An Automatic Alert Unification Method for Heterogeneous Alert Signatures

Ouissem Ben Fredj
The Department of Information Technology, Taif University
P.O.Box: 888, Hawiyah, Taif, Zip Code: 21974, Kingdom of Saudi Arabia
(Email: Ouissem.BenFredj@gmail.com)

(Received Aug. 6, 2015; revised and accepted Nov. 27 & Dec. 8, 2015)

Abstract

Several monitoring systems are usually composed by heterogeneous monitoring sensors. Each sensor raises thousands of alerts to be saved and analyzed in a centralized station. Most of alerts raised by different sensors are almost the same but have various formats and various descriptions. The system administrator must identify manually similar alerts in order to decrease the number of generated alerts and to improve the data quality. This paper proposes an alert unification method that automatically creates meta-alerts from a set of heterogeneous alert sets coming from different security monitoring sensors. Instead of dealing with several sets of alerts, this method allows the administrator to use a unique set of meta-alerts.

Keywords: Data pre-processing, language processing, network security applications, record linkage

1 Introduction

Several systems are usually composed by heterogeneous monitoring sensors. Each sensor has its philosophy, its functional method, and its alert definitions. One important task in such environment is the multisensor data fusion which integrates objects that relate to the same entities from several databases. This task help to improve data quality by producing clean, sanitized, refined, and accurate data ready for fast and simple analysis and good knowledge extraction.

There are several definitions of the architectures of multisensor data fusion system in the literature. Luo and Kay [24, 25, 26] defined functional roles of multisensor integration and multisensor fusion composed of three-level fusion category. Dasarathy [7] proposed an I/O pair-based fusion architecture. [15, 16] provided an introduction to multisensor data fusion based on the architecture of the Joint Directors of Laboratories (JDL) data fusion model [34], which was originally developed for military applications. [10] gave a review of different models of multisensor data fusion system. [32] gave an overview of multisensor fusion techniques relating to different fusion levels of the JDL framework, and discussed the weaknesses and strengths of the approaches in different applications. [17] proposed a framework of logical sensor which treats the multisource information in a multisensor system based on the viewpoint of logical software programming. [14] imported the JDL processes to the cyber security context.

The most used model is the JDL model [15, 16, 34] which divides the data fusion process into four levels: Level 1 for object refinement, Level 2 for situation assessment, Level 3 for threat assessment, and Level 4 for process assessment. Level 1 contains processes of data registration, data association, position attribute estimation, and identification. Level 2 fuses the kinematic and temporal characteristics of the data to infer the situation of the environments. Level 3 projects the current situation into the future. In Level 1, the parametric information is combined to achieve refined representations of individual objects. Levels 2 and 3 are often referred to information fusion, while Level 1 is data fusion. Level 4 is an ongoing assessment of other fusion stages to make sure that the data fusion processes are performing in an optimal way.

Bass adapts the JDL model to data fusion in the field of computer security (see Figure 1) [3]. The proposed model adds Level 0 called data refinement. In this level, data acquired from a set of network security sensors (IDSs, network sniffers, application logs), is filtered and calibrated to generate a set of objects. Level 1 correlates all measurements using common spatial and temporal metrics. Level 2 correlates the objects using high level features like their behavior, dependencies, targets, origins, protocols, attack rates. The output of this step is the situational knowledge. Level 3, threat assessment, assesses the situation against known intrusion detection templates and suggests or identifies future threats. Level 4, resource management, analyses the outputs of lower levels (Levels 2, 1, and 0) to define processing priorities to some objects or situations.

This paper proposes a Level 1 method. After gathering alerts from different intrusion detection sensors, the pro-

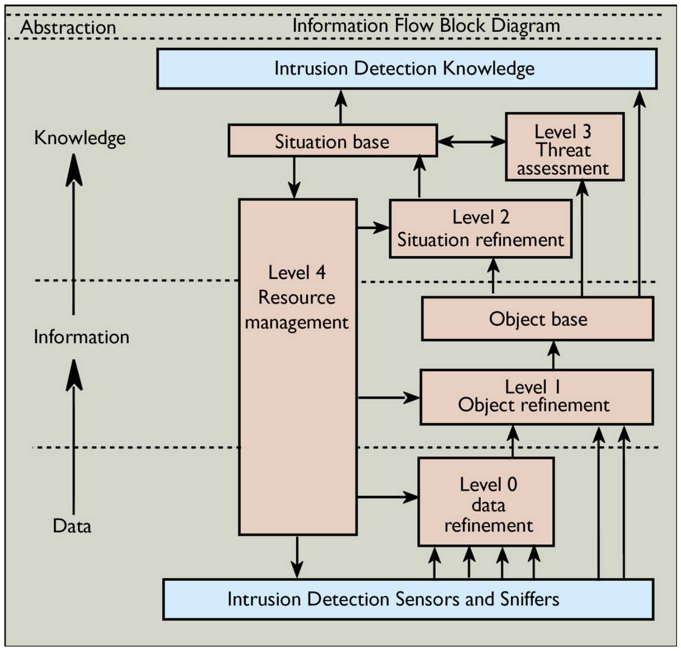
Abstraction
Information Flow Block Diagram
Intrusion Detection Knowledge
Knowledge
Situation base
Level 3 Threat assessment
Level 2 Situation refinement
Level 4 Resource management
Information
Object base
Level 1 Object refinement
Level 0 data refinement
Data
Intrusion Detection Sensors and Sniffers

Figure 1: The JDL multisensor data fusion system

posed method detects duplicated and related alerts and creates a new set of refined and clean alerts. The method makes use of record linkage techniques in order to detect related alerts.

Record linkage aims to identify links between records that refer to the same real-world entities. Most record linkage methods are based on the method of Newcombe and Kennedy in 1962 that was formalized by Fellegi and Sunter in 1969. The literature includes several works in record linkage from various domains [6]. Security specialists have used record linkage to remove duplicated alerts raised by Intrusion detection sensors [3, 4, 20]. It has been also used to identify fraudsters and criminals national security databases [19]. The database of historical census data was subject of many investigations that aim to identify links between individuals and even create a complete genealogy tree over a long period of time [1, 12, 29]. Record linkage was also used to eliminate duplicate records from the result of search engines [21, 33]. Recent researches [11, 31] have employed record linkage methods to determine all bibliographic of an author in large publication databases.

