NIST Internal Report
NIST IR 8414

Making Semantic Structures Explicit:

Developing and Evaluating Tools and Techniques to Support Understanding of Large Cybersecurity Corpora

Ira Monarch
Jacob Collard
Sangjin Shin
Eswaran Subrahmanian
T. N Bhat
Ram Sriram

This publication is available free of charge from:
https://doi.org/10.6028/NIST.IR.8414

NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY
U.S. DEPARTMENT OF COMMERCE

NIST Internal Report
NIST IR 8414

Making Semantic Structures Explicit:

Developing and Evaluating Tools and Techniques to Support Understanding of Large Cybersecurity Corpora

Ira Monarch
Independent Consultant
Pittsburgh
Jacob Collard
Software and Systems Division
Information Technology Lab
Sangjin Shin
Software and systems
Information Technology Lab

Talapady N. Bhat
Biosystems and Biomaterials
Material Measurement Lab
Eswaran Subrahmanian
Software and Systems Division
Information Technology Lab
Ram Sriram
Software and systems division
Information Technology Lab

This publication is available free of charge from:
https://doi.org/10.6028/NIST.IR.8414

December 2022

U.S. Department of Commerce
Gina M. Raimondo, Secretary

National Institute of Standards and Technology
Laurie E. Locascio, NIST Director and Under Secretary of Commerce for Standards and Technology

NIST Technical Series Policies
Copyright, Fair Use, and Licensing Statements
NIST Technical Series Publication Identifier Syntax

Publication History
Approved by the NIST Editorial Review Board on 2022-02-04

How to Cite this NIST Technical Series Publication
Monarch I, et al. (2022) Making Semantic Structures Explicit: Developing and Evaluating Tools and Techniques to Support Understanding of Large Cybersecurity Corpora. (National Institute of Standards and Technology, Gaithersburg, MD), NIST Internal Report (IR) 8414. https://doi.org/10.6028/NIST.IR.8414

NIST Author ORCID iDs
Ira Monarch: 0000-0002-2587-173X
Jacob Collard: 0000-0002-4794-8363
Talapady Bhat: 0000-0002-8396-6622
Eswaran Subrahmanian: 0000-0002-4794-8363
Sangjin Shin: 0000-0003-3687-7587
Ram Sriram: 0000-0001-8602-4748

Abstract

This report describes the adaptation, composition and use of natural language processing, machine learning and other computational tools to help make implicit informational structures in very large technical corpora explicit. The tools applied to the corpora automatically build normalized multi-word term structures, which in turn are used to build taxonomies, semantic schema, topic models, and knowledge graphs. In our cybersecurity use case, we apply these tools to help us understand the threat landscape as exhibited in the Common Vulnerabilities and Exposures (CVE) and the National Vulnerability Database (https://nvd.nist.gov/) corpora with the aim of proactively anticipating threats. The use case provides the context for the development, use, and evaluation of the automated tools and processes based on them. The latter are incrementally and iteratively evaluated and improved as they are developed and used. Local evaluation and improvement come before global evaluation. We believe that performing a global evaluation of text processing methodologies and processes currently exemplified by the Text REtrieval Conference (TREC), Document Understanding Conference (DUC), and Text Analysis Conference (TAC) is worth pursuing, and we have done so using TREC. However, this report will mostly focus on local development, evaluation, and improvement. We will articulate various aspects of our approach by describing and showing 1) how the multi-word term based process for topic modeling is supported by semantic schema, taxonomic structures, and knowledge graphs; 2) the heuristic methods for performance evaluation of this modeling that include suggestions for measuring how well a topic is represented in the documents that are indexed to it; 3) how these local heuristic methods might be transformed into a new full-blown rigorous evaluation standard like TREC, DUC, and TAC, but with emphasis on their contribution to interpretation and understanding of very large corpora. With respect to 3), we will also briefly explore what parts of the heuristic process would need to be automated and an indication of the algorithms needed.

Key Words: Natural language processing, Cybersecurity, Topic models, Common Vulnerabilities and Exposures, Semantics, Root and Rule-based, Knowledge Graphs

Table of Contents

1. Introduction

We are bringing together tools and techniques that can be composed and developed into processes that support cybersecurity researchers and others who work in cybersecurity in various cognitive tasks. These cognitive tasks include: 1) enhanced search based on terminological structures – taxonomic, topical, and ontological – implicit in very large corpora but made explicit for search [1], 2) identifying, interpreting, and visualizing the contents and patterns contained in these corpora using taxonomies, semantic schema, topic models, topic trends and knowledge graphs ([2,3,4] for earlier efforts), and 3) carrying out trend analysis to detect emerging or escalating technology and research fronts. The aim is to support proaction either to advance new concepts and practices or, as in the case of cybersecurity, to thwart emerging or mounting threats. To help make implicit informational structures in very large technical corpora explicit, we adapt, compose and use Natural Language Processing (NLP) and Machine Learning (ML) tools. The NLP tools automatically form syntactically based normalized single-and multi-word term structures, which in turn are used to create normalized semantic schema, taxonomies, topic models, and knowledge graphs.

Our current work has been applied to the Common Vulnerability and Exposures (CVE) List developed and maintained by Mitre [5]. A subset is also supported with enhancements by NIST, which provides other useful information, such as linking each CVE to a CWE and assigning the CVE a severity score. The CVE entries (numbering over 122,000 circa 2019) provide a basis for gaining an understanding of the cybersecurity threat landscape. The CVE List currently includes 1) enhanced searching of the CVE list based on semantic structures derived from R&R terminologies (see https://randr.nist.gov/mgi/Default.aspx?dataSource=CVE); 2) aids for understanding and interpreting the threat landscape as captured in CVE text entries. These conceptual aids include: taxonomies, semantic schema, topical models, topic trends, and knowledge graphs; and 3) use of 1 & 2 to detect emerging or spiking cybersecurity risks and vulnerabilities to support proactive threat management.

During CVE's lifetime (1999 to the present), a tacit, de facto standard schema for reporting cybersecurity vulnerabilities and exposures developed. We are working on making this implicit schema more explicit to evaluate how well it works, where it does not work, where it is inconsistent, where it is incomplete, and finally to improve it. Current suggestions for improvement involve more diverse and elaborated partitions of the schema and taxonomies of expressions instantiating the schema. Much like a case frame or argument structure, the schema specifies an n-ary relation to the main verb of the sentence. In CVE, it is usually some form of the verb "allow" where other schema elements are found in a sentence (see section 3.2 Semantic Schema for more details).

In addition to making suggested improvements to the implicit schema guiding a large number of CVE descriptions, we also critically evaluate the vulnerabilities by type managed at the CVE Details website (see especially the interactive table, Vulnerabilities by Type (https:// www.cvedetails.com/vulnerabilities-by-types.php) (see section 4.2). We describe how our work uses, refines, elaborates, and improves the current CVE Details Types partitioning and characterization of vulnerabilities. While CVE is an important cybersecurity information source, and while CVE Details is an example of how information can be organized for such an information source, there are other cybersecurity information sources (Exploit Database, https://www.exploit-db.com/) and information organizing (Common Attack Pattern Enumeration and Classification (CAPEC, http://capec.mitre.org/about/index.html). There are also more freewheeling hacker websites and mailing lists that can also be analyzed and contribute further information to be classified. The ultimate goal would be to support an evolving data and community driven standard for cybersecurity reporting as a basis for producing cybersecurity knowledge.

Our work gets its impetus from two innovations: 1) the use of NLP to generate normalized multi-word term structures, in addition to single-word terms, to be the constituents for automatic generation of semantic assemblages that can be visualized like taxonomies, topic models [6,7] and knowledge graphs and 2) a new direction for evaluation of topic models constructed with the addition of multi-word terms, emphasizing a) how the use of topic models can contribute to a better understanding of corpora (see [8] whose work is a precursor to ours) and b) how from locally positioned users and developers who together evaluate and improve the development of the tools and methods for using them, more generic and standardized evaluation methods can grow.

Concerning topic modeling in point (1) above, while there have been attempts to enhance such modeling using multi-word units, for example, n-grams or phrases, there have been very few, if any, approaches where constituents of topics are syntactically derived multi-word units or grammatical phrases. The ones we found were based on *non-syntactic* approaches [9,10,11,12,13,14]. Concerning point (2a), intrinsic methods such as some approaches to held-out likelihood [15,16] do measure whether terms that have a high probability of belonging to a topic in a training set will have a similarly high probability of belonging to that topic in the test set. However, they only provide a relative measure of performance, comparing one topic modeling approach to another, without acknowledging that the difference among models may not be significantly different, nor that for other tasks the difference may be reversed. In addition, such measures do not provide insight into how topics are informative to users.

Similarly, while extrinsic tasks such as document classification and information retrieval do provide quantitative measures, they often depend on so-called "ground truth" classification. These latter are actually subjective and error prone. Also, such quantitative measures as precision and recall by themselves provide little insight into how topics help users understand a corpus. Understanding also requires evaluating the inferred topics themselves [16] and how documents share meaning with the topics they are indexed to. More generally, extrinsic evaluation tasks are concerned with search that brings back documents with thus and so words in them or brings back documents that have been labeled as belonging to one set of documents or another and none that do not. They are not concerned with bringing back documents that contribute to an increased understanding of the concepts that correspond to the words or labels, nor how the documents brought back introduce related concepts that increase users' understanding of the corpus or the domain of the corpus.

In our view, while quantitative measures are essential in evaluating topic modeling, ways of increasing understanding also need to be evaluated. Not much attention has been devoted to constructing evaluations that try to take this into account. The objective of this report is to argue that in very large corpora, making tacit semantic structures like taxonomies, topic models, and knowledge graphs explicit for search, mining and discovery provides a basis for constructing at least qualitative and even quantitative evaluations of increases in understanding. Our quantitative view for evaluating increases in understanding relies on two premises: 1) semantic structures, such as the semantic frames we are focusing on, are tacitly constructed via social-practical-linguistic activity, and 2) making these tacit semantic structures explicit to a considerable extent depends on them being recorded in text, especially very large corpora, and having the means to process such texts automatically and accurately.

Critical to this data-driven and quantitative approach is the tacit role of scorekeeping in norm instituting social practices governing language use.[1] Scorekeeping is meant to express the idea of keeping track of one's own and others' commitments and entitlements concerning how singular terms, predicates, sentences, etc., are used in discourse. In our approach, the idea of scorekeeping applies to written

[1] See Robert Brandom, *Making It Explicit: Reasoning, Representing and Discursive Commitment* [17], for a related quantitative but non-computational account of the relation of thinking and understanding to language use via the idea of scorekeeping.

discourse in a specific domain or sublanguage. Also very important is that such tacit practice-based and normative semantic scorekeeping alters and manages scores systematically based on the written performances produced by each sublanguage practitioner in a given domain. We explore whether such tacit scorekeeping can be made explicit via ranking terms, topics, and documents in a topic model based on probabilistic modeling. The use of probability in topic modeling will be discussed more in the Topic Models section 3.3.

Such language-based scorekeeping is similar to the sublanguage approach taken by Harris and his colleagues in The Form of Information in Science: Analysis of an Immunology Sublanguage [18]. The book analyzes a sample of articles from the subscience of immunology using a formal method of capturing word combinations in formulaic structures. The method was an explicit way of keeping track of the regularities of word combinations and in relation to other word combinations. The sampling of articles was small because the analysis was done by hand rather than computationally. However, subsequent work in sublanguage analysis has shifted to automating the analysis tasks involved [19]. Similarly, semantic networks were introduced into cognitive science and NLP in the 1960s and 1970s and have developed in various ways since then. The standard representation of semantic networks in conventional computers uses frames as data structures [20,21]. What we call a semantic schema is very similar to a version of frame systems called case frames. Knowledge graphs we will also discuss in the report are a kind of semantic network. Both are directed graphs consisting of vertices, labeled appropriately with terms used to express represented concepts, and edges, representing conceptual relations between concepts. Semi-automated and automated approaches to generating semantic networks or knowledge graphs began in early 2000's and continue up to the present [22, 23]. Automated approaches that combine sublanguage approaches with semantic networks have also been recently explored [24]. Knowledge graphs will be discussed in section Knowledge Graphs.

The report is divided into three main sections and a Conclusion (section 5) that summarizes the main points but with an eye to future work. Section 2 describes the term structure building blocks that are the constituents of the semantic structures as exhibited in the patterns of language use in a corpus. It includes a summary of our NLP implementation using spaCy, a popular library for advanced NLP in Python (https:// spacy.io/), and our approach to generating term structures based on the parsed output of very large corpora. Section 3 describes the four semantic structures exhibited in patterns of language use, taxonomies, CVE semantic schema, topic models, and knowledge graphs and how they are derived. Section 4 describes how to evaluate the detected patterns of language. In other words, how do identifiable semantic structures presented to a researcher or other human interpreter help them to navigate better and search huge corpora like CVE? Subsection 4.1 describes how the current NIST project did these evaluations, and section 4.2 describes how these evaluations could be made more generally applicable. Our overall aim is to make explicit language patterns such as semantic structures and use them to support interpretation, understanding, and knowledge discovery in very large technical corpora.

2. Semantic Building Blocks: The R&R-Based NLP Approach

Some natural languages (such as Sanskrit, Inuktitut, Georgian, and German) create compound noun structures on demand using well-defined rules. R&R emulates this kind of language. The approach uses NLP powered by a high-performance neural dependency parser to parse every sentence in the input corpus. The R&R process then applies the emulated natural language rules to extract constituency trees from parser output and converts them into normalized terms. Semantic structures (taxonomies, semantic schema, topic models, and knowledge graphs) are automatically derived using semantic rules, computational techniques, and pipelines of modules applied to R&R terms creating term structures that capture domain concepts and their relationships [1].

2.1. R&R Parsing: Tree-like structures represent phrases

The phrases to be represented are either noun or verb phrases. The initial example shows how parsing works is the noun phrase, “several falls out of bed.” Fig.s 1-4 show the four phases of the parsing process.

