DOI: 10.15587/978-617-7319-98-5.CH4

Kateryna Potapova, Mykola Nalyvaichuk, Vasyl Meliukh,
Stanislav Gurynenko, Kostiantyn Koliada, Alexandre Scherbyna

CHAPTER 4

SEMANTIC ROLE LABELLING AND ANALYSIS IN ECONOMIC AND CYBERSECURITY CONTEXTS USING NATURAL LANGUAGE PROCESSING CLASSIFIERS

ABSTRACT

Semantic Role Labeling (SRL) is a crucial task in Natural Language Processing (NLP) that plays a vital role in extracting meaningful information from text. In the fields of economics and cybersecurity, accurately identifying and analyzing semantic roles within text is crucial due to the rapid increase in the amount and complexity of textual information. This abstract examines the significant role of SRL and its application in economic and cybersecurity contexts. It discusses the state-of-the-art NLP classifiers used for this purpose. By examining the relationship between language processing and these important areas, we aim to emphasize the importance of SRL in extracting useful information and improving decision-making in a constantly changing digital environment.

The aim of the findings is to emphasize the significance of SRL in extracting valuable insights from text, as it serves as a fundamental technique in NLP. It is utilized in the economic context to analyze financial reports, news articles, and economic texts. It assists in decision-making and market analysis. It aids in identifying important participants, actions, and objects in economic discourse, leading to better decision-making and market analysis. In the field of cybersecurity, SRL assists parse and comprehending text data related to security, enabling faster responses to threats. NLP classifiers and machine learning models utilize SRL to automate the analysis of large amounts of text. These techniques are practically significant as they improve the ability to extract actionable insights, assess risks, and make informed decisions by organizing unstructured text data.

The process of determining relevant information from a large corpus of data requires an optimal methodological basis. Relevant textual data is collected from sources such as financial reports, news articles, or cybersecurity incident reports. Textual data is cleaned, tokenized, and tagged with part-of-speech labels in preparation for NLP analysis. Human annotators label semantic roles in the text, identifying actors, actions, and objects. This creates a dataset that can be used to train classifiers. NLP classifiers, including machine learning models, are trained using annotated datasets to identify semantic roles. The accuracy and performance of the trained classifiers are evaluated using various metrics. NLP classifiers are used to automatically identify and label semantic roles in new, unseen textual data. The output helps extract insights, such as market trends or security threats, depending on the specific field. Researchers improve classifier models by iteratively training and applying them to increase accuracy.

KEYWORDS

Semantic Role Labelling, NLP Techniques, Economic Data Analysis, Cybersecurity Text Mining, Semantic Role Extraction, Sentiment Analysis, Information Security.

In an age dominated by digital information and economic interdependence, the realms of economics and cybersecurity have gained unprecedented significance. The vast amount of textual data generated in these fields calls for advanced techniques to extract valuable information. Semantic Role Labelling (SRL) has emerged as a powerful Natural Language Processing (NLP) approach to analyze the complex relationships and roles of entities within text [1]. It aims to label words in a sentence with different semantic roles for the verb in the sentence. SRL has gained significant attention in computational linguistics and has become a leading task in the field [2]. It plays a crucial role in understanding the meaning and structure of sentences, enabling various downstream applications such as information extraction, question answering, and machine translation.

The significance of understanding the semantic roles in economic and cybersecurity texts cannot be overstated. In economics, we often deal with massive datasets, financial reports, and market sentiment analysis, where identifying roles like "buye", "seller", "investor", or "regulator" is crucial for informed decision-making. By accurately identifying the semantic roles of predicates in these texts, NLP classifiers can assist in automating tasks such as sentiment analysis, trend prediction, and risk assessment. Similarly, in the cybersecurity context, SRL can be used to analyze security-related documents, incident reports, and online discussions to identify potential threats, vulnerabilities, and malicious activities. By understanding the semantic arguments of predicates in these texts, NLP classifiers can aid in detecting patterns, identifying key actors, and predicting future cyberattacks.

The use of NLP classifiers in these domains can provide valuable insights and assist in decision-making processes. The text discusses the importance of efficient inference and structured learning techniques for accurate labeling in SRL (Structured Learning). It emphasizes the need for efficient inference and structured learning to achieve accurate labeling. These techniques can enhance the accuracy and reliability of SRL classifiers in economic and cybersecurity applications [1].

One approach to SRL is the use of data-driven models based on supervised learning, which have become the method of choice for semantic role labeling [2, 3]. These models leverage neural architectures and adapt them to the SRL task. Lample et al. adapted a similar model used for Named Entity Recognition and applied it to the SRL task. This approach demonstrates the potential of leveraging neural networks for SRL in different domains [4].

To achieve accurate and reliable Semantic Role Labeling (SRL), it is essential to leverage existing resources such as annotated corpora of semantic roles and ontologies [4]. These resources provide valuable knowledge and semantic mappings that can improve the performance of NLP classifiers. It proposes a method that leverages WordNet's upper ontology mapping and PropBank-style Semantic Role Labeling (SRL) for parsing long texts. This approach demonstrates the potential of

integrating ontological knowledge into SRL classifiers for a more comprehensive analysis. Semantic Role Labelling and Analysis using NLP classifiers have significant potential to enhance decision-making and situational awareness. By accurately labeling semantic roles and analyzing the relationships between entities and predicates in texts, these classifiers can provide valuable insights and support decision-making processes in these domains.

4.1 CONVENTIONAL PARADIGMS OF EMPIRICAL AND RATIONAL APPROACHES TO THE AMBIGUITY OF LINGUISTIC DURING THE DEVELOPMENT OF NATURAL LANGUAGE PROCESSING

Conventional paradigms in empirical and rational approaches to the ambiguity of language during the development of natural language processing have been extensively studied in the field of computational linguistics and artificial intelligence. Researchers have explored various aspects of language processing, including statistical natural language processing, individual differences in language processing, unsupervised learning of morphological paradigms, and ambiguity resolution [5].

