Data & Knowledge Engineering 86 (2013) 224–241

Contents lists available at SciVerse ScienceDirect

Data & Knowledge Engineering

journal homepage: www.elsevier.com/locate/datak

Subject-based semantic document clustering for digital forensic investigations

Gaby G. Dagher *, Benjamin C.M. Fung

Concordia University, Computer Science, Montreal, Quebec H3G 1M8, Canada

ARTICLE INFO

Article history:
Received 23 September 2011
Received in revised form 21 March 2013
Accepted 22 March 2013
Available online 3 April 2013

Keywords:
Clustering
Classification
Data mining
Information retrieval
Forensic analysis
Crime investigation

ABSTRACT

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

1. Introduction

The process of investigating digital devices for the purpose of generating digital evidence related to an incident under investigation is commonly referred to as *Digital Forensic Investigation* (*DFI*). According to Carrier et al. [5], digital evidence of an incident is any digital data that supports or refutes a hypothesis about the incident. The task of analyzing persistent documents found on a storage device of a suspect's computer is an essential part of the DFI process to gather credible and convincing evidence. However, this task is daunting due to the large number of documents usually stored on a hard disk. The continuously increasing size of storage devices makes the task even more difficult.

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

In this paper, we propose a new document clustering model that allows an investigator to cluster all documents on a suspect's computer according to certain subjects he is interested in (e.g. hacking, child pornography). Once the documents are clustered in groups, each corresponding to a subject, the investigator can search for documents that belong to a certain subject.

There have been several attempts to define a digital forensic model that abstracts the forensic process from any specific technology, such as the Digital Forensics Research Workshop (DFRWS) model for digital forensic analysis [31], Lee's model of scientific crime scene investigation [25], Casey's model for processing and examining digital evidence [6], and Reith's model for

* Corresponding author. Tel.: +1 514 638 4884.
E-mail addresses: daghir@ciise.concordia.ca (G.G. Dagher), fung@ciise.concordia.ca (B.C.M. Fung).

0169-023X/$ – see front matter © 2013 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.datak.2013.03.005

digital forensic analysis [7]. DFRWS is a pioneer in developing the forensic process. It defined Digital Forensic Science as a linear process:

> *The use of scientifically derived and proven methods toward the preservation, collection, validation, identification, analysis, interpretation, documentation and presentation of digital evidence derived from digital sources for the purpose of facilitating or furthering the reconstruction of events found to be criminal, or helping to anticipate unauthorized actions shown to be disruptive to planned operations.*
>
> [31]

Fig. 1 illustrates the *Digital Forensic Investigation* (*DFI*) process as defined by DFRWS. After determining items, components, and data associated with the incident (Identification phase), the next step is to preserve the crime scene by stopping or preventing any activities that can damage digital information being collected (Preservation phase). Following that, the next step is collecting digital information that might be related to the incident, such as copying files or recording network traffic (Collection phase). Next, the investigator conducts an in-depth systematic search of evidence related to the incident being investigated, such as filtering, validation, and pattern matching techniques (Examination phase) [7]. The investigator then puts the evidence together and tries to develop theories regarding events that occurred on the suspect's computer (Analysis phase). Finally, the investigator summarizes the findings by explaining the reasons for each hypothesis that was formulated during the investigation (Presentation phase). In the examination phase, investigators often utilize certain forensic tools to help examine the collected files and perform an in-depth systematic search for pertinent evidence. However, there are three problems with today's computer forensic tools:

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

Evidence-oriented design. Existing DFI tools are designed for solving crimes committed against people, in which the evidence exists on a computer; they were not created to address cases where crimes took place on computers or against computers. In general, DFI techniques are designed to find evidence where the possession of evidence is the crime itself; it is easier to solve child pornography cases than computer hacking cases [13].

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

Our solution attempts to address these problems by answering the question of whether or not evidence for events defined by the investigator, such as hacking or child pornography, is present in the documents collected from the suspect's computer. The investigator initially defines the subjects (events) he is interested in investigating by providing a set of terms to describe each subject, such as vocabularies that are commonly used in the subject. We introduce a novel subject-based semantic document clustering algorithm that groups (clusters) all documents into a set of overlapping clusters, each corresponding to one unique subject. Fig. 2 illustrates the overall process of our solution.

The general intuition of our clustering approach is to generate a set of expansion vectors for each given subject using its initial subject definition. Each expansion vector consists of a set of weighted terms related to the subject, where each term is generated

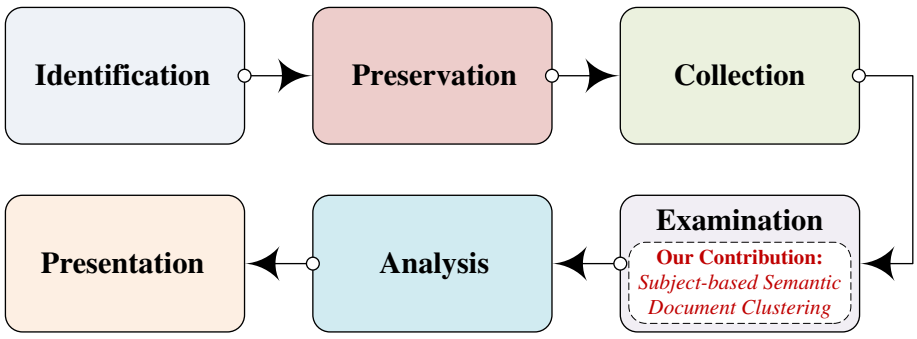

Fig. 1. Digital Forensic Investigation (DFI) process as defined by DFRWS.

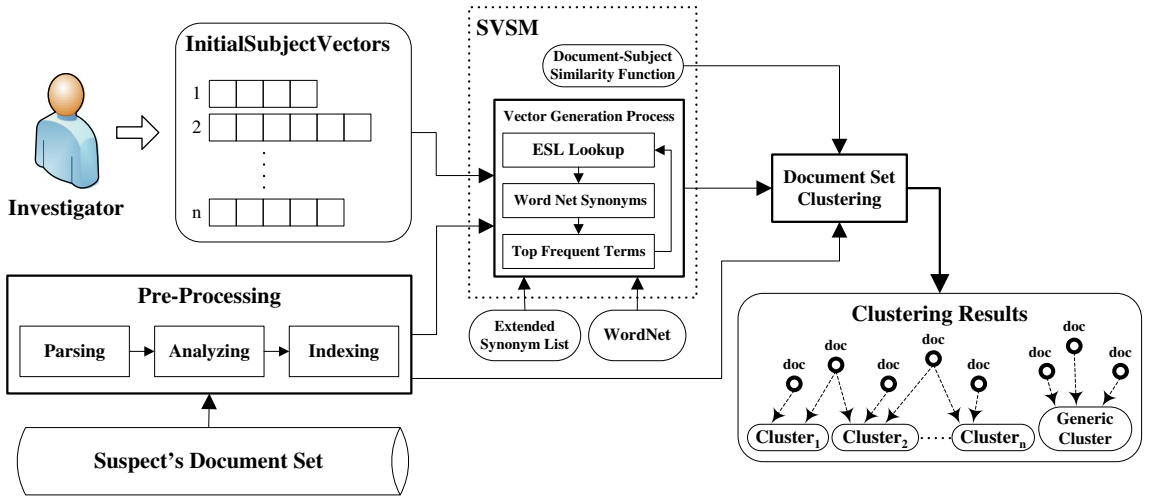

Fig. 2. Pipeline diagram illustrating the overall process of our clustering solution.

using WordNet [29]. Our clustering algorithm also supports integration with the *Extended Synonym List* (*ESL*), a list of forensic-specific synonyms and related terms provided by the digital forensic investigation team at *Sûreté du Québec* (*SQ*) for the purpose of generating terms that are related to the subject. Once the expansion vectors for a subject are generated, they will be used with the initial subject definition to construct a vector of weighted term frequencies called *subject vector* such that the problem of measuring the similarity between a document and a subject is reduced to measuring the similarity between the document and the subject vector.