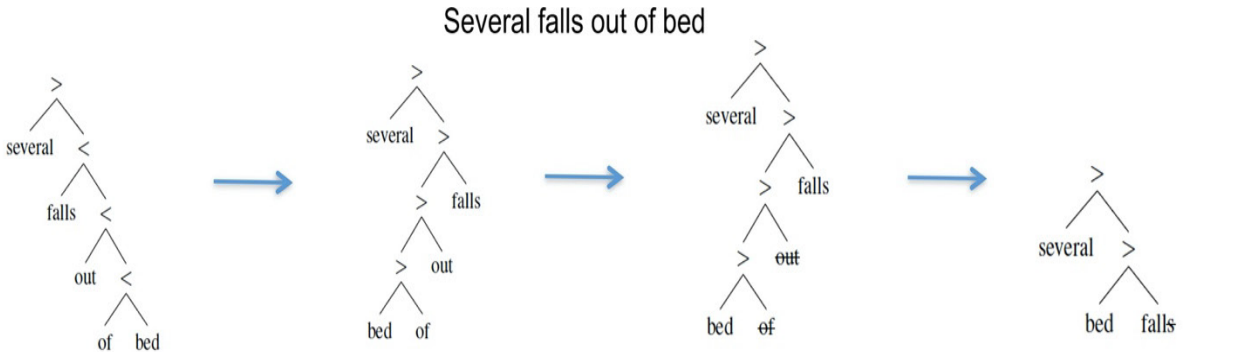

Fig. 1: A simple binary consistency tree Fig. 2: A normalized binary consistency tree Fig. 3: A stop-filtered constituency tree Fig. 4: A lemmatized constituency tree

Figs 1-4: Transformation of Constituency Trees

Fig. 1 shows a simple binary constituency tree. Each node in the tree marks the location of the phrase’s head, which determines the grammatical categories for the phrase. In this example, “several falls out of bed” is a noun phrase, defined by its head “falls,” The subphase “out of bed” is a prepositional phrase, defined by its head “out” and so on. The marker ‘<” indicates that the head appears in the left daughter of the phrase, and in Fig. 4 indicates that the head appears in the right daughter of the phrase. Following these markers allows for the identification of the head of any phrase. Fig. 2 shows the transformation into a normalized binary constituency tree. To normalize the word order of the tree: For each node in the tree, if the head is in the left subtree, the order of the two subtrees is switched so that the left subtree appears on the right. This normalization allows for multiple expressions of the same basic concept to be represented in the same way, like “several falls from bed” or “several bed falls,” and more generally, prepositional phrases like “denial of service” and ”service denial.”

Fig. 3 shows the transformation of the tree into a stop-filtered constituency tree. For stop words, we remove words according to the stop list provided by spaCy, as well as all URLs, pronouns, prepositions, and determiners. Fig. 4 provides the final tree, a lemmatized constituency tree. Words are also lemmatized, removing extraneous morphological information such as plural morphemes and verbal inflections. Both stop-filtering and lemmatization are part of normalization.

2.2. Term Formation: Rule and Root Based NLP Approach to Terminology

To see how to convert a lemmatized constituency tree into a formal term, we turn to a more complex example, the noun phrase, “repeated episodes of severe chest pain.” Fig. 5 again shows a simple binary constituency tree that is transformed into a lemmatized constituency tree, Fig. 6, which is the normalized form of Fig. 5. Figs 7 & 8 show another example of transforming the verb phrase, “the patient was vomiting,” into a normalized verb phrase.

repeated episodes of severe chest pain

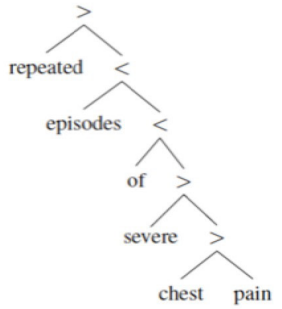

Figure 5: A more complex example

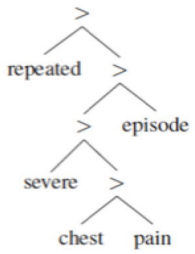

Figure 6: The normalized form of Figure 5

repeated:3:severe:1:chest:0:pain:2:episode

the patient was vomiting red material

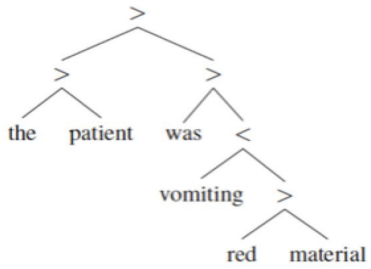

Figure 7: A verb phrase example

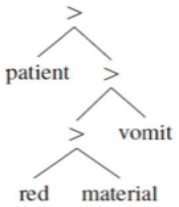

Figure 8: A normalized verb phrase

patient:2:red:0:material:1:vomit

Figs. 5-8: Converting Lemmatized Constituency Trees in Normalized Form into a Formal Term

The structured formalized R&R term is at the bottom of each figure. This term consists of a string of "roots" (or words) delimited by specialized markers of the form, :n: where n is an integer of 0 or greater, indicating constituency structure. For example, :0: is used to indicate that the two adjacent roots are leaves in the tree. The delimiter :1: is more distant than :0: but closer than :2: and so on. These delimiters can be used to reconstruct the original tree by combining roots into constituents, beginning with those delimited by :0: and continuing in ascending order. The formalized R&R term, repeated:3:severe:1:chest:0:pain:2:episode is a representation of "repeated episodes of severe chest pain." "patient:2:red:0:material:1:vomit" is a representation of "the patient was vomiting red material, and for the earlier example, the formalized R&R term "several:1:bed:0:fall" is a representation of "several falls out of bed."

2.3. Using *Latent Semantic Indexing (LSI)* and *Latent Dirichlet Allocation (LDA)* with R&R term structures for a Text Retrieval Conference (TREC) evaluation

While this report offers a new approach to topic modeling evaluation, we have used extrinsic evaluation techniques to assess the normalized terms created. Using Text Retrieval Conference (TREC) facilities, we summarize in this report work that we have already published [23]. In this work, R&R term structures are supplemented with output from *Latent Dirichlet Allocation (LDA)* and *Latent Semantic Indexing (LSI).* The resulting system performs competitively relative to the 2016 TREC Clinical Decision Support (CDS) track participants, improving the ability of a system to retrieve relevant results. The primary metric used in TREC 2016 is Inferred Normalized Discounted Cumulative Gain (infNDCG). This metric, shown in Equations (1), (2), and (3) in Table 1, takes into account the three levels of relevance judgment included in the gold-standard evaluation for TREC 2016 (relevant, possibly relevant, and not relevant) [23].

$$DCG_p = \sum_{i=1}^{p} \frac{rel_i}{log_2(i+1} \quad (1)$$

$$IDCG_p = \sum_{i=1}^{|REL_p|} \frac{2^{rel_i} - 1}{log_2(i+1} \quad (2)$$

$$nDCG_p = \frac{DCG_p}{IDCG_p} \quad (3)$$

Model	infNDCG
LDA+RR	0.2943
AutoSummary1	0.2815
MrkUmlXgb [Ira & Shin 2020]	0.2493
LSI+RR	0.2422
Cbnus	0.2382
UDellInfoCDS5	0.2362
ECNUrun5	0.2334
DUTHsaRPF	0.2265
SUmES	0.2222
CCNUSUMRI	0.2179
Mayoas	0.2146
LAD	0.2011
LSI	0.1806

Table 1: R&R + LDA/LSI extended-term structures appear twice in the top ten of 2016 TREC CDS

3. Semantic Structures: Making the Tacit Explicit

We next discuss the representation of four semantic structures: Taxonomic structures, Semantic schema, Topic modeling, and Knowledge graphs.

3.1. Taxonomic Structures

Taxonomies can be *automatically* built from the formally structured R&R terms, starting from the most general single word representing the most general class and the modifier:0:root primary unit representing one of its highest level subclasses. Representations of further subclasses are formed based on modifiers delimited by ascending :n:s greater than 1 that appear for that root and modifier:0:root primary unit. See Fig. 9 for the cybersecurity taxonomic structures attack:0:vector or attack:1:vector (attack:0|1:vector). Note, remote:0:attack:1:vector and local:1:attack:0:vector represent subclasses of attack:0| 1:vector and that the former has several interesting terms representing subclasses under it. These include remote:0:attack:0|1:vector, unknown:2:remote:0:attack:0|1:vector, 1:2:remote:1:attack:0|1:vector, user:0:assist:2:attack:0|1:vector, impact:2:attack:0|1:vector, etc. Moreover, two of these subclasses have terms representing sub-subclasses under them. On the other hand, local:1:attack:0:vector has no terms representing subclasses under it. In other words, there is a much more specific understanding of the former than of the latter.

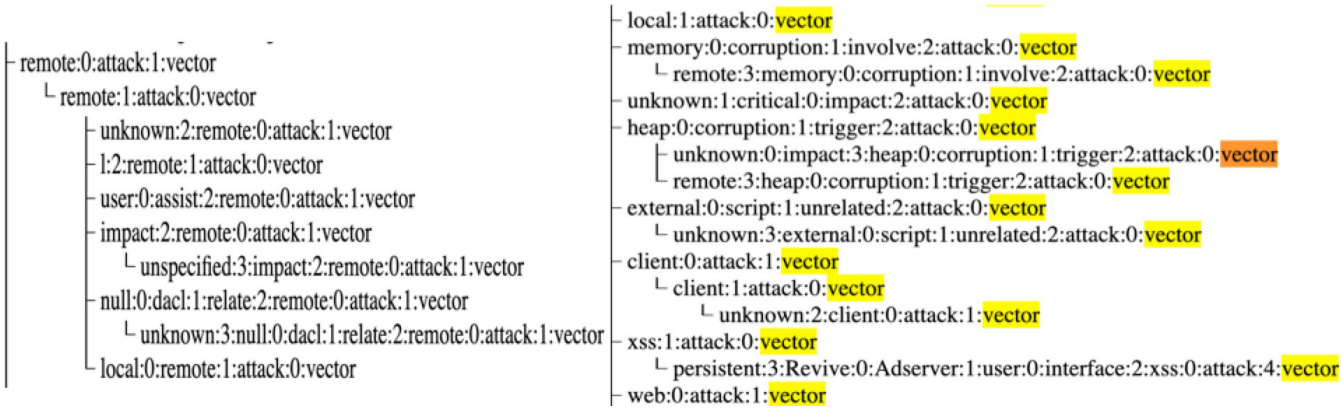

Fig. 9: Two cybersecurity taxonomic structures

3.2. Semantic Schema

While the taxonomic structures currently made explicit by R&R tools can be seen to represent classes, they are a literal rather than semantic classification in the sense that the structures are based on literal overlapping of words in terms. In semantic classification, two terms can belong to the same class even if they do not share the same classified or classifier word or phrase in common, e.g., not all named vulnerabilities share the word “vulnerability.” The above case also holds for attackers, attacks, and various exploitation techniques and methods. Semantic classes derive from sentences of repeating syntactic structure determined by the position in a sentence (e.g., subject position at the beginning of a sentence) or linguistic markers (e.g., prepositions), indicating that the noun or verb phrases delineated play the same semantic roles in sentences of the same syntactic structure. Semantic schemas allow the capture of the semantic classes that fill these repeating structures.

For example, let’s look at the following CVE record.

CVE-2010-1396: Use-after-free vulnerability in WebKit in Apple Safari before 5.0 on Mac OS X 10.5 through 10.6 and Windows, and before 4.1 on Mac OS X 10.4 allows remote attackers to execute arbitrary code or cause a denial of service (application crash) via vectors related to the contentEditable attribute and removing container elements.

Notice that CVE-2010-1396 has the following structure:

Use-after-free vulnerability comes at the beginning of the sentence designating a vulnerability. WebKit in Apple Safari comes after the preposition “in” and is the place or product in which the vulnerability is located, where the next “in” further specifies the location or product 5.0 on Mac OS X 10.5 through 10.6, and Windows before 4.1 on Mac OS X 10.4 comes after the preposition “before” and specifies that the vulnerability applies to Mac and Windows (using “on”) versions before some previous Mac and Windows versions; there is also a sub-schema specifying the range of versions using “through.” “allows” is the main verb instantiating the schema and is a signal for activating the schema that specifies a vulnerability in something and before something enabling someone to do something via something remote attackers is the noun phrase that immediately follows “allows” and is the subject (playing the role of an actor) of the subordinate clause to execute arbitrary code or cause a denial of service (application crash) is the compound infinitive verb (to “execute” or to “cause”) of the subordinate clause as what the subject or actor (“remote attackers”) brings about – either “to execute” –and what is being used in the executing, “arbitrary code,” or “to cause” something a “denial of service,” which is a consequence of the causing; a parenthetical phrase “(application crash)” following what, “denial of service” is asserted as caused is a

more proximate cause of it, or perhaps a kind of “denial of service.” Via (a prepositional marker indicating how the actor is going to bring off what is being done and used in the attack), which is in this case “vectors” characterized by the phrase “related to the contentEditable attribute and removing container elements.”

So, in a nutshell the Schema in the form of a case frame [41] is

> vulnerabilities
>
> located in something … before something … allow(s)
>
> someone (an attacker)
>
> to do something (an attack, what is being used in the attack, the attack basis, a consequence of the attack often part of a causal chain involved in the attack)
>
> via something, a method, or something utilized, an instrument or technique.

We found that around 80% of CVE records containing “denial of service” or “service:0:denial” instantiated a semantic schema. Initial investigation suggests that 65% or more of all CVE records exhibit this semantic schema with some slight variations, like the location of a vulnerability assuming the initial subject position of the sentence.

3.3. Topic Models

Topic modeling algorithms are statistical methods that analyze the terms in a collection of texts to discover the themes that run through them, how those themes are connected to each other (by sharing literal words and words with the same meaning), and how they change over time (topic trend analysis). The topics consist of terms that are associated with one another that can be interpreted as meaningful themes by human interpreters and that are repeated in documents across a corpus. Topic modeling algorithms do not require any

prior annotations or labeling of documents in a collection—the topics emerge from the analysis of the original texts. Topic modeling enables us to organize and summarize electronic archives at a scale that would be impossible by human annotation. However, this does not and should not mean that topic modeling is independent of human evaluation.

3.3.1. How Are Topic Models Generated?

LDA (see section 2.3) is the most familiar topic model. It is a Bayesian two-tiered mixture model that identifies (usually single) word co-occurrence patterns, which are interpreted as topics. LDA and other topic models are part of a larger field of *probabilistic modeling* (see [25]). In generative probabilistic modeling, data are treated as arising from a generative process that includes *hidden variables.* The observed variables are the words of the documents, and the hidden variables are the topic structures. The latter is a distribution of topics containing subsets of the words whose probability of belonging to the topic is not 0 or above some cutoff value. The generative process defines a *joint probability distribution* over both the observed and hidden random variables. Data analysis uses the joint distribution to compute a *conditional distribution* of the hidden variables given the observed variables. This conditional distribution is also called the *posterior distribution.* Inferring hidden topic structure from documents is a matter of calculating the posterior distribution, which is the conditional distribution of the hidden variables given the documents.

A topic model views topics as generators of documents in a corpus in the sense that all the content words come from one or another collection of words that with a high probability indicate topics. The model assumes that the topics are already specified before the documents have been generated. For each document in the corpus, its words are generated in a two-stage process: for each word in a given document 1) randomly choose a topic from a distribution, called the Dirichlet distribution,[2] of already configured topics that have a high probability of generating a particular document and 2) randomly choose a word from the selected vocabulary that constitutes the topic. In this way, each document exhibits a variety of topics, with some contributing more words to a given document or collection of documents than others. While documents in a collection draw from the same topics, they do so with different probabilities and from different distributions of the topics. The central computational problem for topic modeling is to reverse the generative process – instead of focusing on the hidden structure as likely generating the observed collection, rather reverse the focus on inferring the hidden topic structure from the observations. This process could be thought of as an abductive inference from the documentary data to a distribution of topics that accounts for this very data.