Early approaches to SRL relied on rule-based systems, although they have undergone significant changes over time. These systems had a limited scope and necessitated extensive linguistic expertise. The development of machine learning methods, especially deep learning, has transformed SRL by enabling models to learn from vast text corpora. With increased accuracy and scalability resulting from this change, SRL is now more suitable for a variety of fields, including economics and cybersecurity.

The aim of linguistic science, in contrast, is to be able to describe and explain the wide range of linguistic observations that we frequently come across in speech, writing, and other media. An essential part of this process is recognizing the connections between linguistic expressions and the outside world, the linguistic structures through which language expresses meaning, and the cognitive aspects of how people acquire, produce, and comprehend language [6]. It has been suggested that there are rules used to shape language utterances as a way to address the final issue with rules. This fundamental method has a long history that dates back at least 2000 years. However, in this century, it became more formal and rigorous as linguists delved into complex grammars that aimed to determine which utterances in a language were grammatically correct and which were not.

In the field of natural language processing, there has been extensive debate on the empiricist and rationalist approaches to language. While the empiricist approach stresses the value of experience and sensory information, the rationalist approach places more emphasis on the role of innate knowledge and mental structures in language acquisition [5, 6]. A rationalist approach completely dominated the fields of linguistics, psychology, AI, and natural language processing from around 1960 to 1985. In the context of artificial intelligence, rationalist ideas can be regarded as supporting the effort to develop intelligent systems by manually hand-programming a large amount of initial knowledge and reasoning processes into them in an effort to replicate what the human brain initially contains. The rationalist school of thought contends that universal grammar and natural linguistic talents make learning a language easier. They argue that humans are born with a set of cognitive

mechanisms that are unique to language and facilitate the process of language learning [6]. The theory of generative grammar contends that all languages share underlying syntactic features, which supports this point of view [7]. The rationalist approach also emphasizes the importance of reflection and introspective methods for language comprehension [8].

Conversely, empiricists argue that language acquisition is largely influenced by exposure to linguistic information and the statistical regularities present in the environment. According to them, language can be learned without the use of innate information or mental processes through a process of generalization and pattern recognition [9]. Empiricists emphasize the importance of corpora and data-driven methodologies in language research. They also emphasize how the linguistic environment influences the development of language [8, 9].

Both empiricist and rationalist approaches have had an impact on the field of natural language processing. The development of formal grammars and rule-based language processing systems was influenced by the rationalist perspective [10]. To analyze and produce language, these systems rely on explicit linguistic structures and rules. In contrast, the empiricist method has sparked the growth of statistical and machine learning methods for language processing [11]. These methods create predictions about language and learn patterns from large amounts of data. The fundamental principle of empiricist approaches, however, is to assume that the mind does not initially possess comprehensive collections of rules and guidelines specific to different aspects of language and other cognitive domains (such as theories of morphological structure, case marking, and so on), and that the mind instead develops over time. An empiricist approach to NLP contends that by selecting an appropriate generic language model and then determining the parameter values through statistical pattern recognition, we can acquire a deep and comprehensive understanding of the structure of language. An empiricist approach to NLP contends that by selecting a suitable general language model and then applying statistical, pattern recognition, and machine learning techniques to a vast amount of language data, we can learn the complex and extensive structure of language.

An NLP system must understand the structure of the text, typically to a sufficient extent to answer the question "Who did what to whom"? Conventional parsing algorithms simply attempt to answer this question by considering potential grammatical structures for a specific set of words within a specific category [12]. For instance, a typical NLP system will indicate that the sentence in **Fig. 4.1** has multiple syntactic analyses, also referred to as parses, based on a valid grammar. The given parse tree represents a complex sentence stating that the central bank raised interest rates in an attempt to manage inflation. This tree structure provides a visual representation of the grammatical and syntactic elements of the sentence, facilitating understanding of its hierarchical structure and the relationships between its various parts. A useful NLP system must be proficient in distinguishing between words with similar meanings and words with different categories, syntactic structures, and semantic scopes. However, when using symbolic NLP systems, expanding the language coverage to include obscure formulations actually leads to more unwanted interpretations for common phrases, and vice versa. This means that the goal of maximizing coverage while minimizing ambiguity is fundamentally incompatible.

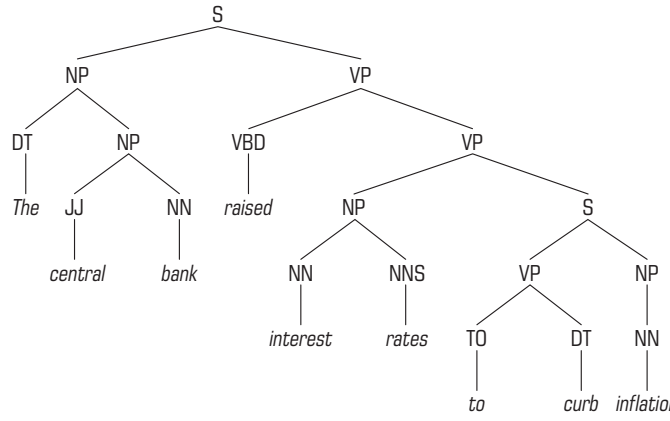

Fig. 4.1 A parse tree for a sentence "The central bank raised interest rates to curb inflation"

In the end, two opposing viewpoints on language learning can be seen in the rationalist and empiricist approaches to natural language processing. The empiricist method places more emphasis on the role of experience and statistical regularities than the rationalist approach, which emphasizes innate knowledge and mental structures. Both approaches have influenced the development of language processing tools, with empiricist approaches guiding the creation of statistical and machine learning techniques, and rationalist approaches influencing rule-based systems [13].

