in \ddot{b}$ to solve the ambiguity problem as follows:

1. Define the context around target term $t_l \in \ddot{b}$ to be all the terms in the initial subject definition vector s_i^0.

2. For each context term $t_k \in s_i^0$, list all the possible senses $\{\mathbb{S}_j \in Synset_{WN}(t_k) | PoS_{\mathbb{S}_j} = PoS_{t_k}\}$.

3. For the target term t_l, as well as each context term $t_k \in s_i^0$, list the following: its own WordNet gloss/definition, the concatenated glosses of all hypernym synsets, the concatenated glosses of all hyponym synsets, the concatenated glosses of all of the meronym synsets, and the concatenated glosses of all of the troponym synsets.

4. Measure the relatedness between each gloss of target term t_l with each gloss from each term $t_k \in s_i^0$ by searching for overlaps. The overall score of each sense of a target term is the sum of the scores for each gloss pair.

5. Once each combination has been scored, assign the synset of the corresponding sense with the highest score to the target term.

6. Repeat this process for every term $t_l \in \ddot{b}$ to determine the most appropriate sense for each term.

5.2.3 Expansion Using Synonyms

Having assigned a synonym set $\mathbb{S}_t$ to each term $t \in \ddot{b}$, the subject expansion vector s_i^{r+1} can now be generated by assigning the synonym terms of each term $t \in \ddot{b}$ to it:

$$\forall t \in \ddot{b} \; : \; s_i^{r+1} = \{t'_i \in Synonym(\mathbb{S}_t) | \, \mathbb{S}_t = wsd(t_l), \, t' \notin \ddot{b}\}$$

where $wsd(t_l)$ represents the best fit synset in the context of subject s_i for each term t_l. The terms in expansion vector s_i^{r+1} will be assigned a weight that equals half the weight of the terms in s_i^r.

5.3 Top Frequent Terms (Lines 24-30)

The goal in this step is to generate an expansion vector s_i^{r+2} for subject s_i based on the expansion vector s_i^{r+1} that was generated in the previous step. First, we determine the documents in D that are the most related to already generated expansion vectors in Section 5.3.1, then we compute *top frequent terms* (*tft*) in Section 5.3.2. For each term in *tft* we then apply the WSD algorithm to determine the context dominant sense in Section 5.3.3. Finally we apply the Jiang-Conrath similarity measure [JC97] to determine the most related terms to the initial subject definition s_i^0 in Section 5.3.4.

5.3.1 Compute Top Documents

The input for this section is a set of terms $\ddot{q}$ that represents all the system-generated subject vectors for subject $s_i \in S$ that have already been created:

$$\ddot{q} = \{t \mid t \in \bigcup_{l=1}^{l=r+1} s_i^l\}$$

Our algorithm utilizes $\ddot{q}$ as a query vector. Hence, it computes the score between each document vector $d \in D$ and $\ddot{q}$ by computing the dot-product between each pair of vectors as follows:

$$score(\ddot{q}, d) = \sum_{t \in \ddot{q}} tf(t, d) \cdot idf(t, D) \tag{5}$$

The score of all documents will then be normalized to be a real number between 0 and

1. Given a threshold $\epsilon \in \mathbb{N}^+$, we consider the top ϵ of the documents with the highest score as the most related set of documents denoted by D'.

5.3.2 Compute Top Frequent Terms

To determine the top frequent terms in D', we first build a term-document matrix $\hat{\mathcal{M}}$ in which each row corresponds to a document $d \in D'$ and each column corresponds to a term $t \in D'$. The entries in the matrix are the *term frequency−inverse document frequency* (*tf-idf*) [SB88] of each term in each document.

Based on matrix $\hat{\mathcal{M}}$, we then determine the set of top frequent terms using the function (*tft*). Let $\hat{\mathcal{M}}$ be a term-document matrix where each entry corresponds to a term frequency−inverse document frequency (tf/idf) of a term $t \in D'$ in a document $d \in D'$, and let σ be a minimum support threshold. We define the function *top frequent terms* (*tft*) of matrix $\hat{\mathcal{M}}$ as follows:

$$\mathit{tft}(\hat{\mathcal{M}}) = \{t \in \hat{\mathcal{M}} \mid \sum_{d \in \hat{\mathcal{M}}} tf(t, d) \times idf(t, D') \geq \sigma\}$$

5.3.3 Word Sense Disambiguation

Even though we have extracted the set of frequent terms $\mathit{tft}(\hat{\mathcal{M}})$ from the document set D', it is not clear – for each term – which sense was used the most in the contexts where the term appeared. We call such sense a *dominant sense*, and our goal here is to determine the dominant sense for each term $t \in \mathit{tft}(\hat{\mathcal{M}})$.

As previously, we use the adapted version of the word sense disambiguation method based on Lesk's algorithm [Les86]. However, the main difference in this case is that each term exists in multiple contexts X. We define each context $x \in X$ of term t as a frame of e terms that appear to the left and right of the term in each of its occurrences.