According to Blei [24], LDA was developed to fix an issue with a previously developed probabilistic model called *probabilistic latent semantic analysis*. That model was itself a probabilistic version of the seminal work on *Latent Semantic Analysis [LSA]*, [26, 27] which revealed the utility of the singular value decomposition (SVD) of the document-term matrix. By applying SVD, a linear algebra dimensionality reduction technique, the document-term matrix was reduced to latent factors. The goal was to identify broad semantic (correlation) structure within the documents by removing noise from uninformative factors (see also [28]). However, LSA could not account for uses of the same word in different contexts. To address this problem, Hofmann [29] proposed to model each word in a document as a sample from a mixture model, where the mixture components are multinomial random variables that can be viewed as representations of "topics." Words in documents are generated from topics, and a word that has different meanings in different documents can be generated from different topics.

LSA and probabilistic LSA prepared the way for topic models based on LDA. LDA extended probabilistic LSA from the word level to include the document level so that documents are seen as mixtures of topics for LDA. For LDA, each document is represented as a list of these mixture components in different proportions and thereby reduced to a probability distribution on a fixed set of topics. This distribution is the "reduced description" associated with the document. Observed words are modeled as being generated by the joint probability of two mixtures: 1) documents being a mixture of topics and 2) topics being a mixture of words. In addition to the document-term matrix, there is the term-topic matrix. The word-topic matrix provides a conditional probability for every term (row) given each hidden topic (column). Using these word probability distributions, any word can be rank ordered by a topic. Similar to SVD used in LSA, the probabilistic (mixture) nature of LDA acts similarly as a dimensionality reduction process by reducing the information about each document from the large number of columns (terms) to a much smaller number of columns (topics).

We disagree with the assumption that the order of documents does not matter. Topics change over time. Dynamic topic models [30] respect the ordering of the documents and give a richer posterior topical structure than LDA.

3.3.2. Structured Topic Modeling (STM): Including Document Covariates

STM, developed by [30], provides facilities for including metadata or covariates, such as names of technical journals or dates that can affect the prevalence with which a topic is exemplified. So, some

[2] Dirichlet allocation consists of the words of a document coming from the different topics of the Dirichlet distribution.

topics can be more frequently associated with specific journals than others, or topics can be exemplified more or less frequently in different time periods. In the case of our analyses of CVE, the only metadata we will use is the time a vulnerability record was assigned. Other security databases, such as the Common Weaknesses Enumeration (CWE) list, potentially provide much more metadata than CVE (though not temporal metadata). So, the description of a security weakness can be associated with, for example, its status, whether it is a draft, incomplete or stable, its mode of introduction, consequences, mitigations, or the attack patterns it is related to. Such covariates can also be used to affect the probability of a word within a given topic. So, documents associated with certain consequences would employ certain of its terms at one probability, but that same topic associated with another consequence would employ certain of its terms at a different probability. Not only does STM support the inclusion of metadata covariates that affect how and which topics vary with respect to the documents that exhibit them, they consistently outperform LDA in covariate-generated topic processes [31]. Moreover, STM provides the option of spectral initialization that dispenses with LDA's need for explicit model evaluation and training data to ensure (faster) convergence in the creation of the topic model.

While STM exhibits a number of advantages, it has some limitations. STM remains intractable when a covariate has multiple values instead of a single value. While temporal covariates can be used to plot the changes in prevalence a topic undergoes through time – thus supporting the identification of trends – topics exist throughout the time period being considered. A more important limitation is that there is no way of identifying when a topic emerges, expires, or develops into another topic. This is an important problem that we plan to take up in future work. A solution for topics consisting of single words has been proposed [32] that seems promising. Basically, they divide a corpus into several year time periods that overlap. We have already done this by hand with good results, but it is very time consuming. Some automation is needed. Their tool produces a list of topics automatically identifying those that are born, die, split, or mix between two adjacent time periods. To interpret, they eyeball the adjacent pairs of time periods. They use several well-known similarity measures, all based on the changing probabilities of single words belonging to a topic. They do not use multi-word terms, let alone syntactically principled meaningful multi-word terms, nor has anyone else tried this in connection with topic modeling. These similarity measures will have to be extended, and/or additional procedures will have to be added to address the addition of multi-word term probabilities. These are weighted more highly than single-word terms. Moreover, some multi-word terms are rated more highly than others based on their role in a corpus, which we currently base on the CVE semantic schema.

Like other topic modeling approaches, STM topic models lack stability over multiple runs. Over multiple iterations, the model does not generate all the same topics for a set number of topics. Some topics are more stable than others. Methods and perhaps tools need to be introduced to decide which topics ranked in terms of stability over multiple runs are most important for interpretation and understanding. Some of these issues are discussed in Section 1. Some research has also focused on the stability issue, including Yang et al. [33] and Belford et al. [34]. Both have measures showing the degree of stability or instability. Yang et al.[33] discuss incorporating stability constraints on the generation of topics, and Belford et al. [34]provide algorithms for generating the best topic model based on term stability and document stability of those generated in multiple runs. In future work, we will compare stability achieved by STM with other methods, such as LDA.

3.3.3. Combining STM and R&R

We combine two approaches to computer assisted text analysis that are often considered separately or even in competition: natural language processing (NLP) and statistical-based algorithms like embedded

word or topic models. In particular, we have experimented with loading R&R output – both phrases and multi-word terms – into topic modeling environments such as STM. Instead of loading the natural language documents into STM, we substitute parsed R&R output from each document, consisting of both single-word and multi-word terms, headed by both nouns and verbs, in place of the English language sentences.

When English language sentences are uploaded and analyzed by standard topic modeling applications, resulting topics consist of single words. When R&R terms are substituted, topic results consist of both single-word and multi-word terms – not only single words. We contend that the addition of phrases and multi-word terms provides a more informative basis for interpreting topics and are still competitive or better at computational efficiency and accuracy (for the latter, see section 2.3 on TREC above).

Various structures were built making use of the R&R building blocks as constituents of taxonomic structures and automatically generated topic models and knowledge graphs, the topic models also being used for automating trend analysis. As will be seen in section 4, automated topics and trends fit the initial benchmarks we use reasonably well, but these appraisals can be improved, as we will describe. A much more comprehensive investigation using not only CVE but other cybersecurity information sources like CWE, CAPEC, and hacker mailing lists and websites also need to be investigated in wider studies beyond the scope of this report.

3.4. Knowledge Graphs

Knowledge graphs (KGs) have been used employing some Semantic Web techniques in diverse applications such as intelligent chatbot services and search engines, where the demand for well-refined and structured data has been gradually increasing. A Knowledge Graph (KG) can be specified as a set of Resource Description Framework (RDF) triples <subject, predicate, object>. The subject and object exist as nodes with labels, and the predicate represents a relationship existing as an edge between the nodes for subject and object. In this report, we apply KGs to help visualize both RDF triples using R&R multi-word terms to label the nodes and single-word terms to label the edges. Relationships between RDF triples are presented as a series of RDF triples (e.g., N1R1N2 and N2R2N3, where N stands for Noun and R stands for Relation and numbers distinguish different Ns and Rs – see Fig. 8 below for an example derived from CVE R&R terms). We intend to use these KG diagrammatic relationships as a way of visualizing the relationships among the R&R terms in the topics of a topic model (see section 4.1).

In order to construct a KG, we adopt Neo4j as a graph database. Neo4j has some advantages over other tools for implementing a knowledge graph database in terms of detailed visualization and scalable plug-in features. The R&R output used as input is analyzed and translated into an RDF triple line by line. For RDF triples derived from the same sentence and document, we generate a tiny graph where a common node is shared by different RDF triples that contain the node as their subject or object. And each tiny graph is merged to generate a whole large KG. To do this, we utilized the py2neo library (https://py2neo.org/2020.0/)

The following is an example of some of the steps we take. Table 1 shows all analyzed information of a sentence from a CVE record. The sentence is: “Some telnet clients allow remote telnet servers to request environment variables from the client that may contain sensitive information or remote web servers to obtain the information via a telnet: URL.”

Line number	R&R Term	POS	Doc. Id	Date in sequence
41474	url	NOUN	2423	200000883
41475	telnet:0:client	NOUN	2423	200000883
41476	telnet	NOUN	2423	200000883
41477	client	NOUN	2423	200000883
41478	telnet:1:information:0:obtain	VERB	2423	200000883
41479	information:0:obtain	VERB	2423	200000883
41480	information	NOUN	2423	200000883
41481	obtain	VERB	2423	200000883
41482	remote:1:web:0:server:2:sensitive:0:information:3:contain	VERB	2423	200000883
41483	remote:1:web:0:server:2:sensitive:0:information	NOUN	2423	200000883
41484	remote:1:web:0:server	NOUN	2423	200000883
41485	web:0:server	NOUN	2423	200000883
41486	web	NOUN	2423	200000883
41487	server	NOUN	2423	200000883
41488	sensitive:0:information	NOUN	2423	200000883
41489	contain	VERB	2423	200000883
41490	remote:1:telnet:0:server:2:environment:0:variable:1:reque st	VERB	2423	200000883
41491	remote:1:telnet:0:server	NOUN	2423	200000883
41492	telnet:0:server	NOUN	2423	200000883
41493	environment:0:variable:1:request	VERB	2423	200000883
41494	environment:0:variable	NOUN	2423	200000883
41495	environment	NOUN	2423	200000883
41496	variable	NOUN	2423	200000883
41497	request	VERB	2423	200000883
41498	allow	VERB	2423	200000883

Table 2: Analyzed information of a sentence from a CVE record for conversion to RDF triples

The first column of the table indicates a line number, the second column represents an R&R term, the third column shows whether the R&R term belongs to a NOUN phrase or VERB phrase, the fourth column denotes the document ID, and the last column indicates the publishing date of the document.

From the R&R output, as shown in the above table, we generate the materials to be translated to RDF triples. To do this, we simply find two possible NOUN phrase terms with the maximum length for each VERB phrase term. For example, two noun phrase terms can be split from the VERB phrase term "remote:1:web:0:server:2:sensitive:0:information:3:contain." These are the NOUN phrase terms "remote:1:web:0:server" and "sensitive:0:information" because there are two separately generated NOUN terms for these terms (See the 41484 and 41488 lines of the above table). The leftmost NOUN term of maximum length of the VERB phrase term represents a subject of the RDF triple, and the rightmost noun term of maximum length is treated as an object of the RDF triple. Therefore, we can generate the RDF triple <remote:1:web:0:server, contain, sensitive:0:information> from this VERB phrase term. Table 2 shows detailed results. The third, fourth, and fifth columns indicate subject, predicate, and object of an RDF triple, respectively. As a result, we get the RDF triples <telnet, obtain, information>, <remote:1:web:0:server, contain,

sensitive:0:information>, <remote:1:telnet:0:server, request, environment:0:variable> and <environment, request, variable> based on our heuristic.

Doc. ID	Date sequence	Object	Relation	Object
2423	20000884	telnet	obtain	information
2423	20000884	remote:1:web:0:server	contain	Sensitive:0: information
2423	20000884	remote:1:telnet:0:server	requests	Environment:0:variable
2423	20000884	environment	requests	Variable.

Table 3. Example RDF triples.

When a large volume of sentences is converted into RDF triples, the KG is able to show related predicates for the same subject or object in an intuitive way. In the current version of the report, we just utilize the KG to see a visualized graph that can provide partial visualizations of topics and aid topic model interpretation. With respect to CVE, this means KGs can supplement topic trend analysis for anticipating cybersecurity threats by showing relationships among the terms associated with a topic. In addition, as a further aid to interpretation of topics and understanding of the "lay of the land" of very large corpora, KG can provide question-answering capabilities [35].
Currently, converting CVE sentences into RDF triples is a work in progress. We successfully converted about 30-40% of the sentences. We are currently exploring ways of increasing this coverage.

4. Benchmarking and Evaluation of Semantic Structures

The proceeding sections have described how semantic structures tacit in a corpus can be made explicit for human interpreters using the described techniques and processes. R&R term structures are used as building blocks or input to ML applications and supporting algorithms to produce semantic structures and diagrams. We use 1) the R STM package to produce topic models consisting of single and multi-word R&R structured terms, 2) a simple algorithm associated with R&R term generation to build a taxonomy of terms based on term structure, 3) the neo4j graph database platform to build a term network with nodes named by terms playing the role of nouns and links named by terms playing the role of predicates or relations and 4) familiarity with CVE records and use of R&R term generation to identify the CVE semantic schema. The semantic structures made explicit by these means are a way of keeping track of the roles that the terms play in the corpus, conferring conceptual status on the terms for suitable interpreters. We are now in a position to show how these terms, that can be taken as concepts, provide the basis for understanding a relatively large corpus like the CVE List that provides insight into the cybersecurity threat landscape.

4.1. Interpreting Explicit Semantic Structures in CVE

Of the four types of semantic structures, taxonomy, semantic schema, topic models, and knowledge graphs, we have experimented most with topic modeling and the use of the

CVE semantic schema for understanding the CVE threat landscape. So, our main focus will be on topic models.

For purposes of exposition, we will discuss a topic model of 20 topics derived from the CVE list from 1999 through much of 2019. While choosing the number of topics is not completely arbitrary - having too few will mean that too many documents or records will only be superficially similar to one another according to the topics they share while having too many will have the opposite effect of too few being similar to one another - the choice depends on the perspective and interests of users of the STM tool. In any case, a range of topic models with different numbers of topics should be investigated depending on time constraints and other situational exigencies to inform the user's choice. We held out about a year's worth of records for possible later use in testing some of our predictions.

Each of the 20 topics contains terms ordered from the most to least representative of the topic. For purposes of showing how the interpretive process works, only the top 50 terms will be presented as a basis for discussing each topic and interpreting its meaning. The 20 topics are ordered according to the estimated proportion of the content of the corpus that belongs to each topic. Topic proportion is an estimate because topics cover CVE records even when there is no overlap among terms in the topic and terms in the record. There may be common terms that both terms in the topic and terms in the record point to, e.g., synonyms or terms that would be deemed similar in meaning, but at some point in the topic list, these would become less and less discernible until a reasonable cutoff point is reached. Using 50 terms is well above any such cutoff point. In any case, for purposes of providing an estimate of the proportion of records in a topic, STM makes an estimate of the cutoff point based on things like the size of the corpus, number of topics specified, etc. In our experience using STM, these estimates seem reasonable, but they are in need of testing and evaluation. Section 4.2 will discuss how such testing and evaluation can be done for this and other matters.

To gain further insight into the overlapping meaning of topics and records, we use the most representative document indexed to the topic – showing it both in its form as natural language text and in its form as a collection of single and multi-word R&R structured terms. In section 4.2, we will discuss representative records of topics and show examples of how well the terms in them overlap with the terms in the topic. Heuristic measures for the comparison will be provided and discussed, showing the intimate relationship between interpretation and evaluation.

Fig. 10 shows expected proportions of topic coverage listing the Topic number ID and the First 3 terms of the topic, often just single words because they are the most prevalent terms in the topic.