In this paper, we consider a security monitoring system encompassing a distributed IDS (Intrusion Detection System), however, the proposed work could be used in any system that encompasses several heterogeneous monitoring sensors. Usually, large companies deploy several IDS sensors in different locations to gather information about possible threats and attacks. The IDWG (Intrusion Detection Working Group) is a major working group that defined a general distributed IDS architecture [13] (see Figure 2). The E blocks are (Event-boxes) is composed of sensor elements that monitor the target system. The D blocks (Database-boxes) are intended to store information from E blocks for subsequent processing by A and R boxes. The A Blocks (Analysis-boxes) are processing modules for analyzing events and detecting potential hostile behavior, so that some kind of alarm will be generated if necessary. The R blocks (Response-boxes) executes an intrusion reaction.

Figure 3 shows a near-real-time distributed IDS for high-speed networks called $(\phi|\pi)$ [30]. $(\phi|\pi)$ aims to be generic, scalable, and adaptive distributed IDS which acts in real-time depending on traffic, alerts, past traffic, and predicted traffic and alerts.

It is clear that almost all the DIDS architectures include distributed agents/sensors and a global manager which collects information. In order to have a global view of the system, the DIDS must gather events from sensors. The events should be correlated to provide a simple and accurate view to the administrator. The correlation step usually makes use of statistics and data mining in order to improve the monitored site.

The main contributions of this paper are:

- A state of the art of the current architectures of data fusion systems;
- Identification of the relation between data fusion systems and record linkage methods;
- A study on how to use record linkage techniques in the security context;
- A record linkage method that improve data quality of alerts generated by security monitoring sensors;
- A semantic similarity method to compare between alerts;

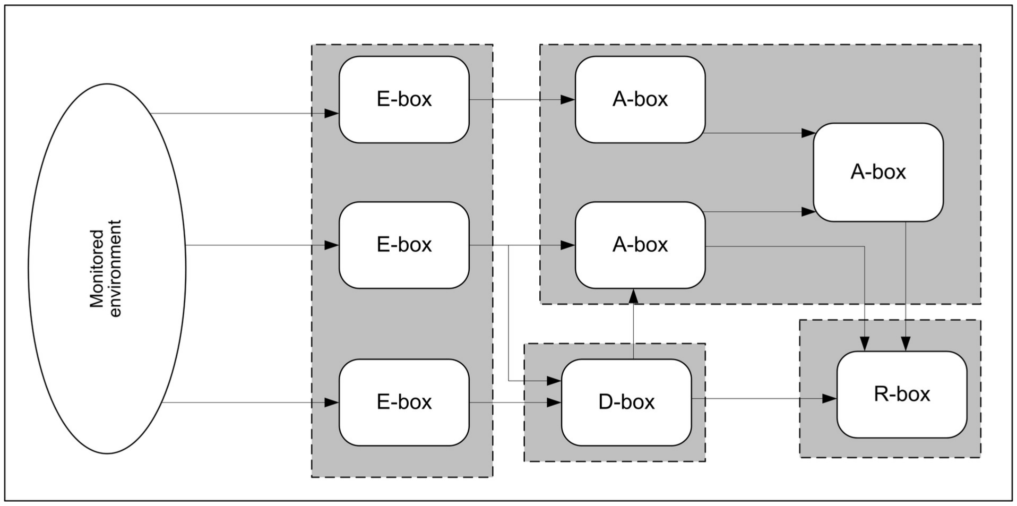

Figure 2: The general IDS architecture as defined by IDWG

- An optimization model and solution that solves ambiguity between related alerts.

The remaining paper is organized as follow: next section defines the problem of record linkage in the context of security monitoring. Section 3 is an overview of the proposed solution called alert unification. Section 4 details the structure of the global algorithm for the alert unification process. Section 5 introduces the semantic-based similarity method to measure the similarity between alerts. Section 6 deals with the introduction of a new alert in the existing system. Section 7 solves the problem of ambiguous related alerts using an original method. Section 8 analyzes the performance of the method, and the last section concludes and discusses the future works.

2 Problem Definition

The alerts defined by a given set of monitoring sensors are different even they refer semantically to the same effective alert. Hence, an alert unification step is required to minimize the total number of generated alerts and to maximize their quality. The alert unification process will combine several alert sets, corresponding to several monitoring sensors, into one unified alert set. Note that this step would be done offline, means that this step would run even before the deployment of the sensors. Thus, sophisticated similarity algorithms could be used to unify alert names. Since each monitoring sensor has its monitoring philosophy and its specific alert format, our method suppose that each generated alert has only one attribute called alert description. Indeed, this simplification makes our method very generic and could be applied with a wide range of monitoring sensors.

Formally, in the current work, we consider a set of sn different sensors monitoring one subject. Each sensor s_i has an alert set $AS_i = A_{(0,i)}, A_{(1,i)}, \dots, A_{(n_i-1,i)}$ where $A_{(j,i)}$ is the alert number j that belongs to the alert set AS_i and n_i is the size of AS_i i.e. $n_i = |AS_i|$. Each alert set AS_i must verify the following conditions:

- There are no duplicate alerts within the same alert set;
- Each alert in AS_i has a unique identifier ($A_{i,j}$);
- Each alert has an attribute called description.

The result of the unification process is an alert set $U = u_0, u_2, \dots, u_{q-1}$ containing q unified alerts such that each unified alert u_k is associated with at least one alert $A_{i,j}$. in this case, u_k and $A_{i,j}$ correspond to the same effective alert. u_k could be a unified alert of several alerts from different alert sets (i.e. different sensors). The goal of the unification process is to maximize the number of alerts that a unified alert u_k is associated with. In other words, our goal is to minimize q.

3 Solution Overview

Consider the problem definition above, our proposed solution (see Figure 4) starts with an indexing step that gives for each alert a unique identifier. During Step 2, each alert is split into word tokens. In order to calculate the similarities between tokens, we propose a semantic similarity approach; given two word tokens, each token has several senses taken from the Wordnet [35] database (Step 3). In order to choose the best sense of a given token, we apply a word sense disambiguation method called the Mickeal Lest algorithm [23] (Step 4). Then, a similarity value is computed between the selected senses of both tokens using the Resnik method [27] (Step 5). The Resnik similarities between tokens senses create a similarity matrix between the tokens of two alerts. The similarity matrix is transformed to an optimization problem