Since the terms are not tagged, we associate with each term (in each context) the synsets of both its verb and noun senses:

$$Syn(t) = \{\mathbb{S}_j | \mathbb{S}_j \in Synset_{WN}(t),\ PoS_{\mathbb{S}_j} \in [Verb, Noun]\}$$

We first determine the most appropriate sense for a term in each context $x \in X$:

$$wsd^x(t) = \mathbb{S}_t \mid \mathbb{S}_t \in Syn^x(t)$$

and then we determine the sense to be assigned to the term by finding the dominant one among all senses in all contexts:

$$wsd(t) = S_t \mid S_t {\in}_{dominant} \{ \bigcup_{x \in X} wsd^x(t) \}$$

5.3.4 Relatedness Distance Measure

The set of terms *tft*$(\hat{\mathcal{M}})$ we extracted from the document set to capture the suspects' terminologies might be unrelated (or weakly related) to the initial subject definition. To reduce noise and avoid a reduction in both the recall and precision of our clustering algorithm, we use a similarity measure to determine the relatedness of each term $t \in$ *tft*$(\hat{\mathcal{M}})$ to the initial subject definition vector s_i^0. The Jiang-Conrath similarity measure (*jcn*) [JC97] is used for this purpose. According to Pedersen et al. [PPM04], it enriches the information content of the least common subsumer of two concepts with the sum of the information content of the individual concepts by subtracting the information content of the least common subsumer from this sum, and then takes the inverse to convert it from a distance to a similarity measure.

Using *jcn*, we compute the distance between each term $t \in$ *tft*$(\hat{\mathcal{M}})$ and all terms in s_i^0. The terms that meet a certain threshold σ' will be used to construct the expansion vector

s_i^{r+2}:

$$s_i^{r+2} = \{t \in \mathit{tft}(\hat{\mathcal{M}}) \mid \sum_{t_c \in s_i^0} jcn(t, t_c) \geq \sigma'\}$$

The expansion vector s_i^{r+2} will be assigned a weight (real value between 0 and 1) that equals half the weight of the previous expansion vector for the same subject, i.e.,

$$\mho_{s_i^{r+2}} = 0.5 \times \mho_{s_i^{r+1}}$$

and the weight of each term $t \in s_i^{r+2}$ is equal to the weight of the expansion vector itself $\mho_{s_i^{r+2}}$. Once s_i^{r+2} has been computed, it will then be used to start a new iteration of subject vector expansion by starting to look up all the terms in s_i^{r+2} in the Extended Synonym List (ESL) in Section 5.1.

Once the generation of expansion vectors for all subjects $s_i \in S$ is completed, our algorithm starts the clustering process by applying the similarity function $Sim(d, s_i)$ between each document $d \in D$ and each subject $s_i \in S$ and generates a set of overlapping clusters $\mathcal{C}$ accordingly. Algorithm 2 provides an overview of the clustering process used to assign each document to one or more clusters.

Algorithm 2 Document Clustering

```
Require: $|s_i| < \delta$
for each subject $s_i \in S$ do
    for each document $d \in D$ do
        if $Sim(d, s_i) > \tau$ then
            $\mathcal{C}_i \leftarrow d$
        end if
    end for
end for
return $\mathcal{C} = \{\mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_n\}$
```

Chapter 6

Experimental Evaluation

Based on our proposed clustering solution, we have developed a tool called *Cyber Forensic Search Engine* (*CFSE*) for the Súreté du Québec (SQ) as a proof of concept. CFSE is a Java-based application that is composed of three components:

Indexing Engine. This engine parses documents on the suspect's computer, analyzes the documents by applying the preprocessing steps, namely tokenization, stemming, stop word removal, and text normalization, and then indexes the documents. We used Apache Tika [1] and Lucene [2] to parse, preprocess, and index the documents.

Clustering Engine. This engine groups all the documents into a set of overlapping clusters according to the algorithm proposed in this thesis, in which each cluster is associated with one and only one pre-defined subject, such that the similarity between a document and its assigned subject is maximized.

Search Engine. This engine allows an investigator to search the resulting clusters and retrieve relevant documents according to a given search query and a specific subject.

The objective of the experiments is to evaluate the performance of our proposed subject-based semantic document clustering algorithm implemented in the clustering engine in

[1] Apache Tika. http://tika.apache.org/

[2] Apache Lucene. http://lucene.apache.org/

terms of accuracy, efficiency, and scalability.

6.1 Data Sets

We use two data sets in our experiment: *Classic3* and *Forensic*. The pre-classification of each document will be used to measure the accuracy of the clustering results; however, during the clustering process, this information will be hidden.

Below is a brief summarization of each data set's characteristics:

- *Classic3* is a benchmark data set used in text mining[3]. It consists of 3893 documents from 3 disjoint classes: 1400 aeronautical-system documents (CRAN), 1033 medical documents (MED), and 1460 information-retrieval documents (CISI).

- *Forensic* is a data set we collected for validating our clustering algorithm against a set of crime-related documents. This data set consists of 90 documents from 3 different classes: 30 drug-related documents, 30 hacking-related documents, and 30 sexual assault-related documents.

6.2 Evaluation Method

We use *F-measure* [LA99] to measure the accuracy of the clustering solution produced by our method.

Let $\mathcal{C} = \{\mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_n\}$ be the set of clusters generated by our system against a document set D. Let $\mathcal{K} = \{\mathcal{K}_1, \mathcal{K}_2, \ldots, \mathcal{K}_l\}$ be the natural classes of the document set D. We compute the $Precision$ and $Recall$ of cluster $\mathcal{C}_j \in \mathcal{C}$ with respect to class $\mathcal{K}_i \in \mathcal{K}$ as follows:

$$Precision(\mathcal{K}_i, \mathcal{C}_j) = \frac{|\mathcal{K}_i \bigcap \mathcal{C}_j|}{|\mathcal{C}_j|} \tag{6}$$

[3] Classic3. ftp://ftp.cs.cornell.edu/pub/smart/

$$Recall(\mathcal{K}_i, \mathcal{C}_j) = \frac{|\mathcal{K}_i \bigcap \mathcal{C}_j|}{|\mathcal{K}_i|} \tag{7}$$

where $|\mathcal{K}_i|$, $|\mathcal{C}_j|$, and $|\mathcal{K}_i \bigcap \mathcal{C}_j|$ denote the number of documents in class $\mathcal{K}_i$, in cluster $\mathcal{C}_j$, and in both $\mathcal{K}_i$ and $\mathcal{C}_j$ respectively.