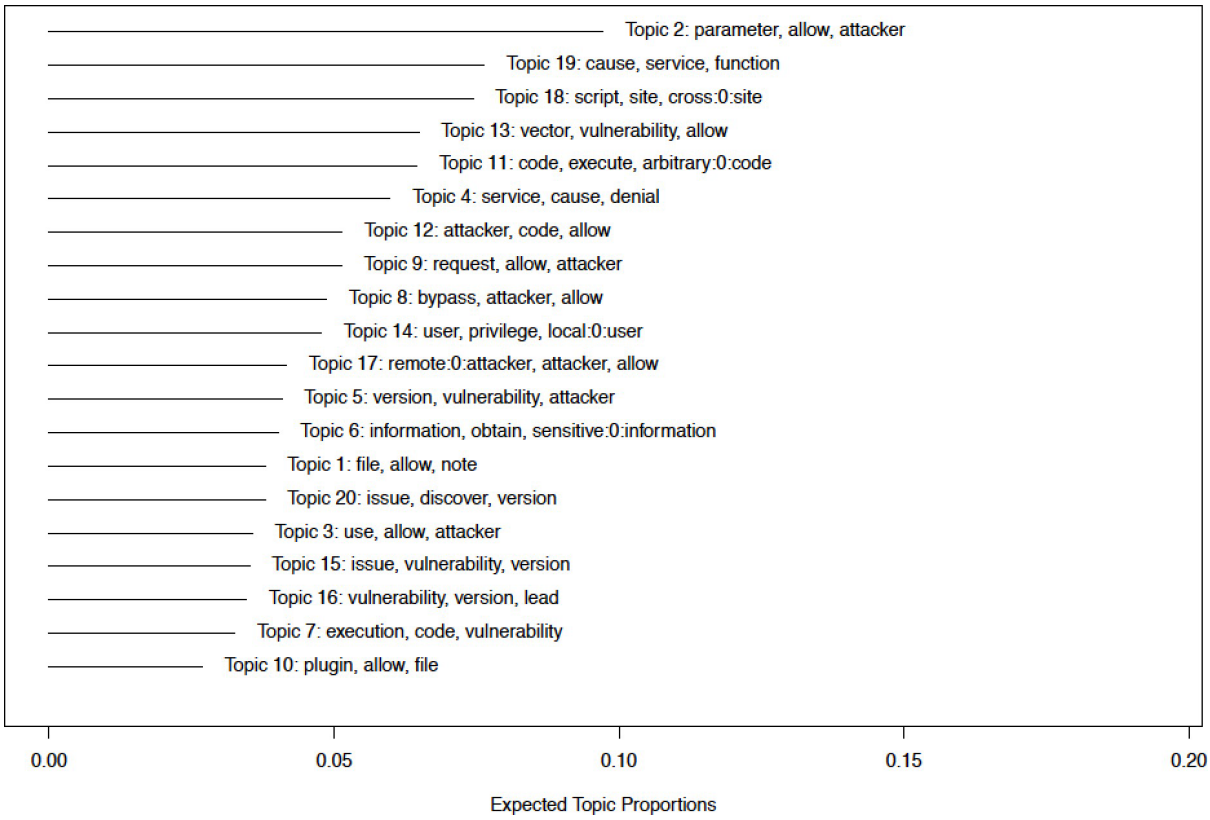

Fig. 10: Expected proportions of CVE topic coverage

Fig. 10 shows the First 50 terms for the First 4 topics in the order of their expected coverage of the content of the corpus.

Topic 2:
parameter, allow, attacker, remote:0:attacker, vulnerability, execute, command, injection, sql:0:injection, sql:0:injection:1:vulnerability, sql:0:command, arbitrary:1:sql:0:command, arbitrary:1:sql:0:command:2:execute, remote:0:attacker:3:arbitrary:1:sql:0:command:2:execute, file, traversal, directory, code, php:0:code, arbitrary:1:php:0:code, url, arbitrary:1:php:0:code:2:execute, traversal:0:vulnerability, remote:0:attacker:3:arbitrary:1:php:0:code:2:execute, inclusion, remote:0:file, dot, arbitrary:0:file, arbitrary:0:file:1:read, dot:0:dot,read, inclusion:0:vulnerability, remote:0:attacker:2:arbitrary:0:file:1:read, include, action, d:0:parameter, parameter:0:url, directory:0:traversal, directory:1:traversal:0:vulnerability, local:0:file, remote:0:file:1:inclusion:0:vulnerability, component, metacharacter, shell:0:metacharacter, note, sequence, module, arbitrary:1:local:0:file, remote:0:file:1:inclusion, remote:0:file:1:inclusion:2:vulnerability

Topic 19:
cause, service, function, denial, crash, allow, attacker, remote:0:attacker, application, read, file, application:0:crash, pointer, crash:0:service, crash:0:service:1:denial, crash:0:service:1:denial:2:cause, memory, value, integer, service:0:denial, bound, kernel, dereference, trigger, application:0:crash:1:service, application:0:crash:1:service:2:denial, linux:0:kernel, application:0:crash:1:service:2:denial:3:cause, overflow, remote:0:attacker:3:crash:0:service:1:denial:2:cause, user, integer:0:overflow, impact, remote:0:attacker:4:application:0:crash:1:service:2:denial:3:cause, error, pointer:0:dereference, buffer, implementation, length, image, null:1:pointer:0:dereference, write, relate, bound:0:read, lead, set, possibly:0:unspecified:1:impact, size, datum, demonstrate

Topic 18:

script, site, cross:0:site, scripting, inject, web, html, web:0:script, arbitrary:1:web:0:script,
html:2:arbitrary:1:web:0:script, vulnerability, remote:0:attacker, attacker, allow,
html:2:arbitrary:1:web:0:script:3:inject, remote:0:attacker:4:html:2:arbitrary:1:web:0:script:3:inject,
parameter, cross:0:site:1:scripting, cross, xss:2:cross:0:site:1:scripting,
xss:2:cross:0:site:1:scripting:3:vulnerability, xss:0:vulnerability,
unspecified:0:vector:3:html:2:arbitrary:1:web:0:script,
unspecified:0:vector:3:html:2:arbitrary:1:web:0:script:4:inject, field, page, action, module,
scripting:0:vulnerability, search, message, title, url, plugin, d:0:parameter, vulnerabilitie, query, error,
xss:2:cross:0:site:1:scripting:3:vulnerabilitie, note, unspecified:0:parameter, tag,
scripting:1:xss:0:vulnerability, email, function, comment, crafted:0:url, admin,
cross:0:site:2:scripting:1:xss:0:vulnerability, version

Topic 13:
vector, vulnerability, allow, unspecified:0:vulnerability, user, authenticate, relate,
unspecified:0:vector, remote:0:authenticate, remote:0:authenticate:1:user, unknown:0:vector,
component, affect, remote:0:attacker, attacker, impact, integrity, attack, confidentiality,
different:0:vulnerability, availability, unknown:0:impact, remote:0:authenticate:1:user:2:allow,
attack:0:vector, authenticate:0:user, remote:1:authenticate:0:user, confidentiality:0:integrity,
involve, unspecified:0:impact, module, confidentiality:0:affect, local:0:user, note,
integrity:0:confidentiality, update, integrity:0:confidentiality:1:affect,
permission, comment, claim, issue, attack:0:vector:1:unknown:0:impact, remote:0:attack,
remote:0:attack:1:vector, unknown:0:vector:1:availability, unknown:1:attack:0:vector, previous:0:information,
attack:1:unknown:0:impact, availability:1:confidentiality:0:integrity,
unknown:0:vector:1:availability:2:confidentiality:0:integrity, cpu

Fig. 6: First 4 of the 20 CVE topics ordered as to expected coverage

We use the CVE semantic schema described in section 3.2 to aid in interpreting these and the other topics. The schema in the form of a case frame is

vulnerabilities

located in something … before something … allow(s)

someone (an attacker)

to do something (an attack, what is being used in the attack, the attack basis, a consequence of the attack often part of a causal chain of actions involved in the attack)

via something, a method, or something utilized, an instrument or technique. We focus on the topic with the highest proportional coverage first.

4.1.1. CVE Topic with Highest Estimated Proportional Coverage

Topic 2 is quite rich in multi-word terms that fill in parts of the CVE schema. It includes terms for three vulnerabilities prominent[3] in the corpus

1. the most prevalent, sql:0:injection:1:vulnerability,
2. the second most prevalent, directory:1:traversal:0:vulnerability, and
3. the least prevalent, remote:0:file:1:inclusion:0:vulnerability.

[3] Prominence and prevalence are based on counts of these terms using simple searches of the terms in the corpus.

It is unusual for a CVE topic to include three named vulnerabilities. As we shall see, this will need to be considered in the evaluation phase. An even longer term in Topic 2 includes other parts of the schema for the sql injection vulnerability, namely

remote:0:attacker:3:arbitrary:1:sql:0:command:2:execute.

This and other such correlations have been checked by a regular expression search. Remote:0:attacker fills the someone slot of the schema, and arbitrary:1:sql:0:command:2:execute fills the attack or action slot. Looking at the most representative natural language record for Topic 2, we can see that its First sentence includes all of these:
ID 13657: “Multiple SQL injection vulnerabilities in vBulletin 3.0.7 and earlier allow remote attackers to execute arbitrary SQL commands…”

An interesting question then becomes: are the remote attacker and attack terms relevant to the other vulnerabilities in Topic 2? This will be a question to ask in the evaluation phase, as well, again showing the connection between interpretation and evaluation. In how many records does this combination occur?

Searching the CVE corpus of R&R terms, we find that of the 6,518 records that contain sql:0:injection:1:vulnerability,5,095 also,
remote:0:attacker:3:arbitrary:1:sql:0:command:2:execute

and of these,

5094 contain allow.
Neither directory:1:traversal:0:vulnerability nor remote:0:file:1:inclusion:0:vulnerability records contain it. Do any containing one of the 3 vulnerabilities in Topic 2 contain

remote:0:attacker:3:arbitrary:1:php:0:code:2:execute?

It turns out that neither records containing sql:0:injection:1:vulnerability, nor ones containing directory:1:traversal:0:vulnerability contain it, but records for remote:0:file:1:inclusion:0:vulnerability do contain it 1,291 times along with allow.

So the vulnerabilities, sql:0:injection:1:vulnerability, and remote:0:file:1:inclusion:0:vulnerability, each Fill 4 of the 6 slots of the CVE semantic schema according to these searches.

Up to this point in the analysis of the three vulnerabilities in Topic 2, only directory:1:traversal:0:vulnerability has not been seen to have any semantically related terms associated with it using regular expression search. We could try the term, remote:0:attacker:2:arbitrary:0:file:1:read since it is the only other term that has the structure that is a candidate for filling in the CVE semantic schema after allow, i.e., vulnerability allows attacker to do something. In this case, of the 803 records directory:1:traversal:0:vulnerability occurs in, 303 contain remote:0:attacker:2:arbitrary:0:file:1:read.

Note that the R&R term-based knowledge graph database also contains a graph suggesting a relationship between directory:1:traversal:0:vulnerability and remote:0:attacker:2:arbitrary:0:file:1:read (see Fig.s 11 and 12). The graph in 11 depicts a read relationship between directory:0:transversal and arbitrary:0:Pile and the graph in Fig. 8 depicts a relationship between arbitrary:0:Pile and remote:attacker as well. We can now conclude that all 3

vulnerabilities included in topic 2 fill 4 out of 6 CVE semantic schema slots according to the semantic structures that have been made explicit.

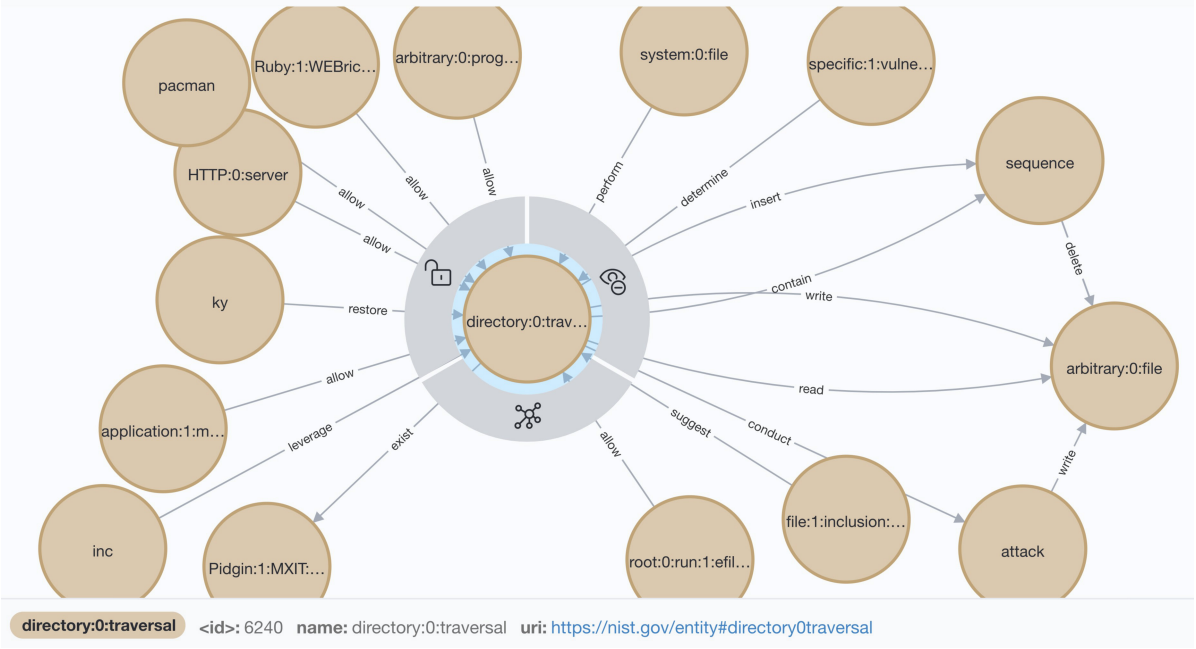

Fig. 11 Relationship between directory:0:traversal and arbitrary:0:Pile

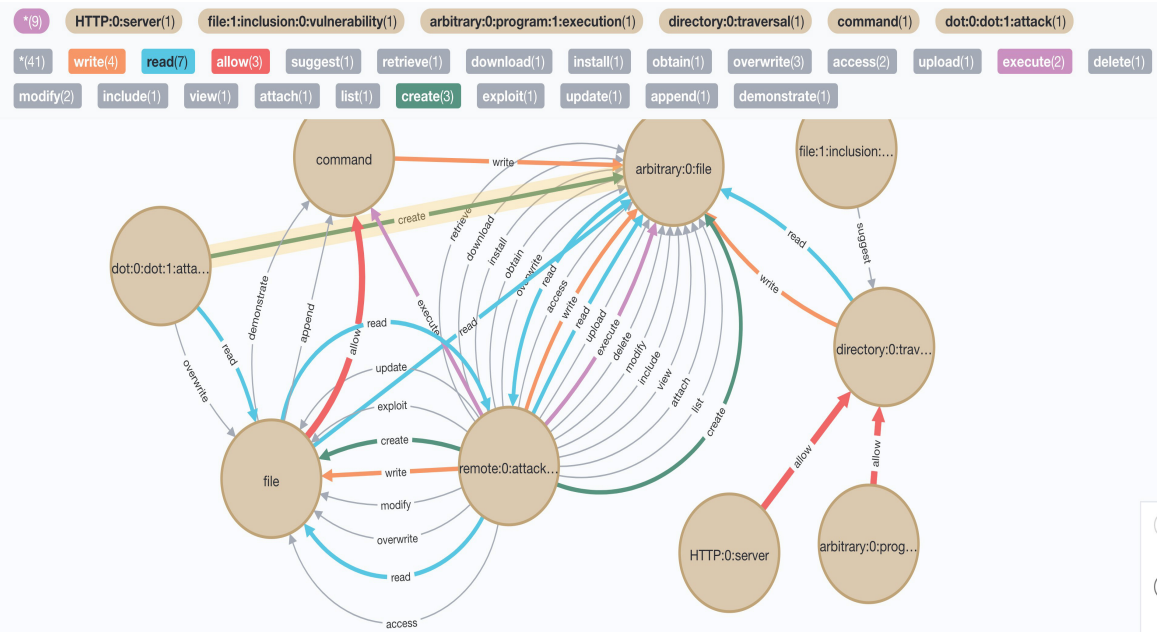

Fig. 12 Read (among other relationships) between remote:0:attqacker and :arbitrary:0:Pile

However, in Fig. 12, although remote:0:attacker:2:arbitrary:0:file:1:read is represented in the graph by the remote attacker reading an arbitrary Tile, the connection between directory:0:traversal and arbitrary:0:file is also linked by read.