These issues are addressed by a statistical NLP technique that automatically learns lexical and structural preferences from corpora. Instead of relying exclusively on syntactic categories, such part of speech labels, to parse sentences, we understand that the associations between words – that is, which words tend to cluster together – contain a wealth of information [5, 8]. It is possible to use this collocational knowledge as a window into deeper semantic linkages. Because statistical models are stable, generalize well, and behave graciously in the presence of errors and new data, they are particularly useful for solving the ambiguity problem. In order to successfully provide disambiguation in large-scale systems that use naturally occurring text, statistical natural language processing (NLP) approaches have taken the lead. Statistical NLP models' parameters can often be automatically estimated from text corpora. This not only reduces the reliance on human labor in the development of NLP systems but also raises interesting scientific questions about the process of language acquisition in humans [5].

For groups of words W in a vocabulary V that number in the tens or hundreds of thousands, a statistical language model calculates the prior probability values $P(W)$. Typically, the string W is divided into sentences or other pieces that are presumed to be conditionally independent, like utterances in automatic speech recognition. Decomposing the sentence probability in accordance

with the chain rule and ensuring that the end-of-sentence symbol is predicted with a non-zero probability in any context are two straightforward and sufficient methods to ensure proper normalization of the model. $W = w_1, w_2, \ldots, w_n$ results in:

$$P(W) = \prod_{i=1}^{n} P(w_i \mid w_1, w_2, \ldots, w_{i-1}). \tag{4.1}$$

The language model is required to group the context $W_{k-1} = w_1, w_2, \ldots, w_{k-1}$ into an equivalence class determined by a function $\Phi(W_{k-1})$ because the parameter space of $P(w_k \mid w_1, w_2, \ldots, w_{k-1n})$ is too large. The result is:

$$P(W) \cong \prod_{i=1}^{n} P(w_k \mid \Phi(W_{k-1})). \tag{4.2}$$

In a real application, word strings of limited length are encountered. The support of $P(W)$ should consist of strings of finite length. The probability distribution $P(W)$ should assign a probability of 0.0 to strings of words of indefinite length. A language model must anticipate the specific end-of-sentence symbol since the text is divided into sentences in a real-world context. If the language model is straightforward, or in other words, if $P(w_k \mid \Phi(W_{k-1})) > \epsilon > 0, \forall w_k, W_{k-1}$, after that we have $P(< /s > \mid \Phi(W_{k-1})) > \epsilon > 0, \forall W_{k-1}$. It guarantees that the model gives the set of word sequences with a fixed length a probability of 1.0 [13].

Finding appropriate equivalence classifiers and ways to assess the likelihood are the focus of language modeling research. The $(n-1)$-gram equivalence classification, which defines $\Phi(W_{k-1})$, is the most effective paradigm for language modeling, leaving the problem of estimating probability from training, and it defined like that:

$$\Phi(W_{k-1}) = w_{k-n+1}, w_{k-n+2}, \ldots, w_{k-1}. \tag{4.3}$$

Natural language processing (NLP) is one of the many fields where Zipf's law has been seen statistically. It claims that a word or term's rank is negatively correlated with how frequently it occurs in a corpus [14]. In other words, the majority of terms only occasionally occur, but a select minority do. For applications like semantic labeling, this law has been frequently used in NLP [11]. The generative modeling of natural languages is one area in which Zipf's law is applied in NLP. Researchers can create models that produce realistic and cohesive writing and get insights into the structure and qualities of a language by examining the frequency distribution of words in a corpus.

We can examine the correlation between the frequency of a word, denoted as f, and its position in a list, referred to as its rank, r, by tallying the occurrence of each word (type) in a large corpus. Subsequently, we can arrange the words in ascending order based on their frequency [13].

$$f \alpha \frac{1}{r}, \text{ or } f = \frac{1}{r}. \tag{4.4}$$

Meaning that there is a constant k:

$$f * r = k, \tag{4.5}$$

$$f(r) = \frac{c}{(r+\beta)^{\alpha}}. \tag{4.6}$$

Its idea holds that both the speaker and the hearer are attempting to exert the least amount of effort possible. A short vocabulary of common words saves the speaker's effort, whereas a wide vocabulary of words that are individually rarer saves the hearer's effort by making communications less ambiguous. The kind of reciprocal relationship between frequency and rank that can be seen in the data proving Zipf's rule is said to be the most economically advantageous solution to these conflicting needs. The biggest impact of Zipf's law for us, nevertheless, is the practical issue that most words will have extremely scant usage data. We will only have a large number of samples for a few terms. As we can observe, Zipf's law roughly holds, however it varies significantly for the terms with the highest frequency. Additionally, a phenomenon that may be seen in many of Zipf's own research has been recognized, namely that the product f and r tend to show a tiny bulge for terms of higher ranks [5]. Although it is only a generalization, Zipf's law is helpful in describing the frequency distribution of words in human languages. It claims that there are a small number of highly common words, a substantial number of terms with medium frequency, and a significant number of words with low frequency.

The broader link between rank and frequency in order to achieve a better fit with the empirical distribution of words. that Mandelbrot discovers is as follows:

$$f = P(r+\rho)^{-\beta} \text{ or } \log f = \log P - B * \log(r+\rho). \tag{4.7}$$

P, B are text parameters that together gauge how varied the vocabulary is in the text. As in the original equation, there is still a hyperbolic distribution between rank and frequency. When this equation is plotted on logarithmic axes, it closely resembles Zipf's law as a declining straight line with slope B for large values of r. However, by appropriately selecting the other parameters, one can construct a curve where the expected frequency of the most common terms is lower.

4.2 ANNOTATING LINGUISTIC STRUCTURE

The act of parsing can be seen as a simple application of the concept of chunking, which allows to condense the description of a phrase by identifying higher level structural pieces. Learning a grammar that explains the structure of the chunks one encounters is one approach to grasp the regularity of chunks over various phrases. The issue with grammatical induction is this. The concept of syntactic constituency holds that word groups can function as single entities, or constituents. The process of creating a grammar includes creating a list of the language's components. Due to its

exploration of the empiricist challenge of how to acquire structure from unannotated textual input, grammar induction has received a lot of attention. It suffices to argue that, while context-free or more complicated languages of the size required to handle a respectable percentage of the complexity of human languages, grammar induction approaches are quite difficult to understand. It is simple to introduce organization into a corpus of text. A chunked representation of sentences, which we might understand as a phrase structure tree, will be produced by any algorithm for creating chunks, such as one that recognizes common subsequences. The process of mechanically examining a given sentence, seen as a sequence of words, in order to identify any potential underlying grammatical structures is known as parsing in the field of natural language processing [2].