Based on the above, the generic version of F-measure (F_β) can be presented as follows:

$$F_\beta = \frac{(\beta^2 + 1) \times Precision \times Recall}{\beta^2 \times Precision + Recall} \tag{8}$$

where $\beta \in \mathbb{R}^+$ is a balancing parameter between $Precision$ and $Recall$. To make F-measure equivalent to the harmonic mean of $Precision$ and $Recall$, we set $\beta = 1$ and end up with a balanced version of F-measure called F_1 measure. We use F_1 in our experiments to compute the accuracy of cluster $\mathcal{C}_j$ with respect to class $\mathcal{K}_i$ as follows:

$$F_1(\mathcal{K}_i, \mathcal{C}_j) = \frac{2 \times Precision(\mathcal{K}_i, \mathcal{C}_j) \times Recall(\mathcal{K}_i, \mathcal{C}_j)}{Precision(\mathcal{K}_i, \mathcal{C}_j) + Recall(\mathcal{K}_i, \mathcal{C}_j)} \in [0, 1] \tag{9}$$

where F1 score reaches its best value at 1 and worst score at 0.

6.3 Experimental Results

In this section, we evaluate our subject-based semantic document clustering algorithm in terms of accuracy, as well as efficiency and scalability. All experiments were conducted on an Intel Core2 Quad E6650 3GHz PC with 4GB RAM.

6.3.1 Accuracy

Our clustering algorithm allows for two user-specified thresholds δ and τ, where δ is the maximum length (maximum number of terms) threshold of a subject vector and τ is the minimum similarity threshold for all clusters. A document $d \in D$ is added to cluster $\mathcal{C}_i$ if its normalized score returned by the similarity function is larger than τ (2).

Figure 5 depicts the F-measure values (accuracy) of the clustering algorithm with respect to δ and τ. We set δ to three different values relative to the average document length avg_dl in the data set: $0.25 * avg\_dl$, $0.50 * avg\_dl$, and avg_dl, whereas τ was set to a range of values between 0.05 and 0.5. We observe that the F-measure value spans from 0.61 to 0.72 when the minimum similarity threshold τ increases from 0.05 to 0.5. Assigning a value between 0.1 and 0.175 to τ provides high accuracy for all three different values of δ. Based on the extensive number of experiments conducted, we observed that our clustering algorithm provides a high accuracy value when $\tau \in [0.08, 0.2]$ and δ = avg_dl.

We also observe that the change in δ value affects the accuracy of the algorithm in a minimal way. We argue that the integration of *WSD* technique in step2 and step3 of our subject vector generation algorithm helps reduce the noise and consequently reduces the sensitivity of our clustering algorithm to the input parameter δ. We also observe that when the maximum length of a subject vector is equal to the average document length in the document set (δ = avg_dl), the algorithm in most cases provides higher F-measure values.

6.3.2 Efficiency and Scalability

One major contribution of our work is the development of an efficient and scalable algorithm for semantic document clustering that expands the term vector representing each subject using WordNet. According to Algorithm 1, the runtime complexity of our approach is dominated by the maximum subject vector length δ and the data set size; therefore, we study the runtime under different subject vector lengths and different data set sizes.

Figure 7 presents the runtime on the two data sets *Forensic* and *Classic3*, with respect to δ that ranges between 0.25*avg_dl and 1.25*avg_dl. We observed that the runtime scales linearly with respect to the subject vector length under both data sets. We also observed that regardless of the size of the data set (Classic3 is 43 times larger than Forensic), the

