CRIMINAL NETWORK MINING AND ANALYSIS FOR FORENSIC INVESTIGATIONS

RABEAH ALZAIDY

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

AUGUST 2010

© RABEAH ALZAIDY, 2010

CONCORDIA UNIVERSITY
School of Graduate Studies

This is to certify that the thesis prepared

By: **Rabeah Alzaidy**

Entitled: **Criminal Network Mining and Analysis for Forensic Investigations**

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

M.A.Sc. (Information Systems Security)

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

Signed by the final examining committee:

Dr. A. Awasthi	**Chair**
Dr. S. Li	**Examiner**
Dr. Z. Kabir	**Examiner**
Dr. B. Fung	**Supervisor**
Dr. A. Youssef	**Supervisor**

Approved by ____
Chair of Department or Graduate Program Director

____
Dean of Faculty

Date

ABSTRACT

Criminal Network Mining and Analysis for Forensic Investigations

Rabeah Alzaidy

Criminal network analysis tools are widely used by law enforcement, mainly in cases of organized crime. The data required for a majority of these tools are police records and databases. In many cases, forensically collected data contains valuable information about the suspect's social network. This information is normally obtained by manual inspection of the collected documents using forensic tools' queries and other basic search features. The information is then manually entered in the police database. There are no known tools that provide methods to automatically extract social networks from raw documents on behalf of the investigator, add them to a knowledge base and then analyze them.

In this thesis, we propose a method that is capable of performing these tasks. In our proposed system, we claim three distinct contributions to cyber forensics investigations. The first is by constructing the social network of one or multiple suspects from documents in a file system. Secondly, we provide an analysis of the interactions and structures of these social networks and the communities comprising them. Thirdly, potential evidence and leads are identified by extracting conceptual links between members of the social network across the document set.

Finally, the proposed method is implemented and experimental results are obtained to demonstrate the feasibility of the approach.

Acknowledgments

When writing a thesis, one is by necessity dependent upon many people. The substantive value of each contribution is difficult to assess. Therefore, I wish to convey my gratitude to all who in any way aided me in this endeavor. Particular thanks are to the following:

To my parents, sisters, and brother for their endless encouragement and support.

To my supervisors Dr. Amr Youssef and Dr. Benjamin Fung whose advice, knowledge and guidance from the very early stages of this research made this thesis possible.

Finally, to all my friends and colleagues who made themselves available by providing support in many ways.

Contents

List of Figures

List of Tables

List of Acronyms

CCMS	Criminal Communities Mining System
CRF	Conditional Random Fields
FP	Frequent Pattern
HG	Hypothesis Generation
HMM	Hidden Markov Models
IE	Information Extraction
ME	Maximum Entropy
NER	Named Entity Recognition
NLP	Natural Language Processing
OTS	Open Text Summarizer
SML	Supervised Machine Learning
SNA	Social Network Analysis

Chapter 1

Introduction

The recent availability of storage media in large sizes has increased the average amount of data that can be saved by an individual. When law enforcement investigates a crime, these devices must be seized from the suspects involved in the case for forensic inspection. Inspection of the files involves searching their content for information that can be used as evidence or lead to other sources of information that may assist the investigation process and analyzing this retrieved information. The forensic analysis of this collected data is a tedious task; therefore, many tools have emerged to assist investigators. However, it is typically up to the investigator to search for data that is significant to the case at hand. Significant information such as information about other criminals or the suspect's collaborations is among the most sought after in the analysis process. Social Network Analysis (SNA) tools provide mechanisms to analyze social networks once provided with a database of criminals and their allies. However, in the case of hard drives and forensically seized storage media, contrary to police reports, the individuals in the social networks are not stored in a database but are scattered across the hard drive's documents. The identification of social networks in text documents can be viewed as the task of mining the data collection in search for individuals and their interactions. This approach allows us to address the problem as a data mining one and we can apply data mining techniques to solve it.

1.1 Motivation

The procedures for accomplishing digital forensics involve two key steps: the collection of digital evidence and the analysis of the collected data. Typically, in the collection phase, data is gathered from digital sources in formats varying from common file types to memory dumps and in many cases data is extracted in bitstream formats (a bitstream is a sequence of bits that requires further processing in order to transform it to a human-readable format.) Today's digital forensic practitioners have a large number of tools developed for the aforementioned forensic tasks to choose from. Most of these tools focus solely on physical analysis, i.e, representing the retrieved digital content in a human-readable format. However, once the logical analysis of the data begins, investigators are faced with the monumental task of examining the vast amount of collected data in search for evidence. On this front, toolkits, such as EnCase [26] and Forensic ToolKit [8], provide various forensic features such as data filtering, keyword search, and time line views. These features are designed in an attempt to alleviate the magnitude of the analysis task. Although these features are useful, they cannot be considered sufficient for several reasons:

- They rely mainly on the investigative skills of the data analyst in constructing a proper list of terms related to the case.
- Even so, there remains the fact that on the outset of a case these lists are usually based on hunches and few of them have actual facts to support them, regardless of how skilled the investigator is.
- These tools do not provide any intelligent information retrieval features.
- They have no conceptual content analysis features, i.e., they do not provide any content-based classifications or interpretations for the documents.

The need to provide intelligence features to content analysis tools has initiated studies that have proposed tools that aim to address the conceptual content analysis, such as SNA tools.

Forensic SNA tools attempt to systematically analyze complex social structures between criminals in large data sets. The application of SNA tools for criminal network analysis has been the focus of many studies for many reasons, some of which are:

- In organized crime cases, where crimes are carried out by a group of collaborating criminals, the documents on the hard drive of a criminal in the group are likely to contain names of the other criminals in the group. Texts retrieved from chat sessions and email messages are the most likely to contain these names.

- Names are a common lead in investigations. For example, names of business personnel can be found that might be able to testify for the occurrence of some purchase or interaction of significance. A certain business such as hotel or a bank in which the suspect kept a record of might have other names mentioned with it in textual context, and this can lead to one of the suspect's contacts.

- Novel relationships may be hidden in the data. Criminals tend to use certain terms unknown to anyone other than themselves. These terms can be mentioned in a document with the suspects name and in another document with another name. This link can rarely be found by manually going through the documents.

- According to [22], the failure to objectively analyze criminal networks is one of the major reasons law enforcement and intelligence can fail. By using SNA systems, a more objective analysis is ensured.

- Generally, SNA tools can be effective tools in supporting law enforcement prosecutions if they are reliable enough to serve as evidence. Identifying key players in a criminal network and providing meaningful information about the social structure of the network can prove that a criminal has played a role in a certain network. However, because of the various challenges facing such tools, e.g., the lack of reliable data sources, SNA tools have not reached this level of reliability [22].

- Investigators who are provided with a meaningful criminal network analysis have greater chances to learn more about their opponents. By examining criminal networks closely and studying the behavior of the individuals comprising these networks and the flow of resources among them, investigators enhance their understanding of structures of various crimes. Moreover, they gain more knowledge about how and in what positions the individuals participate in the network to successfully achieve their criminal goals.

1.2 Objectives

The main objective of this thesis is to provide forensic analysis tools with intelligence features. Our focus is on providing these tools with the ability to identify social communities from forensically collected text documents. Moreover, we aim to analyze the social links between members of the networks and extract valuable information about their interactions. Our goal is to provide a means to automate the construction and analysis of the social networks, yet allow for the intervenience of human guidance. Since our analysis is in the context of forensic investigation, the extracted information must be of investigatory and evidentiary value, specifically we select the following information:

1. Members and key role-players of a suspect's social network

2. Analysis of the links between members of the social network

3. Sensitive information related to social communities in the network

4. Indirect links between members of the social network, i.e., links through trails of evidence

Moreover, we aim to provide a visual representation to display the results of the analysis. Exploration of the social networks is most efficient when they are displayed as graphs that map the network members and the links between them.

To meet these objectives, it is necessary to examine information extraction methods, as well as link analysis techniques. Sensitive data such as contact information and topics about the communities add to the value of the extracted communities, which many social network extraction tools do not provide. The display of social networks and the sensitive data found attached to these networks, along with the mining of possible indirect links between these networks is a novel approach that we believe will provide both a powerful and reliable criminal network analyzer.

1.3 Contribution

In order to meet our objectives, we have developed a novel method for social network mining and analysis in a data set of forensically collected documents. We propose two methods for identifying two types of links between social entities from the documents of a file system. Both methods are implemented and successfully tested to identify social networks and perform an analysis of the interactions between members of the network. In many cases the system is able to present new names and links across the data set that the user was not aware of. The system is also capable of suggesting hypothetical links across members of the social network. In many cases the links have been proven to be a valid hypothesis.

Our approach has the following merits:

- We explore the feasibility of Named Entity Recognition in identifying members of a social network in textual content

- We investigate the efficiency of a frequent pattern approach to identify criminal communities in a set of text documents
- We examine the application of Information Extraction techniques in analyzing links between the individuals in a social network
- We explore the employment of Hypothesis Discovery methods in detecting evidential trails between individuals in the network
- We propose a visual representation of the social network and the interaction patterns between the individuals

Figure 1 depicts an overview of our proposed approach to address the criminal network analysis problem. The first task is to read a collection of text documents and extract peoples' names from them. The name extraction task is followed by a normalization process to eliminate duplicate names that refer to the same person. The next task is to extract the prominent criminal communities from the document files. Then, we extract the information that is valuable to investigators such as contact information. After the criminal communities are identified, we extract the indirect links between the criminals from the data set. Finally, the social network is visualized along with the relevant information regarding the actors.

An implementation of our approach, named the Criminal Communities Mining System (CCMS), is developed to test the feasibility of our proposed methods. The system identifies social communities in a data collection and retrieves sensitive information about these communities. The analysis is also capable of defining the structure of the social communities by identifying the sub-groups existing in larger communities. We implement the analysis methods such that various parameters of each method can be set manually by the

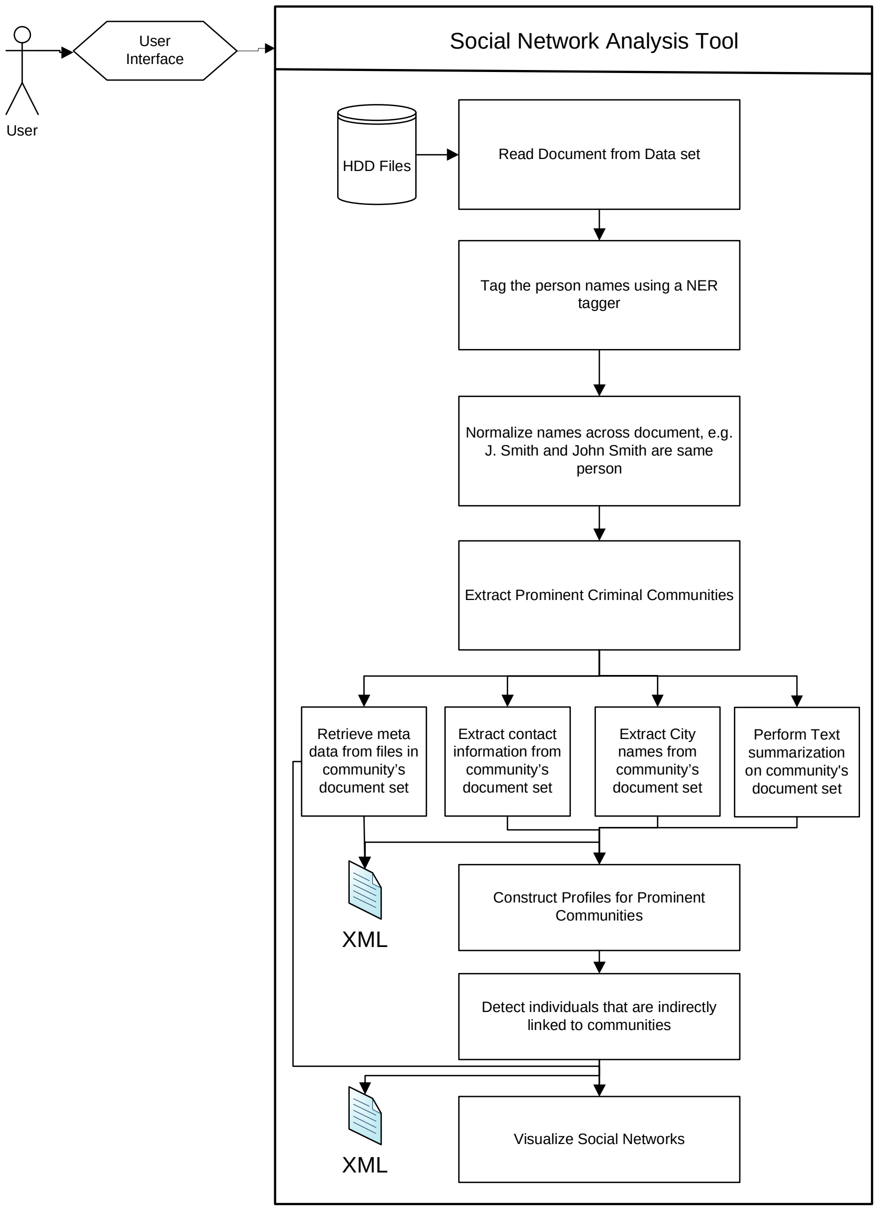

User
User Interface
Social Network Analysis Tool
HDD Files
Read Document from Data set
Tag the person names using a NER tagger
Normalize names across document, e.g. J. Smith and John Smith are same person
Extract Prominent Criminal Communities
Retrieve meta data from files in community's document set
Extract contact information from community's document set
Extract City names from community's document set
Perform Text summarization on community's document set
XML
Construct Profiles for Prominent Communities
Detect individuals that are indirectly linked to communities
XML
Visualize Social Networks

Figure 1: Proposed Criminal Communities Mining System

user. Integrating the professional expertise in the proposed systematic methods allows for finer tuning of the analysis results.

1.4 Thesis Organization

The remainder of this thesis is organized as follows.

Chapter 2 presents definitions of digital forensic terms. It describes the significance of data analysis in digital forensics procedures and some major challenges in this domain. It also addresses limitations of some popular state-of-the-art forensic tool kits in analysis performance. Moreover, it provides definitions of data-mining techniques and terms as well as some hypothesis generation concepts. Additionally, we introduce concepts related to social network analysis and data visualization.

Chapter 3 explains our proposed method for extracting social networks. It demonstrates how the criminal communities are identified and the methods employed to extract sensitive information about them.

Chapter 4 addresses the detection of indirect relationships. It explains our proposed discovery algorithm that is developed to address this problem.

Chapter 5 introduces our social network extraction tool. This tool is developed to assist investigators in analyzing digitally collected data by detecting social networks and their interaction data. Finally, chapter 6 summarizes our research and provides conclusions and future research directions.

Chapter 2

Preliminaries

In this chapter, we explain the background information that is relevant to the field of digital forensics, especially in the areas of digital evidence analysis and criminal network analysis. In order to analyze criminal networks, the information required for the analysis must first be extracted from the storage media. Therefore, we discuss Information Extraction (IE) techniques and models, focusing on the ones that are adopted in our proposed system. Since data mining techniques are employed to identify the criminal networks, we briefly explain the relevant algorithms. We also explain the Hypothesis Generation (HG) algorithms, the basis of our proposed indirect relationship extraction method. Finally, we present an overview of key concepts and techniques of data visualization.

2.1 Digital Forensics

The science of digital forensics can be defined as the use of scientifically derived and proven methods used for 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 criminal events [40]. In order to present a formal methodology to accomplish these tasks, many proposed methodologies emerged. In 2001, the Digital Forensics Research Workshop [40] proposed a process for digital investigations that involves the following six procedures:

1. Identification - identify the incident and specify its type
2. Preservation - preserve and secure digital and physical evidence
3. Collection - document the crime scene and produce forensic copies of digital evidence
4. Examination - in-depth examination and search for evidence and preparation of detailed documentation for analysis.
5. Analysis - filter and detect important data either manually or through analysis tools
6. Presentation - summarize and explain the conclusions.

Since criminal network analysis is a task that is conducted during the analysis phase, we specifically discuss this phase in the remainder of this section.

In an investigation, the analysis phase is the one that most relies on the investigator's skills and experience. To analyze the data collected about a case, an investigator wants to understand and know where the suspect might have hidden the data and in what formats and what application he might have used. Some patterns in the data are important; once found and fully examined, they can lead to more evidence. In order to achieve a successful analysis, many tools are adopted to aid investigators analyze the collected data. Common tools include EnCase [26] and the Forensic ToolKit [8] which comes with a searching tool *dtSearch*. These tools display the files of a storage media and allow the user to navigate through the files similar to traditional file explorers. However, they provide additional features that are useful in forensics context such as displaying file headers and opening compressed files. Some of these tools provide more contextual analysis features such as queries and a time-line view of the files. However, the investigator is responsible for manually performing the analysis and gathering knowledge from the extracted data. Thus, the main shortcoming of these tools is that they do not provide automated intelligent analysis of the information they are developed to extract.

2.2 Information Extraction

Information Extraction (IE) refers to the automatic identification of information, such as entities, relationships between entities, and attributes describing entities, often referred to as structured information, from unstructured sources [38, 48]. For over two decades, researchers have been addressing the challenging task of extracting structures from noisy, unstructured data collections. Extracting information involves many tasks including the identification of named entities such as people and organization names from natural language textual content. The advances in information extraction techniques were strongly influenced by both the Message Understanding Conference (MUC) and the Automatic Content Extraction (ACE) program [48].

2.2.1 Named Entity Recognition

Named Entity Recognition (NER) is an IE task that involves identifying entities of predefined types in unstructured texts. Entities can be names of persons, locations, etc. The ability of NER tools to extract names is based on recognition and classification rules that are provoked by features associated with positive and negative examples. NER techniques can be categorized into supervised, semi-supervised, and unsupervised machine learning techniques. According to [48], supervised learning approaches consist of systems that read a large annotated corpus, memorize lists of entities, and create disambiguation rules based on discriminative features. Examples of models that are based on this approach are Hidden Markov Models (HMM) [42], Maximum Entropy (ME) [35], and Conditional Random Fields (CRF) [31]. Semi-supervised learning, as the name indicates, involves a small degree of supervision for the learning process. A common technique used for this approach is bootstrapping. The last category, unsupervised learning, relies on a clustering approach that makes use of lexical resources such as WordNet [37]. Techniques from either category can be used depending on the application and they can also be combined to yield better results.