While this may be indirectly the case, since the directory:1:traversal:0:vulnerability allows a remote:0:attacker to read an arbitrary:0:file, the direct link should preferably be allow. Again this is a case of evaluation of the performance of a work product posing a problem for the development of a deliverable product. Is it a problem of R&R parsed output or the term generation based on it, or is it a problem for how the knowledge graph database application converts the generated terms into graphs? This problem and ones like it are discussed in sections 3.4 and 4.1. Since 4 out of 6 slots of the CVE semantic schema have been Filled for all three of the vulnerabilities in the topic, what about the other two slots - located in something … before something … and also the slot via something, a method, or something utilized? For the former, topic 2 has some candidates for located in something but not before something. Before something presents the problem of representing numbers as terms since the same number appearing in different places depends on what it is counting. Although numbers can be pinned down in multi-word terms just as descriptive words can, they are often quite unique to a record and not repeated and therefore are as dispersed as leaves of a taxonomy, rarely seen as part of a prevalent term in a topic and rarely the content of a node or link in a knowledge graph. This is true of some unique words; they will also suffer the same fate. We will not be further addressing this uniqueness problem here. It is something to consider down the road.

What about terms in topic 2 that are candidates for the slot located in something? In this case, the vulnerability referenced is located in something. We have run regular expression search, extraction, and frequency scripts on CVE natural language text records searching on the <name> “vulnerability in” and extracting everything that comes after the named vulnerability up to the word "allow’. For sql injection vulnerability, there are 5115 records containing it in CVE 1999-2019. We found that one prevalent type of object this vulnerability is in PHP (PHP: Hypertext Preprocessor) scripts. Of the 5115 records containing the vulnerability, 2009 are of this kind divided into 715 subtypes. The top 10 are

index.php	730
login.php	67
search.php	57
news.php	37
admin.php	31
view.php	23
category.php	21
proTile.php	20
forum.php	17
member.php	17

Table 4. Top 10 PHP subtypes

Note, none of these are covered by terms in Topic 2. Unfortunately, it is a known problem for the R&R parser and term generator to miss converting words connected by dots into terms. However, as we have seen, Topic 2 does contain terms with PHP like

arbitrary:1:php:0:code:2:execute, but these are associated with the remote:0:file:1:inclusion:0:vulnerability.

The other slot, via something, a method, or something utilized, is filled by a term in Topic 2 that is the most prevalent of its terms, "parameter,' albeit a single word, that is also contained in the significantly less prevalent terms d:0:parameter and parameter:0:url. It is also mentioned quite prominently in Topic 2's most representative document, along with many .php's. Here is the whole record:

Multiple SQL injection vulnerabilities in vBulletin 3.0.7 and earlier allow remote attackers to execute arbitrary SQL commands via the (1) announcement parameter to announcement.php, the (2) thread[forumid], or (3) criteria parameters to thread.php, (4) userid parameter to user.php the (5) calendarcustomfieldid (6) calendarid (7) moderatorid (8) holidayid (9) calendarmoderatorid or (10) calendar[0] parameters to admincalendar.php (11) the cronid parameter to cronlog.php (12) user[usergroupid][0] parameter to email.php (13) help[0] parameter to help.php the (14) limitnumber or (15) limitstart parameter to user.php the (16) usertitleid or (17) ids parameters to usertitle.php (18) rvt[0] parameter to language.php (19) keep[0] parameter to phrase.php (20) dostyleid parameter to template.php (21) thread[forumid] parameter to thread.php or (22) usertools.php.

4.1.2. Topics of Next Highest Proportional Coverage

To interpret Topic 2, we focused strategically on terms that were most productive in filling the CVE semantic schema. These turned out to be named vulnerabilities and terms that expressed what the named vulnerability allowed attackers to do. Topic 2 contained three sets of such terms. Let's see if this is typical of the next three topics of highest estimated proportion, Topics 19, 18, and 13.
Topic 19 does not name a vulnerability. In fact, it does not contain any terms with the word "vulnerability.' It does contain terms referring to denial of service, the longest being application:0:crash:1:service:2:denial:3:cause, and remote:0:attacker:3:crash:0:service:1:denial:2:cause. This term is sometimes used in connection with the discussion of vulnerabilities. It can be thought of as referring to those vulnerabilities that allow someone to cause a denial of service crash or application crash.

There are 7,263 records of this kind. In some cases, records (26 of them) actually refer to denial of service vulnerability or service0:denial:1:vulnerability. A website we will discuss in more detail in later section 4.2 actually classifies denial of service as a vulnerability type. Topic 19 does contain terms that may help characterize the vulnerabilities that allow someone to cause a denial of service. These consist of two sets of terms: 1) pointer, read, memory, bound, dereference, buffer, pointer:0:dereference, null:1:pointer:0:dereference, write, and bound:0:read and a much smaller set 2) linux and linux:0:kernel. The former will be discussed in section 4.2 and, most specifically, in the concluding section 5. In the latter, a more ontological approach to classification is pursued, including distinguishing between labels and concepts with clear distinctions among bugs, vulnerabilities, attacks, and consequences of attacks based on distinctions between states and actions and introducing causal chains. Many of these distinctions are necessary for making CVE reporting practices more standardized and useful. Records in

CVE can be divided into different clusters. They can be labeled cluster 1, cluster 2, etc., where the meaning of vulnerability is not carefully delineated or adhered to. They can be classified roughly into what are called vulnerabilities but are a mix of vulnerabilities, actions that are part of attacks, or the consequences of these actions. On the other hand, they can be distinguished more carefully, or ontologically, according to principles that do not mix concepts that must be kept distinct and adhered to consistently.

Unlike Topics 2 and 19, Topic 18 is clearly about one vulnerability, the xss:2:cross:0:site:1:scripting:3:vulnerability, where XSS is an acronym for Cross Site Scripting. It also clearly allows (in the semantic schema sense) remote:0:attacker:4:html:2:arbitrary:1:web:0:script:3:inject as the natural language text version of the most representative of Topic 18's records makes clear:

"Multiple cross-site scripting (XSS) vulnerabilities in … allow remote attackers to inject arbitrary web script or HTML."

Topic 13 is different from the other topics so far discussed in that one of its most prevalent terms is *unspecified:0:vulnerability*, closely followed by unspecified:0:vector, unknown:0:vector, different:0:vulnerability, unknown:0:impact, unspecified:0:impact and more specifically, though less prevalent, attack:0:vector:1:unknown:0:impact, unknown:0:vector:1:availability, unknown:1:attack:0:vector, attack:1:unknown:0:impact and unknown:0:vector:1:availability:2:confidentiality:0:integrity. It also introduces the idea of a remote:0:authenticate:1:user that is probably allowed (remote:0:authenticate:1:user:2:allow) by the unspecified or unknown vulnerability to do something with the attack:0:vector:1:unknown:0:impact have something to do with integrity:0:confidentiality:1:affect. This is somewhat borne out by the topic's most representative document, though without reference to the remote:0:authenticate:1:user: "Unspecified vulnerability in the Java Runtime Environment (JRE) component in Oracle Java SE 7 Update 6 and earlier allows remote attackers to affect confidentiality integrity and availability via unknown vectors related to Beans."

Again this topic will be discussed further in connection with a comparison between all the topics discussed in this 20-topic model and the CVE Details collection of 20 vulnerabilities by type.

4.1.3. Using Semantic Structures in Trend Analysis for Proactivity

Topic trend analysis shows how the relative prevalence of topics for a given corpus unfolds over the lifetime of a corpus or for selected portions of it. There can be upward or downward changes, sometimes steep, sometimes gradual, or a plateau reached for various periods of time. We are investigating the conditions under which changes in prevalence occur, or a plateau is maintained. If such conditions can be identified for past changes, they may be projected into the future. If so, proactive steps can be taken to eliminate or minimize vulnerabilities before they can be utilized in cybersecurity attacks [36,37]. From a cybersecurity standpoint, proactivity means taking advantage of germination periods. These periods are the time it takes to identify, name, understand, exploitation methods to be designed and tested, and finally actual attacks implemented. It takes time for descriptions of vulnerabilities to take hold and, at some point, to take on a publicly known name. Additional time is needed for named vulnerabilities to become well understood

enough in terms of where they might be found and the conditions surrounding them. Also, choices are involved. What does the would-be hacker want to achieve? Which method of exploitation could produce consequences that would achieve the would-be attackers' goal? What sort of skills does a would-be attacker need, and who else might have to be involved? Finally, what tools and methods are available or could be relatively easily developed that are needed for a successful attack? This section focuses on how this proactivity frame on topic trend analysis helps evaluate our development of topic modeling for CVE. To what extent can topic modeling in the form of topic trend analysis help, in a timely fashion, identify the conditions under which the prevalence of a vulnerability identified and characterized in a topic spikes upwards? Similarly, when it spikes downward, what were the conditions of its mitigation? Also, why does it plateau at a relatively high or a relatively low prevalence? In these regards, what can be achieved currently using our R&R based topic modeling topic approach, and what are the chances for its improvement in the future if specific developmental steps are taken?

We begin by looking at how the prevalence of topics that include characterization of vulnerabilities changing over time. We proceed from the premise that when there are upward surges, this indicates that either the vulnerability or changes to how the vulnerability is exploited have not been characterized quickly enough to thwart attacks. Even in this case, there may be time to thwart further surges. The primary goal of our work to date has been to show that such surges occur and that they can be characterized to some extent using the R&R terms that Fill CVE semantic schema and presented to users via topic model trends and knowledge graphs. We will raise questions as to whether characterizations that can be currently provided suffice for proactivity and whether the tools in their current stage of development and processes of use can provide any useful information within a germination period.

Fig. 5 shows the changes in the proportional prevalence of each of the First 7 of the 20 topics (except for Topic 13), some already discussed. As a handy memory aid, we provide key terms from each topic.

Topic 2, SQL Injection, Directory Transversal, and File Inclusion,

Topic 19, Denial of Service, Remote Attacker, Memory, Application Crash, Null Pointer Dereference, and Linux Kernel

Topic 18, Cross-Site Scripting, Inject Arbitrary Web Script or HTML

Topic 11, Remote Attacker Executes Arbitrary Code, User Assisted Remote Attacker, Stack and Heap Based Buffer Overflows

Topic 4, Denial of Service, Memory Consumption, CPU Consumption, Remote Attacker,

Topic 12, Remote Attacker Executes Arbitrary Code or Causes Denial of Service Memory Corruption via a Crafted WebSite

We have already seen that topic 2 contains three vulnerabilities. It also the highest in topic prevalence among the 20 topics. Topics 19, 4, and 12 all include Denial of Service as key, all work with Remote Attacker alone, not including User Assisted Remote Attackers as

does Topic 11. Also, they all include operations on memory. For trend analysis purposes, we divide these 6 topics into two groups. One group, consisting of Topics 2, 18, and 11, makes prominent references to named vulnerabilities or vulnerabilities that are fillers of the vulnerability slot in the CVE semantic schema, 2 - SQL Injection, Directory Transversal and File Inclusion vulnerabilities, 18 – Cross-Site Scripting vulnerability and 11, Overflow Vulnerabilities. This group's trend analysis is shown in Fig. 9. The other group depicted in a trend analysis graph (Fig. 10) shows all topics (19, 4, and 12), where denial of service figures prominently.