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

Example 1.1. Let us assume that the investigator is interested in investigating whether hacking events occurred on the suspect's computer. He might provide the following set of terms (along with their PoS tag) to define the subject:

<Hacking>hack:Verb,security breach:Noun,login:Noun,Nmap:Noun,permission:Noun,exploit:Verb</Hacking> ■

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

1.1. Contribution

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

- Subject Vector Space Model: We model our clustering solution by proposing *Subject Vector Space Model* (*SVSM*), a new model based on *Vector Space Model* (*VSM*) [36] and *Topic-based Vector Space Model* (*TVSM*) [3]. In SVSM, each dimension represents a subject, where terms and documents are represented in the space according to their relations to all subjects. This allows a more realistic representation of the terms because terms inherently are not are orthogonal to each other. This representation also allows us to reduce the clustering problem of a document to determine the coordinate of this document on each subject vector. This is reflected in our proposed similarity function $Sim(d,s_i)$ that measures the similarity between any document $d \in D$ and subject $s_i \in S$, where D is a collection of documents and S is a set of subjects.
- Novel subject-based semantic document clustering algorithm: We introduce an efficient and scalable subject-based semantic document clustering algorithm that expands the term vector representing each subject using *WordNet* [29]. *Word Sense Disambiguation* (*WSD*) algorithm [9] is integrated in the process to determine the appropriate sense (and accordingly, the synonym set) of a term in WordNet in the context of the initial terms that define a subject. The integration of WSD improves the precision of our clustering algorithm by reducing the polysemy affect [10,40].

- Dynamically capturing suspects' terminologies: We make use of the document set to incrementally expand the subject vector by adding top frequent terms from the most similar documents to the subject. WSD is also integrated in this phase to determine the dominant sense of the term, which is the sense used the most in the several contexts in which the term appears.
- Experiments on real-life data: We conduct an extensive experimental study over three real-life data sets and examine the effectiveness of the algorithm according to subject vector length and document-score thresholds. We also demonstrate that our approach is highly scalable for large data sets.

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

2. Related work and background knowledge

The traditional document classification algorithms [11,39,28,21] cannot solve our subject-based clustering problem because there is no training data available to train the classifiers of the classification algorithms. Similarly, semi-supervised document clustering approaches [14,35,23,17,16] are unable to solve our clustering problem, regardless of whether they are search-based or similarity-based. This is because the initial definition of a subject, a short set of representative terms, cannot be mapped to labeled documents, constraints, or used as a feedback to adjust the resulting clusters. Zhao and Karypis [42] proposed a semi-supervised clustering approach to address this problem; however, their approach assumes that there exists prior knowledge of the natural (major) clusters within the document set. This is not applicable to our case as investigators are unlikely to have prior knowledge of all criminal events that have already occurred on a suspect's computer. In our case, the investigator provides an initial vector for each subject (topic) that he would like to investigate, regardless of whether or not this subject exists in the document set.

The traditional partitional [18,19,30] and hierarchical [12,41,8] document clustering algorithms are not designed to accept any feedback from the user; therefore, they cannot solve our subject-based clustering problem because they cannot take advantage of the user-provided initial subjects definitions.

Kuropka et al. [3] proposed an algebraic model called Topic-based Vector Space Model (TVSM) for information retrieval. TVSM is a d dimensional space where each dimension represents one fundamental topic, and all fundamental topics are independent from each other (orthogonal). Terms, documents, and queries are represented as vectors, each represented by its coordinates on all dimensions (topics). TVSM does not define the term-topic relation (length of each term vector and the inter-term vector angles). This is important in order to determine the similarity between a document and topic. *Enhanced Topic-based Vector Space Model* (*eTVSM*) [33] attempted to define the term-topic relation by providing an algorithm that uses WordNet ontology as the source of semantics. Our proposed Subject Vector Space Model (SVSM) on the other hand is also based on TVSM, where each dimension corresponds to an interested subject originally provided by the investigator. However, we propose a novel approach to define the subject vector for each dimension in SVSM, consequently allowing us to determine the term-subject relation of any term and subject in SVSM. In addition, we propose a novel similarity function that calculates the similarity (angle) between any document vector and subject vector in SVSM.

WordNet [29], a lexical database for the English language, is utilized to establish semantic relations between terms during several phases of our clustering algorithm. WordNet's lexicon is divided into four major categories: nouns, verbs, adjectives, and adverbs. The basic unit of a WordNet lexicon is *synonym set* (*synset*), in which each synset includes a word, its synonyms, definition, and sometimes example. Any word is assumed to have a finite number of discrete meanings, where each meaning under one type of part of speech (PoS) is called a sense. Each sense of a word in WordNet is represented in a separate synset. WordNet supports two types of relations: semantic and lexical.

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

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

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

[1] http://plato.stanford.edu/entries/leibniz/.
[2] Two senses of a word are considered *homonyms* if they have the same spelling but completely different meanings.
[3] A word is considered polysemous if all of its senses are various shades of the same basic meaning.

3. Problem definition

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

3.1. Initial subject definition

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

For each input term $t \in s_i^0$, the investigator should provide its part of speech PoS_t. Providing PoS for each term is not mandatory; however, it is highly recommended to help increase the accuracy of our clustering algorithm by avoiding homonymous word problems, where more than one distinct meaning exists for the same word. Moreover, PoS_t should be either "verb" or "noun." We limited ourselves to only verbs and nouns because they are the most content-bearing types of words [27].

3.2. Extended Synonym List (ESL)

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

3.3. Preprocessing

The following preprocessing procedures are applied on the initial vectors and the document set.

3.3.1. Initial vectors preprocessing

In each initial vector, words that have morphological forms are normalized to their canonical form. We employed the *Porter* stemming algorithm [34] for this purpose. This algorithm, for example, reduces the words "exploiting", "exploited", and "exploitation" to the root word "exploit".

3.3.2. Document set preprocessing

Stopwords removal, *stemming*, and *tokenization* preprocessing procedures are applied on each document with the goal of reducing its dimensionality, noise, and computational complexity while avoiding a significant loss of information.

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

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

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

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

Weight assignment. As we will see in Section 4.2, each document is represented by a vector whose coordinates are the weights of the document in each subject dimension. To achieve such representation, we first determine the weight of each term in each document using *term frequency–inverse document frequency* (*tf–idf*) [37]. Specifically, we determine each weighted term frequency $\upsilon_{t,d} = tf(t,d) \times idf(t,D)$, where $tf(t,d)$ is the frequency of term t in document d and $idf(t,d)$ is the *inverse document frequency* of term t in the document set D:

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

where $|D|$ is the number of documents in the document set D, and $Freq_{t,d}$ is the number of documents in D that contain the term t. The *tf–idf* method was chosen because it considers both the local and global frequencies of a term when computing its weight. In other words, the weight of a term in a document is computed not only based on its frequency in this document, but also on the frequency of the term within the document set.