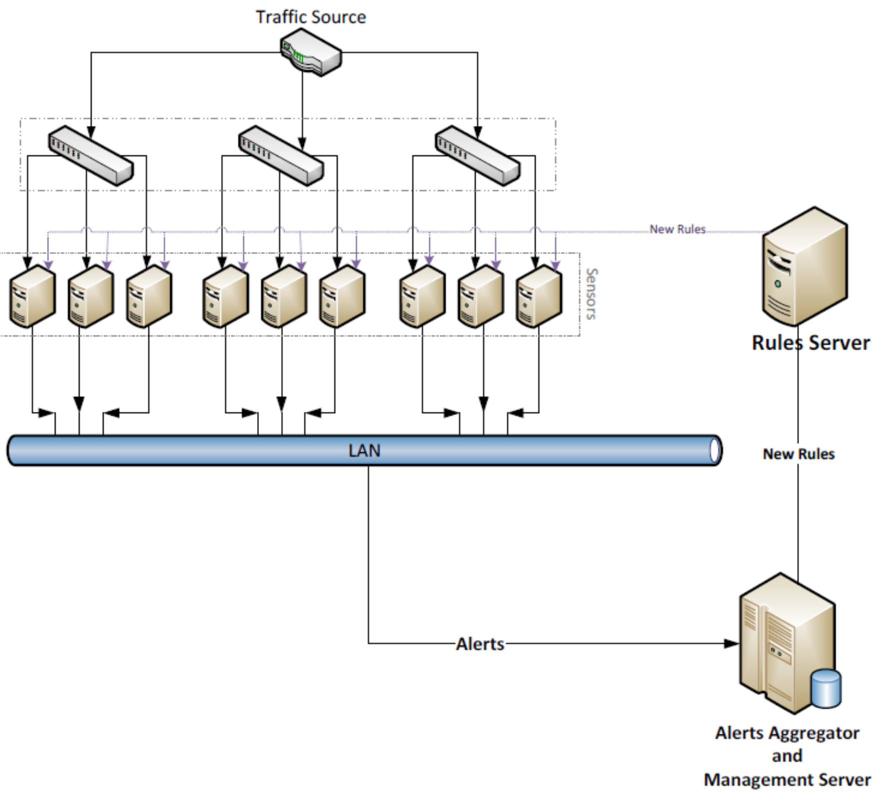

Figure 3: The global architecture of a phi DIDS

aiming to maximize the overall similarities between the tokens of both alerts in order to choose the best matching between tokens. The optimization problem is solved using the polynomial Hungarian method [22] (Step 6). The output of the later step is the similarity value between two alerts (Step 7). The previous steps are repeated to compute the similarity between each couple of alerts. In order to minimize the complexity of the global algorithm, the unified alert set U is created incrementally; each new alert is compared to the existing unified alerts (using the previous 7 steps). The result of the previous comparison is one of the following three cases:

- The new alert is not similar to any of the unified alerts (using a high similarity threshold). In this case the new alert become a new unified alert and added to the unified alert set U;

- The new alert is similar to only one unified alert. In this case the new alert is considered as a duplicate of the unified alert;

- The new alert is similar to several unified alerts. This case raises an ambiguity problem between the alerts. However, the new alert must be a duplicate of only one unified alert. The ambiguity problem is solved by an optimization method called Hopcroft-Karp algorithm as explained in Section 7.

4 The Alert Unification

In this section, the main steps of the alert unification are detailed. Given sn different IDS sensors. Each sensor s_i has an alert set $AS_i = A_{(0,i)}, A_{(1,i)}, \ldots, A_{(n_{i-1},i)}$ where $A_{(j,i)}$ is the alert number j that belongs to the alert set AS_i and n_i is the size of AS_i i.e. $n_i = |AS_i|$. The result of the unification process is an alert set $U = u_0, u_2, \ldots, u_{q-1}$ such that each u_k is associated with at least one alert $A_{i,j}$. in this case, u_k and $A_{i,j}$ correspond to the same alert. u_k could be the unified alert of several alerts from different alert sets AS_i (i.e. different sensors). Let UM is the unification matrix. Hence, UM_k is the unification vector of the unified alert u_k. $UM_{k,p}$ is the alert number from the alert set S_p that corresponds to the unified alert u_k. For example $UM_{5,2} = 7$ means that the fifth alert of U corresponds to the seventh alert of the alert set of the IDS sensor 2. $UM_{5,3} = -1$ means that the unified alert u_5 has not any corresponding alert with the alert set of the sensor 3.

$$UM_{k,j} = \begin{cases} i & if \quad u_k \approx AS_{i,j} \\ -1 & if \quad \forall i, \quad u_k \not\approx AS_{i,j} \end{cases}$$

For two alerts u and v, $u \approx v$ means that the alert u and v relate to the same alert even they differ in their respective descriptions, and $u \not\approx v$ means that the alert u and v are different. Note that UM has q (i.e. $|U|$) rows and sn (the number of different IDS sensors) columns. Also, q is initially unknown. $q \in [\max_{i \in [0,sn-1]} n_i, \sum_{i=0}^{sn-1} n_i]$

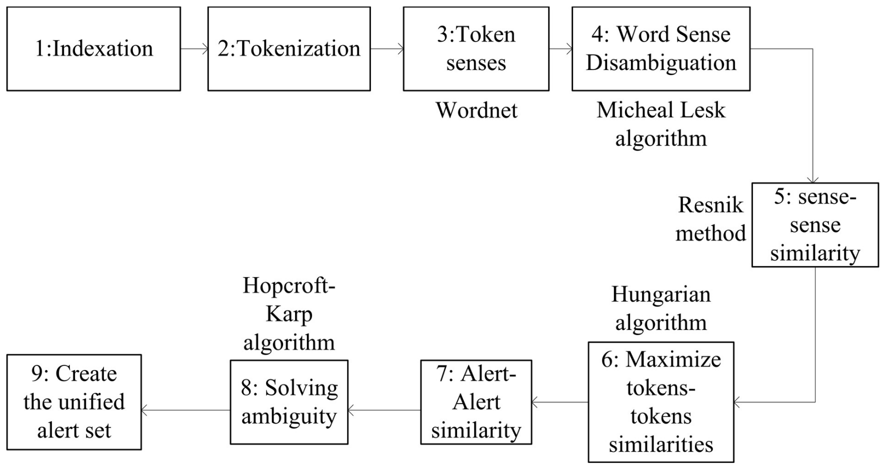
1:Indexation
2:Tokenization
3:Token senses
Wordnet
4: Word Sense Disambiguation
Micheal Lesk algorithm
Resnik method
5: sense-sense similarity
Hungarian algorithm
6: Maximize tokens-tokens similarities
7: Alert-Alert similarity
Hopcroft-Karp algorithm
8: Solving ambiguity
9: Create the unified alert set

Figure 4: the main steps of the alert unification process

The goal of the unification process is to maximize the number of alerts that a unified alert u_k is associated with. Thus, minimizing the (-1)s in the unification matrix UM. The goal of the unification process can be viewed also as minimizing q. The proposed approach builds UM column by column. The steps are summarized in the following:

Inputs:

- sn alert sets: $AS_0, \cdots, AS_{sn-1}$;
- Each alert set AS_i is composed by n_i alerts: $S_i = A_{i,0}, A_{i,1}, \cdots, A_{i,n_{i-1}}$. It is preferable that the alert sets are decreasingly ordered according to their size, $n1 \geq n2 \geq \cdots \geq n_{sn-1}$.