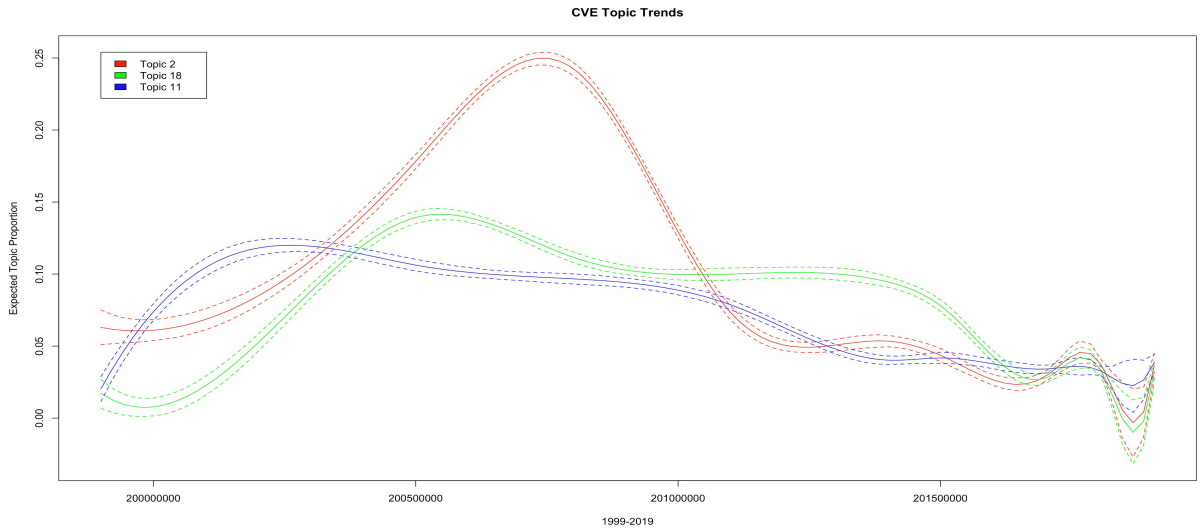

Fig. 13: Trend Analysis for topics identifying vulnerabilities – 2, 18 and 11

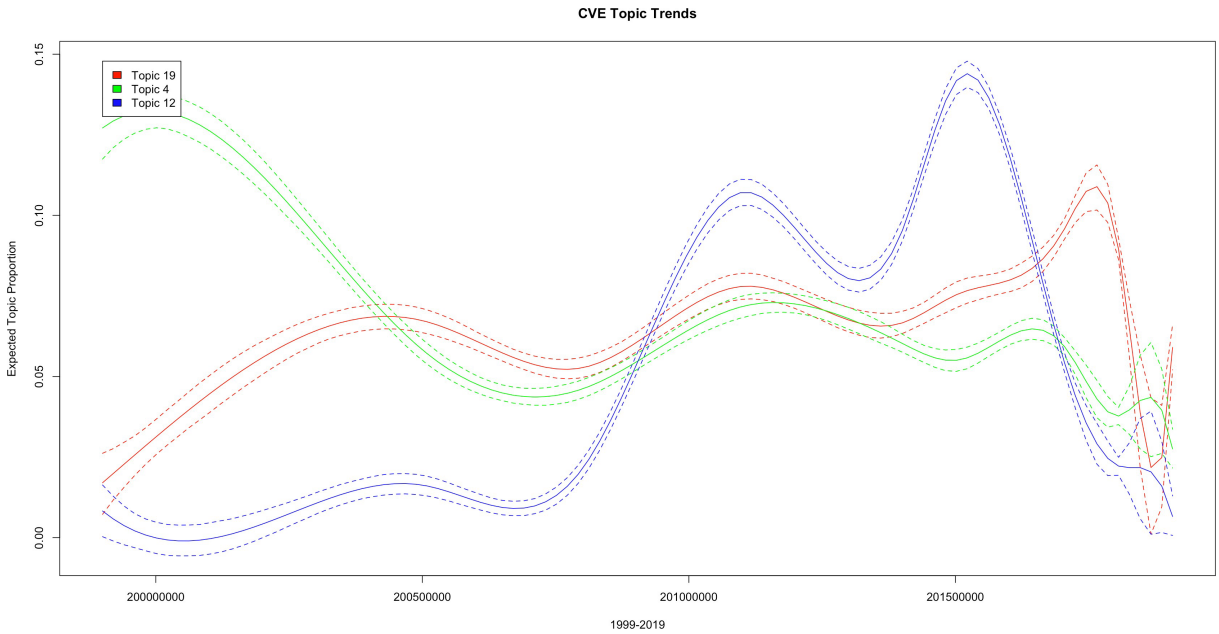

Fig. 14: Trend Analysis for topics involving Denial of Service – 19, 4 and 12

Topic 2 (Fig. 13), referencing the three vulnerabilities, does stand out from the other two topics in this group, reaching the highest prevalence amongst them sometime in the year

2007. It then drops even faster than it rose, reaching a mildly bumpy plateau starting in 2011, competing with Topic 18 (Cross-Site Scripting vulnerability) for the lowest prevalence. In fact, all three topics are at their lowest expected proportion at the end of the time period examined. In Fig. 14, Topics 19 (Null Pointer Dereference) and 12 (Execute Arbitrary Code, Memory Corruption) of the Denial of Service topics start with low prevalence and rise at different rates, whereas Topic 4 (Memory/CPU Consumption, Memory Leak) starts at a high expected proportion, but dips down quickly intersecting with Topic 19 just before 2005, and these two intersect with Topic 12 just before 2010. In this group, Topic 12 stands out in its rise from around 2007 until an initial peak and fall in 2011 and then another rise starting sometime in 2008, peaking just after 2015 and then having a precipitous fall. Topic 19 reaches its peak, though somewhat lower than the highest peak for Topic 12 in 2018, then also falls even more quickly than Topic 19.

An account of Topic 2's rise in the time period 2003 to 2008 seems to indicate a germination period, where a proactive intervention could have flattened the rise at any time between 2003 and 2007; the earlier, the better of course. An intervention seems to have happened in 2008 because of the steep decline in reports on this topic; at least, this is the prediction that can be drawn from the trend analysis. Further investigation would be needed to see if the prediction is borne out. An extended but shorter germination period also seems to be indicated for Topic 12 starting in 2008, where an intervention could have flattened the rise in prevalence between then and early 2010. Some intervention does seem to have happened around the latest time because of the decline after that. However, unlike the case of Topic 2, Topic 12 started a downward decline and then rose again. This happened between 2011 and 2014 with the second rise beginning in 2014 and lasting about a year. While the germination period is relatively short for the second spurt, at least some of the groundwork for proaction had already been done for the previous spurt, and these vulnerabilities should have been on the radar for this sort of threat. Again these are predictions that can be investigated.

4.2. From Local to Global Topic Modeling Evaluation: Benchmark

As we have seen, local development is a dialog with local evaluation and improvement. It has even involved global evaluation, but only in the sense of using already installed global evaluation set-ups (e.g., our use of TREC described in section 2.3). We will now argue that global evaluation designs are already implicit in local evaluation. In the following sections, we aim to make this more explicit with a discussion of the role of benchmarks. In collaboration with the IR division of NIST, there is a potential to develop this work into an evaluation standard in the area of augmenting human understanding of very large corpora. This would include providing new processes, methods, and tools that would characterize and support these new standards for evaluating and testing.

A critical task for developing evaluation standards is characterizing what needs to be tested and identifying or constructing benchmarks against which relevant testing is done. We have brought together tools to develop the building blocks, R&R formal term structures and using these building blocks as inputs into tools for gaining a better understanding of the information contents of very large corpora. In particular, we have

described how we have applied the tools to information sources such as the CVE List of over 120,000 vulnerability reports. We have also described how *developing* a topic model of CVE List records interconnects with *evaluating* how helpful the output is for interpreting and understanding the CVE List to anticipate cybersecurity risk (for a precursor approach to risk identification and mitigation focused on software [38,39].

4.2.1. Benchmarks that Scale Up

In this and the following sections, we begin to generalize some of our local evaluation practices toward a more generic approach for evaluating our tools. The evaluative frame is to gauge how well they contribute to increased understanding of large corpora. To take a step in this direction, we want to consider the role of benchmarks in measuring progress in understanding the patterns and themes of the CVE List. Understanding, in this case, means gaining insight into the cybersecurity landscape to anticipate and mitigate the threat. Early in this project, we constructed benchmarks by hand to appraise the topics and their trends generated automatically. Benchmarks were painstakingly constructed using regular expressions in Emacs to collect CVE records into clusters typified by the terms they shared. During this process, we noticed other regularities like named vulnerabilities were often at the beginning of a sentence. These were followed by the preposition "in' that marked the place in the sentence where the vulnerability was found (usually a product or piece of software – functions or modules). Sometimes there would be a sequence of phrases beginning within, marking, for example, a certain kind of software in a specific product. These were often followed by another prepositional marker, often the word "before' followed by a phrase or series of phrases marking the versions of products or software before a given one. These sequences were often followed by the word "allow' that served as a marker for a clause that followed containing a noun phrase like "remote attacker' and a verb phrase like to "cause a denial of service."

Finally, there was the last marker, "via' followed by a phrase or a clause describing the role of an inserted piece of crafted software. The role of such inserted software was to carry out what had been described in other parts of the sentence, e.g., an exploit or attack allowed by a vulnerability. These considerations became the basis for identifying the CVE semantic schema, which covers 65-75% of all CVE records and probably another 10% or more using slight schema variations. This approach to benchmarking included identifying and ranking different classes of instances of different elements of the schema. So, for example, named vulnerabilities could be ranked by their frequency of occurrence in the CVE List as a whole or in different time periods, showing their rise and fall. For the 1999-2009 time period, we discovered 5,982 different named vulnerabilities. This rose to 9,423 in 2010-2019 time period. See Fig. 15 for the 13 most frequent. named vulnerabilities in the two time periods. Similar tables were generated for products, attackers, attacks, and methods of attack. Such classifications provide a benchmark for testing topic trend analysis.

	CVE 1999-2009	
2	Named Vulnerabilities	Frequencies
3	sql injection vulnerability	3437
4	cross-site scripting (xss) vulnerability	3347
5	buffer overflow	2399
6	unspecified vulnerability	2085
7	directory traversal vulnerability	1589
8	multiple cross-site scripting (xss) vulnerabilities	1522
9	php remote file inclusion vulnerability	1306
10	multiple sql injection vulnerabilities	1065
11	stack-based buffer overflow	846
12	multiple php remote file inclusion vulnerabilities	507
13	heap-based buffer overflow	500

	CVE 2010-2019	
2	Named Vulnerabilities	Frequencies
3	unspecified vulnerability	3976
4	cross-site scripting (xss) vulnerability	3107
5	sql injection vulnerability	1233
6	multiple cross-site scripting (xss) vulnerabilities	1218
7	use-after-free vulnerability	982
8	directory traversal vulnerability	815
9	buffer overflow	785
10	cross-site request forgery (csrf) vulnerability	605
11	stack-based buffer overflow	518
12	heap-based buffer overflow	481
13	integer overflow	423

Fig. 15: Changing Prevalence of Named Vulnerabilities

The new appearance of the use-after-free and untrusted search path vulnerabilities at the top of the most frequently occurring vulnerabilities in 2010-2019 was used as a benchmark to test topic modeling trend analysis. They had been far down the list for 1999-2009, indicating a spurt in prevalence. This was, in fact, born out by the other relevant classes for 2010-2019 included in the benchmark. Topic trend analysis did show a spurt in both of these named vulnerabilities starting in 2010 and also indicated comparable changes in the R&R terms each was correlated with. However, this initial approach to benchmarking did not scale up. It took a long time to construct. Just as important, focusing on particular named vulnerabilities did not cover the whole CVE topic landscape. Topic modeling includes not only the relative proportions of topics throughout the corpus but also their interrelationships relative to other elements of the CVE semantic schema. Creating benchmarks, in this fashion, for all the topics implicit in a corpus that topic modeling purports to capture would be very time consuming, especially without a guide independent from topic modeling itself. However, such a guide, CVE Details Vulnerabilities by Type (CDVT), is available for use (on https:// www.cvedetails.com/vulnerabilities-by-types.php), and it could be turned into a benchmark or collection of benchmarks, provided certain issues with it are resolved.

4.2.2. Using CVE Details Types as an Initial Benchmark to Evaluate CVE Topic Models

CVE Details provides a web interface to CVE vulnerability data called Vulnerabilities By Type.

Vulnerabilities By Type

Year	# of Vulnerabilities	DoS	Code Execution	Overflow	Memory Corruption	Sql Injection	XSS	Directory Traversal	Http Response Splitting	Bypass something	Gain Information	Gain Privileges	CSRF	File Inclusion	# of exploits
1999	894	177	112	172			2	7		25	16	103			2
2000	1020	257	208	206		2	4	20		48	19	139			
2001	1677	403	403	297		7	34	123		83	36	220		2	2
2002	2156	498	553	435	2	41	200	103		127	74	199	2	14	1
2003	1527	381	477	371	2	49	129	60	1	62	69	144		16	5
2004	2451	580	614	410	3	148	291	111	12	145	96	134	5	38	5
2005	4935	838	1627	657	21	604	786	202	15	289	261	221	11	100	14
2006	6610	893	2719	663	91	967	1302	322	8	267	271	184	18	849	30
2007	6520	1101	2601	954	95	706	884	339	14	267	324	242	69	700	44
2008	5632	894	2310	699	128	1101	807	363	7	288	270	188	83	170	74
2009	5736	1035	2185	700	188	963	851	322	9	337	302	223	115	138	738
2010	4652	1102	1714	680	342	520	605	275	8	234	282	238	86	73	1493
2011	4155	1221	1334	770	351	294	467	108	7	197	409	206	58	17	557
2012	5297	1425	1459	843	423	243	758	122	13	344	389	250	166	14	624
2013	5191	1455	1186	859	366	156	650	110	7	352	511	274	123	1	205
2014	7946	1598	1574	848	420	305	1105	204	12	457	2106	239	264	2	401
2015	6484	1791	1826	1083	749	218	778	150	12	577	748	367	248	5	127
2016	6447	2028	1494	1324	717	94	497	99	15	444	843	600	87	7	1
2017	14714	3154	3004	2495	745	508	1518	279	11	629	1639	459	327	18	6
2018	16556	1853	3041	2368	400	517	2042	531	11	708	1424	247	461	31	4
2019	12174	919	2277	1247	296	410	1593	280	4	495	900	129	398	40	
Total	122774	23603	32718	18081	5339	7853	15303	4130	166	6375	10989	5006	2521	2235	4333
% Of All		19.2	26.6	14.7	4.3	6.4	12.5	3.4	0.1	5.2	9.0	4.1	2.1	1.8	

Fig. 16: CVE Details Vulnerabilities by Type

The CVE Details table shown in Fig. 16, accessed on the website, shows a typology of the CVE List with clickable dates and the number of occurrences underlined in blue. Types are in 13 columns, and dates 1999-2019 are in rows. Note that the number of occurrences adds up to the number of records in CVE List at the time the web page was accessed. The table also provides a cross-section of CVE records by type and year. As is, the table provides a benchmark for evaluating the R&R based topic model. Since the CVE R&R based taxonomy and knowledge graph are combined with the R&R-based topic modeling, it could also be the basis for benchmarking the combined use of these tools. However, we concentrate for the most part on benchmarking for topic modeling here. One way we have used CDVT was to match its types with the 20-topic model described in section 4.2.2. For topics already described in that section, eight different vulnerabilities by type are matched, i.e.,

Topic 2, SQL Injection, Directory Transversal and File Inclusion, matches these three vulnerabilities by type.

Topic 19, Denial of Service, Remote Attacker, Memory, Application Crash, Null Pointer Dereference and Linux Kernel, matches the types, DoS or Denial of Service.

Topic 18, Cross-Site Scripting, Inject Arbitrary Web Script or HTML, matches the type, XSS or Cross-Site Scripting.

Topic 11, Remote Attacker Executes Arbitrary Code, User Assisted Remote Attacker, Stack and Heap Based Buffer Overflows, matches the types, Code Execution and Overflow.

Topic 4, Denial of Service, Memory Consumption, CPU Consumption, Remote Attacker, matches DoS type again and also covers Memory Consumption and CPU Consumption that play similar roles to the vulnerability topic Memory Corruption in the CVE corpus, though Topic 12 covers the latter.

Topic 12, Remote Attacker Executes Arbitrary Code or Causes Denial of Service Memory Corruption via a Crafted Web Site, matches already covered named vulnerabilities by type, code execution and DoS, but also covers the type, Memory Corruption

Four more types are covered by the following topics.

Topic 9, Remote Attacker, Cross-Site Request Forgery, Hijacking Administrator Authentication, matches the type, CSRF or Cross-Site Request Forgery.