3.4. Clustering system definition

We can now define our clustering system as a tuple:

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

where:

- $D = \{d_1,d_2,\ldots,d_{|D|}\}$ is a set of input documents to be clustered.
- $S = \{s_1,s_2,\ldots,s_n\}$ represents a set of n subjects, where each subject s_i is a set of weighted terms that are semantically generated based on the initial definition vector s_i^0.
- *SVSM* is a framework for representing terms, documents, and subjects as vectors, as well as defining the relationship between them.
- $Sim(d,s_i)$ is a similarity function that measures the similarity between a document $d \in D$ and a subject $s_i \in S$, and returns a real value that ranges between 0 and 1.
- $\mathcal{C} = \{\mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_n\}$ is a set of n overlapping output clusters, each of which is associated with one subject. In other words, each cluster $\mathcal{C}_i \in \mathcal{C}$ contains a set of retrieved documents for subject $s_i \in S$ given a similarity threshold τ:

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

3.4.1. Problem statement

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

4. Solution modeling

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

4.1. Subject vector generation

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

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

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

4.2. Subject Vector Space Model (SVSM)

We introduce the *Subject Vector Space Model* (*SVSM*), an algebraic model generated based on the *Vector Space Model* (*VSM*) [36] and the *Topic-based Vector Space Model* (*TVSM*) [3]. SVSM is an n-dimensional vector space in which each dimension (axis) represents a subject $s_i \in S$. Similar to TVSM, all axis coordinates in SVSM are positive, and all axes are orthogonal to each other:

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

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

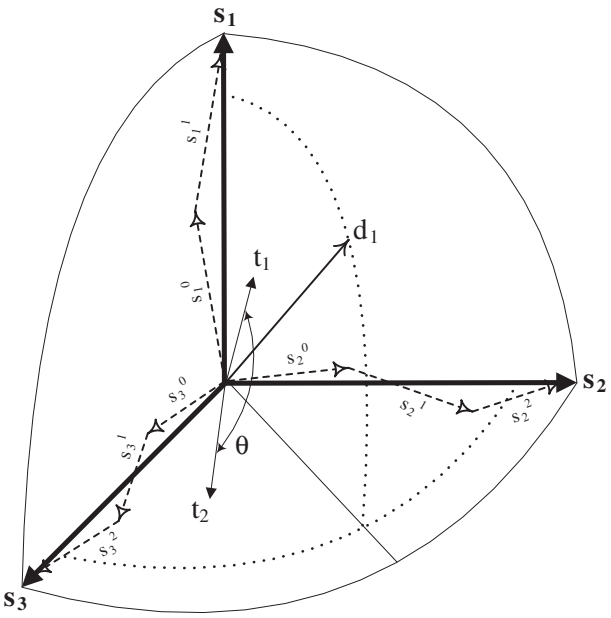

Fig. 3. Subject Vector Space Model (SVSM).

Fig. 3 illustrates an example of the Subject Vector Space Model. The representation of terms and documents in SVSM is analogous to their representation in TVSM.

4.2.1. Term representation

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

$$\vec{t} = \left\{ \omega_{t,s_i} | i = 1, 2, \ldots, n \right\}$$

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

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

$$|| \vec{t} || = \sqrt{\sum_{i=1}^{n} \left(\omega_{t,s_i} \right)^2}.$$

To measure the similarity between term vectors, we utilize the cosine similarity measure [38]. Other similarity measures could have been used instead, such as Pearson coefficient or Euclidean distance; however, we chose the cosine similarity measure because it abstracts the length of the documents and focuses on the correlation between their vectors. Since all axis coordinates are positive, the cosine similarity value between any two term vectors is always between zero and 1:

$$0 \leq \theta\left(\vec{t_i}, \vec{t_j}\right) \leq \pi/2 \Rightarrow cos(\theta) \in [0, 1]$$

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

4.2.2. Document representation

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

$$\vec{d} = \frac{1}{\psi_d} \sum_{t \in \mathbb{T}} \vec{t} \,.\omega_{t,d} | \psi_d = || \sum_{t \in \mathbb{T}} \vec{t} \,.\omega_{t,d} ||$$

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

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

4.3. Document-subject similarity function

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

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

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

5. Semantic clustering algorithm

In this section, we illustrate our approach for semantically clustering a document set D based on initial subject definitions $\{s_1^0, s_2^0, \ldots, s_n^0\}$. The objective is to utilize the initial subject definition vector of each subject $s_i \in S$ to semantically generate expansion vectors such that the final representation of the subject can be expressed as defined by Eq. (3). To achieve this objective, we apply two computational linguistics techniques — *Word sense disambiguation (WSD)* and part-of-speech (PoS) tagging. WSD is used with words that have multiple meanings in order to determine the most appropriate sense of a word within a sentence. PoS tagging, on the other hand, is used to label a word in a sentence with its corresponding lexical category (Noun, Verb, etc.). Brill's tagger [4] is a rule-based tagger employed in our algorithm to perform the PoS tagging tasks.

5.1. Slang words processing

If the slang word to be processed is provided by the investigator as part of the initial subject vector, then no PoS tagging is required as the investigator must provide the PoS for each word in the initial vector. However, to determine the most appropriate sense of a slang word, our algorithm uses WordNet while applying Lesk WSD algorithm (as we will see in Section 5.1.2). WordNet can still be used in this case because it partially supports informal words. For example, if we search WordNet for the keyword "coke", the sixth noun sense we get is: "*coke, blow, nose candy, snow, C — (street names for cocaine)*." If the slang word does not exist in WordNet, then the word remains in the subject vector, and does not contribute to the expansion of the vector, but can still be used to cluster the document set. On the other hand, if the slang word was captured from the suspect's documents by identifying it as a frequent term, then we use WordNet and Lesk algorithm to perform WSD and PoS (as we will see in Section 5.1.3). If the word does not exist in WordNet, then our tagger will attempt to identify its PoS using the context (sentence) in which the word exists.

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

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

Step 1 *ESL Lookup* (lines 6–11): Generate an expansion vector by looking up synonyms/related words in the *Extended Synonym List* (*ESL*) of each input term of this step.

Step 2 *WordNet Synonyms* (lines 17–18): Utilize WordNet to determine the synsets of each input term of this step, and then generate an expansion vector using the synonym terms resulting from applying Lesk's [26] word sense disambiguation technique to determine the appropriate synonyms of each term.

Step 3 *Top Frequent Terms* (lines 24–30): Compute frequently used terms from the top-ranked documents of the document set, and then generate an expansion vector by extracting top frequent terms (*tft*) using Jiang–Conrath's [20] relatedness distance measure. Word sense disambiguation technique is utilized for part-of-speech (PoS) tagging and sense determination in a context to address the problem of homonyms and polysemous words.