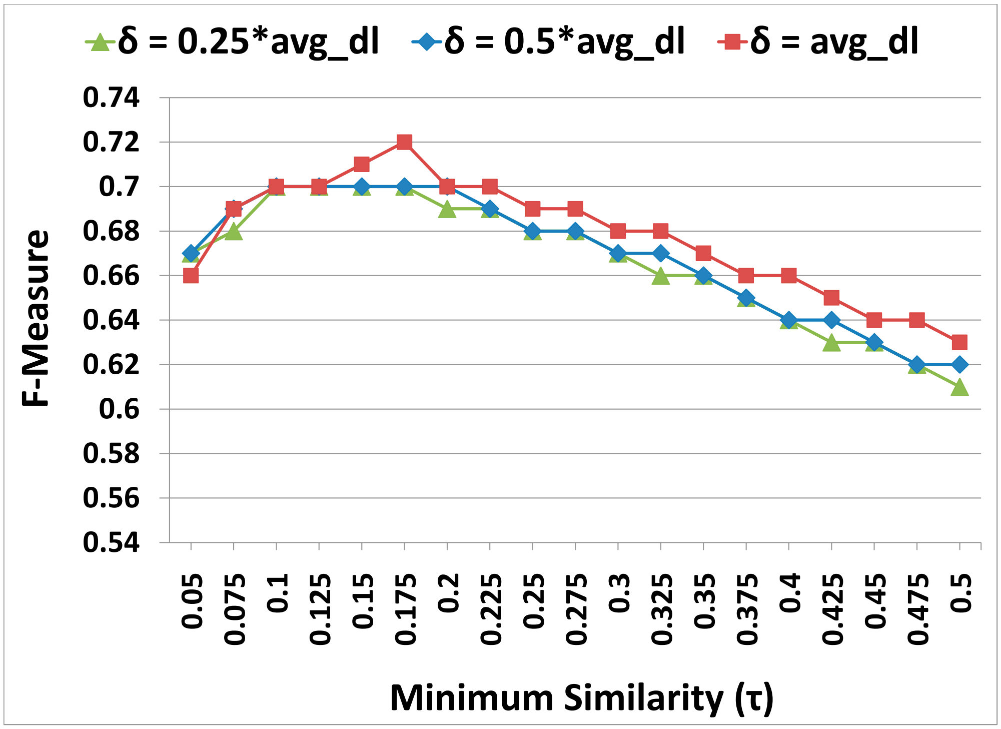

Figure 5: Sensitivity to cluster's minimum similarity (τ) in relation to the maximum subject vector length (δ)

gradient (slop) of each data set's runtime remains the same.

We choose Classic3, the larger data set with 3893 files, to examine the runtime with different data set sizes. The files in the data set are duplicated so scalability can be measured starting from 10,000 documents, going up to 100,000 documents. To ensure a balanced duplication of the data set, we define the *scaleup factor* α as follows:

$$\alpha = \lfloor \frac{Target\#ofdocuments}{|D|} \rfloor \tag{10}$$

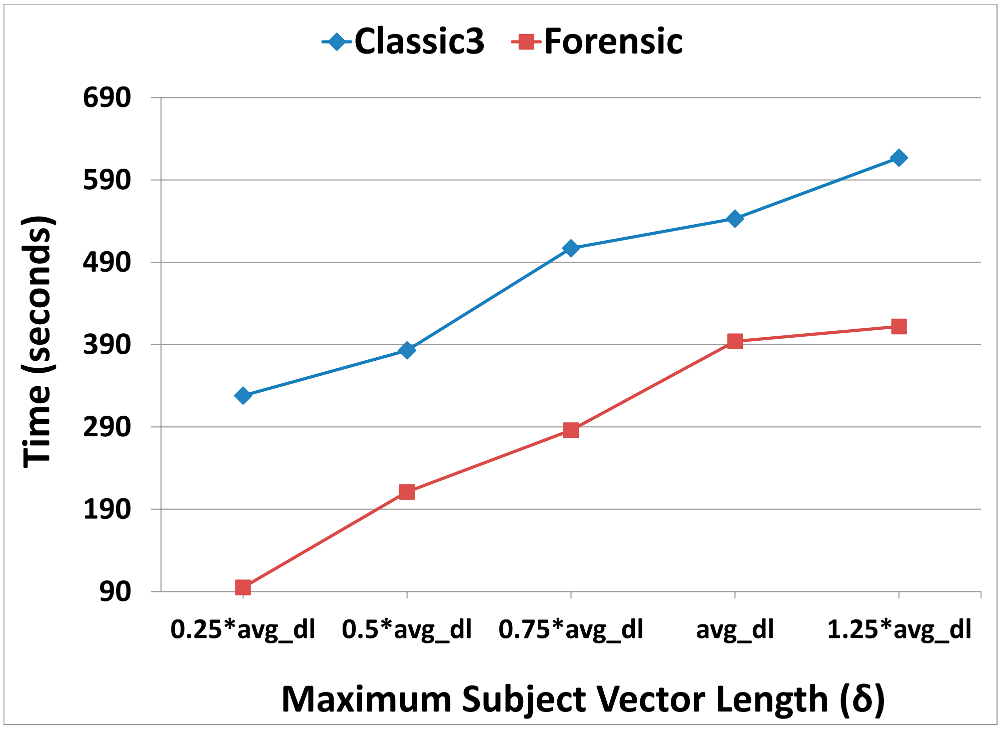

Figure 6: Efficiency with regard to *Classic3* and *Forensic* document sets

We also define the *remainder factor* α' as follows:

$$\alpha' = (Target\#ofdocuments)\ \%\ |D| \tag{11}$$

where $\lfloor\ \rfloor$ and $\%$ are the floor function and remainder function, respectively. We first copy all files in Classic3 $(\alpha - 1)$ times, and then we copy α' random files 1 time. The total number of files, including the original files in Classic3, is equal to the target number of documents.

Our subject vector generation algorithm consists of three phases: *ESL Lookup*, *WordNet Synonyms*, and *Top Frequent Terms*. The objective is to measure the runtime of each phase to ensure it does not grow proportionally to the total number of documents in the data set. Figure 7 depicts the runtime of the three phases with respect to the total number of documents being clustered. The total runtime for processing 100,000 documents is 313s, where 3s are spent looking up synonyms from *ESL*, 46s are spent generating synonyms using WordNet, and 264s are spent analyzing the document set to extract frequent terms that capture the suspects' terminologies. The runtimes of both the ESL Lookup phase and WordNet Synonyms phase are independent of the total number of documents. As for the Top Frequent Terms phase, the runtime grows as the total number of documents increases. This is due to the internal parameter ϵ that is used to capture the suspect's terminologies and is set to 1% of the data set size. The runtime scales linearly with respect to the data set's size. Since each phase of the algorithm is either independent or grows linearly with respect to the total number of documents, the experimental results on real-life data sets suggest that our algorithm is scalable.