Other important aspects of NER to consider are domain specific challenges. For instance, extracting people's names from a corpus of news articles might not be efficient when applied to less formal texts such as instant message logs and email messages. Informal texts involve more complex methods to recognize the named entities within them. This is due to the fact that these documents do not normally follow strict grammatical and punctuation conventions.

Several open source NER tools have emerged, which employ Natural Language Processing (NLP) techniques. The Stanford Named Entity Recognizer performs well on newswire documents [13] and the Illinois Named Entity Tagger has also shown high performance rates [7,44].

2.2.2 Text Summarization

Text summarization is the process of distilling the most important information from a source or multiple sources to produce an abridged version for a particular user and task.

Automatic text summarization tools take textual documents and provide a synopsis of the input document. A synopsis can vary from being a set of keywords that represent the main topic of the document, to a list of sentences that grouped together represent a coherent summary of the content of the document [47]. According to [20], the summarization process can be broken down into three stages:

1. Identification: Identify the important topics.
2. Interpretation: The identified topics are outputted after related topics are generalized and redundancies are removed.
3. Generation: Construct a new text from the extracted topics and produce a coherent summary.

Various approaches are used to address these tasks which can be classified into two main categories depending on whether the summary is an abstraction or extraction. Extractions are summaries that are based on selecting salient parts of the text as the summary, whereas abstracts are summaries that are produced by understanding the texts and reproducing a shorter, more focused version of the source document [14].

Extract summaries are often produced by approaches that are based on lexical and statistical relationships between the text units. The summary is produced by concatenation of certain parts of the original text. These approaches usually follow a linear weight model, that is relatively easy to implement. The linear weight is a model that assigns a weight to each text unit by summing the weight values based on some features. Examples of such approaches are ones that are based on features such as term frequency, location of the term, and other statistically significant features. The Open Text summarizer [46] is an example of an extract summarizer that is developed based on a linear model.

The main disadvantage of abstract summaries is that they require heavy computations. They are usually obtained through either a syntactic parsing approach, or the NLP approach. The first approach produces parse trees of the text and then re-groups parts of the tree based on some criteria to produce the summaries. The NLP approach also involves syntactic parsing but does not produce parse trees as a result. Instead, it uses the parse trees as a conceptual representation of the source document. These are assembled and used to build a text knowledge base. The Identification step is performed on the knowledge base to produce a less detailed conceptual representation of the document, which is synthesized as the output in the generation step. Many systems using the abstract approaches require training corpora.

Examples of the most impressive existing text summarizing tools are the MEAD system [43], the SUMMARIST system [21], [20], and the Newsblaster [9], which were all developed by employing extractive summarization techniques. Meanwhile, abstraction based approaches are still mostly theoretical [34].

2.3 Frequent Itemset Mining

In data mining, frequent itemsets are used to find frequent patterns in a large set of data. For instance, in a set of documents R, where each document must contain at least one name, each name is an item, and each document is referred to as a transaction. In a single transaction, each set of items (names) is referred to as an itemset. An itemset that consists of k items is a k-itemset. The set $\{Bob, Alice\}$ is a 2-itemset, whereas $\{Bob, Alice, John\}$ is a 3-itemset. The occurrence frequency of an itemset is the number of transactions (documents) in the data set that contain the itemset, this is called the support count of the itemset. If the support count of an itemset satisfies a previously specified minimum support value, it is called a frequent itemset. The set of frequent itemsets with k items is denoted by L_k. Frequent itemset mining methods provide a means of efficiently finding frequent itemsets in large data sets. In the following subsections, we present two of these methods and compare them in terms of efficiency and scalability.

2.3.1 The Apriori Algorithm

The Apriori algorithm is a bottom-top algorithm, proposed by R.Agrawal and R. Srikant in 1994 for mining frequent itemsets [1] [3]. As the name indicates, this algorithm makes use of prior knowledge of a property of frequents itemsets that is: all nonempty subsets of frequent itemsets must be frequent. To illustrate this property, consider the 3-itemset $\{Bob, Alice, John\}$. If we assume this itemset satisfies the minimum support, then by definition, its subset $\{Bob, Alice\}$ must also satisfy the minimum support threshold.

The Apriori algorithm uses this property in a depth first approach where frequent k-itemsets are used to find frequent $(k + 1)$-itemsets. The first step is to scan the database and accumulate the support count for each item. Those that pass the minimum support threshold are added to the set of frequent 1-itemsets denoted L_1. Next, L_1 is used to find the set of frequent 2-itemsets, L_2. This is turn is used to find L_3, and so on, until no more frequent k-itemsets can be found. For each of these L_k sets, a full scan of the database is required.

The Apriori algortihm has been employed in many applications for its accuracy in mining frequent patterns. However, its main disadvantage is that it needs to generate a huge number of candidate sets, which reduces its scalability. For instance, in order to find a frequent itemset of size 100, it has to generate at least 2^{100}-1 $\approx 100^{30}$ candidates in total. Another drawback of this algorithm is that it must repeatedly scan the database, which is costly.

Figure 1 outlines the steps of the Apriori algorithm. The algorithm requires a transaction database D and the minimum support count threshold $min\_support$, as user input. The Apriori algorithm generates the L_k sets of frequent itemsets from the set L_{k-1} by following a sequence of pruning and joining steps iteratively, until no more L_k sets can be found. The joining and pruning steps are as follows:

Joining For each iteration k in the Apriori algorithm, a set of candidate frequent itemsets

```
Input:
   - A database of transactions ($D$)
   - The minimum support count threshold ($min\_support$)
Output:
   $L$, frequent itemsets in $D$.
Method:
   $L_1$ = find_frequent_1-itemsets(D);
   for ($k = 2$ ; $L_{k-1} \neq \phi$; $k++$ ) do
      $C_k$ = apriori_gen($L_{k-1}$)
      for each transaction $t \in D$ do
         $C_t$ = subset($C_k$, $t$)
         for each candidate $c \in C_t$ do
            c.count++;
         end for
      end for
      $L_k$={$c \in C_k | c.count \geq min\_support$}
   end for
   return $L = \cup_k L_k$;
Procedure apriori_gen($L_{k-1}$ : frequent $k-1$-itemsets)
   for each itemset $l_1 \in L_{k-1}$ do
      for each itemset $l_2 \in L_{k-1}$ do
         if ( $l_1[1] = l_2[1]$ ) $\wedge$ ($l_1[2] = l_2[2]$) $\wedge \ldots \wedge$ ($l_1[k-2] = l_2[k-2]$) $\wedge$ ($l_1[k-1] < l_2[k-1]$)
         then
            $c = l_1 \bowtie l_2$
            if has_infrequent_subsets($c$, $L_{k-1}$) then
               delete $c$;
            else
               add $c$ to $C_k$
            end if
         end if
      end for
   end for
   return $C_k$
Procedure has_infrequent_subsets($c$: candidate $k$-itemsets, $L_{k-1}$: frequent ($k-1$)-itemsets)
   for each ($k-1$)-subset $s$ of $c$ do
      if $s \notin L_{k-1}$ then
         return TRUE;
      end if
   end for
   return FALSE
```

Algorithm 1: Apriori Algorithm [15]

C_k is generated from the set of frequent itemsets found in the previous iteration $k-1$. The set C_k is the result of joining the set of frequent $k-1$ itemsets, namely L_{k-1} with itself, $L_{k-1} \bowtie L_{k-1}$. In order to join two itemsets l_1 and l_2 in L_{k-1} their first $k-2$ items must have the same values. For instance, let $l_1[i]$ denote the ith item in l_1. Then, for l_1 and l_2 to be joinable, they must satisfy :

$(l_1[1] = l_2[1]) \wedge (l_1[2] = l_2[2]) \wedge \ldots \wedge (l_1[k-2] = l_2[k-2])$

It is important to note that in order to avoid generating duplicates, the Apriori algorithm assumes that items are lexicographically ordered within the itemsets. By joining the sets l_1, l_2 that satisfy the joinable condition, we get the candidate itemsets containing the items: $l_1[1], l_1[2], \ldots, l_1[k-1], l_2[k-1]$.

Pruning The purpose of the joining step is to generate a list of itemsets that are candid to be frequent, based on the knowledge that they are constructed from itemsets that are frequent. The set of frequent k-itemsets, L_k, is a subset of C_k. The pruning step removes all non-frequent itemsets that occur in C_k, the resulting list after the pruning is L_k. To count the support of each candidate itemset in C_k, a scan of the transaction database D is required, where the support count for each itemset l_i in C_k is calculated. As a result of the large number of itemsets generated in the joining step, we eliminate from our scan all candidates that contain a non-frequent subsets. For such candidates, we know it is not frequent prior to counting its support in the database because at least one of its subsets is infrequent. This frequent subset check is useful since it reduces the number of itemsets in C_k that are looked up in the database scan.

For a transaction database D and a preset value for the minimum support count threshold $min\_support$, the joining and pruning steps are adopted to find the frequent 1-itemsets, 2-itemsets, …, k-itemsets denoted $L_1, L_2, \ldots, L_k$ respectively. The Apriori algorithm proceeds as follows:

1. Start Apriori by a scan of the transaction database D to create a list C_k of all items in D and count their support.

2. All items in C_k whose minimum support count does not pass the $min\_support$ threshold are removed from C_k. The resulting list of frequent 1-itemsets is L_1.

3. If L_1 is a non-empty set, use the method explained in the joining step to generate the list of frequent 2-itemset candidates, namely C_2.

4. Apply the pruning step to remove any non-frequent itemsets from C_2. The result is a set of the frequent 2-itemsets, L_2.

5. To find the sets of frequent itemsets for the remaining values of k, steps 3 and 4 are repeated until they yield either an empty L set or an empty C set. An empty set value for either C or L terminates the Apriori algorithm.

2.3.2 Frequent-Pattern Growth Algorithm

The Frequent Pattern (FP) Growth method [16] is yet another method for mining frequent itemsets. One of the main advantages of this method compared to the Apriori algorithm is the fact that it does not involve the costly step of candidate generation. In this approach, the database representing frequent itemsets is compressed in a frequent-pattern tree, or FP-tree. The tree is constructed after scanning all the transactions in the database. Thus, the problem of mining frequent itemsets in the database is transformed to that of mining the FP-tree. This approach is more scalable in the sense that it does not include the costly step of generating candidates. It also requires less scans of the database. Yet, for very large databases, the construction of a memory-based FP-tree is impractical.

The FP-growth algorithm for mining frequent itemsets consists of two steps: (1) constructing an FP-Tree, (2) mining frequent patterns using FP-tree. For a transaction database

D, and a minimum support threshold $min\_support$, mining the frequent itemsets using the FP-growth algorithm is achieved through the following steps:

1. Constructing the FP-tree:
    (a) The tree consists of a root node labeled "null", a set of item-prefix subtrees as the children of the root, and a frequent item-header table.
    (b) Each node in the item-prefix subtree carries three values: the name of the item, a pointer value, and a support count value. The pointer links the node to any other node in the tree that has the same value for the node name, this value is set to null if no such node exists. The support count value represents the number of transactions in the database that contain the items in the portion of the path from the root down to this node as an itemset in that transaction.
    (c) The frequent item-header table consists of records, each record contains two values: the item-name and a pointer to the first node in the FP-tree carrying an item with this name. Thus the size of the table is equal to the number of unique items in the database.
2. The second step is to traverse the tree to find the frequent itemsets.
    (a) To extract the frequent itemsets from the FP-tree a bottom-up algorithm is used, it starts from the leaf nodes and traverses upwards toward the root.
    (b) Divide and conquer: first it looks for frequent itemsets ending with a specific item ,e for instance. Then, it searches for the itemset ending with de, and so on until all possible prefixes are searched. Next, it continues to d, then cd, and so on.
    (c) First, extract prefix path sub-trees ending in an item(set) using the linked lists.

(d) Each prefix path sub-tree is processed recursively to extract the frequent itemsets. Solutions are then merged.

2.4 Hypothesis Generation

Generating hypotheses [27] is a major step in making new discoveries. The generation of hypothesis requires prior knowledge, experience, and intuition [54]. The more data to generate the hypothesis from, the more challenging the task becomes. Many studies made use of text mining techniques to generate propositions or hypothesis that require further verification. Text mining is a field similar to data mining, but differs from it in that it involves mining unstructured texts instead of databases [17]. In the biomedical domain Swanson at al. were able to use text mining techniques to propose several hypothesis from MEDLINE, the National Library of Medicine's premier bibliographic database, that were later confirmed by bioscientists. Ever since, this knowledge discovery approach has been studied by other researchers. One major contributor was Srinivan who after two decades proposed the open and closed discovery methods [50] that were built within the discovery framework initiated by Swanson [51, 52] and Smalheiser [53]. The main goals of the discovery algorithms are to process a large set of documents of textual content and generate interesting hypotheses concerning the associations of specific terms, thus discovering new knowledge [50].

The open and closed discovery algorithms presented in [50] are based on representing topics in a structure referred to as a profile. These profiles are built as a set of MeSH terms related to the topic of interest. MeSH terms, short for Medical Subject Headings terms, are the metadata applied to MEDLINE records. By selecting different MeSH terms different profiles can be generated for a specific topic. Consequently, the algorithms expect two inputs: topics of interest and the type of profile to build. The algorithms were then tested and evaluated with various parameters.

2.4.1 Profiles

The profile for a term A that is extracted from a collection of texts specifies all terms that are statistically related to this term within this textual context. The assumption in this case is that a statistical association implies a conceptual association. To illustrate this assumption, consider the profile built for the topic *Barack Obama* extracted from newswire article texts. The profile contains names of places he visited, issues he is involved in, and the events he participates in. In order to build a profile for a given topic, the type of relevant information must first be clearly identified. Here, the type of information refers to its semantic type that can be extracted from a textual collection through Information Extraction methods. Phone numbers can be a type of relative terms, city names are also another type of terms. Each term is assigned a weight, which represents the strength of the association between the term and the topic. Weights can be calculated as the distance between the term and the topic in the documents, i.e., the number of words between them, or as the term frequency. Moreover, the terms can be further classified within the profile by their semantic types. Thus, a term is represented by its semantic type and its weight. Each semantic type is represented as a vector of weighted terms. Basically a profile for a topic T_i is a vector of semantic terms vectors,

$$Profile(T_i) = \{\{w_{i,1,1}m_{1,1}, \ldots, w_{i,1,k}m_{1,k}\}, \ldots, \{w_{i,j,1}m_{j,1}, \ldots, w_{i,j,k}m_{j,k}\}\}$$

where $m_{x,y}$ represents the y^{th} related term of semantic type x and $w_{i,x,y}$ is the computed weight for $m_{x,y}$. Within each semantic type vector, the terms are sorted with terms of highest weight first. This way, a profile represents the terms most relevant to the topic T_i in a view that can be chosen by varying the semantic types accordingly.

2.4.2 Discovery Algorithms

Given a topic of interest *A*, a normal query of this term should return terms *B* that are related to *A*. If a user is interested in finding terms *C* that are indirectly connected to term *A* through a collection of terms *B*, a simple query will not be sufficient. Hypothesis discovery algorithms address such connections, which are both indirect and novel. An indirect relationship between two terms *A* and *C* is identified as one where *A* and *C* are connected through an intermediate term *B*, whereas a novel relationship is one where terms *A* and *C* occur in two different contexts that do not overlap. It is important to note the significant role of the *B* terms, as they act as conceptual links between *A* and *C*. Thus, the selection of the *B* terms is key to the accuracy and novelty of the association between the initial terms. We refer to the initial term as the *A* term, the intermediate terms as the *B* terms, and the second term as *C* term. The following are the open and closed hypothesis discovery algorithms proposed by [50].

Open Discovery Algorithm

Algorithm 2 shows the steps of the Open Discovery algorithm. The algorithm requires the following input: an initial topic A, a set of semantic types ST-B that are used to construct the profile of topic A, a set of semantic types ST-C that are used to select the C terms, and a value N that specifies the number of terms that are selected from the profile of the initial topic A.

Starting with topic A, a search in the data set is conducted to build a profile for A. This profile is built according to the semantic types specified in ST-B. All terms that are of a semantic type in ST-B and occur in the same document with topic A are added to the profile in the corresponding semantic type vector in the profile. The terms in the profile are sorted in descending order by weight within each semantic type vector. After the profile is populated with terms related to A, the size of the resulting profile will typically be large.

```
Input:
  (1) A topic of interest ($A$),
  (2) A set of UMLS semantic types of interest ($ST$-$B$) for selecting $B$ terms.
  (3) A set ($ST$-$C$) for selecting $C$ terms.
  (4) A parameter $N$ specifying the number of $B$ terms in the profile .
Method:
  Step 1: Conduct an appropriate PubMed search for topic $A$ and build its MeSH profile limited
          to the semantic types in $ST$-$B$. Call this profile $AP$.
  Step 2: For each semantic type in $ST$-$B$, select the $N$ top ranking MeSH terms from $AP$. These
          are designated the $B$ terms.
  Step 3: Conduct an independent PubMed search for each term $B_i$ and build its profile limited
          to the semantic types $ST$-$C$. Call these profiles $BP_i$.
  Step 4: Compute a final combined profile where the combined weight of a MeSH term is the
          sum of its weights in $BP_1$, $BP_2$, etc. Call this initial profile $CP$.
  Step 5: For each term $B_i$ in $CP$, if a MEDLINE search on topic $A$ AND $B_i$ returns non empty
          results, eliminate $B_i$ from $CP$.
Output:
  The MeSH terms in $CP$ ranked by combined weight, for each semantic type in $ST$-$C$. (These
  are the $C$ terms organized by semantic type and ranked by estimated potential.)
```

Algorithm 2: Open Discovery Algorithm [50]

In order to reduce the size of this profile, we select only the N first terms for each semantic type in the profile and discard the rest. These are the terms with the greatest weight values. Let these terms be the intermediate terms B_i. Once the B_i terms are specified, we conduct a search in the data set for each of these terms. The purpose of this step is to find all terms that are of semantic types ST-C and occur in the same document with the term B_i. The resulting profiles are called BP_i. The next step is to combine the BP_i profiles into one new profile called CP. This is achieved by combining all terms in all the semantic types accordingly. If a term occurs in more than one BP profile, its weights are summed and then the term is added to CP as one term and is assigned a weight value equivalent to the combined weights. To ensure that only novel relationships are returned, for each term B_i in CP, a search is conducted in the data set for: (B_i AND A). If the search returns documents in which B_i occurred with A, remove B_i from CP. The reason for this step is that the occurrence of B_i with A in the same document represents a direct relationship between the two terms. By filtering CP using the query mentioned in the previous paragraph, we now have a list of terms categorized by their semantic types and ranked by their weights. These terms are the C terms and the weights represent their estimated potential as linking terms.