Outputs:

- The unified alert set $U = u_0, u_2, \cdots, u_{q-1}$. q is the number of unified alerts.
- The unification matrix UM.

Procedures:

1) Clone AS_1 to U. U is initialized with the first alert set AS_1. Thus,
   - $U := AS_1$
   - $q := |AS_1|$
   - $UM_{i,1} := i$, for each $i \in [0, |AS_1| - 1]$. Each alert of AS_1 is similar with itself.
   - $j := 2$
2) Build the similarity matrix SS_{U,S_j} between each couple of alert sets U and AS_j. $SS_{U,S_j}[a, b]$ is the similarity rating between the alerts u_a and $A_{j,b}$. We assume that all the scores of the similarity matrix are in the range $[0, 1]$, which means that if the score gets a maximum value (equal to 1) then the two alerts are absolutely similar. The algorithm used to build the similarity matrices is explained below (see Section 5).
3) Set a hard similarity threshold ST for example (0.9) which guarantees the similarity between the alerts. In other words, if $SS_{U,S_j}[a, b] > ST$, then we assume that the alerts u_a and $A_{j,b}$ could be similar.
4) Set to zero the values of the similarity matrix SS_{U,S_j} which are less than ST. i.e. $SS_{U,S_j}[a, b] = 0$ if $SS_{U,S_j}[a, b] < ST$. $SS_{U,S_j}[a, b] = 0$ means that the alerts u_a and $A_{j,b}$ could not correspond to the same alert.
5) if $SS_{U,S_j}[a, b] > ST$ and $\nexists$ c such as $SS_{U,S_j}[a, c] > ST$ then $UM_{a,j} = b$. i.e. if the alert u_a is similar to only one alert $A_{j,b}$, then we assume that the unified alert of $A_{j,b}$ is u_a:
   - $UM_{a,i} := 0$, for each $i \in [0, sn - 1]$, the new unified alert has not any corresponding alert except $A_{j,b}$.
   - $UM_{a,j} = b$
6) if $SS_{U,S_j}[a, b] = 0$ for all $b \in [0, |AS_j| - 1]$ i.e. the alert $A_{j,b}$ has not any corresponding unified alert, then, add $A_{j,b}$ as a new unified alert:
   - $q := q + 1$, increment the number of unified alerts
   - $u_q := A_{j,b}$, the new unified alert corresponding to $A_{j,b}$
   - $UM_{q,i} := 0$, for each $i \in [0, sn - 1]$, the new unified alert has not any corresponding alert except $A_{j,b}$.
   - $UM_{q,j} := b$
7) if $SS_{U,S_j}[a, b] > ST$ and $\exists$ c such as $SS_{U,S_j}[a, c] > ST$. i.e. the alert u_a is a unified alert of several eventual alerts from the same alert set S_j. This is problematic because several alerts from S_j are ambiguous while u_a must correspond to only one alert from the set S_j. Hence, we propose to find a support to choose the most appropriate alert. See details of the

approach in Section 7. Let the alert $A_{j,b'}$ the supported alert. Thus, the unified alert is voted to correspond to $A_{j,b'}$:

- $UM_{a,i} := 0$, for each $i \in [0, sn-1]$, the new unified alert has not any corresponding alert except $A_{j,b'}$.
- $UM_{a,j} := b'$

8) If $j < sn$ then increment $j (j := j+1)$ and go to Step 2.

5 Similarity Measure Between Alerts

As introduced in the last section, the similarity between alert sets requires the similarity matrix SS that summarizes the similarities between alerts. This step is the major step of the unification process since it may affect the overall accuracy of the whole process. Optimistic similarity measure may lead to incorrect fusion between different alerts giving the administrator a false view of the system. Pessimistic similarity measure supports dissimilarity between alerts which may increase the number of unified alerts; each one corresponds to a few effective alerts. The worst case is that each unified alert corresponds to only one effective alert. In our approach, we propose to take advantage from the semantic similarity approaches in order to compare the descriptions of the alerts that have different names. Thus, we compare senses and not words. We propose an approach based on semantic similarity. Given two sentences, semantic similarity gives a score that reflects the semantic relation between the meanings of them. The semantic similarity algorithm may take advantage from the WordNet [35] semantic dictionary. WordNet is a lexical database of English. English words (Nouns, adjectives, verbs and adverbs) are grouped into sets of synsets which are cognitive synonyms. Each synset expresses a distinct concept. Wordnet graph's edges are the relations between synsets. Relations are set up by means of lexical relations and conceptual-semantic that results in a network of related words and concepts. These relations vary based on the type of word. For the current work, we limit the considered word types to nouns and verbs and we limit the relations to the following:

For nouns: hypernyms, hyponyms, holonym, and meronym.

For verbs: hypernym, troponym, entailment, and coordinate terms.

The steps involved in the semantic similarity are the following:

Inputs:

- Let X and Y two alerts;
- DX the sentence that correspond to the description of the alert X;
- DY the sentence that correspond to the description of the alert Y.

Output:

- Similarity score $sim(X,Y)$ between X and Y

Procedures:

1) If $X = Y$ then $sim(X,Y) = 1$; exit!
2) Tokenization of DX and DY;
   - Remove the stop words;
   - Remove the articles;
   - Split the sentences into a list of words (tokens);
   - We denote m to be the number of tokens of DX, and n to be the number of tokens of DY.
3) Identify the eventual senses of each token using Wordnet.
4) Identify the best sense of each token. This step takes advantage of the Word Sense Disambiguation (WSD) algorithms to identify the most appropriate sense of a word used in a given sentence, when the word has multiple senses (polysemy).
5) Similarity $sim(t_i, t_j)$ between each couple of two senses (t_i, t_j) where ti is the i^{th} sense in the description of the first alert and t_j is the j^{th} sense in the description of the second alert. The step makes use of the Resnik Method as explained later.
6) Similarity of tokens without senses. If a word does not exist in the dictionary, such that in the case of abbreviations and acronyms, we use the following binary similarity measure:
$$sim(t_1, t_2) = \begin{cases} 1 & if \quad t_1 = t_2 \\ 0 & Otherwise \end{cases}$$
7) Build the similarity matrix between all the tokens of both alerts.
8) Identify the best similarity matching: for each token from X, identify a token from Y that maximizes the similarities between all token of X and Y.
9) Compute the similarity between X and Y: $\sim (X,Y)$.