Algorithm 1. Subject vector generation

Require: $|s_i^0| > 0$
1: $s_i \leftarrow s_i^0$
2: $r \leftarrow 1$
3: **for** each subject $s_i \in S$ **do**
4: $\ddot{a} \leftarrow s_i^0$
5: **while** $|s_i| \le \delta$ **do**
6: $s_i^r \leftarrow Lookup_{ESL}(\ddot{a})$
7: **if** $r == 1$ **then**
8: $\mho_{s_i^r} = 1$
9: **else**
10: $\mho_{s_i^r} = \lambda \times \mho_{s_i^{r-1}}$
11: **end if**
12: **if** $|s_i + s_i^r| > \delta$ **then**
13: break
14: **else**
15: $s_i \leftarrow s_i + s_i^r$
16: $r \leftarrow r + 1$
17: $s_i^r = Synonym\left(wsd_{WordNet}\left(\ddot{a} + s_i^{r-1}\right)\right)$
18: $\mho_{s_i^r} = \lambda \times \mho_{s_i^{r-1}}$
19: **if** $|s_i + s_i^r| > \delta$ **then**
20: break
21: **else**
22: $s_i \leftarrow s_i + s_i^r$
23: $r \leftarrow r + 1$
24: $\ddot{q} \leftarrow \left\{\cup_{l=1}^{l=r-1} s_i^l\right\}$
25: $D' \leftarrow \{d | d \in D, score_\epsilon(q,d)\}$
26: $\hat{\mathcal{M}} \leftarrow tfidf\left(t \in D' \times d \in D'\right)$
27: $tft\left(\hat{P}\right) \leftarrow \left\{t \middle| \sum_{d \in \hat{\mathcal{M}}} tf(t,d) \cdot idf\left(t, D'\right) \ge \sigma\right\}$
28: $wsd(t) \leftarrow \mathbb{S}_t \subset_{dominant} \cup_{x \in X} wsd^x(t) : t \in tft\left(\hat{P}\right)$
29: $s_i^r = \left\{t \middle| t \in tft\left(\hat{P}\right), \sum_{t_c \in s_i^0} jcn(t, t_c) \ge \sigma'\right\}$
30: $\mho_{s_i^r} = \lambda \times \mho_{s_i^{r-1}}$
31: **if** $|s_i + s_i^r| > \delta$ **then**
32: break
33: **else**
34: $s_i \leftarrow s_i + s_i^r$
35: $\ddot{a} \leftarrow s_i^r$
36: $r \leftarrow r + 1$
37: **end if**
38: **end if**
39: **end if**
40: **end while**
41: **end for**
Output: $S = \{s_1, s_2, \ldots, s_n\}$

We define a threshold δ to be the maximum length allowed for any subject vector. That is, our algorithm keeps expanding each subject vector as long as its length (the number of terms) does not exceed the threshold δ.

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

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

Based on an extensive number of experiments conducted, we observed that our clustering algorithm provides better results when $\lambda = 1$ in Step 1 — ESL Lookup. This is because introducing new terms using ESL imposes a minimal risk of adding noise because the list is provided by the investigator. We also observed that our clustering algorithm provides better results when $\lambda = 0.5$ in Step 2 — WordNet Synonyms and Step 3 — Top Frequent Terms. This is because using ontology terms or external source terms as additional features may introduce noise [15].

Fig. 4 illustrates the subject s_i vector generation process.

5.1.1. ESL Lookup (lines 6–11)

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

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

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

5.1.2. WordNet Synonyms (lines 17–18)

The next phase is to utilize WordNet to determine the synsets of each input term and to generate an expansion vector s_i^{r+1} using the synonym terms resulting from applying Lesk's word sense disambiguation technique [26] to determine the appropriate synonyms of each term.

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

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

Our lexical semantic expansion approach using WordNet involves three stages: First, we identify all the senses for each term $t \in \ddot{b}$ (Section 5.1.2.1), then use *WSD* algorithm to select the most appropriate sense for each term (Section 5.1.2.2), and finally generate the expansion vector s_i^{r+1} by assigning the synonym terms (Section 5.1.2.3). Fig. 5 illustrates the lexical semantic expansion approach using WordNet.

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

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

where $PoS_{\mathbb{S}_j}$ denotes the part of speech of all synonyms in synset $\mathbb{S}_j$. We observe that associating only the synsets of the senses with matching *PoS* for each term has a major impact on the WSD algorithm, as it helps improve the overall disambiguation

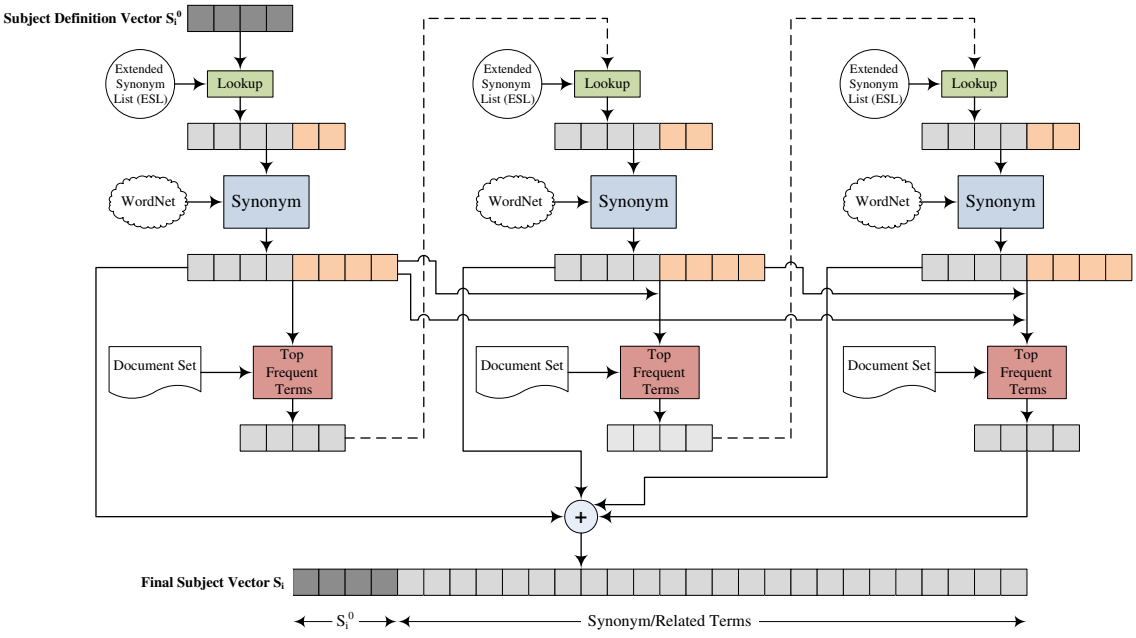

Fig. 4. Subject s_i vector generation process.