Parsing requires a formal, grammar-based, mathematical representation of the syntax of the target language. A formal grammar is made up of a set of rules that define how language constituents, such as words, may be put together to make sentences and how sentences should be put together. The subject-verb agreement, word order, and other purely syntactic details may be the focus of rules, but other models may additionally include considerations like lexical semantics. There are many different grammatical formalisms that rely on different syntactic theories, and the structures that are produced by parsing, or parses, can vary greatly between these formalisms. A phrase structure is an ordered, labeled tree that expresses hierarchical interactions among specific groupings of words called phrases [6]. It is one of the many formalisms that explain the syntactic analysis of a sentence. Dependency structure, which indicates binary grammatical relationships between words in a phrase, is another option for representation. There exist "shallow" representations of syntactic structures, in contrast to "deep" representations, where the maximum depth is strongly constrained. Finite state approaches are commonly used to create such representations.

Parse structures are primarily significant because of the grammatical information they provide to modules that carry out semantic, pragmatic, and discourse processing. This information is essential for tasks such as text summarization, question-answering, and machine translation. Parsing can be seen as a crucial element of traditional NLP systems, and the accuracy of the parses can greatly affect the overall effectiveness of an application.

A natural language grammar often allows for a wide range of parses for a given input sentence. This is because formal grammars often overlook important aspects of a language's structure, meaning, and usage, leading to numerous parses that people would not consider logical. By computing isolated portions of the parses and saving them in a table, significant practical issues in computation and storage can be avoided. This has the benefit of allowing multiple parses to utilize the same fragment. Tabular parsing is the correct term for this. Numerous tabular parsing techniques can compute and store exponentially many parses while only requiring polynomial time and space.

Probabilistic parsing, which depends on the attribution of probabilities to grammatical rules, is one special case of this. A parse's probability is calculated as the sum of the probabilities of the rules used to construct it. By selecting the parse with the highest likelihood, disambiguation is achieved. Contrary to approaches that rely on a deep understanding of linguistics for syntactic disambiguation, probabilistic parsing and weighted parsing are successful due to their adaptability and scalability.

Context-free grammar, or CFG, is a widely used formal method for representing constituent structure in natural language. Phrase-structure grammars are another name for context-free grammars, and Backus-Naur form, often referred to as BNF, is the formalization used. It wasn't until Chomsky that the concept of basing a language on constituent structure was formalized [6].

To the left of the arrow, each context-free rule has a single non-terminal symbol that denotes a cluster or generalization, and to the right, an ordered list of one or more terminals and non-terminals. The lexical category or part of speech for each word is connected to a non-terminal in the dictionary [6].

A context-free grammar (CFG) can be viewed in two different ways: as a mechanism for generating new sentences and as a means of assigning a specific structure to a sentence. We can interpret the arrow as meaning to "rewrite the symbol on the left with the string of symbols on the right" if we think of a context-free grammar as a generator [15].

The formal language, as defined by a CFG, consists of the collection of strings that can be derived from the selected start symbol. Each grammar must have a specific start symbol, commonly referred to as *S*. *S* is typically considered the "sentence" node because context-free grammars are often used to generate sentences. The other nodes are defined in **Table 4.1**. Grammatical sentences are those that can be produced from a grammar and are part of the formal language specified by that grammar. Ungrammatical sentences are those that cannot be generated by a specific formal grammar and do not belong to the language that the grammar defines. This is because context can often determine whether a sentence belongs to a specific natural language.

A lexicon consists of words and symbols, while a collection of rules or productions defines the different ways in which the symbols of a language can be combined and arranged. Together, these components form a context-free grammar. Rule-free context: we can combine the previous rules with others that describe details about the vocabulary specified by the four parameters *T*, *N*, *R*, and *S* because *G* can be hierarchically embedded:

$$G = < T, N, S, R > . \tag{4.8}$$

Each of the parameters contains the following data:

– *T* is set of terminal symbols that corresponds to words in the language (lexicon);

– *N* is set of non-terminal symbols that express abstractions over the terminals (or variables);

– *S* is start symbol (one of the non-terminals);

– *R* is rules/productions of the form $X \rightarrow \gamma$, where *X* is a nonterminal and γ is a sequence of terminals and non-terminals (may be empty);

– *A* grammar *G* generates a language *L*.

As shown in the **Fig. 4.2**, *h* a given context-free grammar defines the rules for constructing valid sentences. Context-free grammar rules specify the relationships between different constituents, such as subjects, verbs, objects, and modifiers. The parsing process disambiguates such sentences and selects the most appropriate syntactic structure.

Table 4.1 Token label definitions of lexical parsing processes

Token Label	Description
ADJP	Adjective Phrase
ADVP	Adverb Phrase
CONJP	Conjunction Phrase
FRAG	Fragment
INTJ	Interjection
NAC	Not a constituent
NP	Noun Phrase
NX	Head subphrase of complex noun phrase
PP	Prepositional Phrase
QP	Quantifier Phrase
RRC	Reduced Relative Clause
S	Simple declarative clause (sentence)
SBAR	Clause introduced by complementizer
SBARQ	Question introduced by wh-word
SINV	Inverted declarative sentence
SQ	Inverted yes/no question
UCP	Unlike Co-ordinated Phrase
VP	Verb Phrase
WHADJP	Wh-adjective Phrase
WHADVP	Wh-adverb Phrase
WHNP	Wh-noun Phrase
WHPP	Wh-prepositional Phrase

The idea of derivation provides a language with its definition. If a set of rule applications can transform one string into another, then the first string derives the second. More precisely, derivation is a generalization of direct derivation, according to Hopcroft and Ullman [16].