The first two steps of the algorithm are easy. Regarding Step 3, there are three categories of the proposed techniques; the dictionary-based methods, the completely unsupervised methods, and the supervised machine learning methods based on a corpus of manually sense-annotated examples. An example of this first approach is the Micheal Lesk algorithm [2, 23]. The objective of the algorithm is to count the number of words that are shared between two glosses (definitions). The more overlapping the words, the more related the senses are. Given a word to be disambiguated, the dictionary definition of each of its senses is compared to the glosses of every other word in the phrase. The sense whose gloss shares the largest number of words with the glosses of the other words is selected. The method begins anew for each word and does not utilize previously assigned senses. Formally, given two words w_1 and w_2, the score of each pair of word senses $S_1 \in Senses(w_1)$ and $S_2 \in Senses(w_2)$:

$$score_{Lesk}(S_1, S_2) = |gloss(S_1) \cap gloss(S_2)|$$

where $gloss(S_i)$ is the set of words in the definition of sense S_i of word w_i. The senses that maximize the score formula are assigned to the respective words. The WordNet glosses (definitions) might be used as dictionary definition database. The following pseudo code describes the original Lesk algorithm [23].

Algorithm 1 Lesk algorithm

```
Begin
for every word $w[i]$ in the alert do
    let $best\_score = 0$
    let $best\_sense = null$
    for every sense $sense[j]$ of $w[i]$ do
        let score = 0
        for every other word $w[k]$ in the alert, $k! = I$ do
            for every sense $sense[l]$ of $w[k]$ do
                $score = score + number$ of words that occur
                in the gloss of both $sense[j]$ and $sense[l]$
            end for
        end for
        if $score > best\_score$ then
            $best\_score = score$
            $best\_sense = w[i]$
        end if
    end for
    if $best\_score > 0$ then
        return $w[i]$ the word w[i] has is the best sense.
    else
        return nil
    end if
end for
End
```

The Lesk algorithm requires the calculation of $|Senses(w_1)|.|Senses(w_2)|$ gloss overlaps. In a context of n words, we need to compute $\prod_{i=1}^{n} |Senses(wi)|$ overlaps, which require an exponential number of steps. However, this is not problematic since the unification process is offline. The goal of Step 4 is to compute the similarity $sim(t_i, t_j)$ between each sense couple (t_i, t_j) where t_i is the i^{th} sense in the description of the first alert and t_j is the j^{th} sense in the description of the second alert. In WordNet, if a word has more than one sense, it will appear in multiple synsets at various locations in the graph. WordNet defines relations between synsets and relations between word senses. Figure 5 shows an example of the graph generated by WordNet including nouns and verbs.

Furthermore, we take advantages from the Resnik approach [27] to compute similarity between senses. The approach is based on the notion of information content. The method assumes that one criterion of similarity between two concepts is "the extent to which they share information in common", which in an IS-A taxonomy can be computed by examining the relative position of the most-specific concept that subsumes them both. An information content $IC(c)$ of a concept c is the probability $p(c)$ of finding an instance of the concept c in a given corpus. Following the standard argumentation of information theory [28], the information content of a concept c can be quantified as negative the log likelihood:

$$IC(c) = -logp(c)$$

IC has a lower value for the more abstract a concept.

Regarding the similarity between concepts, Resnik stated that, the more information two concepts share in common, the more similar they are. That means that, given the taxonomy graph, the shorter the path from one node to another, the more similar they are. Thus, the Resnik method only considers the information content (IC) of lowest common subsumer (LCS). A LCS is a concept in taxonomy (WordNet in our case), which has the shortest distance from the two given concepts. That is, the LCS of two synsets is the most specific common subsumer of the two synsets (most specific ancestor node). The similarity between two concepts $c1$ and $c2$ is then defined as:

$$sim(c1, c2) = -logp(lcs(c1, c2))$$

Note that the Resnik method assume that a root node of the taxonomy graphs exists which is not always true because the different top nodes of each word type taxonomy are not joined. For that, we create a root node that joins all the top node of the different taxonomies. Then, a path will certainly exist between any two concepts.

The probability $p(c)$ of finding an instance of the concept c in a given corpus is calculated as the following:

$$p(c) = \frac{\sum_{n \in W(c)} count(n)}{N}$$

where $W(c)$ is the set of words in the WordNet corpus whose senses are subsumed by concept c, and N is the total number of word (noun) tokens in the corpus that are also present in WordNet. Resnik used the Brown Corpus of American English as the corpus.

Now, given two words w_1 and w_2, the similarity between them is computed as follow:

$$sim(w_1, w_2) = \max_{c1 \in s(w_1), c2 \in s(w_2)} sim(c1, c2)$$

where $s(w)$ is the set of concepts in the taxonomy that are senses of word w [27]. That is, the similarity between two words is equal to that of the most-related pair of concepts that they denote.

Step 6 aims to build a matrix that summarizes the similarity between the tokens of the two alerts. The result is the similarity relative matrix $R[m, n]$ of each pair of token senses, $R[i, j]$ is the similarity between the token

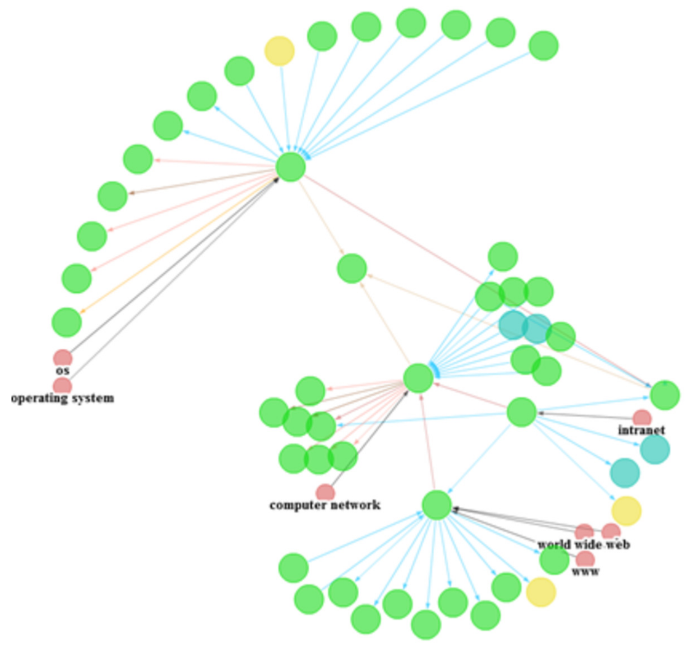

Figure 5: An example of wordNet taxonomy that includes the word types noun and verb and the relation between them. Image generated by Wordnet Editor [36]

sense i of the first alert and the token sense j of the second alert. The matrix R is the result of similarity measures of Steps 4 and 5.