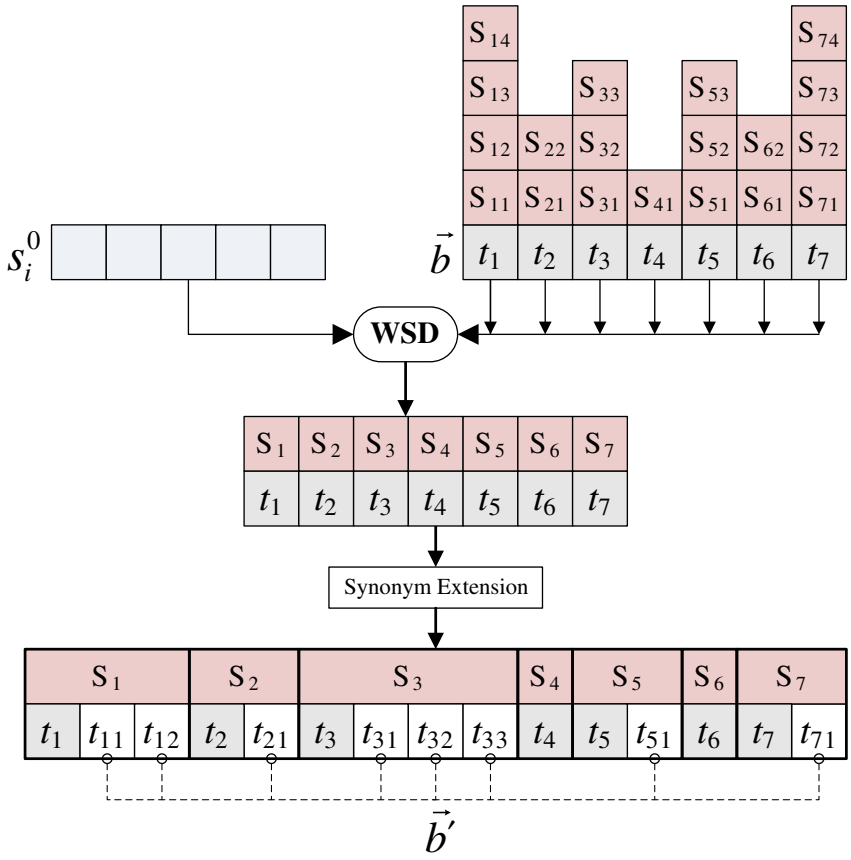

Fig. 5. Lexical semantic expansion using WordNet.

accuracy while reducing the computational time of the algorithm due to the reduction in search space. However, if the term is not tagged (no PoS is assigned to it), then we associate the synsets of both verb and noun senses with that term:

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

Note that $Syn(t) = \varnothing$ when term t cannot be recognized by WordNet. This case could occur since WordNet lexicon contains the majority (but not all) of the English words. This could also happen if the initial subject definition vector s_i^0 contains special terms, e.g., slang and special expressions, that are commonly used within criminal society but do not exist in standard English dictionaries.

5.1.2.2. Unique synset assignment. In the previous step, we associated a set of synsets with each term t. In this step, the objective is to identify the best fit synset in the context of subject s_i for each term $t \in \ddot{b}$. To achieve that, we use an adapted version of a word sense disambiguation method based on Lesk's algorithm [26]. Lesk's algorithm disambiguates a word by comparing the gloss of each of its senses to the glosses of every other word in a phrase. A word is assigned to the sense whose gloss shares the largest number of words in common with the glosses of the other words. The adapted version [9] of Lesk's algorithm will be applied to each term $t \in \ddot{b}$ to solve the ambiguity problem as follows:

(1) Define the context around target term $t_l \in \ddot{b}$ to be all the terms in the initial subject definition vector s_i^0.
(2) For each context term $t_k \in s_i^0$, list all the possible senses:

$$\left\{ \mathbb{S}_j \in Synset_{WN}(t_k) \middle| PoS_{\mathbb{S}_j} = PoS_{t_k} \right\}.$$

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

(4) Measure the relatedness between each gloss of target term t_l with each gloss from each term $t_k \in s_i^0$ by searching for overlaps. The overall score of each sense of a target term is the sum of the scores for each gloss pair.
(5) Once each combination has been scored, assign the synset of the corresponding sense with the highest score to the target term.
(6) Repeat this process for every term $t_l \in \ddot{b}$ to determine the most appropriate sense for each term.

According to Banerjee [2], the adapted version of Lesk's algorithm to WordNet can achieve F-measure values 0.421 for nouns and 0.239 for verbs.

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

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

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

5.1.3. Top frequent terms (lines 24–30)

The goal in this step is to generate an expansion vector s_i^{r+2} for subject s_i based on the expansion vector s_i^{r+1} that was generated in the previous step. First, we determine the documents in D that are the most related to already generated expansion vectors in Section 5.1.3.1, then we compute *top frequent terms* (*tft*) in Section 5.1.3.2. For each term in *tft* we then apply the WSD algorithm to determine the context dominant sense in Section 5.1.3.3. Finally we apply the Jiang–Conrath similarity measure [20] to determine the most related terms to the initial subject definition s_i^0 in Section 5.1.3.4.

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

$$\ddot{q} = \left\{ t \middle| t \in \bigcup_{l=1}^{l=r+1} s_i^l \right\}.$$

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

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

The score of all documents will then be normalized to be a real number between 0 and 1. Given a threshold $\epsilon \in \mathbb{N}^+$, we consider the top of the documents with the highest score as the most related set of documents denoted by D'.

5.1.3.2. Compute top frequent terms. To determine the top frequent terms in D', we first build a term-document matrix $\hat{\mathcal{M}}$ in which each row corresponds to a document $d \in D'$ and each column corresponds to a term $t \in D'$. The entries in the matrix are the *term frequency–inverse document frequency* (*tf–idf*) [37] of each term in each document.

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

$$tft\left(\hat{\mathcal{M}}\right) = \left\{ t \in \hat{\mathcal{M}} \middle| \sum_{d \in \hat{\mathcal{M}}} tf(t, d) \times idf\left(t, D'\right) \geq \sigma \right\}$$

■

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

As previously, we use the adapted version of the word sense disambiguation method based on Lesk's algorithm [26]. However, the main difference in this case is that each term exists in multiple contexts X. We define each context $x \in X$ of term t as a frame of e terms that appear to the left and right of the term in each of its occurrences.

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

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

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

$$wsd^x(t) = \mathbb{S}_t \left| \mathbb{S}_t \in Syn^x(t) \right.$$

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

$$wsd(t) = S_t \left| S_t \in_{dominant} \left\{ \bigcup_{x \in X} wsd^x(t) \right\} \right.$$

5.1.3.4. Relatedness distance measure. The set of terms $tft\left(\hat{\mathcal{M}}\right)$ we extracted from the document set to capture the suspects' terminologies might be unrelated (or weakly related) to the initial subject definition. To reduce noise and avoid a reduction in both the recall and precision of our clustering algorithm, we use a similarity measure to determine the relatedness of each term $t \in tft\left(\hat{\mathcal{M}}\right)$ to the initial subject definition vector s_i^0. The Jiang–Conrath similarity measure (*jcn*) [20] is used for this purpose. According to Pedersen et al. [32], it enriches the information content of the least common subsumer of two concepts with the sum of the information content of the individual concepts by subtracting the information content of the least common subsumer from this sum, and then takes the inverse to convert it from a distance to a similarity measure.

Using *jcn*, we compute the distance between each term $t \in tft\left(\hat{\mathcal{M}}\right)$ and all terms in s_i^0. The terms that meet a certain threshold σ' will be used to construct the expansion vector s_i^{r+2}:

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

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

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

and the weight of each term $t \in s_i^{r+2}$ is equal to the weight of the expansion vector itself $\upsilon_{s_i^{r+2}}$. Once s_i^{r+2} has been computed, it will then be used to start a new iteration of subject vector expansion by starting to look up all the terms in s_i^{r+2} in the Extended Synonym List (ESL) in Section 5.1.1.