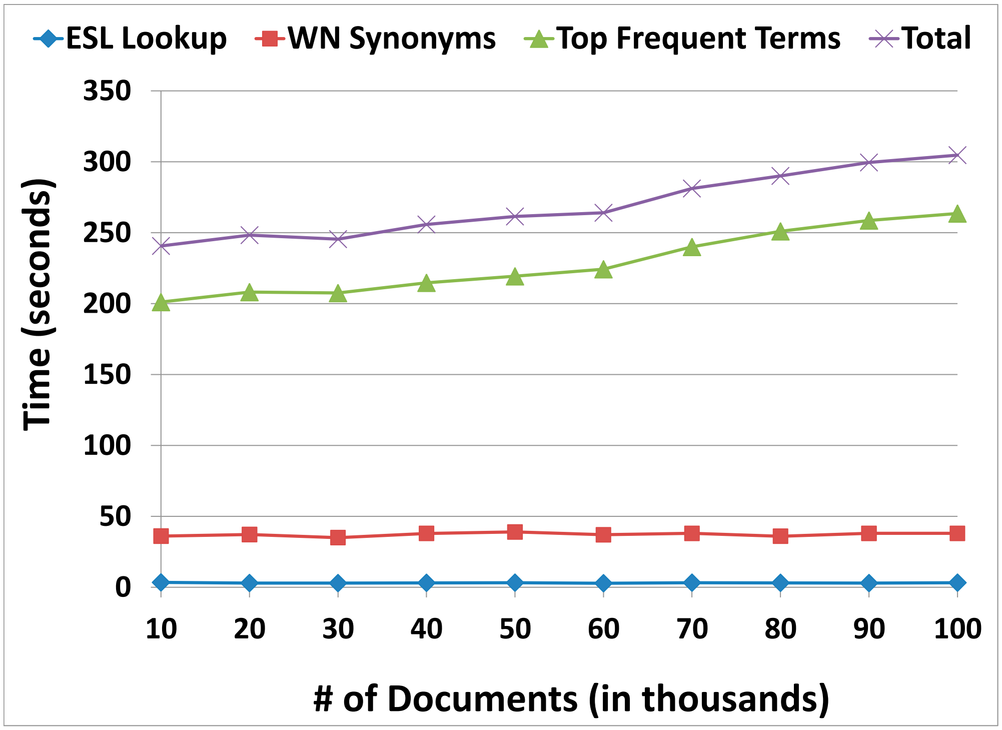

Figure 7: Scalability with the scale-up *Classic3* document set

Chapter 7

Conclusion

This chapter concludes the thesis. First, we give a summary of the contributions, then we describe the research directions that can be conducted as a future work.

7.1 Summary of Contributions

In this thesis, Motivated by the digital forensic process at *Sûreté du Québec* (SQ), we have proposed a subject-based semantic document clustering algorithm for digital forensic investigations with the objective of using data mining to support investigations.

First, we modeled our clustering solution by proposing *Subject Vector Space Model* (*SVSM*). In SVSM, each dimension represents a subject, where terms and documents are represented in the space according to their relations to all subjects. This allows a more realistic representation of the terms because terms inherently are not are orthogonal to each other. This representation also allows us to reduce the clustering problem of a document to the determination of the coordinate of this document on each subject vector. This is reflected in our proposed similarity function $Sim(d, s_i)$ that measures the similarity between any document $d \in D$ and subject $s_i \in S$, where D is a collection of documents and S is a set of subjects.

Second, we introduced an efficient and scalable subject-based semantic document clustering algorithm that expands the term vector representing each subject using WordNet. Word sense disambiguation (WSD) algorithm was integrated in the process to determine the appropriate sense (and accordingly, the synonym set) of a term in WordNet in the context of the initial terms that define a subject. The integration of WSD improved the precision of our clustering algorithm by reducing the polysemy affect.

Third, we dynamically captured suspects' terminologies by making use of the document set to incrementally expand the subject vector by adding top frequent terms from the most similar documents to the subject. WSD is also integrated in this phase to determine the dominant sense of the term, which is the sense used the most in the several contexts in which the term appears.

Finally, we conducted an extensive experimental study over two real-life data sets and examined the effectiveness of the algorithm according to subject vector length and document-score thresholds. We also demonstrate that our approach is highly scalable for large data sets.

7.2 Future Work

For future work, we identify several potential research directions.

SVSM model assumes that all axes are orthogonal to each other. It implies that subjects (dimensions) are completely independent of each other, i.e., the dot product between any two subject vectors is 0. This assumption simplifies the system and makes it easier to understand; however, in most cases, the subjects to be investigated are not completely independent since one or more terms might exist in more than one subject vector. Such dependency between subjects will impact the way the similarity between documents are computed, but more importantly, the similarity function between a document vector and a subject vector will change to take in consideration the dependency between subject vectors.

In the three steps of our algorithm, we integrated word sense disambiguation to improve the accuracy (presision/recall) of the algorithm. It would also be interesting to study how to integrate certain dimensionality reduction techniques, such as Principal Component Analysis [Jol86] or Outlier detection techniques [AY01], with respect to the impact on the accuracy, efficiency, and scalability of the algorithm.

In our algorithm, we limited each term in the initial subject definition to be either a verb or a noun. The reason for that is because verbs and nouns are the most content-bearing types of words [59]. However, it would be also interesting to investigate the impact of allowing other type of words to be part of the initial subject definition. For example, adjectives and adverbs also bear some content meaning, and WordNet already supports these type of words.

Bibliography

[ABKS99] M. Ankerst, M. M. Breunig, H. P. Kriegel, and J. Sander. Optics: Ordering points to identify the clustering structure. *SIGMOD Rec.*, 28:49–60, June 1999.