Let $\alpha_1, \alpha_2, \ldots, \alpha_m$ be strings in $(T \cup N)^*$, $m \geq 1$, such that α_1 derives α_m:

$$\alpha_1 \Rightarrow \alpha_2, \alpha_2 \Rightarrow \alpha_3, \ldots, \alpha_{m-1} \Rightarrow \alpha_m. \tag{4.9}$$

The language L produced by a grammar G can thus be properly defined as the set of strings made up of terminal symbols that can be deduced from the prescribed start symbol S:

$$L_G = \{w \mid w \text{ is in } T^* \text{ and } S \text{ derives } w\}. \tag{4.10}$$

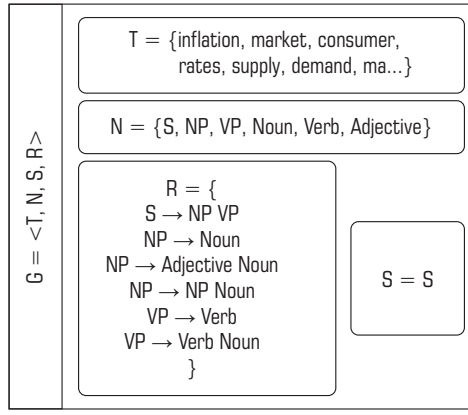

Fig. 4.2 Context-Free Grammar parsing example

A treebank is a corpus in which each sentence is tagged with a parse tree. In parsing, as well as in linguistic studies of syntactic issues, treebanks are crucial. The process of mapping a group of words to their parse tree is known as syntactic parsing.

Treebanks are generally made by parsing each sentence with a parse that is then hand-corrected by human linguists. As shown in the **Fig. 4.3**, the constituency parsing (parse tree) provides a hierarchical structure of the sentence, while the universal dependencies give a flatter representation of the syntactic relationships between words in the sentence. Both methods are used for syntactic analysis in natural language processing, with dependency parsing being more compact and favored for some tasks, while constituency parsing provides a hierarchical structure often used for deeper linguistic analysis.


```
(ROOT
 (S
  (NP (DT The) (JJ central) (NN bank))
  (VP (VBD raised)
   (NP (NN interest) (NNS rates))
   (S
    (VP (TO to)
     (VP (VB curb)
      (NP (NN inflation))))))
  (..)))
```

a

```
det(bank-3, The-1)
amod(bank-3, central-2)
nsubj(raised-4, bank-3)
root(ROOT-0, raised-4)
compound(rates-6, interest-5)
obj(raised-4, rates-6)
mark(curb-8, to-7)
advcl(raised-4, curb-8)
obj(curb-8, inflation-9)
```

b

Fig. 4.3 Parsing type: *a* – constituency parsing (Parse tree); *b* – dependency parsing

It can be helpful to have a normal form for grammars when each production has a certain form. A context-free grammar, for instance, is in Chomsky normal form (CNF) [6] if it is not null and each production also has one of the following forms: $A\ BC$ or $A \rightarrow a$.

In other words, each rule has either two non-terminal symbols on the right side or one terminal sign. Binary branching, or having binary trees (down binary branching to the prelexical nodes), is a feature of Chomsky normal form grammars. A weakly equivalent Chomsky normal form grammar can be created from any context-free grammar. Chomsky adjunction is the production of a symbol A with a possibly infinite series of symbols B with a rule of the type $A \rightarrow AB$.

The context-free nature of our grammar rules gives rise to the advantage of dynamic programming. Once a constituent has been identified in a section of the input, we can record its presence and make it available for use in any subsequent derivation that might require it. This saves time and storage because subtrees can be looked up in a table instead of being reanalyzed. The Cocke-Kasami-Younger (CKY) algorithm, which is the most popular dynamic programming-based parsing method, is presented in this section. A comparable strategy is chart parsing [13, 16], and dynamic programming techniques are often referred to as chart parsing techniques with the requirement of being in CNF.

Any context-free grammar (CFG) can be transformed into Chomsky normal form (CNF) while still preserving the language, as long as it does not generate the empty string. Some definitions of CNF allow the symbol S as an admissible rule to account for the empty string, as long as S does not appear on the right-hand side of any rule. We'll use T to represent the table used in the CKY algorithm. The table's elements, or items as we'll call them, are represented by the notation $P[i, A, j]$, where $A \in N$ and $0 \leq i \leq j \leq n$, and n is the length of the input string $w = a_1 \ldots a_n$. It is best to think of the numbers i and j as input positions: the position 0 comes before a_1, the position i with $1 \leq i \leq n-1$, w separate the symbol placement a_{i-1} and a_i in w, and the position n comes after a_n.

The substring $a_{i+1} \ldots a_j$ of w can be obtained from non-terminal A if an item $P[i, A, j]$ is added to the table. Considering that the algorithm's primary objective is to add item $P[0, S, n]$ to the table, this could be seen as a partial recognition outcome. Only when the input string is correct, will this item be discovered at the end of the process.

Fig. 4.4 shows the display of the algorithm.

The algorithm considers substrings that end in j for each input location and determines the non-terminals from which the substrings can be derived. The substrings a_j of least length are considered first, then the Chomsky normal form suggests that any parse of such a substring must contain a single instance of the rule in the form $A \rightarrow a_i$. After that, substrings $a_{i+1} \ldots a_j$ of greater lengths ($j > i+1$) are taken into consideration. The CNF suggests that if such a substring can be derived from A, then there is a rule $A \rightarrow BC$ (some B and C), where $i < k < j$, and $a_{i+1} \ldots a_k$ and $a_{k+1} \ldots a_j$ to a_k and a_{k+1} to a_j can be derived, respectively, for some choice of k.