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

Algorithm 2. Document clustering.

Require: $|s_i| < \delta$
1: **for** each subject $s_i \in S$ **do**
2: **for** each document $d \in D$ **do**
3: **if** $Sim(d,s_i) > \tau$ **then**
4: $\mathcal{C}_i \leftarrow d$
5: **end if**
6: **end for**
7: **end for**
Output: $\mathcal{C} = \{\mathcal{C}_1, \mathcal{C}_2, ..., \mathcal{C}_n\}$

6. Experimental evaluation

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

Indexing engine. This engine parses documents on the suspect's computer, analyzes the documents by applying the preprocessing steps, namely tokenization, stemming, stop word removal, and text normalization, and then indexes the documents. We used Apache Tika[4] and Lucene[5] to parse, preprocess, and index the documents.

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

[4] Apache Tika. http://tika.apache.org/.
[5] Apache Lucene. http://lucene.apache.org/.

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

The objective of the experiments is to evaluate the performance of our proposed subject-based semantic document clustering algorithm implemented in the clustering engine in terms of accuracy, efficiency, and scalability.

6.1. Data sets

We use three data sets in our experiments: *Classic3*, *Forensic-1* and *Forensic-2*. The pre-classification of each document will be used to measure the accuracy of the clustering results; however, during the clustering process, this information will be hidden.

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

- *Classic3* is a benchmark data set used in text mining.[6] It consists of 3893 documents from 3 disjoint classes: 1400 aeronautical-system documents (CRAN), 1033 medical documents (MED), and 1460 information-retrieval documents (CISI).
- *Forensic-1* is a data set we collected for validating our clustering algorithm against a set of crime-related documents. This data set consists of 90 documents from 3 different classes: 30 drugs-related documents, 30 hacking-related documents, and 30 sexual assault-related documents.
- *Forensic-2* we collected more data and extended *Forensic-1* to build a larger data set. Forensic-2 consists of 320 documents from 3 different classes (topics): 80 drugs-related documents, 80 hacking-related documents, 80 sexual assault-related documents, and 80 overlapping documents that belong to more than one class (contain more than one topic).

6.2. Evaluation method

We use *F-measure* [24] to measure the accuracy of the clustering solution produced by our method.

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

$$Precision\left(\mathcal{K}_i, \mathcal{C}_j\right) = \frac{true\_positive}{true\_positive + false\_positive} = \frac{\left|\mathcal{K}_i \cap \mathcal{C}_j\right|}{\left|\mathcal{C}_j\right|} \tag{6}$$

$$Recall\left(\mathcal{K}_i, \mathcal{C}_j\right) = \frac{true\_positive}{true\_positive + false\_negative} = \frac{\left|\mathcal{K}_i \cap \mathcal{C}_j\right|}{|\mathcal{K}_i|} \tag{7}$$

where $|\mathcal{K}_i|$, $\left|\mathcal{C}_j\right|$, and $\left|\mathcal{K}_i \cap \mathcal{C}_j\right|$ denote the number of documents in class $\mathcal{K}_i$, in cluster $\mathcal{C}_j$, and in both $\mathcal{K}_i$ and $\mathcal{C}_j$ respectively. We use F_1 in our experiments to compute the accuracy of cluster $\mathcal{C}_j$ with respect to class $\mathcal{K}_i$ as follows:

$$F_1\left(\mathcal{K}_i, \mathcal{C}_j\right) = \frac{2 \times Precision\left(\mathcal{K}_i, \mathcal{C}_j\right) \times Recall\left(\mathcal{K}_i, \mathcal{C}_j\right)}{Precision\left(\mathcal{K}_i, \mathcal{C}_j\right) + Recall\left(\mathcal{K}_i, \mathcal{C}_j\right)} \in [0, 1] \tag{8}$$

where F_1 score reaches its best value at 1 and worst score at 0.

6.3. Experimental results

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

6.3.1. Accuracy

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

Fig. 6 depicts the F-measure values (accuracy) of the clustering algorithm with respect to δ and τ. We set δ to three different values relative to the average document length *avg_dl* in the data set: $0.25 * avg\_dl$,$0.50 * avg\_dl$, and *avg_dl*, whereas τ was set to a range of values between 0.05 and 0.5. We observe that the F-measure value spans from 0.61 to 0.72 when the minimum similarity threshold τ increases from 0.05 to 0.5. Assigning a value between 0.1 and 0.175 to τ provides high accuracy for all three different values of δ. Based on the extensive number of experiments conducted, we observe that our clustering algorithm provides a high accuracy value when $\tau \in [0.08, 0.2]$ and $\delta = avg\_dl$.

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

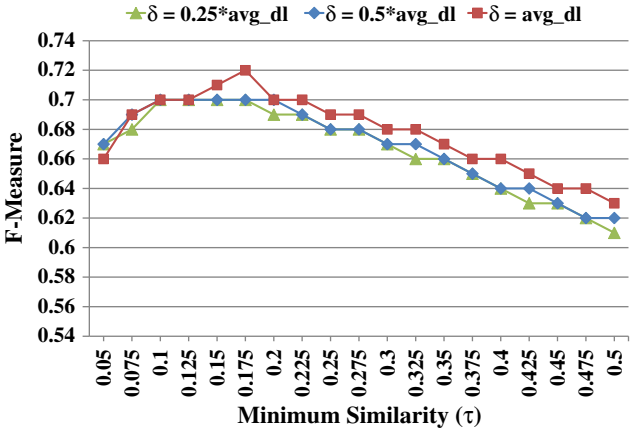

Fig. 6. Sensitivity to cluster's minimum similarity (τ) in relation to the maximum subject vector length (δ) using *Classic3* data set.

We also observe that the change in δ value affects the accuracy of the algorithm in a minimal way. We argue that the integration of *WSD* technique in step 2 and step 3 of our subject vector generation algorithm helps reduce the noise and consequently reduces the sensitivity of our clustering algorithm to the input parameter δ. We also observe that when the maximum length of a subject vector is equal to the average document length in the document set ($\delta = avg\_dl$), the algorithm in most cases provides higher F-measure values.

Having determined that higher accuracy is obtained when $\delta = avg\_dl$, we perform additional accuracy experiment on *Forensic-2*. Fig. 7 depicts the F-measure values of the clustering algorithm with respect to τ, where $\delta = avg\_dl$. We observe that the F-measure values span from 0.59 to 0.75 when the minimum similarity threshold τ increases from 0.05 to 0.5. This accuracy range slightly differs from the accuracy range we obtained in the previous experiment when *Classic3* was used. Based on these results, we conclude that our algorithm is insensitive to whether the classes in the data set are disjoint (*Classic3*) or overlapping (*Forensic-2*).

Fig. 7 also depicts the sensitivity of the algorithm with regard to the two major steps, namely *WordNet Synonyms* and *Top Frequent Terms*. We observe that the accuracy of our algorithm decreases dramatically when WordNet (*WordNet Synonyms* step) is eliminated. That is, the accuracy drops by an average of 65% when we eliminate WordNet from the process. We also observe that the higher the minimum similarity threshold τ is, the higher the drop in accuracy. That is, when $\tau = 0.05$, the drop percentage is 45%; however, when $\tau = 0.5$, the drop percentage increases to 81%. Similarly, we observe that the accuracy of our algorithm decreases when TFT (*Top Frequent Terms* step) is eliminated. The average drop in accuracy is 14%, a much smaller value compared to the impact of the elimination of WordNet; however, unlike WordNet, the drop in accuracy does not seem to correlate with the value of the minimum similarity threshold τ.