The term weight can be measured in various ways, depending on several factors, such as the properties of the data set and the selection of the semantic types. An example of a method to calculate the weights is by using the term frequency value [50]. By experimenting with various methods and measuring the ranking accuracy, a suitable weight calculation scheme can be reached.

Many variations of the open discovery algorithm were used to develop automated discovery systems. Examples of such systems is the LitLinker system [61], which discovers interesting and novel connections between biomedical terms in the biomedical literature. Several studies involved the employment of the open discovery algorithm for forensic purposes. The *ATHENS* system, proposed in [49], mines the world wide web for novel relationships between two query topics. This system searches for topics that are related to the user-entered initial topic, but cannot be directly found using only the initial topic.

Closed Discovery Algorithm

The input for the closed discovery algorithm differs from the open algorithm in that it takes two initial topics A and B instead of one. The purpose behind the closed algorithm is to find a link between these two initial topics A and B. One set of semantic types ST-B is required, as well as a parameter N to specify the maximum number of terms to consider in the intermediate list of terms.

As shown in Algorithm 3, the first step is to build a profile for each of the input topics A and C, name them AP and AC, respectively. The profile is built by searching the data set for these topics, and only considering results that belong to one of the semantic types in ST-B. The terms are sorted in the profile by weight within each semantic type in descending order. After the profile is built its size is reduced by considering only the first N terms in

```
Input:
  (1) Two topics of interest denoted $A$ and $C$
  (2) A set of semantic types of interest ($ST$-$B$) for selecting intermediate $B$ terms.
  (3) A parameter $N$ specifying the number of terms to be considered in the combined profile
Method:
  Step 1: Conduct independent PubMed searches for $A$ and $C$. Build the $A$ and $C$ MeSH profiles.
          Call these profiles $AP$ and $AC$ respectively.
  Step 2: Limit $AP$ and $CP$ to component vectors for semantic types listed in $ST$-$B$. Also for
          each $ST$-$B$ semantic type retain only the highest weighted $P$ MeSH terms within $AP$
          and $CP$.
  Step 3: Compute a $B$ profile ($BP$) composed of terms $B_i$ that are in common between $AP$
          and $CP$. The weight of a MeSH term in $BP$ is the sum of its weights in $AP$ and $CP$.
  Step 4: Eliminate terms from $BP$ that do not represent novel associations. That is, if a search
          for $A$ AND $B_i$ AND $C$ returns non empty results, then eliminate $B_i$ from $BP$. The
          remaining MeSH terms are the potential $B$ concepts of interest. (This constraint may be
          relaxed if the user is also interested in connections that are not necessarily novel).
Output:
  A ranked list of MeSH terms for each semantic type in $ST$-$B$. Each term represents a potential
  conceptual connection between $A$ and $C$.
```

Algorithm 3: Closed Discovery Algorithm [50]

each semantic type. For each semantic type in ST-B, we select all the terms B_i that occur in both AP and AC, calculate the sum of their weights and add them to a new profile BP with their combined weight. The new profile BP must be built with the same semantic types $ST - B$, and each term should be added to the corresponding semantic type in BP. The final step is to eliminate from BP all terms that occur in the same document with both A and C. Thus, for every term B_i in BP, search the data set for (B_i AND A AND C). If this query returns non empty results, remove B_i from BP. The remaining terms in BP are used to define an indirect association between A and C as follows: A is linked to B_i which is linked to C, thus A and C are linked through B_i. The association paths are ranked by weight, where the most meaningful relationships are the ones with highest weight values.

This algorithm has been employed in various applications. For instance, in investigative applications, [27] proposed a tool to identify links between two initial topics. This was tested on the 9/11 counterterrorism commission report as a corpus. To rank the links are extracted, concept association graphs are employed to rank the links by relevance [28].

2.5 Criminal Network Analysis and Visualization

Criminal network analysis is a task that is related to organized crime. The analysis is conducted by gathering data from various incidents and sources related to the case under investigation and identifying the patterns, structures and flow of information in the criminal network.

A good start to the analysis process is to map the criminals' activities to a visualized graph that displays the associations between the criminals [36]. In this section we present the trends in criminal network analysis methods and tools. We also discuss the process of data visualization with a focus on the visualization of criminal networks.

2.5.1 Criminal Network Analysis

The social structure of criminal communities affects how they should be approached for analysis. Consequently, this impacts many law enforcement crime-fighting strategies. Recent studies have proposed the notion of adopting a social network paradigm for criminal community analysis as a replacement for the traditional hierarchical paradigm. This paradigm has been widely accepted by both the research community and law enforcement agencies [32].

In this section, we address Social Network Analysis (SNA) terminology and features. We refer to criminal networks and social networks interchangeably. A social network can be defined as a specific set of linkages among a defined set of persons with the additional property that the characteristics of these linkages as a whole may be used to interpret the social behavior of the person involved [32].

The main feature of social network analysis approaches is that the subjects of analysis are the characteristics of the relationships between the social entities rather than the characteristics of the entities themselves. Therefore, the actual and potential flow of resources between the individuals can be inferred by examining the characteristics of the relationships. Moreover, hierarchical relationships such as leadership and authority can also be encompassed by analyzing the relationships.

According to [58], extracting the structural properties of criminal networks such as subgroups and roles of individuals is an important goal of the networks analysis process. Generally, SNA involves trying to measure key attributes of the social network. In [32] the authors describe six basic attributes of social networks: network size, network composition, the multiplicity of the ties between the entities, the relative duration of ties, the network density, and the frequency of contact between the entities.

2.5.2 Data Visualization and Criminal Network Analysis

Data visualization [55] is the process of visually representing data. Large sets of data need to be explored and analyzed when the user has previous knowledge of the existence of valuable information within that data. Visualization methods provide a means to perform this analysis with a better chance of finding this valuable information. For this reason, data visualization is commonly associated with data mining. Some advantages of visual representations of data are:

- They allow for involvement of the human factor in the data exploration process
- They go hand in hand with the hypothesis generation process. The visualization of data provides the user with a perceptual view of the data in which they can develop a hypothesis
- They produce good results for data sets in which the data is of a varying and noisy nature
- They have been proven to be useful when the user is exploring a large database that he knows little about, and does not know exactly what he is looking for

- They provide a faster means to explore large amounts of data

Thus, applications that involve the management of large data sets which need to be analyzed have adopted data visualization methods. In [29] the author emphasizes the importance of integrating the human user in the data exploration process, that is to gain insight from the large amounts of data, and directly interact with it.

The data visualization process depends on the goal that the designer intends to achieve by the visualization system. Based on the task the visualization is required to perform, visualization systems can be categorized into three main categories. These are Exploratory Visualization, Confirmatory Visualization, and Production Visualization [10].

This is a rather general categorization, which classifies the visualization systems according to the task at hand. However, more technical details are involved in the visualization process. In [29], the author specifies three dimensions that should be addressed when designing the visualization. The dimensions are: the type of the input data to the visualization method, the visualization method, and the interaction and distortion techniques. The data types can be one-dimensional data, two-dimensional data, multi-dimensional data, text and hypertext, hierarchies and graphs, and algorithms and software.

The author also classifies visualization methods into five main categories. The first category is the standard 2D/3D displays, such as pie and bar charts. The second is techniques that employ geometrically transformed displays. Third, techniques that are icon-based. These displays map the features of the data to those of an icon, such as stick figures or little faces. The fourth class of techniques are the dense pixel displays. Finally, stacked displays visualize data partitioned in a hierarchical structure.

The third dimension to data visualization is the interaction and distortion technique used. This part is incorporated into the visualization process if the users are to interact with the visualization and guide the exploration and visualization process. Examples of these are interactive projection techniques, and interactive zooming techniques.

Any visualization may be designed to display one or more of the data types mentioned. Multiple visualization techniques and interaction and distortion techniques can be used within the same system. It is up to the designer of the visualization to determine which combinations provide a better means to explore the data.

When visualizing a criminal network a sociogram is commonly used. A sociogram displays the network as nodes and edges. This is the common graphical representation of social networks where the attributes of the relationships between social entities can be mapped as graphical attributes of the edges, i.e. color and shape properties.

Earlier in the criminology literature, Anacapa charts were used to display relationships between criminals that have been analyzed by constructing an association matrix [58]. Association matrices are built by mapping relationships between a set of criminals, which were gathered from raw data, to a matrix that is then visualized as an Anacapa chart. These charts were widely deployed by investigators for their effectiveness, yet they can become impractical when the size of the set of criminals is large, or in the case of a large data collection.

In [30] the visualization of criminal networks is classified into three generations:

- First generation: this is the generation where no automation of the analysis has been introduced yet. Hand-drawn Anacapa charts and maps with colored pins were used.
- Second generation: in this generation, network analysis tools are characterized by their focus on graphically representing raw data, leaving the analysis of the content to the user.
- Third generation: these tools are more focused on the implications and content of the data.

Second generation analysis tools include visual analysis tools that are able to graphically represent large sets of raw data. However, the actual analysis of the content of the data is left to the user. Most of the common currently used analysis tools belong to this generation. Some of the most famous ones are the Analyst's notebook [23] and XANALYS Link Explorer [33].

Third generation tools are the most intelligent in terms of deriving meaningful knowledge from the data. The COPLINK project [6] is an example of third generation tools. Its analysis component, the CrimeNet Explorer [59], is used to discover knowledge in criminal network data.

Chapter 3

Mining Criminal Networks

Law enforcement used to view organized crime as static groups of criminals operating through hierarchical structures led by "senior bosses" who control the criminal activities. Some crime-fighters assume that the gang is dismantled once the big bosses are down. However, studies on organized crime in the 1990s reveal that the assumed pyramid structure no longer holds [30]. Many criminal organizations adapt a more dynamic paradigm and operate in the form of a social network. Criminal organizations operating in the modern paradigm of collaborative network structures have proven to be more successful in their criminal business than the ones operating in the traditional hierarchical paradigm.

In this chapter, we present an algorithm to identify criminal communities from a set of text documents collected from the computers of some suspect(s) in a forensically sound manner. The text documents can include e-mails, chat logs, blogs, webpages, and any textual data. The objective of our proposed method is to assist the crime investigator in identifying the communities involved with the suspect and determining other members involved in the communities. In many cases, an investigator may have very few clues on the suspects or an organization at the early stage of investigation. In some other cases, an investigator may already know the members involved in an organization, but does not have concrete information on how the organization operates and relationships between the members. Thus, in addition to the discovery of communities, our method also measures the closeness among the members in a community and extracts useful information for crime investigation.

The remainder of this chapter is organized as follows. In section 3.1, we formally define the problem of communities discovery. In section 3.2, we discuss some methods employed to extract the entities from an input textual dataset. The general idea is to use a classifier to determine the person names from each text document in the dataset. Each identified person is a candidate to be an actor (a node) in the resulting criminal network. In section 3.3, we present an efficient method to extract the communities from the identified persons. Intuitively, a group of people is considered to be in the same community if their names frequently co-occur in some documents. In section 3.4, we discuss the extraction of useful information that can describe the relationship between members of a community and that is of value to crime investigation. We also present some methods to extract such information from the identified communities. The extracted information not only provides the investigator with more information about the communities, but it also serves as the labels in the resulting social networks to facilitate effective browsing.

3.1 The Problem of Criminal Communities Discovery

The problem of criminal communities discovery is to identify the hidden communities from a set of text documents that is obtained from one (or multiple) suspect's file systems. In this thesis, a text document is broadly defined to include e-mail messages, chat log sessions, webpages, blogs, and text files. Let D be a set of text documents. Let $U = \{p_1, \ldots, p_m\}$ denote the universe of all person names in D. Each document $d \in D$ is represented as a set of names such that $d \subseteq U$. Let $C \subseteq U$ be a set of person names called a *community*. A document d *contains* a community C if $C \subseteq d$. A community having k person names is a k*-community*. For example, $C = \{p_1, p_2, p_4\}$ is a 3-community. The support of a community C is the percentage of documents in D that contains C. A community C is a *prominent community* in a set of documents D if the support of C is greater than or equal to some user-specified minimum support threshold.

Definition 3.1.1 (Prominent community) Let D be a set of text documents. Let $support(C)$ be the number of documents in D that contain C, where $C \subseteq U$. A community C is *prominent* in D if $support(C) \geq min\_sup$, where the minimum support min_sup is a prespecified threshold. ■

Since the input set of text documents is obtained from a suspect's file system, we call the identified prominent communities to be the criminal communities related to the suspect. The problem of criminal communities discovery is formally defined as follows:

Definition 3.1.2 (Problem of criminal communities discovery) Let D be a set of text documents. Let min_sup be user-specified threshold $[0, 1]$. The *problem of criminal communities discovery* is to identify all prominent communities from D, i.e., all communities $\{C_1, \ldots, C_m\}$ that have $support(C_j) \geq min\_sup$ for $1 \leq j \leq m$, and to extract information that is useful for crime investigation. ■

3.2 Identifying Person Names

The first step is to identify the person names from a given set of text documents. There are many Named Entity Recognition (NER) tools and methods available in the market to identify person names. We briefly describe the Stanford NER [11], which is a promising tool to identify English names.

Named entities are recognized by using classification rules that are generated based on distinctive features of positive and negative examples of named entities. As mentioned in Chapter 2, recent NER studies make use of Supervised Machine Learning (SML) algorithms. These algorithms are used to study the features of examples of named entities in a large collection of annotated texts called the *training corpus* [39]. The rules are then constructed based on these extracted features for a given entity type such as person, organization, and location. After the set of rules are constructed , they are applied to tag the words in a text document in a *testing corpus*. The performance of the SML method is evaluated by recall, the number of entities identified from the total number of entities that actually exist in the testing corpus, and by precision of recognition, which is the number of identified entities that actually are entities from the total number of tagged entities the method was able to identify. If the set of classification rules yields a reasonably good performance in terms of recall and precision, the rules can be used to identify names in some future unseen text documents. Common SML methods employed for NER are Hidden Markov Models (HMM) [42], Maximum Entropy (ME) [35], and Conditional Random Fields (CRF) [31].

HMMs are statistical models used in machine learning. The assumption in this method is that the state of the system being modeled is not directly visible but the output dependent on the state is visible. Each state has a probability distribution over the possible outputs. This model is used as a generative model to construct a probabilistic tagging model to identify the part of speech for words or to find the names in texts that are not previously known [2]. A main feature of HMM is that they consider future and previous observations of the word. Future and previous observations, also called forward and backward observations, refer to the sequences of words in the text that occur before and after the word that is being processed. This is particulary useful for improving the matching rates. Systems based on HMM have produced accuracy rates up to 90% [57].

ME Models make use of the principle of maximum entropy which states that subject to known constraints (called testable information), the probability distribution which best represents the current state of knowledge is the one with the largest entropy. This principle can be employed to formulate a discriminative model that is based on classifying words in a text document according to a set of predetermined classes of named entities types such as persons and locations [45]. Some NER systems based on ME, such as the MENE system, are able to obtain accuracy levels up to 96% precision and 89% recall [4].

CRF is a probabilistic model that is also used in NER systems. CRF is a probabilistic model that is based on defining conditional probability distribution of a specific sequence of words over sequences of labels applied to these words in the training corpus. This conditional approach is discriminative, which is more desirable in a NER model as opposed to generic approaches [56]. The ME model is also discriminative, but it does not take into consideration future observations of the word being processed, whereas the CRF model does. This makes the CRF a combination of the best features of the previous two models.

The Stanford NER tool uses the CRF model to identify named entities. It then tags the entities into four categories: *person* for person names, *location* for locations such as cities countries or street addresses, *organization* for organization and company names, and *other* for all entities that do not fall into the above three categories. The Stanford NER only tags the word tokens into one of the above four categories, and does not do any other further processing. Thus, we implement the following preprocessing steps to increase the quality of the result:

1. Elimination of duplicates of the same name entity
2. Variants of a name referring to the same individual are merged into one name
3. Elimination of the false positives produced by the tagger.

Each document in the given data set is considered to be a transaction. For each document d, we apply the NER tagger, obtain a bag of entities R, and then remove any duplicates of the same name such that R contains a set of distinct names. Furthermore, variants of the same name are represented as one name. For instance, *John*, *J. Smith*, and *John Smith* are transformed into a common form: *John Smith*. Initials that are followed by only other initials and no names are discarded. This is due to the observation that, through the experiments, initials produced a significant amount of noise in the link identification step. In many cases, the full name of a person occurs in the same document with the initials of the same person. Since the full name is considered, discarding the initials in these cases will cause no information loss.

To eliminate false positives, our method allows the user to incorporate his/her domain knowledge. User interference to guide this step is crucial to improve the quality of the result in the subsequent communities extraction and information extraction steps. In certain cases, an individual can have different names. For instance, some suspects can have nicknames in a chat log and in the same session their real name is mentioned. According to the identification method, these will be considered as two individuals if there is no lexicographical resemblance between the two names. Our method allows the crime investigator to interfere and merge the two names to one in the rest of the analysis process. This will reduce the amount of redundancy in the identified communities, resulting in a more precise analysis.