[AY01] C. C. Aggarwal and P. S. Yu. Outlier detection for high dimensional data. In *Proceedings of the 2001 ACM SIGMOD International Conference on Management of Data*, pages 37–46, New York, NY, USA, 2001. ACM.

[BBM02] S. Basu, A. Banerjee, and R. J. Mooney. Semi-supervised clustering by seeding. In *Proceedings of the 19th International Conference on Machine Learning*, pages 27–34, San Francisco, CA, USA, 2002. Morgan Kaufmann Publishers Inc.

[Bez81] J. C. Bezdek. *Pattern Recognition with Fuzzy Objective Function Algorithms*. Kluwer Academic Publishers, Norwell, MA, USA, 1981.

[BK03] J. Becker and D. Kuropka. Topic-based Vector Space Model. In *Proceedings of the 6th International Conference on Business Information Systems*, pages 7–12, Colorado Springs, July 2003.

[BM03] S. Basu and R. J. Mooney. Semi-supervised clustering: Learning with limited user feedback. Technical report, University of Texas at Austin, 2003.

[Cas00] E. Casey. *Digital Evidence and Computer Crime: Forensic Science, Computers, and the Internet with Cdrom*. Academic Press, Inc., Orlando, FL, USA, 1st edition, 2000.

[CCM03] D. Cohn, R. Caruana, and A. Mccallum. Semi-supervised Clustering with User Feedback, 2003.

[CCX$^+$04] H. Chen, W. Chung, J. J. Xu, G. Wang, Y. Qin, and M. Chau. Crime data mining: A general framework and some examples. *Computer*, 37:50–56, April 2004.

[CH98] W. Cohen and H. Hirsh. Joins that generalize: Text classification using whirl. In *Proceedings of the 4th International Conference on Knowledge Discovery and Data Mining*, pages 169–173, 1998.

[CMSW92] R. H. Creecy, B. M. Masand, S. J. Smith, and D. L. Waltz. Trading mips and memory for knowledge engineering. *Communications of the ACM*, 35:48–64, August 1992.

[CN89] P. Clark and T. Niblett. The cn2 induction algorithm. *Machine Learning*, 3:261–283, March 1989.

[Coh95] W. W. Cohen. Fast effective rule induction. In *Proceedings of the 12th International Conference on Machine Learning*, pages 115–123. Morgan Kaufmann, 1995.

[CRCG02] M. R. Clint, M. Reith, C. Carr, and G. Gunsch. An Examination of Digital Forensic Models, 2002.

[CS99] W. W. Cohen and Y. Singer. Context-sensitive learning methods for text categorization. *ACM Transactions on Information Systems (TOIS)*, 17:141–173, April 1999.

[CS04] B. D. Carrier and E. H. Spafford. An event-based digital forensic investigation framework. In *Proceedings of the 4th Digital Forensic Research Workshop*, 2004.

[Csi96] I. Csiszár. Maxent, mathematics, and information theory. *Maximum Entropy and Bayesian Methods*, 79:0–35, 1996.

[DBE99] A. Demiriz, K. Bennett, and M. J. Embrechts. Semi-supervised clustering using genetic algorithms. In *Proceedings of the Artificial Neural Networks in Engineering (ANNIE-99*, pages 809–814. ASME Press, 1999.

[DDF$^+$90] S. Deerwester, S. T. Dumais, G. W. Furnas, T. K. Landauer, and R. Harshman. Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, 41(6):391–407, 1990.

[DKR97] I. Dagan, Y. Karov, and D. Roth. Mistake-driven learning in text categorization. In *Proceedings of EMNLP-97, the 2nd Conference on Emperical Methods in Natural Language Processing*, pages 55–63, June 1997.

[DM01] I. S. Dhillon and D. S. Modha. Concept decompositions for large sparse text data using clustering. *Machine Learning*, 42:143–175, January 2001.

[DPHS98] S. Dumais, J. Platt, D. Heckerman, and M. Sahami. Inductive learning algorithms and representations for text categorization. In *Proceedings of the 7th International Conference on Information and Knowledge Management*, pages 148–155, New York, NY, USA, 1998. ACM.

[DS10] T. N. Dao and T. Simpson. Measuring similarity between sentences. *The Code Project*, 2010.

[Dun73] J. C. Dunn. A fuzzy relative of the isodata process and its use in detecting compact well-separated clusters. *Journal of Cybernetics*, 3(3):32–57, 1973.

[dVACM01] O. de Vel, A. Anderson, M. Corney, and G. Mohay. Mining e-mail content for author identification forensics. *ACM SIGMOD Record*, 30:55–64, December 2001.

[DWV99] H. Drucker, D. Wu, and V. N. Vapnik. Support vector machines for spam categorization. *IEEE Transactions on Neural Networks*, 10(5):1048 –1054, September 1999.

[ELL09] B. S. Everitt, S. Landau, and M. Leese. *Cluster Analysis*. Wiley Publishing, 4th edition, 2009.

[F9̈9] J. Fürnkranz. Separate-and-conquer rule learning. *Artificial Intelligence Review*, 13:3–54, February 1999.

[FB91] N. Fuhr and C. Buckley. A probabilistic learning approach for document indexing. *ACM Transactions on Information Systems (TOIS)*, 9:223–248, July 1991.

[FGG97] N. Friedman, D. Geiger, and M. Goldszmidt. Bayesian network classifiers. *Machine Learning*, 29:131–163, November 1997.