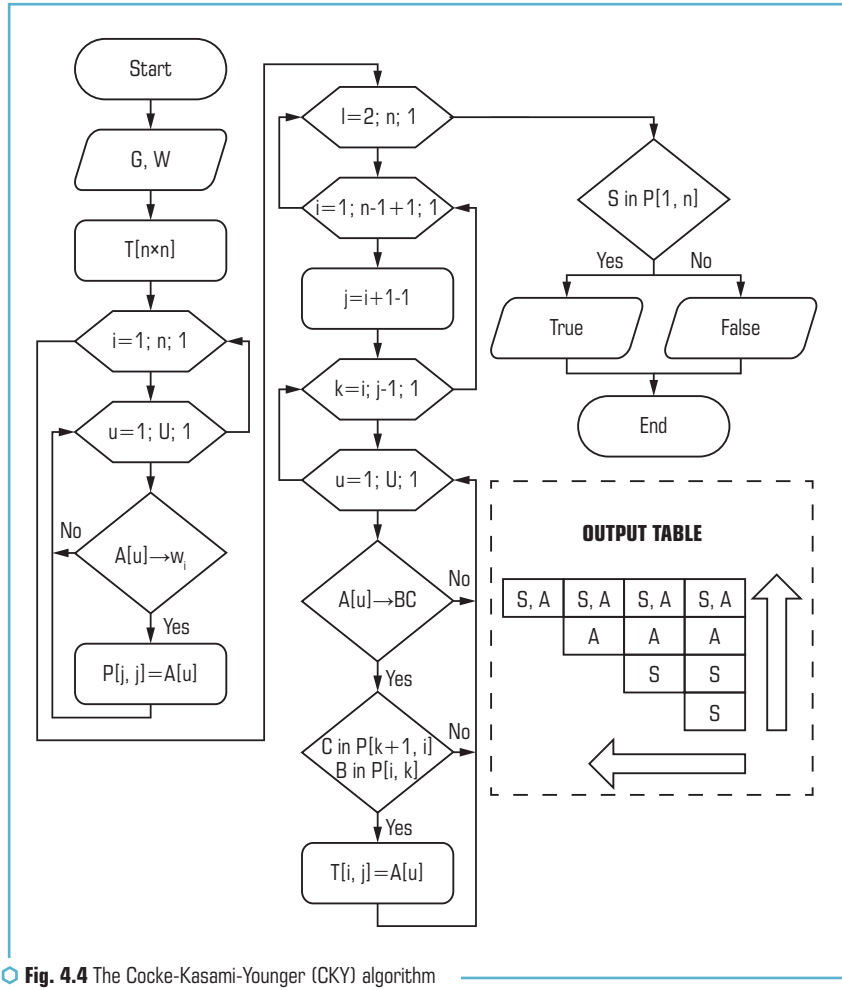

Fig. 4.4 The Cocke-Kasami-Younger (CKY) algorithm

4.3 DEVELOPING MODELS FOR SEMANTIC ROLE LABELING

The first step in any research project involving Natural Language Processing (NLP) classifiers is data collection. It is crucial to identify and gather relevant raw data sources [17]. These sources should encompass a wide range of materials, including financial news articles, annual reports, cyber incident reports, network traffic logs, and cybersecurity threat feeds.

Data collection. Financial news articles serve as a primary source of textual data for the economic aspect of the research. These articles are typically rich in content related to economic events, market dynamics, company performance, and economic indicators. To collect this data, we will consider various reputable financial news outlets and databases such as Bloomberg, Reuters, CNBC, and financial news sections of major newspapers. The selection of these sources will be based on their relevance, coverage, and the availability of structured and unstructured data. Annual reports of companies are another valuable source of economic data. These documents provide in-depth information about a company's financial performance, strategic goals, and risk assessments. The data will be collected from public sources, such as company websites, financial regulatory authorities like the U.S. Securities and Exchange Commission (SEC), and databases of publicly available annual reports. This data will be useful in understanding how companies describe their financial status and outlook, and how they frame economic information within their reports.

NLP systems require both correct and incorrect data for training and testing purposes. Obtaining correct data is relatively easy, but obtaining data with errors like typos and misspellings is challenging. Generating incorrect data can be a solution, but it is difficult to ensure that the generated texts correspond to real human mistakes. In this paper, the authors focused on collecting incorrect texts and misspellings from players through an automated web application. They used this data to build a model of common errors, which can be used to generate a large amount of authentic-looking erroneous texts [18].

The cybersecurity aspect of the research necessitates the collection of data related to cyber incidents. Cyber incident reports, often generated by Computer Emergency Response Teams (CERTs) and security organizations, contain valuable information about security breaches, attack vectors, and mitigation strategies. Sources for collecting cyber incident reports will include both government and private organizations, such as the United States Computer Emergency Readiness Team (US-CERT), the Center for Internet Security (CIS), and various industry-specific CERTs. These reports will provide insights into the language and semantics used to describe cyber threats and incidents. For a deeper understanding of the cybersecurity context, network traffic logs are an essential data source. These logs contain detailed information about network activities, including traffic patterns, protocols, and communication behaviors. Data collection in this category may involve partnerships with organizations that can provide anonymized network traffic data, or accessing publicly available datasets and network monitoring tools. These logs will help in analyzing the language and semantics used in network communications during cyber incidents. To keep up with the rapidly evolving cybersecurity landscape, cybersecurity threat feeds will be a valuable source of real-time data. These feeds provide information on emerging threats, vulnerabilities, and malicious activities. Sources for collecting threat feeds include government agencies, commercial threat intelligence providers, and open-source threat intelligence platforms. The data from these feeds will enable the analysis of the language and semantic roles used in describing emerging threats and vulnerabilities.

It is imperative to ensure that the collected data aligns with the specific research objectives of semantic role labeling and analysis [14]. The data will be chosen in such a way that it encompasses a broad spectrum of economic and cybersecurity language, including event descriptions, indicators, actors, and actions. The alignment will allow for meaningful analysis and classification of semantic roles in the data, furthering our understanding of the language used in these domains [18].