3.3 Extracting Criminal Communities

Once the individuals are identified, the next step is to identify *all* prominent criminal communities. When two or more individuals interact frequently, from our analysis point of view, this indicates a strong relationship or linkage. Analyzing the strength of linkages is a key step for effective crime investigation. The strength of a linkage can be measured by either the absolute strength or the relevant strength. The absolute strength is measured by comparing the frequency of the interaction between the individuals to a fixed threshold. A linkage is strong if the number of interactions passes a given threshold; otherwise, the linkage is weak or there is no linkage. Alternatively, one can measure the strength of a linkage by relevance, which is a more flexible method and requires no prior knowledge about the data set. This is achieved by setting a threshold as percentage of the total number of text documents instead of a fixed value. A group is considered to be a community if its support is equal to or greater than a given percentage threshold.

A naive approach to identifying all prominent communities is to enumerate all possible communities and identify the prominent ones by counting the support of each community in D. Yet, in case the number of identified individuals $|U|$ is large, it is infeasible to enumerate all possible communities because there are $2^{|U|}$ possible communities. To efficiently extract all prominent communities from the set of identified individuals, we employ the Apriori algorithm [1] (also see section 2.3.1), which is originally designed to extract frequent patterns from transaction data.

Let $U = \{p_1, \ldots, p_m\}$ denote the universe of all person names in D. Each document $d \in D$ is represented as a set of names such that $d \subseteq U$. Refer to section 3.1 for the notations. Apriori is a level-wise iterative search algorithm that uses the prominent k-communities to explore the prominent $(k + 1)$-communities. First, the set of prominent 1-communities is found by scanning the documents D, accumulating the support count of each person name, and collecting the person name p's that has $support(p) \geq min\_sup$. The resulting prominent 1-communities are then used to find the prominent 2-communities, which are then used to find prominent 3-communities, and so on, until no more prominent k-communities can be found. The generation of prominent $(k + 1)$-communities from prominent k-communities is based on the following Apriori property.

Property 3.3.1 (Apriori property) All nonempty subsets of a prominent community must also be prominent because $support(C') \geq support(C)$ if $C' \subseteq C$. ■

By definition, a community C' is not prominent if $support(C') < min\_sup$. The above property implies that adding a person name p to a non-prominent community C' will never make it prominent. Thus, if a k-community C' is not prominent, then there is no need to generate $(k+1)$-community $C' \cup p$ because $C' \cup p$ must not be prominent. The strength of

```
Input: A set of text documents $D$.
Input: User-specified minimum support $min\_sup$.
Output: Sets of prominent communities $L_1, \ldots, L_k$ with $support(C_j)$ and $R(C_j)$.
Method:
 1: $L_1$ = all prominent 1-communities in $D$;
 2: for ($k = 2$; $L_{k-1} \neq \emptyset$; $k$++) do
 3:    $Candidates_k = L_{k-1} \bowtie L_{k-1}$;
 4:    for all community $C_j \in Candidates_k$ do
 5:       if $\exists s \in C_j$ such that $s \notin L_{k-1}$ then
 6:          $Candidates_k = Candidates_k - C_j$;
 7:       end if
 8:    end for
 9:    $support(C_j) = 0$ and $R(C_j) = \emptyset$ for every $C_j \in Candidates_k$;
10:    for all document $d_i \in D$ do
11:       for all $C_j \in Candidates_k$ do
12:          if $C_j \subseteq d_i$ then
13:             $support(C_j) = support(C_j) + 1$;
14:             $R(C_j) \leftarrow d_i$;
15:          end if
16:       end for
17:    end for
18:    $L_k = \{C_j \in Candidates_k \mid support(C_j) \geq min\_sup\}$;
19: end for
20: return $L_1, \ldots, L_k$ with $support(C_j)$ and $R(C_j)$;
```

Algorithm 4: Prominent Communities Discovery

the linkages among the members in a prominent community C is indicated by $support(C)$. The presented algorithm can identify *all* prominent communities by efficiently pruning all communities that cannot be prominent based on the Apriori property.

Algorithm 4 summarizes our Prominent Communities Discovery algorithm. The algorithm finds the prominent k-communities from the prominent ($k - 1$)-communities based on the Apriori property. The first step is to find the set of prominent 1-communities, denoted by L_1. This is achieved by scanning the data set once and counting the support count for each 1-community C_j. The support count for C_j, denoted by $support(C_j)$, is the number of documents containing C_j. L_1 contains all prominent 1-communities C_j with $support(C_j) \geq min\_sup$. The set of prominent 1-communities is then used to identify the set of candidate 2-communities, denoted by $Candidates_2$. Then the algorithm scans the database once to count the support of each candidate C_j in $Candidates_2$. All candidates C_j that satisfy $support(C_j) \geq min\_sup$ are prominent 2-communities, denoted by L_2. The algorithm repeats the process of generating L_k from L_{k-1} and stops if $Candidate_k$ is empty.

Two prominent $(k-1)$-communities can be joined together to form a candidate k-community only if their first $(k-2)$ person names are identical and their last $(k-1)$ person names are different. This operation is based on the Apriori property: A community C_k cannot be prominent if any of its subsets is not prominent. Thus, the only potential prominent communities of size k are those that are formulated by joining prominent $(k-1)$-communities. Lines 4-8 describe the procedure of removing candidates that contain at least one non-prominent $(k-1)$-community.

Lines 9-17 describe this procedure of scanning the database and obtaining the support count of each community C_j in $Candidates_k$. Each candidate community C_j is looked up in each document d_i in the document set. Once a match is found the value of $support(C_j)$ is incremented by 1 and the document d_i is added to the set $R(C_j)$. If $support(C_j)$ is larger than the user-specified minimum threshold min_sup, then C_j is added to L_k, the set of prominent k-communities with k members. The algorithm terminates when no more candidates can be generated or none of the candidate communities pass the min_sup threshold. The algorithm returns all prominent communities $L_1, \ldots, L_k$ with support counts and sets of associated documents for each prominent community.

Data Structure for Efficiency Improvement

The most expensive operation in the Prominent Communities Discovery algorithm is to scan the database for counting the supports of every community in $Canadidates_k$. We present a tree structure to improve the efficiency of the support counting process.

In each iteration of Line 2 in Algorithm 4, the number of candidate communities

$Candidates_k$ can be huge. We use a prefix tree structure to represents each candidate community as a tree path from root-to-leaf. The names within each community are sorted in lexicographical order. A prefix tree uses the same path to represent two communities that share the same first n members. Each leaf nodes keeps track of the support count of a community. To count the support, each record is scanned and matched with the prefix tree. The count at the leaf node is increased by one if the entire path is matched.

We describe the details as follows:

Tree Construction The construction of the tree follows the extended prefix-tree structure to store the candidate communities. It is important to note that the names within each community are sorted in lexicographical order. The following are the steps to construct the tree from the set $Candidates_k$.

1. The root node of the tree is set to the value NULL.
2. Each path from the root node to a leaf node represents a community $C_j \in Candidates_k$.
3. All nodes must not have duplicate child nodes. Thus, if two (or multiple) communities C_x and C_y have the same first n members, they are represented in the same branch up to the non-leaf node p_n. The node p_n will have two (or multiple) children: p_{n+1} of both C_x and C_y.

After the tree is constructed, the number of leaf nodes in the tree must be equivalent to $|Candidates_k|$.

Tree Traversal/Support Counting Once the tree is constructed with all its leaf nodes assigned a count value that is reset to zero, the data set is scanned to count the support as follows.

1. For each document in the data set, each person name will be looked up in the tree through a breadth-first scan. If a matching node is found in the children of

the root node it is marked to be searched in a subsequent lookup. The current lookup is terminated at this node for this level and will proceed to the next level in the tree, as explained in the next step.

2. To improve the breadth-first scan of the tree, the next level of nodes are not searched entirely. Only the children of the marked children nodes of the root (and the marked child nodes of those marked nodes) are searched. Once a match is found the node will be marked if it is a non-leaf node. If the matching node is a leaf, the count for that node is increased by one, and it is not marked, since it has no children.

3. For the next document in the data set, the marks on the tree nodes are cleared. However, the count value associated with the leaf nodes is not reset, instead it accumulates during the scan of the entire data set.

4. After the scan is completed, the counter associated with the leaf nodes is tested against the min_sup thresholds. If a leaf node's count is less than the threshold, the corresponding community C_j is removed from the set $Candidates_k$. Since the communities in $Candidates_k$ are sorted in lexicographical order, the correlation between the leaf nodes and the communities is trivial.

Example 3.3.1 (Prominent Communities Discovery) Consider a document set $D = \{d_1, \ldots, d_9\}$, where each document is represented as a vector of person names, as shown in Table 1. Suppose $min\_sup = 2$. The algorithm scans D once to identify all prominent 1-communities, i.e., the names that appear in at least 2 documents in D. During the scan, the algorithm counts not only the support for the prominent communities, but it also keeps track of all the documents that contain a community C, denoted by $R(C)$. In this example, the set of prominent 1-communities is:

$L_1 = \{\{John\}, \{Jenny\}, \{Kim\}, \{Mike\}, \{Alan\}\}$, with

$$support(\{John\}) = 6$$
$$support(\{Jenny\}) = 7$$
$$support(\{Kim\}) = 6$$
$$support(\{Mike\}) = 2$$
$$support(\{Alan\}) = 2$$

and

$$R(\{John\}) = \{d1, d4, d5, d7, d8, d9\}$$
$$R(\{Jenny\}) = \{d1, d2, d3, d4, d6, d8, d9\}$$
$$R(\{Kim\}) = \{d3, d5, d6, d7, d8, d9\}$$
$$R(\{Mike\}) = \{d2, d4\}$$
$$R(\{Alan\}) = \{d1, d8\}$$

Next, the algorithm generates the set of candidate 2-communities by a self join on L_1. This results in:

$$\begin{aligned} Candidates_2 = \{&\{John, Jenny\}, \{John, Kim\}, \{John, Mike\}, \{John, Alan\},\\ &\{Jenny, Kim\}, \{Jenny, Mike\}, \{Jenny, Alan\},\\ &\{Kim, Mike\}, \{Kim, Alan\}, \{Mike, Alan\}\}\end{aligned}$$

Then the algorithms scans the document set D to count the support of each candidate community in $Candidates_2$, and determines

$$\begin{aligned} L_2 = \{&\{John, Jenny\}, \{John, Kim\}, \{John, Alan\},\\ &\{Jenny, Kim\}, \{Jenny, Mike\}, \{Jenny, Alan\}\}\end{aligned}$$

Next, the algorithm generates the set of candidate 3-communities by a self join on L_2. This results in:

Document	Names in d_i
d_1	{John, Jenny, Alan}
d_2	{Jenny, Mike, Susan}
d_3	{Jenny, Kim}
d_4	{John, Jenny, Mike}
d_5	{John, Kim}
d_6	{Jenny, Kim}
d_7	{John, Kim}
d_8	{John, Jenny, Kim, Alan}
d_9	{John, Jenny, Kim}

Table 1: Document Vectors for Data Set D

$$Candidates_3 = \{\{John, Jenny, Kim\}, \{John, Jenny, Alan\}\}$$

Note that $\{John, Kim, Alan\}$, $\{Jenny, Kim, Mike\}$, $\{Jenny, Kim, Alan\}$, and $\{Jenny,$ $Mike,$ $Alan\}$ are not in $Candidates_3$ because some of their subset(s) do not exist in L_2 and they are pruned by Property 3.3.1. The candidate 3-communities have

$$support(\{John, Jenny, Kim\}) = 2,\ R(\{John, Jenny, Kim\}) = \{d_8, d_9\},$$
$$support(\{John, Jenny, Alan\}) = 2,\ R(\{John, Jenny, Alan\}) = \{d_1, d_8\}.$$

Thus,

$$L_3 = \{\{John, Jenny, Kim\}, \{John, Jenny, Alan\}\}.$$

The algorithm performs a self join on L_3, applies the Apriori property, and results in $Candidates_4 = \emptyset$. Finally, the algorithm terminates and returns the prominent communities L_1, L_2, and L_3 with $support(C_j)$ and $R(C_j)$ for each $C_j \in L_1 \cup L_2 \cup L_3$. ■

3.4 Retrieving Information from Criminal Communities

The next phase is to retrieve useful information for crime investigation, such as contact information, from the identified criminal communities (i.e., prominent communities). In the context of this thesis, a group of people are considered to be in the same prominent community if their names appear together frequently in some minimum percentage of text documents. Thus, the topics of the set of documents containing their names are the “reasons" bringing them together. By analyzing the content of the text documents containing the names of the community members, a crime investigator may obtain valuable clues that are useful for further investigation, especially in the early stages of the investigation. For instance, if a set of community member names are all contained in the some chat sessions, then summarizing the topics of the discussion can help the investigator infer the type of relationship the community members share. To facilitate crime investigation, we extract the following types of information from the set of documents $R(C_j)$ for each prominent community C_j:

1. Key topics
2. Names of other people not in C_j
3. Locations and addresses
4. Phone numbers
5. E-mail addresses
6. Website URLs

In some real-life cyber criminal cases, there could be thousands of identified individuals and hundreds of criminal communities. Even with a data mining software, an investigator may still find it difficult to cope with such a large volume of information. The summarized key topics from $R(C_j)$ can provide an investigator with an overview of each community and the related topics. The extracted key topics can be a link label when the communities are visualized on the screen as discussed in chapter 5. Some people names may appear only a few times in $R(C_j)$ but may not be frequent enough to be included as a member in C_j. Identifying these infrequent people names may lead to some new clues for investigation. Locations, addresses, and contact information, such as phone numbers and e-mail addresses, are valuable information for crime investigation because they may reveal other potential channels of communications among the community members. To extract the key topics, we employ an Open Text Summarizer (OTS) [5, 60]. The topic extraction involves four steps:

1. *Stop words removal:* Stop words are common words in a language that do not help differentiate the semantic of the text. Examples of stop words are "a", "the", "he", "them", and "who", etc.

2. *Stemming*: The next step is to conflate words that have common stems into one term since all these words usually share the same semantic. For instance, the words "compute," "computing," and "computer" are all stemmed to "comput." Although "comput" is not a valid English word, it does capture the meaning of compute.

3. *Counting stemmed terms*: Each document in $R(C_j)$ is now represented as a vector of stemmed terms. The next step is to count the frequency of each of the stemmed terms.

4. *Identifying key topics*: The key topics of a document set $R(C_j)$ are the top n frequent terms in $R(C_j)$, where n is a user-specified threshold.

To extract the city names, we search the documents for the cities in the GeoWorldMap database [24]. To extract other addresses, phone number, and e-mail addresses, we use regular expressions [12]. Note, the retrieved information is associated with C_j.

Other useful information may be extracted to further describe the relationship between the members of an identified prominent community, such as the duration of the relationship. The relationship duration is a key piece of information regarding the activity of members of a community. It is especially useful for investigators to provide them with a sense of a time line for the relationships that the communities share.

In order to specify the duration of the relationship between a criminal community identified in a set of textual documents we make use of the metadata of these documents. The metadata of a file is the data linked to this file by the hosting system upon creation of the document. We can define the *duration* of a relationship as all or some of the values of: (1) the starting date of the relationship, (2) the ending date, (3) and the amount of time the relationship lasted. We can identify the starting date of the relationship between members of a prominent community C_j, by the oldest of the dates attached to the documents in $R(C_j)$. The end date of the relationship is the most recent of the dates associated with these documents. The duration of the relationship is then calculated as the difference between the start and end dates.

Algorithm 5 describes the community information retrieval process. We extract the community information of every community C_j from the set of documents $R(C_j)$ that links the members of C_j together. The information extraction method identifies the $phones$, $emails$, $websites$, and $topics$ information from each document d_i in $R(C_j)$ as follows.

First, a document d_i is represented as a set of tokens $\{tok_1, \ldots, tok_y\}$. Next, stop words such as *the*, *a* , *it*, and *I* are removed as they do not contribute any meaning to the topics of the document. The next step is to match the tokens against the regular expressions of phone numbers, email addresses, locations and website URLs. Once a match is found, the token is added to the corresponding set. Then, the next token in d_i is processed. However, if the token does not match any of the regular expressions the token is passed to the summary topics extraction procedure described in Lines 20-24. The topics are identified by first stemming the tokens and then counting their frequency. The top n most frequent tokens are selected as the document summary topics. The Open Text Summarizer (OTS) method

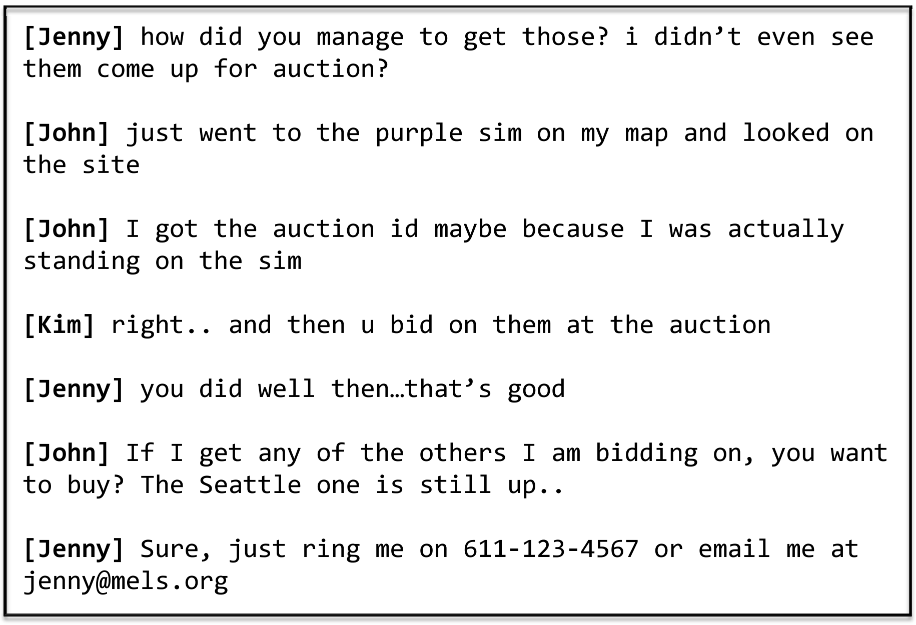
```
Input: Document $d$ as a string literal, $phone\_regex$, $email\_regex$, $website\_regex$, number of topics $n$, a list of location names $Loc$, a list of stop words $S$
Output: $phones$, $emails$, $locations$, and $topics$
Method:
1: initialize a set $V = \emptyset$;
2: split $d$ into $tokens = \{tok_1, \ldots, tok_y\}$;
3: $Count(tok_i) = 0$ for $1 \leq i \leq y$;
4: for tokens $tok_i \in tokens$ do
5:     if $tok_i \in S$ then
6:         $tokens = tokens - tok_i$;
7:     end if
8:     if $tok_i$ matches $phone\_regex$ then
9:         $phones \longleftarrow tok_i$;
10:        continue to the next token;
11:    end if
12:    if $tok_i$ matches $email\_regex$ then
13:        $emails \longleftarrow tok_i$;
14:        continue to the next token;
15:    end if
16:    if $tok_i \in Loc$ then
17:        $locations \longleftarrow tok_i$;
18:        continue to the next token;
19:    end if
20:    $tok_i = stem\_pre1(tok_i)$;
21:    $tok_i = stem\_post1(tok_i)$;
22:    $tok_i = stem\_manual(tok_i)$;
23:    $tok_i = stem\_post2(tok_i)$;
24:    $tok_i = stem\_synonym(tok_i)$;
25:    $V \longleftarrow tok_i$;
26:    $Count(tok_i)$++;
27: end for
28: sort $V$ in descending order by $Count(tok_i)$;
29: $topics$ = top $n$ most frequent tokens;
30: return $phones$, $emails$, $locations$, and $topics$;
```

Algorithm 5: Community Information Extraction

we apply for extracting the topics stems the tokens in five distinct steps: stemming punctuation prefixes, stemming punctuation suffixes, manual replacements, stemming inflection suffixes, and synonym replacements.