Topic 8, Remote Attacker Authentication Bypass, Physically Proximate Attacker, Protection Mechanism, matches the type, Bypass something.

Topic 14, Local User Gains Privileges, Root Privilege Gain, Local Attacker, Privilege Escalation, matches the type Gain Privilege.

Topic 6, Obtain Sensitive Information via Crafted Certificate, Man in the Middle Attacker to Spoof Server, matches the type, Gain Information.

So of the 13 CDVTs, 12 were matched by topics in the 20-Topic model. In these 12 cases, the topics increase understanding of the vulnerability types by associating the named vulnerability with terms that fill the CVE semantic schema.

The one type not matched was HTTP Response Splitting, but this type only covers 166 records in the CVE List (See Fig. 8). Also, Topic 2 covers three named vulnerabilities, SQL Injection, Directory Transversal and File Inclusion, Topic 11 covers two, Code Execution and Overflow, Topic 4 covers DoS again and Topic 12 covers Code Execution and DoS again. It would be better if each topic covered one type so that all its associated terms were correlated with one topic. When two or more vulnerabilities are in the same topic, the correlation with associated terms in the topic is less clear. This is not to say that the correlation in the latter case does not need to be tested. It does. We view the topic model as an hypothesis about the CVE corpus as a whole and each individual topic as an hypothesis concerning the documents that are deemed representative of it. There is the possibility that the topics referencing two or more vulnerabilities might be proposing that the vulnerabilities so grouped might have certain characteristics in common. To check to see whether duplication would disappear or at least be reduced, we ran another topic model. This one was specified to have 25 topics instead of 20. We were most interested in whether the three vulnerabilities in topic 2 would be split into separate topics without ruining the performance of the topic model vis-à-vis CDVT. We also wondered whether the remaining type would be covered.

We found that in the topic model with 25 topics a new topic was created in which SQL Injection was split off from, Directory Traversal and File Inclusion, though the latter were

still together in another topic. There was a reference to Cross-Site Scripting in the SQL Injection topic, but Cross-Site Scripting was clearly covered in a topic with no reference to other named vulnerabilities. The inclusion of Cross-Site Scripting with a topic almost solely devoted to SQL Injection makes sense since both are types of injection. As to the continued appearance of both Directory Traversal and File Inclusion in one topic, they are also related. File Inclusion is a way of gaining unauthorized Tile system access, taking advantage of an application building a path to executable code allowing the attacker to control which Tile is executed at run time. Directory or Path Traversal aims to access Tiles and directories that are stored outside the web root folder by manipulating variables that reference Tiles with "dot-dot-slash (../)" sequences or by using absolute File paths. File Inclusion is distinct from Directory or Path Traversal in that the latter is a way of gaining unauthorized File system access, whereas the former subverts how an application loads code for execution. All other coverage remained pretty much the same, except that the type, HTTP Response Splitting, was covered by a topic in the 25-topic model. This topic contained terms corresponding to both HTTP Header and CRLF Injection vulnerability, which together are used for HTTP Response Splitting. While a term for HTTP Response Splitting was not referenced explicitly in the topic, web servers use CRLF (standing for Carriage Return and a Line Feed) to decide when a new HTTP header begins and another one ends. In CRLF injection vulnerability both the carriage return and linefeed characters are inserted into user input to trick the server to respond as if an object is terminated and another one has started. Such CRLF injections can be used for HTTP Response Splitting.

4.2.3. CVE Details Types are Types of Attacks not Vulnerabilities

What CVE Details calls vulnerability types are very often not instanced as or named vulnerabilities in CVE records. Sometimes they are *also* instanced as or named attacks or more specifically, they are instanced as part of a causal chain of attack, often initiating it, but also sometimes being its consequence. From a cybersecurity ontological standpoint [40], the *concept* of a vulnerability encompasses weaknesses often caused by or tantamount to poor programming or bugs in software. They can be exploited and can be the basis for mounting an attack but are not attacks themselves. Part of the confusion results from being able to *label* a vulnerability without making clear conceptually what it is in itself. Rather it is referred to by way of the attacks or actions that it, uncharacterized though it is, allows. This can be useful for purposes of cataloging because there are many different weaknesses that allow a certain kind of attack. Such uniqueness of weaknesses makes sorting CVE records into classes that pick out patterns difficult. What would tend to result would be a long detailed list that would be practically impossible to work with. However, it is also good to avoid confusion. In order to detect both useful patterns and avoid confusion, all it takes is simply to be clear on what is being classified. It turns out, given our semantic analysis, that CVE is not so much about vulnerabilities as such, but more about the causal chains bringing attacks or exploits. DoS or Denial of Service is what attackers cause as a consequence of an Attack, and 'Memory Corruption' most often appears in parentheses after 'Denial of Service,' in other words as a kind of denial of service. Bypass something, Gain Information and Gain Privileges all can be seen as the consequence of an attack, though sometimes can be seen as an earlier element in the causal chain. All the other types either initiate attacks or are an element in the causal chain of an attack.

However, vulnerabilities are referred to in CVE Details and CVE itself, but it is not often clear what they are or what needs to be fixed. Nevertheless, they can be used to build a benchmark. All that is needed is to clarify that every CVE Details type can be characterized as a consequence or as an element in the causal chain of an attack. Once that is done the partitioning exhibited by CVE Details Vulnerabilities by Type can be useful to benchmark and test how well a topic model partitions the contents of CVE into types.

4.2.4. Refining CVE Details Types as a Benchmark to Evaluate CVE Topic Models

To refine and make more rigorous the use of CDVT as a benchmark to evaluate CVE Topic Modeling, much more extensive matching is needed. Matching terms in the topics to the names and the accepted meaning of CVE types, as specified by CDVT, is only the first step in a more refined and rigorous benchmarking process. Also needed is to match the CVE records indexed to a CVE Details type (accessible on the CVE Details website) to the documents indexed to the automatically generated R&R based topics correlated with the CVE Details types. This requires finding a way to measure the amount of overlap between these two sets of documents. This should be relatively straightforward because both are indexed to the same dataset and a common set of IDs. However, CDVT's types are indexed to their documents by what the CDVT site managers call keyword, though what keywords are used is not discussed. There are also disclaimers about how accurately the data in CDVT represents the data in the CVE List as maintained by the NIST National Vulnerability Database (NVD). CDVT is not a perfect benchmark, especially as it stands. No benchmark is. It will have to be thoroughly investigated, and refinements made. Even then it will have limitations and almost surely will not be used as a standard benchmark by itself, but rather as one of the initial steps toward establishing one. It does represent a data-driven way to begin to get an appreciation of how well R&R based topic modeling is doing at partitioning CVE data against categories that people are familiar with.

One obvious limitation of CDVT as a benchmark is how it determines what documents belong to a given type. Just using keywords is a limitation in the sense that after the obvious ones related to the name of the vulnerability are selected, it is not clear what additional keywords need to be added. One of the key features of topic modeling is that it Finds terms related to one another that are not simply literal matches, but nevertheless have some similarity of meaning. However, there are many words that have some probability of meaning overlap, but a very slight one. It is not clear what the cutoff point should be to select the terms that should count in determining a match between the terms in the topic and the terms in the documents selected as representative of the topic. For this comparison, CDVT no longer leads the way because it is based on literal matching. Rather, as long as the topics and their indexed documents cover the types and their indexed documents, the latter lead the way in Finding relevant documents that the types do not. In other words, the tested leads the way in suggesting an update of the benchmark that is used to test it. The relation between development and evaluation works both ways. We are getting a sense of just how, but more investigation is needed.

We propose that at least two capabilities are needed: 1) an exhaustive and rigorous testing of literal matching that measures both a) how well the terms in documents indexed to a topic match the terms in that topic, which is dependent on Finding a satisfactory number

of topics a topic model needs to get a meaningful partitioning of all the documents in a corpus and b) how well the topic model using literal matching is correlated with the respective partitioning of documents indexed to CVE Details types, and if they are so correlated, how much and how well they can then be used to extend understanding of the CVE Details types and 2) an exploratory and perhaps qualitative testing of how well documents indexed to topics which contain few and perhaps no literal matches nevertheless, for example, in the judgment of experts are representative of these topics.

Once these initial benchmarking and testing capabilities are in place, and testing and evaluation take place, processes for interpreting topics and their relation to the documents they index will be incrementally improved. Each improvement will inform the anticipatory capability of topic trend analysis because there will be a better understanding of why spurts are taking place. Anticipations within the germination period supporting successful proaction to mitigate spurts, while they are happening, or preventing them, will also redound to a higher evaluation score of the topic modeling that made it possible. On the other hand, a good score on the topic modeling processes will also encourage improving benchmarking and testing capabilities beyond literal matching.

4.2.5. How representative are the documents indexed to topics?

To determine representativeness, we match the single and multi-word terms in CVE documents to single and multi-word terms in the topic represented in the documents. In the case we are to consider in this section, we use the 20-topic topic model discussed above. We found that creating a topic model with just five more topics improved results without degrading results that had already been achieved. However, we use the 20-topic model because it has been much more discussed in this report, the results are similar except for what was pointed out at the end of section 4.2. The matching measures we discuss apply equally to both models.

Each topic is ranked according to the expected proportion of the contents of the CVE corpus it covers (see section 4.1) and the terms in each topic are ranked in order of their frequency of occurrence in documents that are representative of the topic. For purposes of discussing approaches to matching topic terms and document terms, we use the top 50 terms for each topic and show how the terms of the single most representative of the documents overlap with the terms in the topic. Looking at Topic 2 and its most representative document (both as a sequence of terms – see Fig. 9 – and natural language text – see end of section 4.1), highlighting shows different grouped items that could be matched for SQL Injection, Directory Traversal, and File Inclusion – green highlighting shows different ways of matching document terms to topic terms. Since the most representative document is about SQL Injection, terms including it and its associates are highlighted in green. Turquoise is used only on the topic terms to highlight Directory Transversal and its associates. Grey is used to highlight File Inclusion and its associates. Terms with no highlighting are either too vague to be of much use or there is no evidence yet from regular expression analysis of documents indexed to Topic 2 of which vulnerabilities, if any, they are associated with (See Fig. 17).

Topic 2: parameter, allow, attacker, remote:0:attacker, vulnerability, execute, command, injection,sql:0:injection, sql:0:injection:1:vulnerability, sql:0:command, arbitrary:1:sql:0:command,arbitrary:1:sql:0:command:2:execute, remote:0:attacker:3:arbitrary:1:sql:0:command:2:execute, file, traversal, directory, code, php:0:code, arbitrary:1:php:0:code, url, arbitrary:1:php:0:code:2:execute, traversal:0:vulnerability, remote:0:attacker:3:arbitrary:1:php:0:code:2:execute, inclusion, remote:0:file, dot, arbitrary:0:file, arbitrary:0:file:1:read, dot:0:dot, read, inclusion:0:vulnerability, remote:0:attacker:2:arbitrary:0:file:1:read, include, action, d:0:parameter, parameter:0:url, directory:0:traversal, directory:1:traversal:0:vulnerability, local:0:file, remote:0:file:1:inclusion:0:vulnerability, component, metacharacter, shell:0:metacharacter, note, sequence, module, arbitrary:1:local:0:file, remote:0:file:1:inclusion, remote:0:file:1:inclusion:2:vulnerability

Topic 2, Term Document 13657 1:0:announcement:1:parameter 1:0:announcement announcement parameter remote:0:attacker:3:arbitrary:1:SQL:0:command:2:execute remote:0:attacker attacker arbitrary:1:SQL:0:command:2:execute arbitrary:1:SQL:0:command SQL:0:command command execute allow multiple:3:vBulletin:2:SQL:0:injection:1:vulnerability vBulletin:2:SQL:0:injection:1:vulnerability SQL:0:injection:1:vulnerability SQL:0:injection injection vulnerability 3:0:criterion:1:parameter 3:0:criterion criterion 5:0:calendarcustomfieldid calendarcustomfieldid 6:0:calendarid calendarid 7:0:moderatorid moderatorid holidayid:2:10:1:9:0:calendarmoderatorid holidayid 10:1:9:0:calendarmoderatorid 109:0:calendarmoderatorid calendarmoderatorid cronid:0:parameter 12:0:parameter 1213:0:parameter13 15:1:14:0:limitnumber 14:0:limitnumber limitnumber limitstart:0:parameter ids:0:parameter 18:0:parameter 19:3:20:1:dostyleid:0:parameter:2:parameter 20:1:dostyleid:0:parameter:2:parameter20:1:dostyleid:0:parameter dostyleid:0:parameter dostyleid 22:1:21:0:parameter 21:0:parameter",

Fig. 17: Example of Matching Terms in Topic to Terms in Document for Topic 2

Matching document terms to topic terms shown here follows, for the most part, literal matching of whole terms – single-word terms in the documents literally match single-word terms in the topics and multi-word terms in the documents literally match multi-word terms in the topics excluding capitalization and possibly numbers between colons, though in this case the numbers also match. However, there are matches shown where a single-word term in a topic matches part of a term either in the topic or the document as in the case of the most prominent term in the topic, the single-word term parameter. This seems a reasonable deviation form exact literal matching in the case of very prominent single-word terms that are repeated in longer terms throughout a document, especially where the single-word term is playing the role of a general term being qualified in various ways with words that may be unique to one or a very few documents. The same goes for multi-word terms that are lower down in the implicit taxonomy of the corpus, here being made explicit, for example, multiple:3:vBulletin:2:SQL:0:injection:1:vulnerability.

By highlighting groups of terms in different groups of colors, we are also showing how terms in a topic model Tit not only taxonomic hierarchies, but also semantic schema slots. Terms that fill both CVE semantic schema slots found in both the topic and the document being matched would be weighted more highly since they have a more specific semantic relation to each other than other terms in the topic. It is also the case that topics that exhibit

the semantic schema in this way tend to be much more prominent in the topic model and certainly provide a better basis for interpreting them than those that do not fill slots in the schema.

It should be noted here again that the CVE semantic schema we discovered through regular expression search covers at least 65% of the records and probably more. An increase in percentage covered can be probably be produced in a number of ways. First, by adding more marker terms (e.g., "by' in addition to "via'). Second, an increase in percentage can also be produced by relaxing the need for certain schema slots to be filled, e.g., where the noun term at the beginning subject position of the sentence is not a named vulnerability but a noun term referring to the location a vulnerability would be in. Moreover, there may be other semantic schema that are not just modifications of the primary one we have focused on, but different ones with a main verb that is not "allow' or not a word that plays the same role as "allow' such as "permit' or even "enable.' Such a schema would also have different marker words and a different clausal structure.

Most corpora may not have a collection of semantic schemas with a dominant one like the CVE semantic schema we have been discussing. However, as has been exemplified by work to formalize information in sublanguages [e.g., most thoroughly in [18] such semantic schema arise in the analysis of other technical domains. Also, scientific and technical journals that collect technical papers online now have document schema and schema for abstracts, from which semantic schema could be derived. Moreover, work on case grammar, and argument structure[4] that have studied such semantic schema or formulaic structures have substantial literatures associated with them [starting with [41, 18, 42]. Computational formalization processes for deriving formulas, case frames, and argument structures from collections of text are still in need of breakthroughs, but see [43]. We hope to contribute to these breakthroughs in future work.