Data preprocessing. Data preprocessing is a crucial stage in NLP research, where we transform the collected raw data into a format that is suitable for analysis and classification [19]. Data cleaning is the initial step in the data preprocessing process. It involves the removal of any noise, irrelevant information, special characters, and inconsistencies present in the raw data. Financial news articles, annual reports, cyber incident reports, and other sources may contain various forms of noise, such as advertisements, metadata, HTML tags, and extraneous information not relevant to the research objectives [20]. Cleaning the data ensures that we are left with only the textual content that matters for our analysis.

Tokenization is the process of splitting the text into individual words or tokens. Tokens are the basic units of text that can be analyzed independently. By tokenizing the data, we break down the text into its constituent parts, making it amenable to further analysis. For example, the sentence "The stock market is volatile" would be tokenized into the tokens: ["the", "stock", "market", "is", "volatile"]. To ensure consistency in the data and prevent case sensitivity issues, all text is converted to lowercase [20]. Lowercasing ensures that words like "Economy" and "economy" are treated as the same word during analysis. This step helps in creating a uniform text corpus for analysis. Common words known as stopwords, such as "and", "the", "in", etc., do not carry significant meaning in the analysis and can be removed from the data to reduce noise and improve processing efficiency. However, it's important to note that in some cases, stopwords may be domain-specific and could carry relevant information. This aspect will be considered when removing stopwords, ensuring that domain-specific stopwords are retained if necessary [21].

Both economic and cybersecurity domains have their unique terminology, jargon, and linguistic patterns. It is crucial to adapt the preprocessing steps to accommodate these domain-specific characteristics. For example, in the economic context, financial terms such as "GDP", "dividends", and "NASDAQ" may need normalization, and currency symbols need to be handled consistently. In the cybersecurity domain, domain-specific jargon like "DDoS attack", "malware", and "firewall" should be treated in a manner that preserves their specific meanings. The data preprocessing pipeline involves a sequential application of the above steps. Once the raw data has been cleaned, tokenized, lowercased, and had stopwords removed, domain-specific preprocessing is applied to tailor the data for each respective context [21]. This pipeline ensures that the data is transformed into a structured, noise-free, and consistent format that can be used in subsequent stages of semantic role labeling and analysis.

Text annotation and Labeling. Text annotation is a fundamental step in NLP that involves marking specific elements of the text to identify and extract target semantic roles or entities. In the context of our research, we aim to identify and label semantic roles in both economic and cybersecurity domains. These roles will provide the foundation for further analysis and classification [22].

In the economic context, the identification of semantic roles is essential for understanding the relationships and interactions between various entities and actions. The following are some of the semantic roles that we will annotate and label in this domain:

1. Buyer: the entity or role responsible for purchasing goods or services.
2. Seller: the entity or role responsible for selling goods or services.
3. Investor: the entity or role involved in providing capital or funds for investment purposes.
4. Regulator: the entity or role responsible for enforcing rules and regulations within the economic domain.

These roles capture the key entities and actions involved in economic activities and transactions. Annotating and labeling them will enable to analyze how these roles are represented and the relationships between them in economic texts.

In the cybersecurity context, semantic role annotation is critical for understanding the dynamics of cyber incidents, threats, and vulnerabilities. The following are some of the semantic roles that we will annotate and label in this domain:

1. Attacker: the entity or role responsible for initiating a cyberattack.
2. Target: the entity or role that is the victim or the primary target of the cyberattack.
3. Exploit: the means or action used by the attacker to compromise a system or network.
4. Vulnerability: the weakness or flaw in a system or software that the attacker exploits.

These roles are central to understanding the various elements involved in cyber incidents and attacks [23]. Annotating and labeling these roles will enable to explore how these roles are depicted in cybersecurity-related texts and the relationships between them.

The annotation process involves manually identifying instances of the defined semantic roles in the preprocessed text. This process may involve using specialized annotation tools or guidelines tailored to the economic and cybersecurity domains. Annotators will mark text spans or assign labels to specific words or phrases that correspond to the roles identified. Understanding the relationships between the annotated roles is a critical aspect of this research. In both economic and cybersecurity contexts, the interactions and dependencies between roles can be complex. Analyzing these relationships will provide insights into the structure and semantics of the text data [22]. For instance, in economic texts, we may explore how buyers and sellers interact, or how regulators oversee financial transactions. In cybersecurity, we can investigate the interactions between attackers, targets, exploits, and vulnerabilities to gain a deeper understanding of cyber incidents.

Ensuring consistency in labeling is crucial to maintain the quality and reliability of the annotated data. Annotation guidelines and inter-annotator agreement assessments may be used to validate the consistency of role labeling, especially in cases where multiple annotators are involved.

Semantic Role Labeling. SRL is a key step that involves the identification and classification of semantic roles within the text. For our research, we will focus on economic and cybersecurity texts and employ appropriate SRL techniques to fulfill this objective.

Selecting the right SRL model is vital to the success of our research. Given the specific nature of our domains (economic and cybersecurity contexts), we must decide whether to use pre-trained SRL models, develop domain-specific models, or adopt a combination of both approaches:

– pre-trained models: pre-trained SRL models, such as those based on large-scale general corpora like OntoNotes or Universal Dependencies, can be a good starting point. They capture a broad range of semantic roles and syntactic structures in natural language, making them potentially useful for our analysis [24];

– domain-specific models: economic and cybersecurity texts often have unique terminologies and linguistic patterns. To address this, we may consider developing or fine-tuning domain-specific SRL models. These models can be trained on domain-specific datasets, making them more attuned to the intricacies of economic and cybersecurity texts [25].