The first two stemming steps, *stem_pre1* and *stem_post1*, lines 20 and 21, remove any punctation and delimiter prefixes and suffixes from the token. This includes the removal of question marks, apostrophes, leading and trailing brackets, and parentheses from the token. Next, the manual replacement step, *stem_manual*, addresses tokens that are irregular verbs. In this step, the irregular verbs that are an inflection of their present tense are replaced by the present tense. For instance, the verb *taught* will be replaced with its present tense *teach*. In line 23, the *stem_post2* is applied to the tokens. This is responsible for stemming inflection suffixes. This type of stemming addresses regular verbs and inflected nouns. The suffixes of regular verbs, such as the "ing" in *trying*, are removed such that the token becomes *try*. Plural nouns, such as *books*, are reduced to the singular form *book*. Next, the synonyms replacing step is applied. Tokens that are synonyms for a more general term are replaced by the general topic. For example, the tokens *pleased*, *satisfied*, and *glad* are replaced by the term *happy*.

After the tokens are stemmed they are collected in a set V and their count is accumulated. We remove all terms that were reduced by the stemming process to words of less than three characters from this set. The user specified value n, determines the number of topics representing the document. The top n most frequent terms in the document d_i are the summary topics for d_i and are added to the set *topics*. It is important to note that words that are used as common criminal terminology or are related by meaning to a crime topic are added to the *topics* list regardless of their frequency. This is due to their relevance to the domain in interest. A user-specified list of criminal terminologies is obtained to facilitate this step. The information extraction method returns the sets of phone numbers, email addresses, website URLs, and summary topics for each document $d_i \in R(C_j)$. The last



```
[Jenny] how did you manage to get those? i didn’t even see
them come up for auction?

[John] just went to the purple sim on my map and looked on
the site

[John] I got the auction id maybe because I was actually
standing on the sim

[Kim] right.. and then u bid on them at the auction

[Jenny] you did well then…that’s good

[John] If I get any of the others I am bidding on, you want
to buy? The Seattle one is still up..

[Jenny] Sure, just ring me on 611-123-4567 or email me at
jenny@mels.org
```

Figure 2: Sample Chat Session

step is to combine the sets of information for each document in $R(C_j)$ into one set for each information type. The following example illustrates the information extraction procedure.

Person Names	Topics	Phones	Emails	Geographic Location
john jenny kim	auction sim bid id	611-123-1234	jenny@mels.org	seattle

Table 2: Link Related Information Retrieved from Chat Session

Example 3.4.1 (Information retrieval) Consider the chat session in Figure 2, which is a slightly edited version of the chat logs obtained from the Linden case [41]. The information retrieval process starts with processing the chat session that is in the form of a single string of characters. The first step is to tokenize the content of the string into word tokens by using white spaces as delimiter. The document (chat session) is now in the form of a vector of word tokens, as listed in the first column of Table 5.

Next, the tokens are matched against regular expressions of emails, websites, and phone numbers. In this example, the email $jenn@mels.org$ and phone number *611-123-4567* are added to the sets *emails* and *phones*, respectively. By matching the tokens against the list of location names, the city $Seattle$ is added to the set *locations*. The stemming proceeds for the remaining tokens by first removing the first type of prefixes and suffixes: punctuation and delimiter prefixes and suffixes. Examples of punctuation and delimiter prefixes are the brackets and parentheses, e.g., "{" and "(". After this step words such as *auction?* will become *auction* and *that's* and *didn't* are reduced to *that* and *did*, respectively. The word tokens of the chat session are shown in columns *stem pre1* and *stem post1* in Table 5, after removing the prefixes and suffixes, respectively. The next step, *stem_manual*, replaces irregular verbs with their present tense. For example, *did* is replaced with *do*. In Table 5, the column *stem manual* contains a list of tokens after they are replaced by the present tense verb.

Next, the tokens undergo the *stem_post2* process. For example, the word *actually* is replaced by *actual* and *bidding* is stemmed to *bid*. The complete list of words modified in this step is listed in column *stem post2* in Table 5. Then, the stemmed tokens are sorted in descending order by their frequency count. The top n tokens with highest frequency form the topics. By setting $n = 4$ for the chat session in Figure 2, we get the summary topics *auction*, *sim*, *bid*, and *id*. Upon the completion of the information retrieval steps in Algorithm 5, the criminal community found in the chat session excerpt in Figure 2 is described by the information in Table 2. ■

Chapter 4

Indirect Relationship Extraction

Our goal in the previous chapter was to identify the prominent criminal communities in documents residing on storage media seized from a suspect for evidence extraction. In this chapter, we examine the documents' contents to further analyze the suspects' social networks. In the case of analyzing criminal communities relationships, it is important to take into consideration hidden and indirect relationships. The direct relationships identified by the proposed method in the previous chapter may not be the only ones existing in the data. By generating hypotheses that are based on knowledge derived from our knowledge base, hidden and indirect relationships between the names may be discovered.

In many cases, due to limited time, investigators cannot possibly examine every document in the suspect's hard drive to search for evidential trails. As a result, some important information may be overlooked. Typically, investigators search the documents on the local hard drive for evidence using search tools such as FTK or Google Desktop [25], which rely mainly on the investigators' skills in constructing the queries. However, by using hypothesis discovery, the queries are constructed in an objective manner such that hidden evidential trails between names in the data set are extracted.

In this chapter, we propose a method to discover the evidential trails between a prominent community identified in a data set and other people in the data set who are not in the community. An evidential trail represents a relationship between the prominent community and other people through a common topic rather than co-occurrence. This trail is extracted as a chain of intermediate terms that link a community to a person. Thus, for a given a prominent community, C_j, and a person name, p, the indirect relationship discovery method identifies a chain of intermediate terms from the data set that links C_j with p. The length of the chain is limited by a user-specified threshold, denoted by d.

For instance, the evidential trail $C_j \rightarrow tickets \rightarrow Bob$, of length $d = 1$, indicates that the prominent community C_j shares an indirect relationship with Bob through the trail $tickets$ which is a common topic the community shares with Bob. This relationship can be considered valuable from an investigator's perspective since it indicates that the community and Bob were involved in some action related to tickets. However, this is not a direct relationship since Bob is not a member of the prominent community C_j, i.e., Bob and C_j never occurred in the same document.

The problem of indirect relationship discovery is to generate possible indirect relationships between the prominent communities and individuals in a set of documents. In a Hypothesis Generation (HG) problem, the process involves providing one or two initial terms, that can be person names, to the hypothesis generation algorithm. Let the names be *A* and *C*. The hypothesis generator searches for transitive relationships of the form: if *A* connects to *B*, and *B* connects to *C*, then *A* connects to *C* via *B*. In order to comply to this input requirement, we provide the hypothesis generator with two objects to link. The first is a prominent community and the second is the name of an individual that is not related to this community by any link. The intention is to search for all evidential trails that exist between the given prominent community and other names of people that occur in the data set. This provides for a more thorough examination of the relationships across the data set. Individuals that share a direct relationship, i.e., belong to the same prominent community, are by definition linked, and by additionally using HG, a more in-depth analysis of the relationships is performed. Thus the criminal network is defined in terms of direct links that resemble co-occurrence relationships (prominent communities), and indirect links that resemble relationships that are conceptual, and may or may not resemble an actual relationship. The relevance and accuracy of the results of the indirect relationships rely on the parameters deployed in ranking or selecting the terms which link the initial terms *A* and *C*. In our proposed method we selected the intermediate term *B* in a manner that ensures that the extracted evidential trails are the most meaningful ones across the data set.

The rest of this chapter is organized as follows. Section 4.1, formally defines the problem of indirect relationships discovery. Section 4.2 discusses the construction of profiles which are a basic structure required for the discovery algorithm. Section 4.3 explains the discovery algorithm used to find indirect relationships.

4.1 The Problem of Indirect Relationship Discovery

Let D be a set of documents and let C be a prominent community in D (see Definition 3.1.1). Let U be the set of distinct person names in D. The problem of indirect relationships discovery between the community C and an individual $p \in U$ is identifying a set of intermediate terms T that depict a conceptual relationship between C and p. Intuitively, we want to identify a set of terms that link a person to a prominent community. We can model the problem of extracting *indirect relationships* using the concept of *hypothesis generation* as follows.

Consider a criminal prominent community C and an individual p in D. Let $R(\cdot) \subset D$ denote the set of documents containing the enclosed argument where the enclosed argument is a community or a term. The problem of discovering hypothetical, conceptual relationships between community C and individuals in U is generating the tuple (C, p) from D, where p is a term with semantic type *person*. This tuple is generated for each person in U by identifying connecting terms t that conceptually link C and p. A term t represents an *indirect relationship* between C and p if t occurs in at least one document for both sets $R(C)$ and $R(p)$.

Definition 4.1.1 (Indirect Relationship) Let D be a set of documents. Let U be a set of distinct names in D. Let C be a prominent community and p be an individual where $C \subseteq U$ and $p \in (U - C)$. Let $R(C)$ and $R(p)$ be the sets of documents in D that contain C and p, respectively. An indirect relationship, of depth d, between C and p is defined by the tuple (C, p) generated by identifying the terms $[t_1, \ldots, t_d]$ such that,

1. $R(C) \bigcap R(p) = \emptyset$
2. $(t_1 \in R(C)) \bigwedge (t_d \in R(p))$
3. $(t_r \in R(t_{r-1})) \bigwedge (t_r \in R(t_{r+1}))$ for $1 < r < d$
4. $R(t_{r-1}) \bigcap R(t_{r+1}) = \emptyset$ for $1 < r < d$

■

According to this definition, an indirect relationship between a community C and an individual p through a set of intermediate terms $[t_1, \ldots, t_d]$ exists if the following conditions apply.

1. Community C and individual p do not occur in the same document, i.e., $R(C) \bigcap R(p) = \emptyset$.
2. The first term t_1 in the set of terms must occur in at least one document containing the community C and the last term t_d must occur in at least one document containing p, i.e. $(t_1 \in R(C)) \bigwedge (t_d \in R(p))$.
3. For indirect relationships of depth $d > 1$, each term t_r must occur in at least one document containing the previous term in the set t_{r-1}. Moreover, it must occur in at

least one document containing the next term in the set t_{r+1}, i.e., $(t_r \in R(t_{r-1})) \bigwedge (t_r \in R(t_{r+1}))$ for $1 < r < d$ and the terms t_{r-1} and t_{r+1} must not occur in the same document, i.e., $R(t_{r-1}) \bigcap R(t_{r+1}) = \emptyset$.

Given the definition of an indirect relationship, we now define the problem of discovering indirect relationships in a document set D.

Definition 4.1.2 (Indirect Relationship Discovery Problem) Let D be a set of documents. Let U denote the set of distinct person names in D. Let G be a set of prominent communities identified in D. The problem of identifying indirect relationships in D is identifying all the indirect relationships between a prominent community C and individuals $p_1, \ldots, p_m$ in U, for all the prominent communities $C \in G$. ■

4.2 Profile Construction

As explained in chapter 2, any term t in a data set D can be profiled by extracting interesting information about it from the textual content of documents in D. For example, if the document set is obtained from newswire documents, the profile of a topic such as *Microsoft* can be: *Corporation*, *Windows*, *Bill Gates*, and *Office*. In the same sense, the profile for a prominent community C existing in a hard drive can be $city$, $phone$, and $email$. This information can be retrieved from the documents in which the prominent community occurs. However, this information should not be chosen randomly because of the importance of the profile information in the hypothesis discovery process. If the profile information is too general, the discovered relationship is unlikely to be significant and the investigator may be overwhelmed by a large number of false hypotheses. Thus, data must only be added to the profile if it satisfies some prespecified constraints or conditions that are set to ensure the usefulness of this data.

The structure of a profile, as mentioned in section 2.4.1, is based on semantic types. The structure ensures that only a specific type of information is added to the profile. In the criminal network analysis context, we select semantic types that are significant to investigations. In particular, the following semantic types are selected: (1) summarized topics of the documents representing the prominent community's interactions, (2) other names of people mentioned in the documents with the prominent communities, (3) cities and locations, (4) email addresses, (5) phone numbers, and (6) website URLs. These semantic types are also used to identify the relationship between the members of prominent communities. For the profiles of the prominent communities, we use the same information that is retrieved for the prominent communities, as explained in section 3.4, for several reasons. First, it is less costly in the information extraction process. Second, these semantic types are extracted as forensically valuable information about the set of related individuals and consequently any other relationships that are found using this information are likely to be valuable as well. Within each semantic type in the profile of a prominent community, each term has a weight associated with it. In order to minimize the computations, we define the profiles for the prominent communities of maximal size, and combine all the profiles of the sub-communities.

Definition 4.2.1 (Profile) A profile for a prominent community C, $P(C)$, is defined by a collection of vectors V_{x_1}, V_{x_2}, …, V_{x_n}, where n denotes the number of semantic types considered. Each vector V_{x_i}, of length l_{x_i}, where l_{x_i} is the number of terms of semantic type x_i, is given by:

$$V_{x_i} = \begin{bmatrix} t_1, f_{x_i}(t_1) \\ t_2, f_{x_i}(t_2) \\ \vdots \\ t_{l_{x_i}}, f_{x_i}(t_{l_{x_i}}) \end{bmatrix}$$

where $f_{x_i}(t_j)$ is the weight of term t_j and is given by

$$f_{x_i}(t_j) = \frac{f'_{x_i}(t_j)}{\max_j f'_{x_i}(t_j)}$$

and

$$f'_{x_i}(t_j) = n_{x_i,t_j} * log(|D|/n_{t_j}),$$

where $|D|$ denotes the the number of documents in the document set D, n_{t_j} denotes the frequency of occurrence for term t_j in the document set D, and n_{x_i,t_j} denotes the frequency of term t_j of semantic type x_i in $R(C)$. ■

The prominent community identification algorithm, presented in Algorithm 4 (section 3.3), produces a relatively large set of prominent communities. This is because it provides the sets of prominent communities of each size $L_1, \ldots, L_k$ where k is the size of the largest prominent community found in the data set. Thus, in order specify which prominent communities are addressed in the indirect relationship discovery process, we create a set of the *maximal prominent communities*, denoted by G.

Definition 4.2.2 (Maximal Prominent Community) A prominent community C is a maximal prominent community if there is no prominent community C' such that $C' \supset C$. ■

The set of maximal prominent communities G is generated as follows. Starting from the set of prominent communities L_k, all the communities $C \in L_k$ are added to G and sorted in decreasing order by their $support(C)$. Next, for the remaining prominent communities C' in the sets $L_{k-1}, \ldots, L_2$, before adding them to G, we must ensure that they are not a subgroup of a community already in G. Thus, if C' happens to satisfy $C' \subset C$ where $C \in G$, then it is discarded. However, in cases where $support(C') > support(C)$, prior to discarding C' we set $R(C) = R(C') \bigcup R(C)$. Finally, we construct the profile $P(C)$ for each prominent community C (see definition 4.2.1).

Community/ Individual	Doc ID	Content
C=John, Jenny, Kim	d_1	**Jenny:** how did you manage to get those? i didn't even see them come up for auction? **John:** just went to the purple sim on my map and looked on the site **John:** I got the auction id maybe because I was actually standing on the sim **Kim:** right.. and then u bid on them at the auction **Jenny:** you did well then.. that's good **John:** If I get any of the others I bid on, you want to buy? The Seattle one is still up.. **Jenny:** Sure, just ring me on 611-123-4567 or email me at jenny@mels.org
	d_2	**Jenny:** can you do this only with purple land, or can you do it with any land that has an auction id? **Kim:** only with purple land **John:** If it goes without bids for long enough, they can release it into general population as first land
Sam	d_5	**Sam:** Enter the active auction number on the page
Bob	d_4	**Bob:** LIM00290 its for 2849 Rensdon Ave. Seattle, WA

Table 3: Document Sets for Prominent Communities and Individuals in Example 4.2.1

The profile construction process is illustrated in the following example.