The Step 7 is little bit tricky, the goal is to identify the best similarity matching between all the tokens of both alerts X and Y in a way that maximizes the overall similarity scores. Computing semantic similarity between two sentences could be formulated as the problem of computing the maximum total matching weight of a bipartite graph. The nodes of the graph are the tokens of X and Y and the edges are $R[i, j]$, the weight of the edge connecting from token $i \in X$ to token $j \in Y$. Let G denotes such a graph. The graph is often non balanced bipartite graph since the number of tokens in the first alert is often not equal to the number of tokens of the second alert ($|X| \neq |Y|$). Also, the graph is a complete bipartite graph because for any two vertices $i \in X, j \in Y$, ij is an edge in the graph G. The goal of the maximum total matching weight of a bipartite graph is to find a matching M in G which maximizes, over all possible matching, the total amount, $(M) = \sum_{ij \in M} R[i, j]$, of cost consummated by M. The maximum total matching weight of a bipartite graph could be formulated as an assignment problem [5].

The assignment problem consists on a number of agents and a number of tasks. Any agent can be assigned to perform any task for a certain cost. The goal is to perform all tasks by assigning exactly one agent to each task in such a way that the total cost is minimized. The main difference between the maximum total matching weight of a bipartite graph and the assignment problem is that the first problem tries to find the matching that maximizes the cost and the last problem tries to find the optimal assignment that minimizes the cost. One famous algorithm that could solve the assignment problem, in polynomial time of the number of nodes of X, is the Hungarian algorithm [22]. However, the method requires two firm conditions:

$$\begin{cases} linear \quad assignment \quad problem : |X| = |Y| & (1) \\ the \quad goal \quad is \quad to \quad minimize \quad the \quad overall \quad cost & (2) \end{cases}$$

The first condition imposes that the number of agents and tasks are equal which means that the number of token in X must be equal to the number of tokens in the second alert. This problem could be overcome by adding dummy tokens in the small token set with minimum cost 0 (see Figure 6).

Whereas, the second condition impose a minimization problem of the cost which is in contrast with our goal to maximize the overall similarity, two hints exist that allow using the Hungarian method for a maximization problem [9]. The first is to multiply the matrix R by -1. The second method suggest to replace each element R_{ij} by $max(R) - R_{ij}$. Where $max(R)$ is the maximum value that exist in R. Since the elements of R are probabilities, the transformation will be reduced to $R_{ij} = 1 - R_{ij}$.

The last Step 8 allows to compute the total similarity score $sim(X, Y)$. This is done by combining the match results of the previous step into a global similarity score for both alerts X and Y:

$$sim(X, Y) = \frac{2 * \sum_{x \in DX, y \in DY}^{min(|DX|,|DY|)} sim(x, y)}{|DX| + |DY|}$$

This global similarity is deduced by dividing the sum of similarity scores of all match tokens of both sentences X and Y (Step 7) by the total number of both tokens.

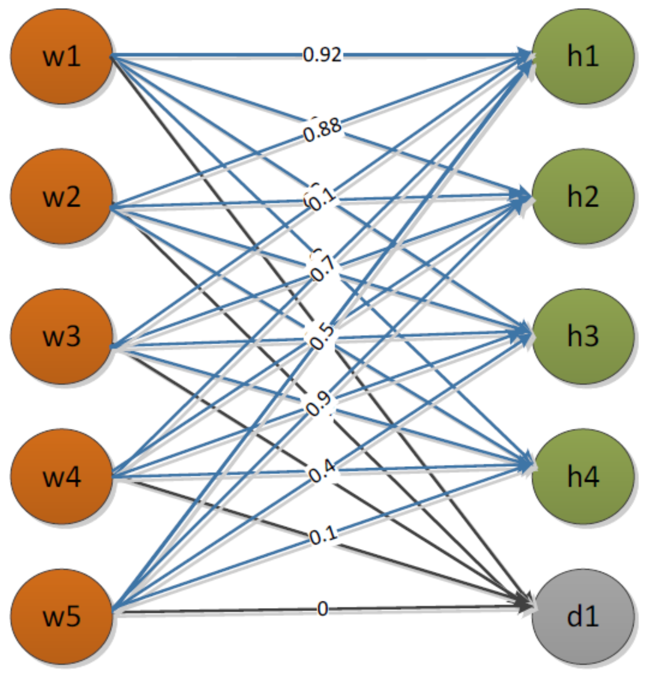

Figure 6: Best matching problem. wi are the token senses of the first alert W, h_j are the token senses of the second alert H. Since $|W| < |H|$, a dummy node d_1 is added in H. The edges oriented to d_1 have cost 0.

6 Classification of New Alert Signatures

When a new alert signature A_a is added to the alert set of a given alert set S_i. The alert A_a is compared to the list of unified alerts U except the alerts that have a corresponding alert in S_i. This is because two alerts in S_i could not be similar and indeed they could not have the same unified alert.

If the alert A_a is similar (the similarity measure is above the threshold ST defined in Step 3 of Section 4) to one or more unified alerts, then let u_b the unified alert that has the maximum similarity measure. The unification matrix is updated to set u_b as the unified alert corresponding to A_a. When a new IDS type is added to the architecture of the DIDS, the algorithm of Section 4 must be applied to the new alert set of the new IDS type. The Algorithm must take as input the unified alert set U and the new alert set.

7 Solving Alert Similarity Ambiguity

In Section 3, when u_a is a unified alert of several eventual alerts from the same alert set S_j, one cannot choose the alert from S_j with highest similarity because the difference between the selected alerts of S_j is minimal. An incorrect alert classification could lead to severe consequences. However, one of the selected alerts of S_j is appropriate since their similarity values are above the hard threshold.

The two alert sets and the similarity relation between their alerts can be described as a bipartite graph $G = (U, V, E)$. U and V denote the two partition nodes and E denotes the edges between U and V. In our case, U is the unified alerts and V is the alerts of S_j. E is the similarity relations that exists between the alerts. G is bipartite since there is no edge connecting two nodes from the same partition. That is, the similarity relation is defined between the alerts of U and the alerts of V. Formally, for every edge $uv \in E$, $u \in U$ and $v \in V$.

Figure 7 shows an example of alert similarity ambiguity formulated as a bipartite graph. The unified alert u_2 is similar to a_1, a_2 and a_3, whereas, the alert a_2 is similar to both alerts u_2 and u_3. To resolve the ambiguity, each unified alert must correspond to only one alert. A good approach is to select the maximum edges possible in a way that each unified alert is similar to only one alert in S_j.

The ambiguity resolution problem could be reduced to finding a set of pairwise non-adjacent edges denoted M. That is, no two edges share a common node. M must contain the largest possible number of edges. This problem is called the maximum matching graph problem or the maximum independent edge set problem.