[For65] E. W. Forgy. Cluster Analysis of Multivariate Data: Efficiency vs Interpretability of Classifications. *Biometrics*, 21:768–769, 1965.

[Fri89] J. H. Friedman. Regularized discriminant analysis. *Journal of the American Statistical Association*, 84(405):pp. 165–175, 1989.

[FU02] U. Fayyad and R. Uthurusamy. Evolving data into mining solutions for insights. *Communications of the ACM*, 45:28–31, August 2002.

[Gar10] S. L. Garfinkel. Digital forensics research: The next 10 years. *Digital Investigation*, 7(1):S64 – S73, 2010.

[GHT07] Y. Guo, T. Hastie, and R. Tibshirani. Regularized Linear Discriminant Analysis and its Application in Microarrays. *Biostat*, 8(1):86–100, January 2007.

[GRS98] S. Guha, R. Rastogi, and K. Shim. Cure: an efficient clustering algorithm for large databases. In *Proceedings of the 1998 ACM SIGMOD International Conference on Management of Data*, pages 73–84, New York, NY, USA, 1998. ACM.

[HK06] J. Han and M. Kamber. *Data mining: Concepts and Techniques*. Elsevier, 2006.

[Hua97] Z. Huang. A fast clustering algorithm to cluster very large categorical data sets in data mining. In *Proceedings of the Research Issues on Data Mining and Knowledge Discovery*, pages 1–8, 1997.

[HZL$^+$09] X. Hu, X. Zhang, C. Lu, E. K. Park, and X. Zhou. Exploiting wikipedia as external knowledge for document clustering. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 389–396, New York, NY, USA, 2009. ACM.

[JC97] J. J. Jiang and D. W. Conrath. Semantic similarity based on corpus statistics and lexical taxonomy. In *Proceedings of the International Conference Research on Computational Linguistics (ROCLING X)*, pages 9008+, September 1997.

[Jen96] F. V. Jensen. *Introduction to Bayesian Networks*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 1st edition, 1996.

[JMF99] A. K. Jain, M. N. Murty, and P. J. Flynn. Data clustering: a review. *ACM Computing Surveys*, 31:264–323, September 1999.

[Joa98] T. Joachims. Text Categorization with Support Vector Machines: Learning with Many Relevant Features. In Claire Nédellec and Céline Rouveirol, editors, *Machine Learning: ECML-98*, volume 1398, chapter 19, pages 137–142. Springer Berlin / Heidelberg, Berlin/Heidelberg, 1998.

[Jol86] I. Jolliffe. Principal component analysis. *Springer Verlag*, 1986.

[KBS97] R. Kohavi, B. Becker, and D. Sommerfield. Improving simple bayes, 1997.

[Kin67] B. King. Step-wise Clustering Procedures. *Journal of the American Statistical Association*, 62(317):86–101, 1967.

[KJ00] R. Klinkenberg and T. Joachims. Detecting concept drift with support vector machines. In *Proceedings of the 17th International Conference on Machine Learning*, pages 487–494, San Francisco, CA, USA, 2000. Morgan Kaufmann Publishers Inc.

[Kon91] I. Kononenko. Semi-naive bayesian classifier. In *Proceedings of the European Working Session on Learning on Machine Learning*, pages 206–219, New York, NY, USA, 1991. Springer-Verlag New York, Inc.

[Kot07] S. B. Kotsiantis. Supervised machine learning: A review of classification techniques. In *Proceedings of the 2007 Conference on Emerging Artificial Intelligence Applications in Computer Engineering: Real Word AI Systems with Applications in eHealth, HCI, Information Retrieval and Pervasive Technologies*, pages 3–24, Amsterdam, The Netherlands, The Netherlands, 2007. IOS Press.

[KR90] L. Kaufman and P. J. Rousseeuw. *Finding Groups in Data: An Introduction to Cluster Analysis*. Wiley-Interscience, 9th edition, March 1990.

[KSB03] C. H. A. Koster, M. Seutter, and J. Beney. Multi-classification of patent applications with winnow. In *Proceedings of the Ershov Memorial Conference*, pages 546–555, 2003.

[LA99] B. Larsen and C. Aone. Fast and effective text mining using linear-time document clustering. In *Proceedings of the 5th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 16–22, New York, NY, USA, 1999. ACM.

[LC94] D. D. Lewis and J. Catlett. *Heterogeneous Uncertainty Sampling for Supervised Learning*, pages 148–156. Morgan Kaufmann, 1994.

[LCH08] Y. Li, S. M. Chung, and J. D. Holt. Text document clustering based on frequent word meaning sequences. *Data & Knowledge Engineering*, 64:381–404, January 2008.

[LdMA98] R. Lopez de Mantaras and E. Armengol. Machine learning from examples: Inductive and lazy methods. *Data & Knowledge Engineering*, 25:99–123, March 1998.

[Les86] M. Lesk. Automatic sense disambiguation using machine readable dictionaries: How to tell a pine cone from an ice cream cone. In *Proceedings of the 5th annual International Conference on Systems Documentation*, pages 24–26, New York, NY, USA, 1986. ACM.

[Lew98] D. D. Lewis. Naive (bayes) at forty: The independence assumption in information retrieval. In *Proceedings of the 10th European Conference on Machine Learning*, pages 4–15, London, UK, 1998. Springer-Verlag.

[LJ98] Y. H. Li and A. K. Jain. Classification of text documents. *The Computer Journal*, 41:537–546, 1998.

[LPM01] H. Lee, T. Palmbach, and M. Miller. *Henry Lee's Crime Scene Handbook*. San Diego: Academic Press, 2001.