Example 4.2.1 (Profile Construction) Consider the documents d_1 and d_2 (shown in Table 3) which contain the prominent community $C = \{John, Jenny, Kim\}$. In order to construct the profiles for this prominent community, we specify the semantic types to be $phones$, $cities$, $topics$, $websites$, $persons$ and $emails$. The profile construction process proceeds by first extracting all the information of types $phones$, $cities$, $topics$, and $emails$ from the community's document set $R(C) = \{d_1, d_2\}$. The value $f'(t)$ is calculated for each extracted term using the formula in definition 4.2.1. Table 4 shows the extracted terms for each semantic type, along with their weight, for the documents d_1 and d_2. Next, we combine the terms of a certain semantic type, e.g. *topics*, from all documents in the

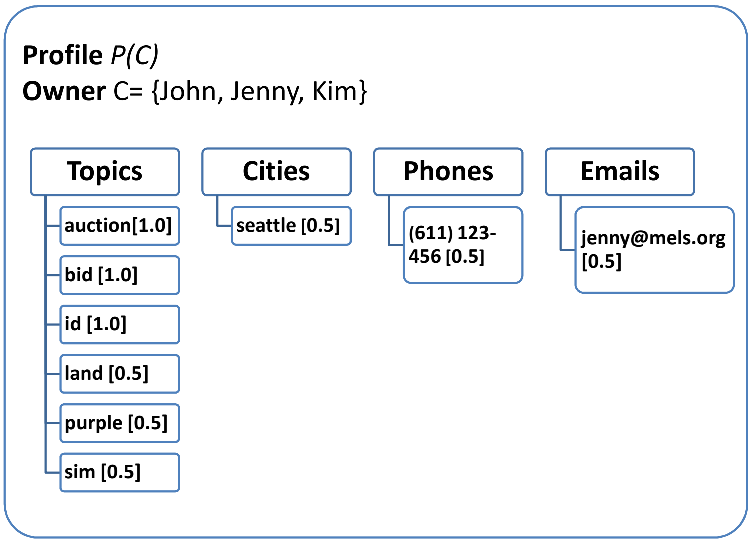


Figure 3: Example of a Profile

community's document set into one vector representing that same semantic type. When combining the terms into one vector, we ensure that the weights are first combined and then normalized as in definition 4.2.1. After the combining step in our example, the terms in V_{topics} are now *auction*, *bid*, *id*, *land*, *purple*, and *sim* with weights: $f(auction) = 1$, $f(bid) = 1$, $f(id) = 1$, $f(land) = 0.5$, and $f(purple) = 0.5$, and $f(sim) = 0.5$. Figure 3 shows the profile built for the prominent community $C = \{John, Jenny, Kim\}$. The profiles can now be processed by the discovery algorithm. ■

4.3 Indirect Relationships Discovery Algorithm

Given a prominent community C with profile $P(C)$, we propose the indirect relationships discovery algorithm to extract indirect relationships between the prominent community C and other individuals in the data set. This algorithm is a hybrid version of both the open and

d_i	x_1=Topics	x_2=Phones	x_3=Emails	x_4=Cities	x_5=Persons
d_1	auction [1.2] sim [0.6] bid [0.6] id [0.6]	(611)123-4567 [0.6]	jenny@mels.org [0.6]	seattle [1.2]	john [0.6] jenny [0.6] kim [0.6]
d_2	purple [0.6] land [0.6] bid [0.6] id [0.6]				john [0.6] jenny [0.6] kim [0.6]
d_3	auction [1.2] active [0.6] number [0.6] page [0.6]				sam [0.6]
d_4	LIM00290 [0.6] rendson [0.6]			seattle [1.2] WC [0.6]	bob [0.6]

Table 4: Extracted Terms of Specified Semantic Types from Documents

closed discovery algorithms described in section 2.4.2. The closed discovery requires two initial terms A and C, and generates hypothetical relationships between A and C through intermediate terms B. On the other hand, open discovery requires the entry of only one initial term, the B and C are provided by the algorithm. We propose a model that is open in the sense that it requires only one initial term to start the discovery process. However, the other end of the relationship must be the name of an individual from the data set.

Algorithm 6 shows the steps of the indirect relationship discovery algorithm. As mentioned above, this method is applied to detect indirect relationships between the prominent communities and other individuals in the data set. The method is applied for each community C_j in the previously defined set G. The algorithm requires the profile of the prominent community $P(C)$ as input, where at least one of its term vectors V_x is of a semantic type “persons". Both the number of intermediate terms, N, and the depth of the indirect relationship, d, are set by the user. If $d = 1$, for example, then the indirect relationship between community a and person c is through one connecting term, e.g. $a \rightarrow b \rightarrow c$. However, if $d = 2$, then the relationship is of the form $a \rightarrow b \rightarrow e \rightarrow c$.

The algorithm proceeds with the truncation of the term vectors, V_x, comprising the profile $P(C)$, by selecting the N top ranking terms in each V_x for all values of $x \in X$.

```
**Input:** Profile $P(C) = \{V_{x_1}, \ldots, V_{x_n}\}$ for community $C$ (see definition 4.2.1), Number of inter-
    mediate terms $N$, user-specified depth $d$
**Output:** Set of person names indirectly related to $C$ through $d$ intermediate terms
**Method:**
  $P = P(C)$
  **while** $d > 0$ **do**
    Let $B_{x_i}$ denote the $N$ top ranking terms in $V_{x_i}$ corresponding to $P$
    Construct profile $P(B_x[y])$, $x \in X$, $y = 1, \ldots, N$
    Combine profiles $P(B_x[y])$ into one profile $P(O)$, where the weight of a term in $P(O)$ is the
    sum of its weights in profiles $P(B_x[y])$, $y = 1, \ldots, N$.
    **for** each $t_j \in V_x$ of $P(O)$, $x \in X$ **do**
      Conduct a query ($t_j$ AND $C$) in $D$
      **if** result $\neq \emptyset$ **then**
        remove $t_j$ from $P(O)$
      **end if**
    **end for**
    $d = d - 1$
    $P = P(O)$
  **end while**
  **return** terms $O_x$ ranked by weight, for semantic type $x$="persons"
```
Algorithm 6: Indirect Relationship Discovery Algorithm

The new truncated vectors are called B_x for each semantic type x accordingly. Next, for each $x_i \in X$ we search the data set for each term $B_x[y]$, $y = 1, \ldots, N$ in order to build its profile. The constructed profiles are $P(B_x[1]), P(B_x[2]), \ldots, P(B_x[N])$. Now, a combined profile is computed where the combined weight of a term is the sum of its weights in each of $P(B_x[1]), P(B_x[2]), \ldots, P(B_x[N])$ in which it occurs. This combined profile is called $P(O)$ and is comprised of vectors V_x for each semantic type $x \in X$. For each term in t_j in the profile $P(O)$, if a search for (C AND t_j) returns a non empty set, the term t_j is removed from $P(O)$. If the depth is set to a value greater than 1, the algorithm iterates again using the value of the profile $P(O)$ produced from the previous iteration as the input profile for the next iteration. Finally, the method returns the terms in $P(O)$ for the semantic type *“persons"* ranked by combined weight and terminates.

Example 4.3.1 (Profile Construction) To illustrate the steps of the algorithm, consider

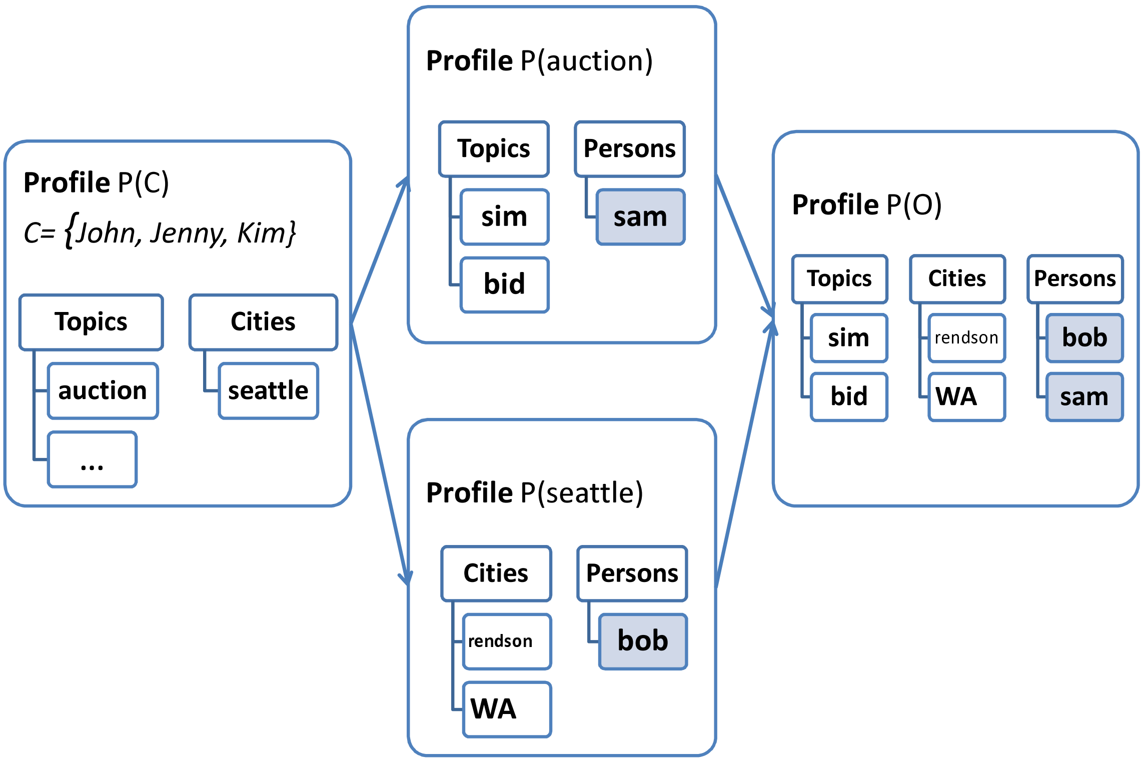


Figure 4: Indirect Relationship Discovery

the community C with profile $P(C)$ from Figure 3. The objective is to discover the possibility of a conceptual, indirect relationship between this prominent community and the individuals in the data set.

The algorithm starts with the profile $P(C)$ for the community $C = \{John, Jenny, Kim\}$. The parameter N specifies the number of terms that are considered in each of the term vectors of this prominent community's profile. Profiles for each term in the profile $P(C)$ are built and obtain the values shown in figure 4. Next, the profiles are combined into one profile $P(O)$ also shown in figure 4.

If $d = 1$, as it is in this example, the *"persons"* vectors in $P(O)$ are used to generate the

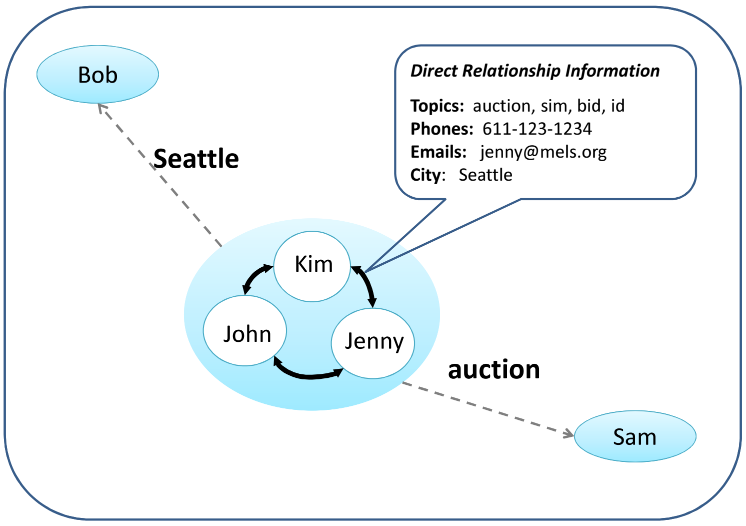


Figure 5: Direct Relationships (–) vs. Indirect Relationships (- -)

indirect relationships. For each name t_j in this vector the document set is searched for documents containing both t_j and C. In this example, the first lookup searches for documents containing all four names: *John, Jenny, Kim,* and *Sam*. If such a document does not exist, then it implies that *Sam* has no co-occurrence relationship with the prominent community. This generates an indirect relationship between the prominent community *John, Jenny, Kim,* and *Sam* through the topic term *auction*. The remaining relationships are extracted in the same manner. Figure 4 shows the final results of the discovery method when applied to this example. Figure 5 shows the prominent community sharing a direct relationship along with the hypothetical indirect relationships to individuals Bob and Sam. ■

Chapter 5

The Criminal Communities Mining System

The proposed methods for mining and analyzing social networks from a suspect's hard drive are implemented in the Criminal Communities Mining System (CCMS). We evaluated the performance of the proposed method in the system with real-life data sets. This chapter explains the capabilities and features of the modules comprising the system. First, we provide an overview of the system architecture and the integration with forensic analysis tools, followed by a description of the communities visualizer. Finally, we present the experimental results to illustrate the performance of the proposed methods in terms of capability of communities identification, efficiency, and scalability.

5.1 Architecture

Figure 6 provides an overview of the CCMS, which is composed of three major components: a content parser, a social network analyzer, and a community visualizer. These are all accessible through a user-friendly and interactive graphical user interface. The extracted information by the three components is stored in a database (knowledge base) at the back-end.

The first component, the content parser, is a file system crawler. This module is responsible for reading the documents from the storage media and parsing their content into a standard textual representation. Once processed by this component, the documents are in the proper format in which the required information extraction techniques can be applied to them.

The second component is the criminal communities mining module, which implements the communities discovery and information extraction algorithms described in chapters 3 and 4. Users can participate in guiding the analysis through the options provided in the user interface. The results of the analysis are fed to the visualization module, which will prepare the data for the display process. This module is designed with both modularity and scalability factors in mind.

The third component is the community visualizer, which provides a graphical view of the results of the identified prominent communities together with the community information. The visualization component is built on top of the Prefuse Library [18, 19] in order to map the communities to geometric shapes. The visualization module is designed to facilitate interactive data exploration.

5.2 Capabilities and Features

5.2.1 File Parsing

The first component is the file system content parser. This component is responsible for constructing and populating the system's dataset/database by reading the user-specified files. Through the user interface, the parser allows the user to specify the source of the documents that will be analyzed. The main functionality of this module is to crawl the

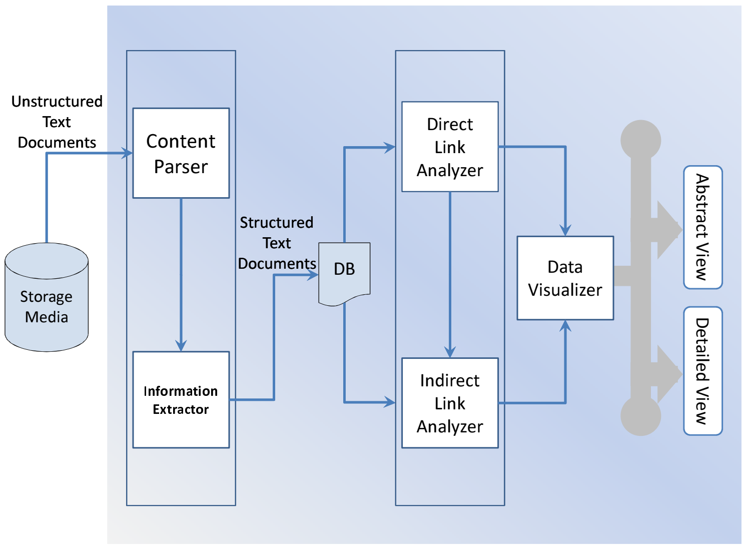


Figure 6: System Architecture

file system, parse the content, and extract the useful information. The preprocessing of the file system files is necessary to represent these files in a uniform structure that can be processed by the subsequent modules. The user can specify a specific type of document or a set of documents in which the analysis is to be performed, otherwise the system default is to analyze the entire file system that is available on the machine including hidden files, for all data file types. In many cases, forensic practitioners obtain more than one file system for a single case. The CCMS tool is capable of parsing the content of one or multiple file systems.

5.2.2 Criminal Communities Discovery

Criminal communities discovery is a major component in the system. This module is responsible for identifying the criminal communities that occur frequently in the documents parsed by the previous component. Moreover, it generates hypotheses for potential indirect relationships between individuals across the data set. These two analysis tasks are performed by two distinct modules in the analysis component. Each module allows the user to specify certain values needed for the analysis.

Subgroup Identification

One key feature of the analysis module is to provide some insight on the strength of the relationships among the members in a community. In the CCMS, closely related individuals are identified by the frequency of the interactions between them. If a community contains a group of individuals who interact more frequently with each other than they do with the rest of the community, they are identified as a sub-group within the community. Similarly, the individual with the highest frequency count in the data set is labeled as the leader in the community. The investigator has the opportunity to explore the structure of a given community by navigating the subgroups. Investigators can accomplish this by manipulating various preferences. The community size parameter allows investigators to view communities of a certain number of members. Another user-specified parameter is the min_sup, which is the minimum support count to form a prominent community (see section 3.3). This allows the user to narrow down the results by increasing the threshold, or expand the results by setting it to as low as 2.

Post-processing of Extracted Names

Extracted names that do not represent names of interest to the investigator can be manually eliminated, through a selection button, from the analysis process. This allows for the elimination of the false positives returned by the Named Entity Recognition (NER) component. The investigator can also manually eliminate irrelevant communities identified by the tool. The summary topics and additional information provided by the system allow the crime investigator to have a quick overview of the documents' contents without actually reading them. The document set can also be viewed by the user along with each community, enabling the user to fully retrieve a document that appears to be of interest.