6.3.2. Efficiency and scalability

One major contribution of our work is the development of an efficient and scalable algorithm for semantic document clustering that expands the term vector representing each subject using WordNet. According to Algorithm 1, the runtime

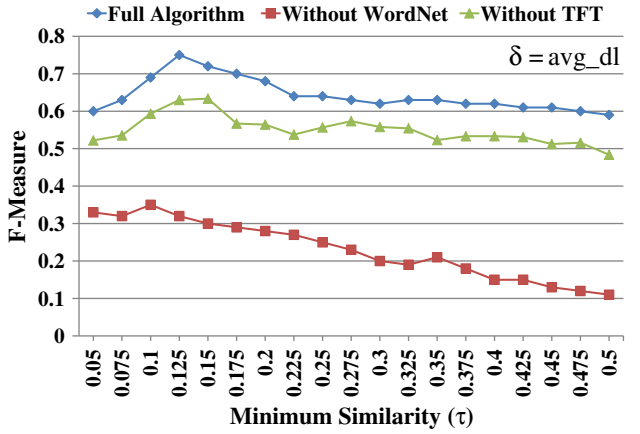

Fig. 7. Sensitivity to cluster's minimum similarity (τ) in relation to WordNet and TFT using *Forensic-2* data set.

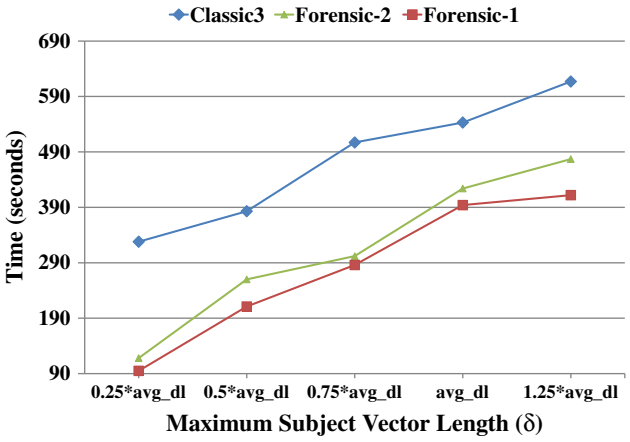

Fig. 8. Efficiency with regard to *Classic3*, *Forensic-2*, and *Forensic-1* data sets.

complexity of our approach is dominated by the maximum subject vector length δ and the data set size; therefore, we study the runtime under different subject vector lengths and different data set sizes.

Fig. 8 depicts the runtime on the three data sets *Forensic-1*, *Forensic-2*, and *Classic3*, with respect to δ that ranges between $0.25 * avg\_dl$ and $1.25 * avg\_dl$. We observe that the runtime scales linearly with respect to the subject vector length under all data sets. We also observe that regardless of the size of the data set (*Classic3* is 43 times larger than *Forensic-1* and 12 times larger than *Forensic-2*), the gradient (slop) of each data set's runtime remains the same.

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

$$\alpha = \left\lfloor \frac{Target\ \#\ of\ documents}{|D|} \right\rfloor. \tag{9}$$

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

$$\alpha' = (Target\ \#\ of\ documents)\%|D| \tag{10}$$

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

Our subject vector generation algorithm consists of three phases: *ESL Lookup*, *WordNet Synonyms*, and *Top Frequent Terms*. The objective is to measure the runtime of each phase to ensure it does not grow proportionally to the total number of documents in the data set. Fig. 9 depicts the runtime of the three phases with respect to the total number of documents being clustered. The

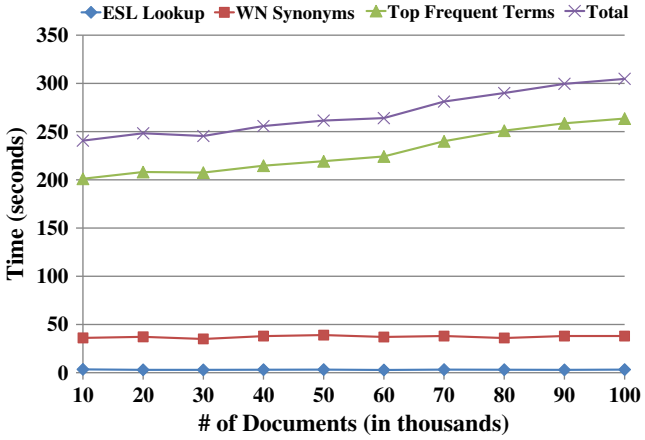

Fig. 9. Scalability with the scale-up *Classic3* data set.

total runtime for processing 100,000 documents is 313 s, where 3 s are spent looking up synonyms from *ESL*, 46 s are spent generating synonyms using WordNet, and 264 s are spent analyzing the document set to extract frequent terms that capture the suspects' terminologies. The runtimes of both the ESL Lookup phase and WordNet Synonyms phase are independent of the total number of documents. As for the Top Frequent Terms phase, the runtime grows as the total number of documents increases. This is due to the internal parameter that is used to capture the suspect's terminologies and is set to 1% of the data set size. The runtime scales linearly with respect to the data set's size. Since each phase of the algorithm is either independent or grows linearly with respect to the total number of documents, the experimental results on real-life data sets suggest that our algorithm is scalable.

7. Conclusions and further work

In this paper, we have proposed a subject-based semantic document clustering algorithm for digital forensic investigations with the objective of using data mining to support investigations. Motivated by the digital forensic process at *Sûreté du Québec* (SQ), we modeled our clustering solution by proposing the Subject Vector Space Model (SVSM). Moreover, we introduced an efficient and scalable algorithm for building full definition of each subject based on the initial definition. This was achieved via the utilization of WordNet to extract term synonyms, the integration of Word Sense Disambiguation (WSD) to determine the appropriate sense for each term, and the dynamic capturing of suspects' terminologies. For future work, it would be interesting to investigate the impact of allowing other types of words, such as adjectives and adverbs, to be part of the initial subject definition. Also, it would be interesting to study how the application of dimensionality reduction techniques [22,1] during the generation of subject vectors can further enhance the accuracy of our algorithm.

Acknowledgment

The research is supported in part by the National Cyber-Forensics and Training Alliance Canada (NCFTA Canada). We would like to thank *Sûreté du Québec* (SQ) for providing us with real-life materials for experimentation.