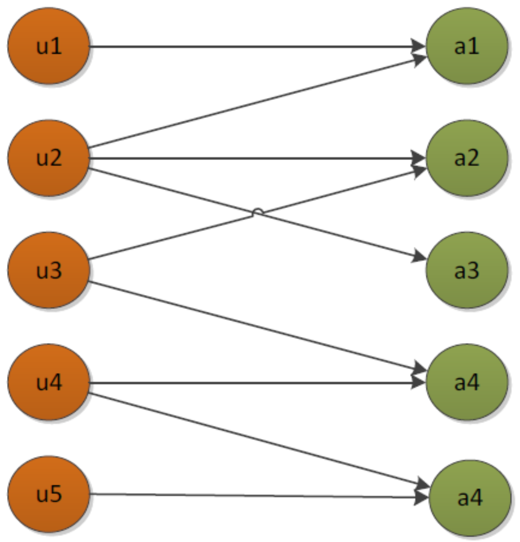

Figure 7: The alert similarity ambiguity is formulated as a bipartite graph. The left set contains the unified alerts and the right set contains the ambiguous alerts of S_j.

The Hopcroft-Karp algorithm [18] can solve the maximum matching problem for bipartite graph as shown in Algorithm 2.

The operator $\ominus$ used for two sets is the symmetric difference. Once the maximum matching M is found, the remaining unmatched alerts ru will be considered as new unified alerts. Formally, $ru = i : i \in Vandui \notin M$.

Algorithm 2 The Hopcroft-Karp algorithm

```
1: Input: − $G = (U, V, E)$ a bipartite graph
2: Output: − The maximum matching M of G
3: Begin
4: $M = \emptyset, U_0 = U, V_0 = V$
5: while $U_0 \neq \emptyset$ do
6:     $L_0 = U_0, k* := k := 0$
7:     while $L_k \neq \emptyset$ do
8:         construct a layered graph:
9:         for all $i \in L_k$ do
10:            $N_i = j : ij \in E \setminus M, j \in L_1 \cup L_2 \cup \cdots L_{k-1}$
11:        end for
12:        $L_{k+1} = \cup_{i \in L_k} N_i$
13:        if $L_{k+1} = \emptyset$ then
14:            return M
15:        end if
16:        if $L_{k+1} \cap V_0 \neq \emptyset$ then
17:            $k^* := k + 1$
18:            $L_{k+2} := \emptyset$
19:        else
20:            $L_{k+2} := i' : i'j \in M, j \in L_{k+1}$
21:        end if
22:        $k := k + 2$
23:    end while
24:    delete all vertices in $L_{k^*} \setminus V_0$
25:    mark all remaining vertices as unscanned
26:    $k := 1$ : path counter
27:    while $L_0 \neq \emptyset$ do
28:        $x_0 = i \in L_0, L_0 = L_0 \setminus i, l := 0$
29:        while $l \geq 0$ do
30:            while $x_l$ has unscanned neighbor in $L_{l+1}$ do
31:                Choose unscanned neighbor $x_{l+1}$
32:                Mark $x_{l+1}$ as scanned
33:                $l := l + 1$
34:                if $l = k^*$ then
35:                    $P_k = (x_0, x_1, \ldots, x_{k^*}); k = k + 1$
36:
37:                end if
38:            end while
39:            if $l < k^*$ then
40:                $l := l - 1$
41:            else
42:                $l := -1$
43:            end if
44:        end while
45:    end while
46:    $P := (P_1, P_2, \ldots, P_{k-1})$
47:    $M = M \ominus P$
48:    Update $U_0$, $V_0$ of unmatched vertices
49: end while
50: return M
51: End
```

8 Performance Analysis

The complexity of the unification process is not a big issue since the process will run offline. However, this section will show that the time complexity of the process is acceptable. Suppose that there are n monitoring sensors, each sensor has m alerts, each alert has an average of p tokens, and each token has an average of q senses. The complexity times of the alert unification is: $c = c(indexation) + c(tokenization) + c(Lesk) + c(Resnik) + c(Hungarian) + c(Hopcroft - Karp)$.

The complexity of the indexation step is proportional to the number of alerts $(n{*}m)$. The tokenization performs an average of p steps for each alert which requires a linear time complexity. Lesk algorithm is applied for each alert and requires qp steps for each alert $(O(n * m * p * qp))$. The Hungarian algorithm runs in polynomial time $O(p4)$ for each alert, and $O(n * m * p4)$ for all the alerts. The Hopcroft-Karp will run occasionally when the alert sets include ambiguities. The time complexity of the algorithm is $O(|E|\sqrt{|V|})$ in the worst case where the E is the set of the edges and V is the set of nodes of the bipartite graph.

The time complexity of Lesk algorithm is the biggest complexity $O(n * m * p * qp)$. However, in real world scenarios, the alerts has about ten tokens, each token has an average of 3 senses [37]. The time complexity of Lesk algorithm is good enough. The overall complexity of the unification process is polynomial.

9 Conclusions and Future Works

The paper has stated the importance and the current uses of the distributed monitoring systems and their current challenges. The problem of heterogeneity of alerts raised by different sensors is widely discussed in the literature. We introduced an approach of an automatic alert unification system that deals with heterogeneous alert signatures. The result is a set of unified alerts. The method used advanced linguistic models and optimization models in order to perform a semantic comparison between alerts. The method resolved also the problem of ambiguity of very similar alerts using an efficient optimization method.

A number of works still opened for future investigations. Regarding the current work, the performance results are acceptable for an offline alert unification. However, there are several monitoring systems that generate unpredictable alert signatures and the alert unification would be done on the fly. The current unification method should be adapted to such scenarios. Second, the paper solves the problem raised by the introduction of a new alert and how to know if it belongs to an existing unified alert. But what about feeding several new alerts at the same time? Is there any method that is faster that processing the new alerts one by one?

References

[1] L. Antonie, K. Inwood, D. J. Lizotte, J. A. Ross, "Tracking people over time in 19th century Canada

for longitudinal analysis," *Machine Learning*, vol. 95, no. 1, pp. 129–146, 2014.

[2] S. Banerjee and T. Pedersen, "An adapted Lesk algorithm for word sense disambiguation using WordNet," in *Computational Linguistics and Intelligent Text Processing*, pp. 136–145, Springer Berlin Heidelberg, 2002.

[3] T. Bass, "Intrusion detection systems and multisensor data fusion", *Communications of the ACM*, vol. 43, no. 4, pp. 99–105, 2000.

[4] M. Bateni, A. Baraani, and A. Ghorbani, "Using artificial immune system and fuzzy logic for alert correlation," *International Journal of Network Security*, vol. 15. no. 3, pp. 190–204, 2013.