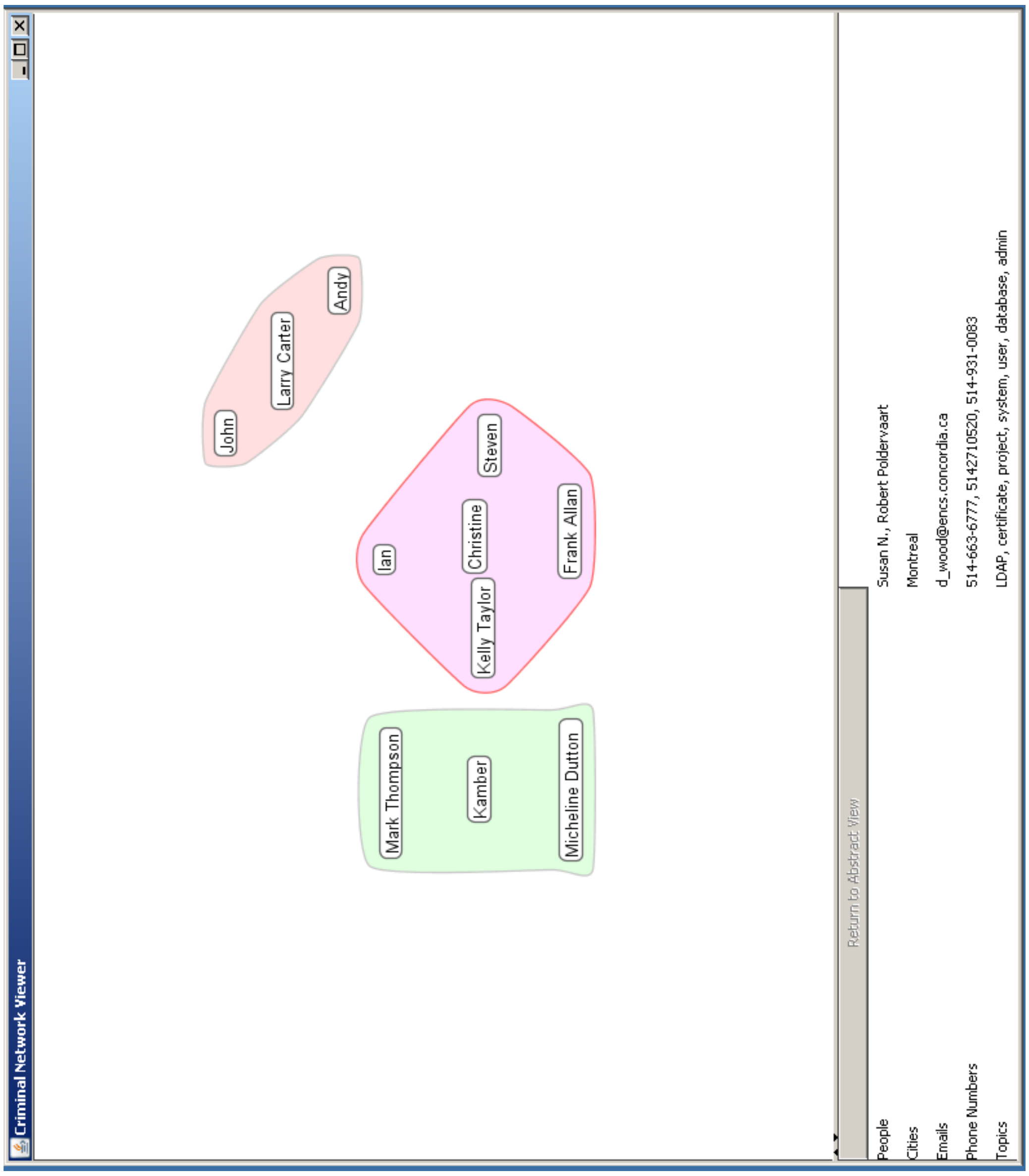
Indirect Relationship Discovery Depth

The second module generates the indirect relationships between the identified individuals from the documents. The relationship identification can be set by the user to various levels. If no links are found through a single intermediate topic, the user can increase the *depth* parameter, thus increasing the length of the evidential trail between the community and the individual.

5.2.3 Data Visualizer

The visualization component displays the findings from the analysis in a graphical representation. The visualizer is built on top of the Prefuse Library [18, 19], and is designed to facilitate the exploration of the analysis results and the file system behind it, if needed. Individuals are mapped to nodes and the relationships are visualized as edges, varying in color to discriminate co-occurrence relationships from hypothetical relationships. In this module, the user is provided with an abstract view and a detailed view.

In the abstract view, the user can view the criminal communities of a certain number of members, or simply view the maximal sizes of communities. Figure 7 depicts an abstract view example. The visualization provides the user with the frequency of each linked group of names. The user can make a quick assessment of the most dominant communities and names in the document set. In the detailed view, the user can select a specific community to view the potential hypothetical links to other names mentioned in the document set. The link label facilitates effective browsing and the community's profile provides a more complete picture of the community's behavior.



People	Susan N., Robert Poldervaart
Cities	Montreal
Emails	d_wood@encs.concordia.ca
Phone Numbers	514-663-6777, 5142710520, 514-931-0083
Topics	LDAP, certificate, project, system, user, database, admin

Figure 7: Criminal Communities Analysis - Abstract View

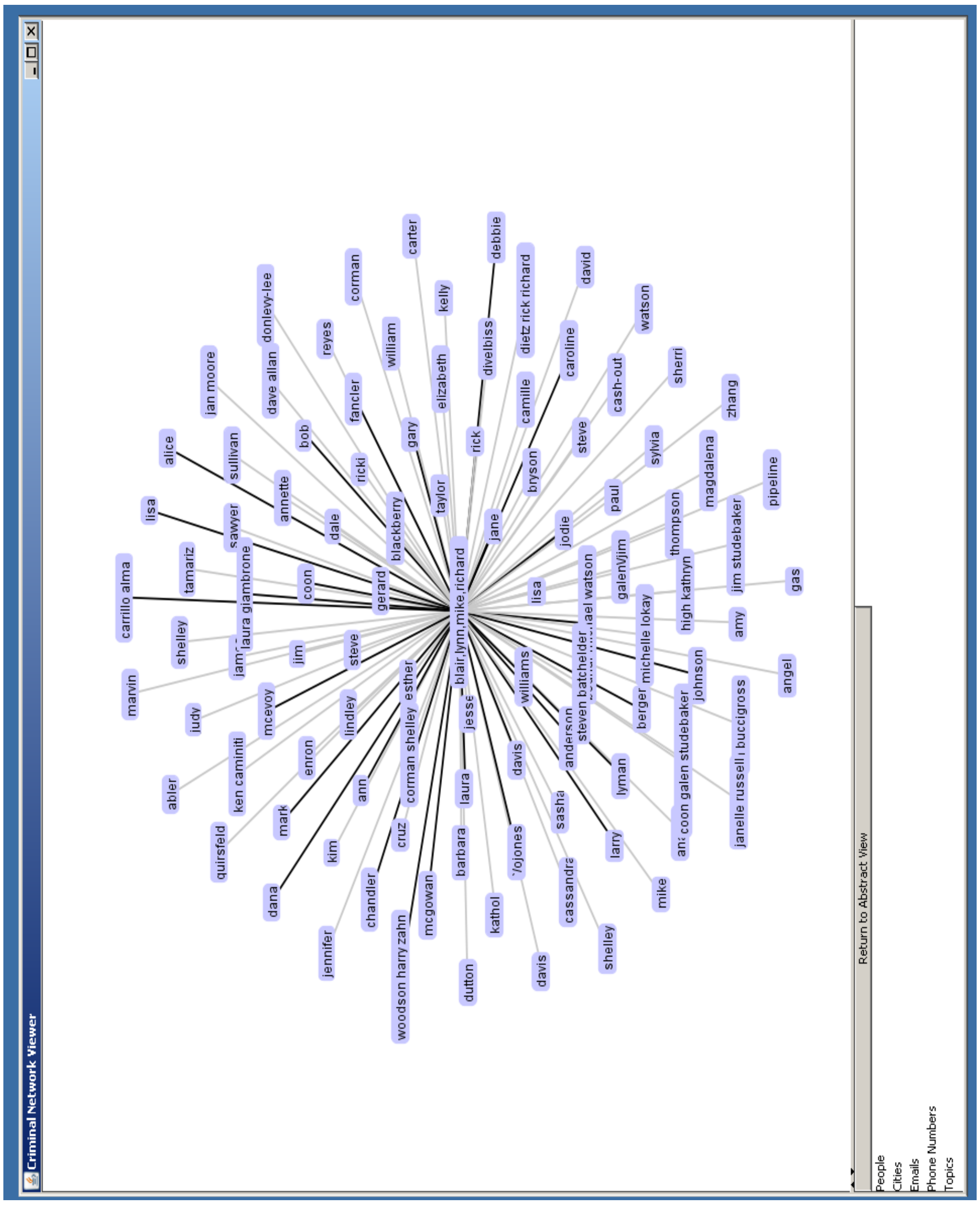


Figure 8: Criminal Communities Analysis - Detailed View

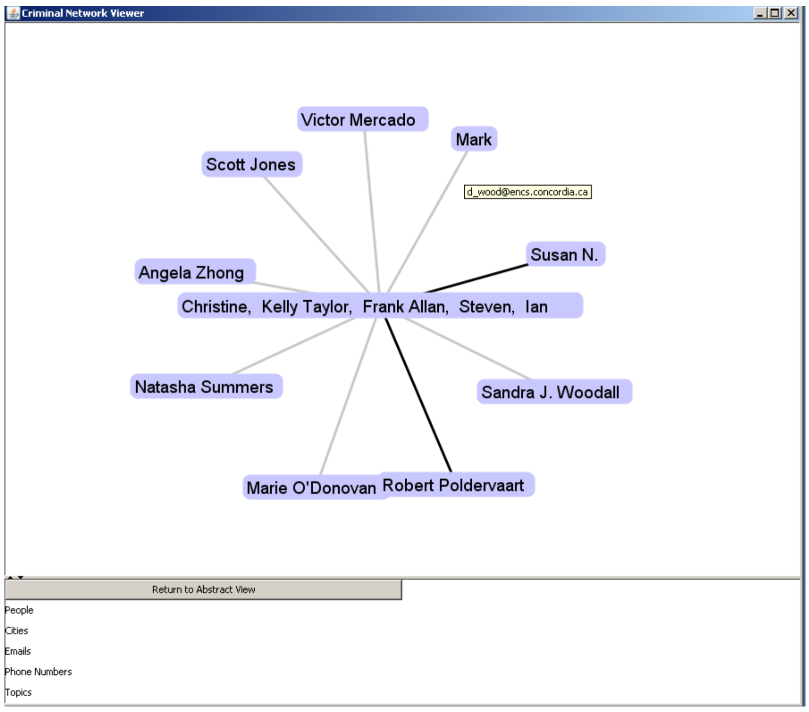


Figure 9: Criminal Communities Analysis - Detailed View with Hint

Figures 8 and 9 show examples for the detailed view of the hypothetical indirect links along with the direct links. The intermediate term(s) linking the individual to the prominent community is displayed as a hint that appears when the investigator hovers over the edge linking the individual to the community. The user can navigate between the different views by selecting the required view, and focus on a particular community by clicking on it.

5.2.4 Efficiency and Scalability

This section describes our experimental results. We use a real-life data set to evaluate the effectiveness and scalability of our proposed methods. The data set is obtained from three machines of a graduate student at Concordia University. The documents are selected from the standard *My Documents* directory on Microsoft Windows. The content of the documents is mainly related to research, tutoring, and course work topics. These include various file types, ranging from Acrobat PDF files to Microsoft PowerPoint slides, from email messages to standard Office documents.

In order to evaluate the effectiveness of our proposed approach, we measure the precision of both the prominent community discovery and indirect relationship discovery modules. By manually inspecting the resulting prominent communities shown in Figure 7, that were obtained from the experiment data described above, we compare them with our prior knowledge of the actual communities of the owner of the machine. The system identified a prominent community comprised of *Christine*, *Kelly*, *Frank*, *Steven*, and *Ian*. The relationship information provided with this community suggested that the documents are related to the topics *LDAP*, *system*, and *project*. Some phone numbers and emails are also correctly identified. This group represents a group of students working on a team project that involved the design of a Lightweight Directory Access Protocol (LDAP) server.

To evaluate the impact of the minimum support threshold on the number of prominent communities, we apply the prominent community discovery method n times, $5 \leq n \leq 12$ on a single set of 350 files over n different min_sup values. Figure 10 shows the number of prominent communities for the minimum support threshold from 5 to 12. As the minimum support threshold increases, the number of prominent communities quickly decreases because the number of documents sharing *all* members in a community decreases very quickly.

Next, we evaluate the scalability of our proposed methods by measuring the runtime required for each method on data sets of various sizes. The proposed communities discovery, information extraction, and visualization modules are applied to data sets starting from 0.25GB up to 2GB with $min\_sup = 3$ and $depth = 1$. Figure 11 shows the runtime of our proposed methods with respect to the size of the document set from 0.25GB to 2GB. In general, the total runtime increases as the data size increases. For instance, the program

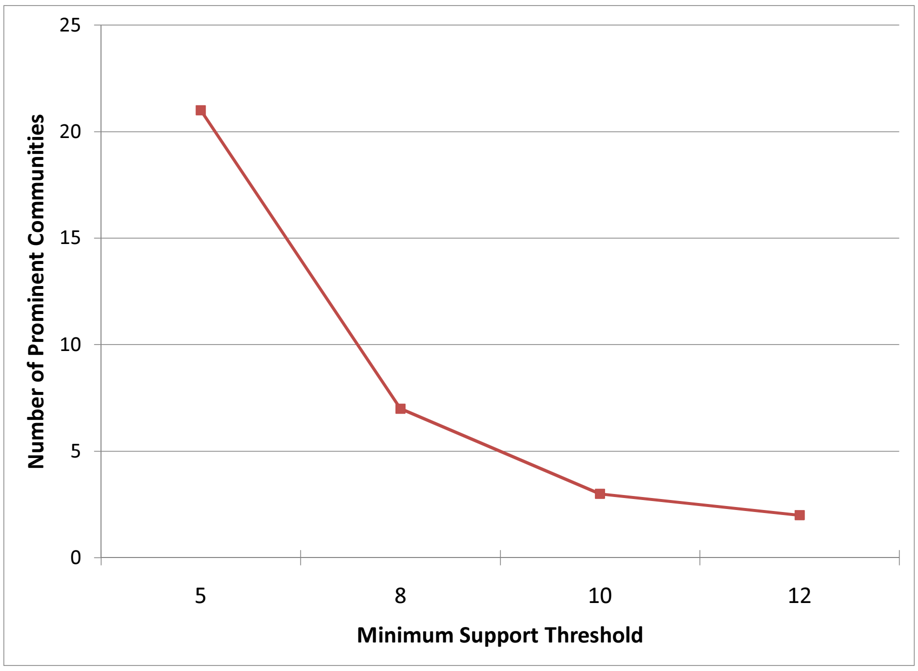


Figure 10: Number of Prominent Communities vs. Minimum Support Threshold

takes 154.69 seconds to complete these three tasks for 2GB of data, excluding the the time spent on reading the document files from the hard drive. The program takes 474 seconds to read 25,000 files with a total size of 1GB from a hard drive.

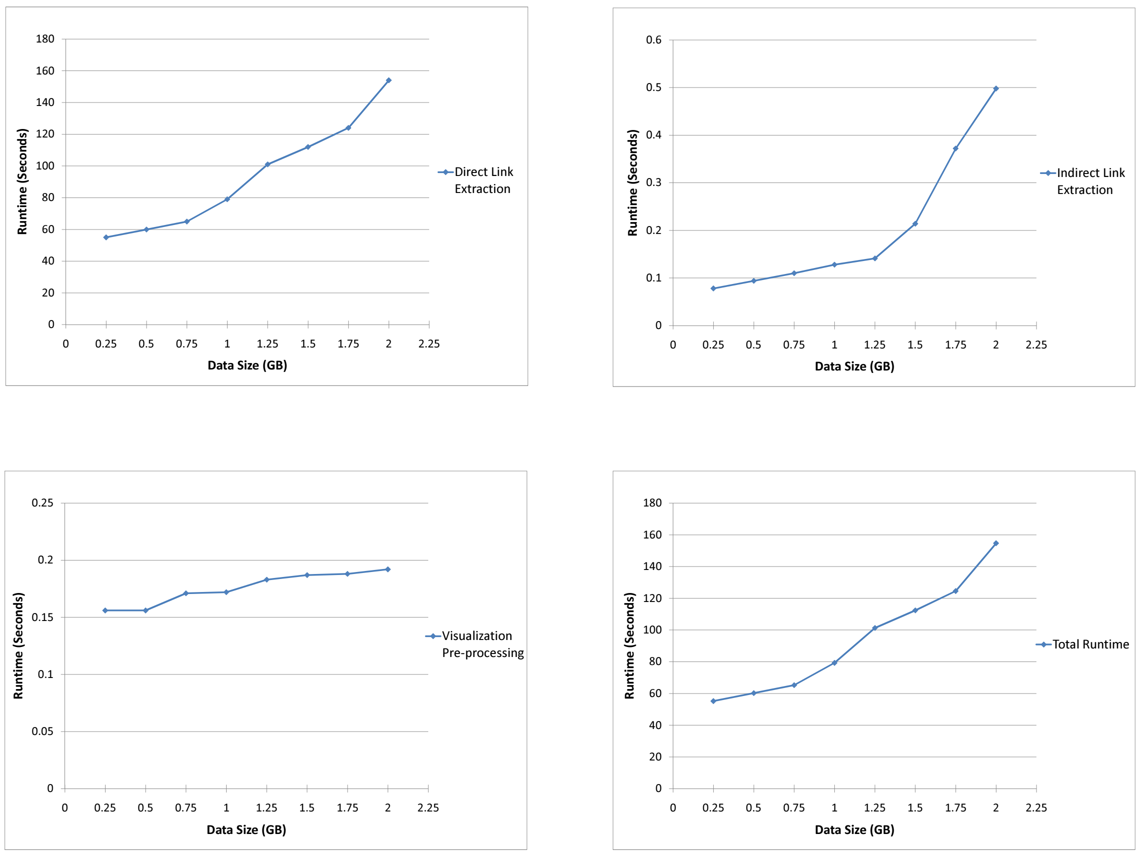


Figure 11: Scalability: Runtime vs. Data Size

Chapter 6

Conclusion and Future Work

6.1 Summary and Conclusions

In this thesis, we have proposed an approach to mine and analyze criminal networks in a collection of text documents. The results of performing criminal network analysis depend on the data that is available for the analysis and on the ability of the analysis method to identify the characteristics of the relationships between the actors in the network. However, obtaining data about social individuals in the case of criminal networks faces a major challenge, which is the scarcity of reliable data about the criminal individuals. In most cases, forensically collected data contains valuable information about a suspect's social network. Typically, investigators try to find this information by manually inspecting the documents using forensic tools' queries and other basic search features. Once gathered, the information is then manually entered into the law enforcement database. To the best of our knowledge, there are no existing tools that provide methods to automatically extract social networks from raw documents, add them to a knowledge base and then analyze them. Our proposed method aims to address two problems: the first is the task of mining raw texts to identify the criminal networks within these texts. Second, we perform criminal network analysis on these networks and provide a visual representation of their interactions and associations.