[LW94] N. Littlestone and M. K. Warmuth. The weighted majority algorithm. *Information and Computation*, 108:212–261, February 1994.

[Mac67] J. B. Macqueen. Some Methods of Classification and Analysis of Multivariate Observations. In *Proceedings of the 5th Berkeley Symposium on Mathematical Statistics and Probability*, pages 281–297, 1967.

[McG11] S. B. 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.

[Mil95] G. A. Miller. WordNet: A Lexical Database for English. *Communications of the ACM*, 38:39–41, 1995.

[Mit97] T. M. Mitchell. *Machine Learning*. McGraw-Hill, Inc., New York, NY, USA, 1 edition, 1997.

[MLW92] B. Masand, G. Linoff, and D. Waltz. Classifying news stories using memory-based reasoning. In *Proceedings of the 15th annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 59–65, New York, NY, USA, 1992. ACM.

[MR02] S. Markovitch and D. Rosenstein. Feature generation using general constructor functions. *Machine Learning*, 49:59–98, October 2002.

[Mur98] S. K. Murthy. Automatic construction of decision trees from data: A multi-disciplinary survey. *Data Mining and Knowledge Discovery*, 2:345–389, December 1998.

[NGL97] H. T. Ng, W. B. Goh, and K. L. Low. Feature selection, perceptron learning, and a usability case study for text categorization. In *Proceedings of the 20th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 67–73, New York, NY, USA, 1997. ACM.

[PC01] G. Palmer and M. Corporation. A Road Map for Digital Forensic Research. In *Proceedings of the 1st Digital Forensic Research Workshop*, August 2001.

[PK07] A. Polyvyanyy and D. Kuropka. *A Quantitative Evaluation of the Enhanced Topic-based Vector Space Model*. Universitätsverlag Potsdam, 2007.

[Por97] M. F. Porter. *An Algorithm for Suffix Stripping*, pages 313–316. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1997.

[PPM04] T. Pedersen, S. Patwardhan, and J. Michelizzi. Wordnet: : Similarity - measuring the relatedness of concepts. In *Proceedings of AAAI*, pages 1024–1025, 2004.

[Qui93] J. R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1993.

[Ros62] F. Rosenblatt. *Principles of Neurodynamics: Perceptron and the Theory of Brain Mechanisms*. Spartan Books, Washington, D.C., USA, 1962.

[Sal89] G. Salton. *Automatic Text Processing: the Transformation, Analysis, and Retrieval of Information by Computer*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1989.

[SB88] G. Salton and C. Buckley. Term-weighting approaches in automatic text retrieval. *Information Processing and Management*, 24:513–523, August 1988.

[Seb02] F. Sebastiani. Machine learning in automated text categorization. *ACM Computing Surveys (CSUR)*, 34:1–47, March 2002.

[SGW$^+$95] T. E. Senator, H. G. Goldberg, J. Wooton, M. A. Cottini, U. Khan, C. D. Klinger, W. M. Llamas, M. P. Marrone, and R. W. H. Wong. The fincen artificial intelligence system: Identifying potential money laundering from reports of large cash transactions. *AI Magazine*, 16:21–39, 1995.

[SHP95] H. Schütze, D. A. Hull, and J. O. Pedersen. A comparison of classifiers and document representations for the routing problem. In *Proceedings of the 18th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 229–237, New York, NY, USA, 1995. ACM.

[SJ88] K. Sparck Jones. *A Statistical Interpretation of Term Specificity and its Application in Retrieval*, pages 132–142. Taylor Graham Publishing, London, UK, 1988.

[SS73] P. H. A. Sneath and R. R. Sokal. *Numerical Taxonomy (by) Peter H.A. Sneath (and) Robert R. Sokal: The Principles and Practice of Numerical Classification*. W.H. Freeman, 1973.

[WAD$^+$99] S. M. Weiss, C. Apte, F. J. Damerau, D. E. Johnson, F. J. Oles, T. Goetz, and T. Hampp. Maximizing text-mining performance. *IEEE Intelligent Systems*, 14:63–69, July 1999.

[War63] J. H. Ward. Hierarchical Grouping to Optimize an Objective Function. *Journal of the American Statistical Association*, 58(301):236–244, March 1963.

[WCRS01] K. Wagstaff, C. Cardie, S. Rogers, and S. Schrödl. Constrained k-means clustering with background knowledge. In *Proceedings of the 18th International*

Conference on Machine Learning, pages 577–584, San Francisco, CA, USA, 2001. Morgan Kaufmann Publishers Inc.

[WPW95] E. D. Wiener, J. O. Pedersen, and A. S. Weigend. A Neural Network Approach to Topic Spotting. In *Proceedings of SDAIR-95, the 4th Annual Symposium on Document Analysis and Information Retrieval*, pages 317–332, Las Vegas, USA, 1995.

[XBP09] P. Xu, G. N. Brock, and R. S. Parrish. Modified linear discriminant analysis approaches for classification of high-dimensional microarray data. *Computational Statistics & Data Analysis*, 53:1674–1687, March 2009.

[YC94] Y. Yang and C. G. Chute. An example-based mapping method for text categorization and retrieval. *ACM Transactions on Information Systems*, 12:252–277, July 1994.

[ZG03] S. Zhong and J. Ghosh. A unified framework for model-based clustering. *The Journal of Machine Learning Research*, 4:1001–1037, December 2003.

[Zho06] S. Zhong. Semi-supervised model-based document clustering: A comparative study. *Machine Learning*, 65:3–29, October 2006.