[5] R. Burkard, M. Dell'Amico, and S. Martello, *Assignment Problems*, Social for Industrial and Applied Mathematics, Philadelphia, PA, USA.

[6] P. Christen, "Data Matching – Concepts and Techniques for Record Linkage, Entity Resolution, and Duplicate Detection," *Data-Centric Systems and Applications*, Springer, 2012.

[7] B. V. Dasarathy, "Sensor fusion potential exploitation Innovative architectures and illustrative applications," *Proceedings of IEEE*, vol. 85, no. 1, pp. 24–38, Jan. 1997.

[8] H. Debar, D. Curry, B. Feinstein, *The Intrusion Detection Message Exchange Format (IDMEF)*, Internet Engineering Task Force, IETF RFC 4765 (Experimental), 2007.

[9] V. R. Dondeti and H. Emmons, "Max-min matching problems with multiple assignments," *Journal of Optimization Theory and Applications*, vol. 91, no. 2, pp. 491–511, Nov. 1996.

[10] W. Elmenreich, "A review on system architectures for sensor fusion applications," in *Software Technologies for Embedded and Ubiquitous Systems*, pp. 547–559, 2007.

[11] A. A. Ferreira, et al., "Self-training author name disambiguation for information scarce scenarios," *Journal of the Association for Information Science and Technology*, vol. 65, no. 6, pp. 1257–1278, 2014.

[12] Z. Fu, M. Boot, P. Christen, and J. Zhou, "Automatic record linkage of individuals and households in historical census data," *International Journal of Humanities and Arts Computing*, vol. 8, no. 2, pp. 204–225, 2014.

[13] P. Garcia-Teodoro, J. E. Díaz-Verdejo, G. Maciá-Fernández, and E. Vázquez, "Anomaly-based network intrusion detection: Techniques, systems and challenges," *Computers & SecurityZ*, vol. 28, no. 1, pp.18-28, 2009.

[14] N. A. Giacobe, "Application of the JDL data fusion process model for cyber security," in *SPIE Defense, Security, and Sensing*, pp. 77100R, International Society for Optics and Photonics, 2010.

[15] D. L. Hall, *Mathematical Techniques in Multisensor Data Fusion*, Boston, MA: Artech House, 1992.

[16] D. L. Hall and J. Llinas, "An introduction to multisensor data fusion," *Proceedings of IEEE*, vol. 85, pp. 6–23, Jan. 1997.

[17] T. C. Henderson and E. Shilcrat, "Logical sensor systems," *Journal of Robotic Systems*, vol. 1, no. 2, pp. 169–193, 1984.

[18] J. E. Hopcroft and R. M. Karp, "An n5/2 algorithm for maximum matchings in bipartite graphs," *SIAM Journal of Computing*, pp.225–231, 1973.

[19] J. Jonas and J. Harper, *Effective Counterterrorism and the Limited Role of Predictive Datamining*, Cato Institute, 2006.

[20] P. Kabiri, A. A. Ghorbani, "A rule-based temporal alert correlation system," *International Journal of Network Security*, vol. 5. no. 1, pp. 66–72, 2007.

[21] G. Kalpana, R. P. Kumar, and T. Ravi, "Classifier based duplicate record elimination for query results from web databases," in *IEEE Trendz in Information Sciences & Computing*, pp. 50–53, 2010.

[22] H. W. Kuhn, "The hungarian method for the assignment problem," *Naval Research Logistic Quarterly*, vol. 2, pp. 83-97, 1955.

[23] M. Lesk, "Automatic sense disambiguation using machine readable dictionaries: How to tell a pine cone from a ice scream cone," in *Proceedings of the 5th Annual International Conference on Systems Documentation*, pp. 24–26, 1986.

[24] R. C. Luo and M. G. Kay, "A tutorial on multisensor integration and fusion," in *Proceedings of 16th Anuual Conference on IEEE Industrial Electronics*, pp. 707–722, 1990.

[25] R. C. Luo and M. G. Kay, "Multisensor fusion and integration in intelligent systems," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 19, no. 5, pp. 901–931, Sep./Oct. 1989.

[26] R. C. Luo and M. G. Kay, *Multisensor Integration and Fusion for Intelligent Machines and Systems*, Norwood, MA: Ablex Publishing, 1995.

[27] P. Resnik, "Using information content to evaluate semantic similarity in a taxonomy," in *Proceedings of the 14th International Joint Conference on Artificial Intelligence*, pp.448–453, 1995.

[28] S. Ross, *A First Course in Probability*, Macmillan, 1976.

[29] C. Rudin, and K. L. Wagstaff, "Machine learning for science and society," *Machine Learning*, vol. 95, no. 1, pp. 1–9, 2014.

[30] H. Sallay, M. Rouached, A. Ammar, O. B. Fredj, K. Al-Shalfan, and M. B. Saad, "Wild-inspired intrusion detection system framework for high speed networks ($\phi|\pi$) IDS framework," in *Privacy Solutions and Security Frameworks in Information Protection*, pp. 241, 2012.

[31] C. Schulz, et al., "Exploiting citation networks for large-scale author name disambiguation," *EPJ Data Science*, vol. 3, no. 1, pp. 1–14, 2014.

[32] D. Smith and S. Singh, "Approaches to multisensor data fusion in target tracking: A survey," *IEEE*

Transactions on Knowledge and Data Engineering, vol. 18, no. 12, pp. 1696–1710, Dec. 2006.

[33] W. Su, J. Wang, and F. H. Lochovsky, "Record matching over query results from multiple web databases," *IEEE Transactions on Knowledge and Data Engineering*, vol. 22, no. 4, pp. 578–589, 2010.

[34] U.S. Dept. Defense, *Data Fusion Lexicon*, Data Fusion Subpanel of the Joint Directors of Laboratories, Technical Panel for C3, 1991.

[35] *WordNet is a Large Lexical Database of English*, 2016. (`http://wordnet.princeton.edu/`)

[36] *Wordnet Editor*, 2016. (`http://wordventure.eti.pg.gda.pl/wne.html`)

[37] WordNet statistics, Aug. 2015. (`https://wordnet.princeton.edu/wordnet/man/wnstats.7WN.html`)

Ouissem Ben Fredj received the BE degree in computer science from the University Manar II, Tunisia in 2002. He obtained the MS in computer science from University of Henri Poincare, France in 2003. He Obtained the PhD degree in computer science from University of Val d'Essonnes, France in 2007. He is currently an assistant professor of computer science at Taif University, Saudi Arabia. His research interests include Vulnerability Assessment, Network Security, and Distributed Systems.