By exploring various Information Extraction (IE) methods, the application of Named Entity Recognition (NER) tools have proven to be effective in identifying members of criminal communities. Text summarization techniques are also beneficial in identifying key topics which can label relationships between members of a criminal community. Moreover, data mining algorithms, such as the Apriori algorithm designed for mining frequent patterns, can efficiently identify criminal communities that are related by co-occurrence. Extracting communities using this algorithm is beneficial in many ways. One of its merits is that it is able to identify criminal communities as well as the interlinked subgroups that comprise them. Thus, the structure of the network is identified in a more fine-grained level, which is a key requirement in criminal network analysis. Another merit of this approach is that it maintains a structure that facilitates an important feature applied in the visualization process. This feature allows for viewing the criminal network with different levels of abstraction.

Additionally, our proposed discovery method provides the investigators with a domain specific link analyzer for possible indirect links between a criminal community and other actors in the data set where the relationship is through intermediate terms. The contact information and document topics are used as the intermediate topics. By selecting terms that are important in investigations as the intermediate linking terms, we are able to ensure that the extracted indirect relationships are the most meaningful ones to the investigator.

The proposed approach, when implemented, produces reliable results and performs in a scalable and efficient manner. By automatically identifying the prominent communities residing on the hard drive and allowing the investigator to explore the attributes and information about the individuals and the links in a simplified visual manner, the efficiency of forensic examinations can be immensely increased. The indirect links are discovered by the system based on information that has been already found in the data, leaving the user with only the task of rationalizing whether the evidential trail is one of interest to him or not. The system has been designed to provide the investigator with the results of the analysis in a visual representation that allows for quick and efficient exploration. The analysis process can be guided by the experience and knowledge of the investigators to either provide a more abstract or detailed analysis of the networks.

6.2 Future Work

The following is a summary of the future work for the approach proposed in this thesis. In our system we are using the Stanford NER developed by Stanford in order to identify person names in the textual content. One problem is that the Stanford NER is trained on a data set obtained from American and British newswire, which relatively limits its performance to the two domains. However, since names on hard drives are mostly found in documents with informal structure, such as email messages and chat logs, the textual content we address is far less structured than newswire documents, which are well structured and written formally. The names found in forensically collected data differ than other more formal contexts in the following:

- They are usually shorter versions of names (nick names, initials) rather than complete names
- They are rarely preceded by titles such as Mr., Dr, etc.
- In this context, fun nick names and gang specific names that do not occur in name dictionaries are used
- Normal name writing conventions such as a leading capital letter are mostly ignored
- The same person can be referred to by more than one name

- Proper punctuation is either ignored or not present because of the format of the documents. For instance, in chat logs different names are not separated by punctuation marks which makes it difficult to infer the end of a single name.

Thus, in order to enhance the accuracy of our system, a forensic-customized named entity recognition model should be designed, developed, and trained.

For the criminal community information extraction, our approach involves some pieces of information that are of evidentiary significance. Studying the extraction methods for more types of forensically significant information such as street addresses, credit card numbers, names of banks, IP addresses, and any other types of data that are useful for investigations can improve the performance of this tool on many levels:

- Identifying the information related to social groups provides the forensic examiners with more options to select what type of data they are interested in viewing. Moreover, in the profiles that are built for the discovery algorithm, more semantic types allow for better fine-tuning and for more experiments to be conducted for finding evidence trails.

- Once more data types can be extracted, more information is available for analysis research. Consequently, additional analysis approaches can emerge and be tested.

Different patterns, for the indirect relationship discovery, can be found by trying different combinations of semantic types for the intermediate terms. Our indirect relationship discovery algorithm can be extended in order to experiment with different semantic types and different values for the depth and intermediate terms.

The CCMS can be tested on an actual data set that is obtained from law enforcement agencies. Experts can verify the results of the analysis, and evaluate the accuracy of the identified communities and evidentiary significance of the extracted information.

Bibliography

[1] R. Agrawal, T. Imieliński, and A. Swami. Mining association rules between sets of items in large databases. *SIGMOD Rec.*, 22(2):207–216, 1993.

[2] D. M. Bikel, S. Miller, R. Schwartz, and R. Weischedel. Nymble: a high-performance learning name-finder. In *Proceedings of the fifth conference on Applied natural language processing*, pages 194–201, Morristown, NJ, USA, 1997. Association for Computational Linguistics.

[3] J. B. Bocca, M. Jarke, and C. Zaniolo, editors. *Fast Algorithms for Mining Association Rules*. Morgan Kaufmann, 1994.

[4] A. Borthwick, J. Sterling, E. Agichtein, and R. Grishman. Nyu: Description of the mene named entity system as used in muc-7. In *Proceedings of the Seventh Message Understanding Conference (MUC-7)*, 1998.

[5] O. Boydell and B. Smyth. From social bookmarking to social summarization: an experiment in community-based summary generation. In *IUI '07: Proceedings of the 12th international conference on Intelligent user interfaces*, pages 42–51, New York, NY, USA, 2007. ACM.

[6] H. Chen, D. Zeng, H. Atabakhsh, W. Wyzga, and J. Schroeder. Coplink: managing law enforcement data and knowledge. *Commun. ACM*, 46(1):28–34, 2003.

[7] University of Illinois at Urbana Champaign Cognitive Computation Group. Illinois named entity tagger. http://l2r.cs.uiuc.edu/ cogcomp/asoftware.php?skey=FLBJNE.

[8] AccessData Corp. Forensic toolkit (ftk). http://www.accessdata.com/forensictoolkit.html.

[9] D. K. Evans, J. L. Klavans, and K. R. McKeown. Columbia newsblaster: multilingual news summarization on the web. In *HLT-NAACL '04: Demonstration Papers at HLT-NAACL 2004*, pages 1–4, Morristown, NJ, USA, 2004. Association for Computational Linguistics.

[10] U. Fayyad, G. Grinstein, and Wierse. A. *Information Visualization in Data Mining and Knowledge Discovery (The Morgan Kaufmann Series in Data Management Systems)*. Morgan Kaufmann, 2001.

[11] J. R. Finkel, T. Grenager, and C. Manning. Incorporating non-local information into information extraction systems by gibbs sampling. In *ACL '05: Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 363–370, Morristown, NJ, USA, 2005. Association for Computational Linguistics.

[12] J. E. F. Friedl. *Mastering Regular Expressions*. O'Reilly Media, 3 edition, August 2006.

[13] The Stanford Natural Language Processing Group. Stanford named entity recognizer (ner). http://nlp.stanford.edu/software/CRF-NER.shtml.

[14] U. Hahn and I. Mani. The challenges of automatic summarization. *Computer*, 33:29–36, 2000.

[15] J. Han and M. Kamber. *Data Mining. Concepts and Techniques*. Morgan Kaufmann, 2nd ed. edition, 2006.

[16] J. Han, J. Pei, Y. Yin, and R. Mao. Mining frequent patterns without candidate generation: A frequent-pattern tree approach. *Data Mining and Knowledge Discovery*, 8(1):53–87, January 2004.

[17] M. A. Hearst. Untangling text data mining. In *Proceedings of the 37th annual meeting of the Association for Computational Linguistics on Computational Linguistics*, pages 3–10, Morristown, NJ, USA, 1999. Association for Computational Linguistics.

[18] J. Heer, S. K. Card, and J. A. Landay. Prefuse information visualization toolkit. http://prefuse.org/.

[19] J. Heer, S. K. Card, and J. A. Landay. prefuse: a toolkit for interactive information visualization. In *CHI '05: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 421–430, New York, NY, USA, 2005. ACM.

[20] E. Hovy and C. Lin. Automated text summarization and the summarist system. In *In Proceedings of the TIPSTER Text Program*, 1998.

[21] E. Hovy and C. Lin. Automated text summarization in summarist, 1999.

[22] R. C. Hulst. Introduction to social network analysis (sna) as an investigative tool. *Trends in Organized Crime*, 12:101–121, 2009.

[23] i2 Group. Analyst's notebook. http://www.i2group.com.

[24] Geobytes Inc. Geoworldmap. http://www.geobytes.com/.

[25] Google Inc. Google desktop. http://desktop.google.com/.

[26] Guidance Software Inc. Encase forensics. http://www.guidancesoftware.com.

[27] W. Jin, R. K. Srihari, and H. Hay Ho. A text mining model for hypothesis generation. In *Proceedings of the 19th IEEE International Conference on Tools with Artificial Intelligence ICTAI 2007*, pages 156–162, 2007.

[28] W. Jin, R. K. Srihari, H. Hay Ho, and X. Wu. Improving knowledge discovery in document collections through combining text retrieval and link analysis techniques. In *Proceedings of Seventh IEEE International Conference on Data Mining ICDM 2007*, pages 193–202, 2007.

[29] D. A. Keim. Information visualization and visual data mining. *IEEE Transactions on Visualization and Computer Graphics*, 8:1–8, 2002.

[30] P. Klerks and E. Smeets. The network paradigm applied to criminal organizations: Theoretical nitpicking or a relevant doctrine for investigators? recent developments in the netherlands. *Connections*, 24:53–65, 2001.

[31] J. D. Lafferty, A. McCallum, and F. C. N. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *ICML '01: Proceedings of the Eighteenth International Conference on Machine Learning*, pages 282–289, San Francisco, CA, USA, 2001. Morgan Kaufmann Publishers Inc.

[32] B. Leighton. The Community Concept in Criminology: Toward a Social Network Approach. *Journal of Research in Crime and Delinquency*, 25(4):351–374, 1988.

[33] Xanalys Limited. Xanalys link explorer. http://www.xanalys.com/solutions/linkexplorer.html.

[34] S. Liu. Experiences and reflections on text summarization tools. *International Journal of Computational Intelligence Systems*, 2-3:202–218, 2009.

[35] A. McCallum, D. Freitag, and F. C. N. Pereira. Maximum entropy markov models for information extraction and segmentation. In *ICML '00: Proceedings of the Seventeenth International Conference on Machine Learning*, pages 591–598, San Francisco, CA, USA, 2000. Morgan Kaufmann Publishers Inc.

[36] J. Mena. *Investigative Data Mining for Security and Criminal Detection*. Butterworth-Heinemann, Newton, MA, USA, 2002.

[37] G. A. Miller. Wordnet - about us. WordNet. Princeton University, 2009. http://wordnet.princeton.edu.

[38] M. F. Moens. *Information Extraction: Algorithms and Prospects in a Retrieval Context (The Information Retrieval Series)*. Springer, 1 edition, October 2006.

[39] D. Nadeau and S. Sekine. A survey of named entity recognition and classification. *Lingvisticae Investigationes*, 30:3–26(24), 2007.

[40] G. Palmer. A road map for digital forensic research. Report From the First Digital Forensic Research Workshop (DFRWS), August 2001. http://www.dfrws.org/2001/dfrws-rm-final.pdf.

[41] Pribanic Pribanic and Archinaco. Evans, spencer & carter v. linden labs. http://virtuallanddispute.com/.

[42] L. R. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257 –286, feb 1989.

[43] D. Radev, T. Allison, S. Blair-Goldensohn, J. Blitzer, A. Çelebi, S. Dimitrov, E. Drabek, A. Hakim, W. Lam, D. Liu, J. Otterbacher, H. Qi, H. Saggion, S. Teufel, M. Topper, A. Winkel, and Z. Zhang. MEAD - a platform for multidocument multilingual text summarization. In *LREC 2004*, Lisbon, Portugal, May 2004.

[44] L. Ratinov and D. Roth. Design challenges and misconceptions in named entity recognition. In *CoNLL '09: Proceedings of the Thirteenth Conference on Computational Natural Language Learning*, pages 147–155, Morristown, NJ, USA, 2009. Association for Computational Linguistics.

[45] A. Ratnaparkhi. A maximum entropy model for part-of-speech tagging. In Eric Brill and Kenneth Church, editors, *Proceedings of the Empirical Methods in Natural Language Processing*, pages 133–142, 1996.

[46] N. Rotem. Open text summarizer. http://libots.sourceforge.net/.

[47] M. Sanderson. Advances in automatic text summarization. *Computational Linguistics*, 26(2):280–281, 2000.

[48] S. Sarawagi. Information extraction. *Found. Trends databases*, 1(3):261–377, 2008.

[49] D. B. Skillicorn and N. Vats. Novel information discovery for intelligence and counterterrorism. *Decision Support Systems*, 43(4):1375 – 1382, 2007. Special Issue Clusters.

[50] P. Srinivasan. Text mining: Generating hypotheses from medline. *Journal of the American Society for Information Science and Technology*, 55:396–413, 2004.

[51] D. R. Swanson. Fish oil, raynaud's syndrome, and undiscovered public knowledge. *Perspectives in biology and medicine*, 30(1):7–18, 1986.

[52] D. R. Swanson. Migraine and magnesium: eleven neglected connections. *Perspectives in biology and medicine*, 31(4):526–557, 1988.

[53] D. R. Swanson and N. R. Smalheiser. An interactive system for finding complementary literatures: a stimulus to scientific discovery. *Artif. Intell.*, 91(2):183–203, 1997.

[54] D. R. Swanson, N. R. Smalheiser, and A. Bookstein. Information discovery from complementary literatures: categorizing viruses as potential weapons. *Journal of the American Society for Information Science and Technology*, 52(10):797–812, 2001.

[55] A. C. Telea. *Data visualization: principles and practice*. A K Peters, 2008.

[56] H. M. Wallach. Conditional random fields: An introduction. Technical Report MS-CIS-04-21, University of Pennsylvania, 2004.

[57] R. Weischedel, R. Schwartz, J. Palmucci, M. Meteer, and L. Ramshaw. Coping with ambiguity and unknown words through probabilistic models. *Comput. Linguist.*, 19(2):361–382, 1993.

[58] J. Xu and H. Chen. Criminal network analysis and visualization. *Commun. ACM*, 48(6):100–107, 2005.

[59] J. J. Xu and H. Chen. Crimenet explorer: a framework for criminal network knowledge discovery. *ACM Trans. Inf. Syst.*, 23(2):201–226, 2005.

[60] V. A. Yatsko and T. N. Vishnyakov. A method for evaluating modern systems of automatic text summarization. *Automatic Documentation and Mathematical Linguistics*, 41:93–103, 2007.

[61] M. Yetisgen-Yildiz and W. Pratt. Using statistical and knowledge-based approaches for literature-based discovery. *Journal of Biomedical Informatics*, 39(6):600 – 611, 2006.

Appendix A

The Stemming Process

This appendix contains the table illustrating the steps of the stemming process explained in section 3.4.

Table 5: Stemming Process Example

Token	stem pre1	stem post1	stem manual	stem post2	stem syn
[jenny]	jenny]	jenny	jenny	jenny	jenny
how	how	how	how	how	how
did	did	did	do	do	do
you	you	you	you	you	-
manage	manage	manage	manage	manag	manag
to	to	to	to	to	-
get	get	get	get	get	get
those?	those?	those	those	thos	thos
i	i	i	i	i	-
didn’t	didn’t	did	do	do	do
even	even	even	even	ev	even
see	see	see	see	se	see
them	them	them	them	them	them
come	come	come	come	com	com

Continued on Next Page…

Table 5: Stemming Process Example

Token	stem pre1	stem post1	stem manual	stem post2	stem syn
up	up	up	up	up	-
for	for	for	for	for	-
auction?	auction?	auction	auction	auction	auction
[john]	john	john	john	john	john
just	just	just	just	just	just
went	went	went	go	go	go
to	to	to	to	to	-
the	the	the	the	th	-
purple	purple	purple	purple	purpl	purpl
sim	sim	sim	sim	sim	sim
on	on	on	on	on	-
my	my	my	my	my	-
map	map	map	map	map	map
and	and	and	and	and	-
looked	looked	looked	looked	look	look
on	on	on	on	on	-
the	the	the	the	th	-
site	site	site	site	sit	sit
[john]	john]	john	john	john	john
i	i	i	i	i	-
got	got	got	get	get	get
the	the	the	the	th	-
auction	auction	auction	auction	auction	auction
id	id	id	id	id	id
maybe	maybe	maybe	maybe	mayb	mayb
because	because	because	because	becaus	becaus
i	i	i	i	i	-
was	was	was	be	be	-

Continued on Next Page…

Table 5: Stemming Process Example

Token	stem pre1	stem post1	stem manual	stem post2	stem syn
actually	actually	actually	actually	actual	actual
standing	standing	standing	standing	stand	stand
on	on	on	on	on	-
the	the	the	the	th	-
sim	sim	sim	sim	sim	sim
[kim]	kim]	kim	kim	kim	kim
right..	right..	right	right	right	right
and	and	and	and	and	-
then	then	then	then	th	then
bid	bid	bid	bid	bid	bid
on	on	on	on	on	-
them	them	them	them	them	them
at	at	at	at	at	-
the	the	the	the	th	-
auction	auction	auction	auction	auction	auction
[jenny]	jenny]	jenny	jenny	jenny	jenny
well	well	well	well	well	well
then..	then..	then	then	then	then
that's	that's	that	that	that	that
[john]	john]	john	john	john	john
if	if	if	if	if	-
i	i	i	i	i	-
get	get	get	get	get	get
any	any	any	any	any	any
of	of	of	of	of	-
the	the	the	the	th	-
others	others	others	others	others	others
i	i	i	i	i	-

Continued on Next Page…

Table 5: Stemming Process Example

Token	stem pre1	stem post1	stem manual	stem post2	stem syn
am	am	am	am	am	-
bidding	bidding	bidding	bidding	bid	bid
on	on	on	on	on	-
want	want	want	want	want	want
to	to	to	to	to	-
buy?	buy?	buy	buy	buy	buy
the	the	the	the	th	-
is	is	is	is	is	-
still	still	still	still	still	still
up..	up..	up	up	up	up
[jenny]	jenny]	jenny	jenny	jenny	jenny
sure	sure	sure	sure	sure	-
just	just	just	just	just	just
ring	ring	ring	ring	r	ring
me	me	me	me	m	-
on	on	on	on	on	-
or	or	or	or	or	-
email	email	email	email	email	email
me	me	me	me	m	-
at	at	at	at	at	-