References

[1] C.C. Aggarwal, P.S. Yu, Outlier detection for high dimensional data, Proceedings of the 2001 ACM SIGMOD International Conference on Management of Data, ACM, New York, NY, USA, 2001.
[2] S. Banerjee, Adapting the lesk algorithm for word sense disambiguation to wordnet, Ph.D. thesis, University of Minnesota, 2002.
[3] J. Becker, D. Kuropka, Topic-based Vector Space Model, Proceedings of the 6th International Conference on Business Information Systems, Colorado Springs, 2003.
[4] E. Brill, A simple rule-based part of speech tagger, Proceedings of the third conference on Applied natural language processing, ANLC '92, Association for Computational Linguistics, Stroudsburg, PA, USA, 1992.
[5] B.D. Carrier, E.H. Spafford, An event-based digital forensic investigation framework, Proceedings of the 4th Digital Forensic Research Workshop, 2004.
[6] E. Casey, Digital Evidence and Computer Crime: Forensic Science, Computers, and the Internet with Cdrom, 1st ed. Academic Press, Inc., Orlando, FL, USA, 2000.
[7] M.R. Clint, M. Reith, C. Carr, G. Gunsch, An Examination of Digital Forensic Models, 2002.
[8] G. Costa, G. Manco, R. Ortale, E. Ritacco, Hierarchical clustering of xml documents focused on structural components, Data & Knowledge Engineering 84 (2013) 26–46.
[9] T.N. Dao, T. Simpson, Measuring similarity between sentences, The Code Project, 2005.
[10] S. Deerwester, S.T. Dumais, G.W. Furnas, T.K. Landauer, R. Harshman, Indexing by latent semantic analysis, Journal of the American Society for Information Science 41 (6) (1990) 391–407.
[11] S. Dumais, J. Platt, D. Heckerman, M. Sahami, Inductive learning algorithms and representations for text categorization, Proceedings of the 7th International Conference on Information and Knowledge Management, ACM, New York, NY, USA, 1998.
[12] B.C.M. Fung, K. Wang, M. Ester, Hierarchical document clustering using frequent itemsets, Proceedings of the 3rd SIAM International Conference on Data Mining (SDM), SIAM, San Francisco, CA, 2003.
[13] S.L. Garfinkel, Digital forensics research: The next 10 years, Digital Investigation 7 (1) (2010) S64–S73.
[14] R. Guan, X. Shi, M. Marchese, C. Yang, Y. Liang, Text clustering with seeds affinity propagation, IEEE Transactions on Knowledge and Data Engineering 23 (2011) 627–637.
[15] X. Hu, X. Zhang, C. Lu, E.K. Park, X. Zhou, Exploiting wikipedia as external knowledge for document clustering, Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, New York, NY, USA, 2009.
[16] Y. Hu, E.E. Milios, J. Blustein, Semi-supervised document clustering with dual supervision through seeding, Proceedings of the 27th Annual ACM Symposium on Applied Computing, SAC '12, ACM, New York, NY, USA, 2012.
[17] R. Huang, W. Lam, An active learning framework for semi-supervised document clustering with language modeling, Data & Knowledge Engineering 68 (2009) 49–67.
[18] A.K. Jain, Data clustering: 50 years beyond k-means, Pattern Recognition Letters 31 (8) (2010) 651–666.
[19] A.K. Jain, R.C. Dubes, Algorithms for Clustering Data, Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1988.
[20] J.J. Jiang, D.W. Conrath, Semantic similarity based on corpus statistics and lexical taxonomy, Proceedings of the International Conference Research on Computational Linguistics (ROCLING X), 1997.
[21] T. Joachims, Text categorization with support vector machines: learning with many relevant features, Proceedings of the 10th European Conference on Machine Learning, Springer-Verlag, London, UK, 1998.
[22] I. Jolliffe, Principal component analysis, Springer Verlag, 2002.
[23] H. Kim, S. Lee, A semi-supervised document clustering technique for information organization, Proceedings of the 9th International Conference on Information and Knowledge Management, ACM, New York, NY, USA, 2000.
[24] B. Larsen, C. Aone, Fast and effective text mining using linear-time document clustering, Proceedings of the 5th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, New York, NY, USA, 1999.
[25] H. Lee, T. Palmbach, M. Miller, Henry Lee's Crime Scene Handbook, Academic Press, San Diego, 2001.
[26] M. Lesk, Automatic sense disambiguation using machine readable dictionaries: how to tell a pine cone from an ice cream cone, Proceedings of the 5th annual International Conference on Systems Documentation, ACM, New York, NY, USA, 1986.

[27] Y. Li, S.M. Chung, J.D. Holt, Text document clustering based on frequent word meaning sequences, Data & Knowledge Engineering 64 (2008) 381–404.
[28] A. McCallum, K. Nigam, A comparison of event models for naive Bayes text classification, Proceedings of the AAAI-98 Workshop on Learning for Text Categorization, AAAI Press, 1998.
[29] G.A. Miller, WordNet: a lexical database for English, Communications of the ACM 38 (1995) 39–41.
[30] R.T. Ng, J. Han, Efficient and effective clustering methods for spatial data mining, Proceedings of the 20th International Conference on Very Large Data Bases, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1994.
[31] G. Palmer, M. Corporation, A road map for digital forensic research, Proceedings of the 1st Digital Forensic Research Workshop, 2001.
[32] T. Pedersen, S. Patwardhan, J. Michelizzi, Wordnet: : similarity — measuring the relatedness of concepts, Proceedings of AAAI, 2004.
[33] A. Polyvyanyy, D. Kuropka, A Quantitative Evaluation of the Enhanced Topic-based Vector Space Model, Universitätsverlag Potsdam, 2007.
[34] M.F. Porter, An Algorithm for Suffix Stripping, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1997. 313–316.
[35] L. Rigutini, M. Maggini, A semi-supervised document clustering algorithm based on EM, Proceedings of the 2005 IEEE/WIC/ACM International Conference on Web Intelligence, IEEE Computer Society, Washington, DC, USA, 2005.
[36] G. Salton, Automatic Text Processing: the Transformation, Analysis, and Retrieval of Information by Computer, Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1989.
[37] G. Salton, C. Buckley, Term-weighting approaches in automatic text retrieval, Information Processing and Management 24 (1988) 513–523.
[38] G. Salton, M.J. McGill, Introduction to Modern Information Retrieval, McGraw-Hill, Inc., New York, NY, USA, 1986.
[39] H. Schütze, D.A. Hull, J.O. Pedersen, A comparison of classifiers and document representations for the routing problem, Proceedings of the 18th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, ACM, New York, NY, USA, 1995.
[40] K. Sparck Jones, A Statistical Interpretation of Term Specificity and its Application in Retrieval, Taylor Graham Publishing, London, UK, 1988. 132–142.
[41] M. Steinbach, G. Karypis, V. Kumar, A Comparison of Document Clustering Techniques, 2000.
[42] Y. Zhao, G. Karypis, Topic-driven clustering for document datasets, Proceedings of the SIAM Data Mining Conference (SDM), 2005.

Gaby G. Dagher is a Ph.D. candidate in Computer Science at Concordia University. He received his Master of Applied Science (M.A.Sc.) in Information Systems Security from Concordia Institute for Information Systems Engineering (CIISE), and his Bachelor of Computer Science (BCompSc) from Concordia University. His research interests are in data security and privacy in cloud computing, with emphases on privacy-preserving data mining and secure data publishing.

Benjamin C. M. Fung is an Associate Professor at Concordia University in Montreal, and a research scientist of the National Cyber-Forensics and Training Alliance Canada. He received a Ph.D. degree in Computing Science from Simon Fraser University in 2007. He has over 60 refereed publications that span across the prestigious research forums of data mining, privacy protection, cyber forensics, web services, and building engineering. His data mining work in authorship analysis has been widely reported by media worldwide. Before pursuing his academic career, he worked at SAP Business Objects as a system software developer for 4 years. Dr. Fung is a licensed professional engineer in software engineering and is currently affiliated with the Computer Security Lab in CIISE.