5. Conclusion: Toward a Cybersecurity Knowledge Sharing and Creation Platform

We have used R&R structured terms derived from the records in the CVE List as the building blocks for automatically making the implicit taxonomic structures, topic models and conceptual networks in the List explicit. Using these resulting semantic structures has helped us better understand the relationships between the conception of vulnerabilities named in the CVE List and classified in CVE Details. It also helped us characterize use of the CDVT interactive online table as a benchmark for initially comparing and evaluating CVE topic models and their indexed documents with the types and documents indexed via the CDVT table. We now turn to how our approach might help us in collaboration with others both inside and outside NIST, including some who have been explicitly working on cybersecurity ontology [Syed et al. 2016] to advance a data driven combined developmental-evaluative approach to refining and elaborating a unified cybersecurity ontology.

[4] Argument structure does not mean structure of reasoning to a conclusion for persuading others. It is the idea that verbs are rarely uttered in isolation but are usually accompanied by certain other words, called the arguments of the verb. Grammars can be used to determine how participants to verbal events are expressed in sentences or clauses.

The idea of cybersecurity ontology is well laid out in the Syed et al. paper. We agree that ontology is useful for unifying information from heterogeneous sources and supporting reasoning and rule writing. The point of making explicit the semantic structures in the CVE List (and from other cybersecurity information sources) is to build a data-driven basis for standard setting ontologies in different technical domains like cybersecurity. These can be used to support an accurate and detailed way of reporting domain specific information and provide a common basis for interpreting the patterns and trends that can be derived using tools like the ones we have described. In other words, such ontologies provide a basis for aiding the specification of standards not only for reporting old and new cybersecurity information, but also a basis for interpreting the patterns and trends and inferring new information (like identifying cybersecurity threat within the germination period) from existing information. The semantic schema that we derived from the contents of the CVE List provides a basis for capturing and expressing the evolving knowledge recorded in the CVE List and other text-based information sources. Ontology classes can be discovered in any technical domain and organized in ontological repositories as a basis for constructing case frames according to heuristic rules and then using the case frames to support user interaction. Users can be supported through visualization and other means of interaction to: 1) formulate and add new information as instances of the case frames; 2) interpret and evaluate the semantic structures produced by the R&R approach or similar approaches and 3) build models and other ways of sharing understanding that can generate new assertions that can accrue evidential support through formal testing or feedback from practice (successful or unsuccessful mitigations of cybersecurity risk).

Toward this ontological vision, our project connects with the NIST Bugs Framework (BF) project, led by Dr. Irena Bojanova. BFs goal is to define classes of software bugs and weaknesses that allow clear descriptions of vulnerabilities that exploit them, which also serves the aim of clarifying the information in CVE records. BF work could use R&R building blocks and semantic structures to extend the semantic schema already identified to additional ones that cover other parts of the contents of the CVE List and refine and elaborate the schema already identified. As discussed above, these advances would benefit future reporting, interpretation and analysis of trends as well as retroactive updates of the current CVE List reporting structure.

Our team and the BF team collaborated on extending the characterization of the memory classes they were working on to include 1000s of instances of attributes, causes, consequences and sites. We used as a starting point for this extension is based on [44]. The paper served as the initial round of collaborative work followed by examination of the instances (data) that our automated tools identified and that could serve as a basis for confirmation, in certain cases, and revision in others. Our collaborative examination of CVE records, showing where bugs actually happen – where in a source code the flaw is – could serve as a basis for evaluation of cybersecurity frameworks. For example, we found that our team and the BF team agreed on where and why CVE vulnerability descriptions were lacking. Based on our team's prior uncovering of the current contours of the tacit CVE semantic schema, our team and the BF team agreed that the schema often fails to provide an accurate and complete enough description of the causal processes involved in characterizing vulnerabilities (see section 4.2.3). Moreover, since that CVE semantic schema also overlaps with Bugs Framework causal descriptions, modifying that schema could be done while maintaining CVE List continuity and comprehensibility. Moreover, combining Bugs Framework conceptual distinctions with our team's data-driven analyses to show where CVE

descriptions are in need of refinement and elaboration, could at the same time provide confirming evidence for any needed modifications.

Such investigations could be extended to cover other frameworks in addition to CVE Details, first and foremost, perhaps, STIX, for Structured Threat Information Expression (STIX™). STIX (https://oasis-open.github.io/cti-documentation/stix/intro.html) is an open source language and serialization format used to exchange cyber threat intelligence (CTI - see OASIS Cyber Threat Intelligence TC - https://www.oasis-open.org/committees/tc_home.php? wg_abbrev=cti). Note that an open source, community-centered platform to store, organize, visualize, share and extend knowledge about cyber threats called OPENCTI is already operating on the web (https://www.opencti.io/en/). STIX, CTI and OPENCTI could greatly benefit from the kind of collaborative investigation our team and the BF team started.

These will be the final points. Currently the BF rests on human experts for the conceptual framework used to classify bugs. The problem is that even for such frameworks, and cybersecurity frameworks are no exception, experts disagree on what conceptual distinctions need to be made. Combining the topdown conceptual distinctions of BF and the bottom up classifications of actual cybersecurity incident reports based on R&R term structures filling semantic schema could bring to bear empirical evidence on the development of such frameworks. Moreover, adding empirical to conceptual analysis provides an additional basis for reaching flexible agreements among experts. However, even establishing agreements among experts is not the final test of the worth of a conceptual framework. Another step would be to put such a framework to the test by using it as a schema for a database platform to store, organize, visualize, share and extend knowledge - in our case cybersecurity knowledge. Perhaps a project using BF memory bugs classes patterns described [44] to evaluate current STIX object and relational semantic structures would be worth considering. The aim would be to modify and extend these concepts for causal structure elaboration to be used in a semantic schema for a database platform like OPENCTI. Use of such a modified OPENCTI, suitably instrumented not only to measure user experience in storing, organizing, visualizing, sharing, and extending cybersecurity knowledge but also allowing users to suggest changes to these processes, especially to the schema itself, would provide insight into its value.

References

[1] Bhat TN., Collard J., Subrahmanian E., Sriram R.D., Elliot JT., Kattner UR., Campbell CE., Monarch I. (2019) "Generating Domain Ontologies Using Root- and Rule-Based Terms," Journal of Washington Academy of Science, December 2019. NIST Information Technology Laboratory.

[2] Coulter N, Monarch I, and Konda S, (1998) “Software engineering as seen through its research literature: A study in co-word analysis,” J. Amer. Soc. Inf. Sci., vol. 49, no. 13, pp. 1206–1223.

[3] Monarch I (2000) “Information Science and Information Systems: Converging or Diverging?,” Proceedings of the Annual Conference of CAIS.

[4] Monarch I, Fisher D, and Nag PK "Augmenting Homeland Intelligence Through Emergent Semantics", Infotech@Aerospace, Infotech@Aerospace Conferences, https://doi.org/10.2514/6.2005-7111, no. 13, pp. 1206–1223.

[5] Common Vulnerabilities and Exposures (2018) CVE is sponsored by US-CERT in the office of Cybersecurity and Communications at the U.S. Department of Homeland Security. Copyright © 1999–2018, The MITRE Corporation.

[6] Blei DM, and Lafferty JD. (2009) Visualizing topics with multi-word expressions. arXiv preprint arXiv:0907.1013.

[7] Chaney AJB, Blei DM (2012) Visualizing topic models. International Conference of Weblogs and Social Media, ser ICWSM '12.

[8] Chang J, Boyd-Graber J, Wang C, Gerrish S, Blei DM (2009) Reading tea leaves: How humans interpret topic models. Neural Information Processing Systems,NIPS.

[9] Wallach HM (2006), Topic modeling: beyond bag-of-words, in: Proc. 23rd Int. Conf. Mach. Learn., ACM, 2006, pp. 977- 984.

[10] Wang X, McCallum A, Wei X (2017) Topical n-grams: Phrase and topic discovery, with an application to information retrieval. Proceedings of the 7th IEEE Intl Conf on Data Mining, ser ICDM '07. 2007:697–702.

[11] El-Kishky, A.; Song, Y.; Wang, C.; Voss, C. R.; and Han, NJ (2014). Scalable topical phrase mining from text corpora. *Proceedings of the VLDB Endowment* 8(3):305–316.

[12] Danilevsky, M.; Wang, C.; Desai, N.; Ren, X.; Guo, J. and Han, J (2014). Automatic construction and ranking of topical keyphrases on collections of short documents. In *Proceedings of the SIAM International Conference on Data Mining (SDM)*.

[13] Ożdżyński P and Zakrzewska (2016 , *A search of significant phrases for building topic models in text documents,* Information Systems in Management (2016) Vol. 5 (2) 205-214

[14] He, Y. (2016) Extracting Topical Phrases from Clinical Documents, Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence (AAAI-16).

[15] Wallach, H. M., Murray, I., Salakhutdinov, R., and Mimno, D. (2009) Evaluation Methods for Topic Models. In Proceedings of the 26th Annual International Conference on Machine Learning, ICML ’09, pages 1105–1112, NewYork, NY, USA. ACM.

[16] Boyd-Graber J, Hu Y, Mimno D (2017) Foundations and Trends® in Information Retrieval 11 (2-3), 143-296

[17] Brandom R (1994) Making It Explicit: Reasoning, Representing and Discursive Commitment, Harvard University Press.
[18] Harris Z et al. (1989) *The Form of Information in Science: Analysis of an Immunology Sublanguage*, Zellig Harris, Michael Gottfried, Thomas Ryckman, Paul Mattick Jr., Anne Daladier, T. N. Harris, and S. Harris, Dordrecht: Kluwer Academic Publishers,, 590
[19] Grishman R and Kittredge R (2014) Analyzing language in restricted domains: sublanguage description and processing: Psychology Press.
[20] Lehmann F (1992) Semantic Networks, *ComputerJ Math. Applic.* Vol. 23, No. 2-5, pp. 1-50.
[21] Wanjawa BM and Muchemi L (2018) Automatic Semantific Network Generation from Unstructured Documents – The Options. October 2018, Conference: 2018 5th International Conference on Soft Computing & Machine Intelligence (ISCMI)
[22] Ji, L, Wang Y, Shi B, Zhang D, Wang Z and Yan J (2019) Microsoft concept graph: Mining semantic concepts for short text understanding. Data Intelligence 1(2019). doi: 10.1162/dint_a_00013
[23] [Zhao et al. 2018] Zhao Y, Fesharaki NJ, Liu H and Luo J (2018) Using data-driven sublanguage pattern mining to induce knowledge models: application in medical image reports knowledge representation, BMC Medical Informatics and Decision Making pp. 18:61
[24] Collard J, Bhat TN, Elliott J, and Sriram R. Monarch I, and Subrahmanian E (2020), Information Retrieval with Root- and Rule-Based Terms. Available at SSRN: https://ssrn.com/abstract=3565983 or http://dx.doi.org/10.2139/ssrn.3565983
[25] Blei, D (2012) "Probabilistic Topic Models". Communications of the ACM. 55 (4): 77–84
[26] Landauer, T., & Dumais, S. (1997) A solution to Plato's problem: The latent semantic analysis theory of acquisition, induction, and representation of knowledge. *Psychological Review*, **104**, 211-240.
[27] Landauer, T., Foltz, P., & Laham, D. (1998), Introduction to latent semantic analysis. *Discourse Processes*, **25**, 259-284.
[28] Deerwester et al. 1990]Deerwester S, Dumais ST, Furnas GW, Landauer TK, and Harshman R(1990) Indexing by latent semantic analysis. Journal of the AmericanSociety of Information Science, 41(6): 391-407, 1990.
[29] T. Hofmann T (1999). Probabilistic latent semantic indexing. *Proceedings of the Twenty-Second Annual International SIGIR Conference*, 1999.
[30] Blei D and J. Lafferty J (2006) Dynamic topic models. In Proc. of the 23rd International Conference on Machine Learning, ACM, 2006.
[31] Roberts ME, Stewart BM, and Airoldi EM (2016) A model of text for experimentation in the social sciences. Journal of the American Statistical Association, 111(515): 988-1003, 2016.
[32] Beykikhoshk A, Arandjelovi´c O and Phung OD (2018) Discovering topic structures of a temporally evolving document corpus, Knowl Inf Syst (2018) 55:599–632, https://doi.org/10.1007/s10115-017-1095-4

[33] Yang Y, Pan S and Song Y (2016) Improving Topic Model Stability for Effective Document Exploration, Proceedings of the Twenty-Fifth International Joint Conference on Artifiicial Intelligence (IJCAI-16).
[34] Mark Belford M, MacNamee B and Greene D (2017), arXiv:1702.07186v2 [cs.IR] 9 Sep 2017.
[35] Sangjin Shin S and Lee K (2020) Processing knowledge graph-based complex questions through question decomposition and recomposition, Information Sciences, Vol. 523, pp. 234-244, 2020.
[36] Chen, HM., Kazman, R., Monarch, I. and Ping Wang, P (2017) "Can Cybersecurity Be Proactive? A Big Data Approach and Challenges," in Proceedings of the 50th Hawaii International Conference on System Sciences.
[37] Chen, HM., Kazman, R., Monarch, I. and Ping Wang, P (2016) "Predicting and Fixing Vulnerabilities before They Occur: A Big Data Approach," Published in: Big Data Software Engineering (BIGDSE), 2016 IEEE/ACM 2nd International Workshop
[38] [Carr et al. 1993] Carr M, Konda S, Monarch I, Walker CF and Ulrich FC, Taxonomy-Based Risk Identification(1993), Publisher: Software Engineering Institute, CMU/ SEI-93-TR-006, 1993.
[39] Monarch, I (1994). An interactive computational approach for building a software risk taxonomy. The Third SEI Conference on Software Risk, Pittsburgh, PA
[40] Syed Z., Padia, A, Mathews M., Finin T., Joshi A (2016) UCO: a unified cybersecurityontology. In: Proceedings of the AAAI Workshop on Artificial Intelligence for Cybersecurity.
[41] Fillmore CJ (1968) "The Case for Case". In Bach and Harms (Ed.): *Universals in Linguistic Theory*. New York: Holt, Rinehart, and Winston, 1-88.
[42] Goldberg AE (1995), *A Construction Grammar Approach to Argument Structure*, Chicago: Chicago UP, 1995.
[43] van Trijp R (2016). *The evolution of case grammar* (Computational Models of Language Evolution 4). Berlin: Language Science Press
[44] Bojanova I and Eduardo Galhardo C (2021) "Classifying Memory Bugs Using Bugs Framework Approach," 2021, IEEE 45th Annual Computers, Software, and Applications Conference (COMPSAC), 2021, pp. 1157-1164, doi:10.1109/COMPSAC51774.2021.00159