The application of SRL techniques involves using the selected models to process the annotated data and extract semantic roles. Some common SRL methods that can be employed for this purpose include:

– dependency parsing: dependency parsing is a technique that analyzes the grammatical structure of a sentence. It identifies the relationships between words and assigns labels to these relationships. In the context of SRL, dependency parsing can be used to identify the roles that different words play in a sentence, such as identifying the subject, object, and other dependents of a verb [26];

– frame-based SRL: frame-based SRL is a method that involves identifying specific semantic frames, which are predefined structures that capture the relationships between roles and their associated arguments in a sentence. This method can be particularly useful for extracting domain-specific roles and their respective arguments in economic and cybersecurity texts [25].

Once the semantic roles have been identified, the next step is to classify them into their respective categories. In economic texts, this might involve classifying roles as "buyer", "seller", "investor", or "regulator". In cybersecurity texts, roles could be classified as "attacker", "target", "exploit", or "vulnerability". Role classification provides a structured representation of the roles in the text, making it easier to analyze and interpret the relationships between these roles.

To ensure the accuracy and quality of the SRL output, it is essential to evaluate the performance of the chosen SRL models. This evaluation can be done using standard metrics such as precision, recall, F1-score, and inter-annotator agreement. The evaluation process helps measure how well the SRL techniques are capturing the semantic roles within the text.

Model training and evaluation. After completing the Semantic Role Labeling (SRL) phase and obtaining annotated data, the focus shifts to the critical stages of model training and evaluation. Model training involves using the annotated data to train SRL models, considering choices such as pre-trained models, domain-specific models, or a combination of both [27]. The training process includes data preparation, model selection, feature engineering (for domain-specific models), training procedures, and leveraging transfer learning techniques to adapt knowledge from pre-trained models to the target domain [28].

Fine-tuning is a crucial step in the model development process. It involves making adjustments to the model's architecture and hyperparameters to ensure it performs optimally on the target task. In the context of economic and cybersecurity texts, fine-tuning may involve adapting the model to understand domain-specific terminology, syntax, and semantics [29]. This step helps the model capture nuances and nuances that are specific to the respective domains.

Model evaluation is essential to measure the performance of the trained and fine-tuned SRL models. It ensures that the models effectively extract semantic roles from economic and cybersecurity texts. The following metrics are commonly used for SRL model evaluation:

– F1 score: the F1 score is a balance between precision and recall and is particularly useful when there is an imbalance between classes. It provides a single score that summarizes the model's performance;

– precision: precision measures the ratio of correctly predicted positive instances to the total instances predicted as positive. In SRL, precision indicates how accurately the model labels semantic roles;

– recall: recall measures the ratio of correctly predicted positive instances to the actual positive instances. In SRL, recall indicates the model's ability to capture all instances of the semantic roles in the text;

– accuracy: accuracy measures the overall correctness of the model's predictions. It is the ratio of correctly predicted instances to the total instances.

To ensure consistent performance and generalizability, cross-validation techniques can be employed, involving the splitting of data into multiple subsets for robust model evaluation. Fine-tuning processes may also optimize hyperparameters, including learning rates, batch sizes, and regularization parameters, aiming to find the optimal configuration for the best model performance. These approaches collectively contribute to the development of effective SRL models tailored to the specific demands of economic and cybersecurity texts.

Feature engineering. Feature engineering involves the extraction of relevant features from the data, which can provide valuable linguistic information to improve the accuracy and effectiveness of the SRL models.

Linguistic features are fundamental to understanding and representing the structure and meaning of text. These features can include:

– part-of-speech (POS) tags: POS tags are labels assigned to each word in a sentence, indicating the word's grammatical category (e.g., noun, verb, adjective). Incorporating POS tags into the feature set can assist SRL models in understanding the syntactic roles of words in a sentence, which is crucial for identifying semantic roles [30];

– syntactic dependencies: syntactic features capture the grammatical relationships between words in a sentence. These dependencies help in understanding how words are connected in terms of subject-verb-object relationships, modifiers, and more. Syntactic dependency trees and labels can be used as features to provide additional context for semantic role identification [31];

– n-grams: n-grams are contiguous sequences of n words from a text. By including n-grams as features, we can capture the local context around a word or phrase, which is particularly useful in disambiguating word senses and identifying semantic roles [30];

– word embeddings: word embeddings, such as Word2Vec or GloVe, represent words as continuous vector spaces, where words with similar meanings are located close to each other. These embeddings can be used as features to provide semantic information, enabling the model to understand word similarity and context [32].

Actionable model creation. The creation of actionable models represents the culmination of the research efforts, where the semantic roles identified in economic and cybersecurity texts are translated into practical applications. These models have the potential to drive informed decisions, enhance security, and provide valuable insights in these domains. In the following chapters, we will explore the practical implementation and real-world impact of these actionable models.

In the economic context, actionable models can be created to support various applications, including:

– investment decisions: semantic roles like "buyer", "seller", and "investor" can be leveraged to inform investment decisions. By tracking the activities and sentiments associated with these roles in financial news articles, the model can provide insights into market trends, potential investment opportunities, or risks;

– sentiment analysis: analyzing the roles associated with sentiment-laden words and phrases in economic texts can enable sentiment analysis. This can help in understanding market sentiment, investor confidence, and public perception, which can be valuable for financial professionals and decision-makers [20];

– sentiment analysis: analyzing the roles associated with Predictive models can be developed by correlating semantic roles with financial indicators. For instance, identifying patterns in how buyers and sellers are described in news articles and their impact on stock prices can aid in market predictions;

– regulatory compliance: models can be created to identify regulatory violations or non-compliance by tracking the actions of regulators and the entities they oversee in annual reports and financial documents.

In the cybersecurity context, actionable models can be developed for a range of applications, such as:

– threat detection: by analyzing semantic roles like "attacker" and "exploit" in network logs and cybersecurity reports, actionable models can be built to detect and respond to security threats in real time;

– incident response: models can be used to identify the roles involved in cyber incidents, helping incident response teams understand the nature of the attack, its impact, and potential countermeasures;

– threat intelligence: by analyzing semantic roles, such as "vulnerability" and "target", actionable models can provide insights into emerging threats, vulnerabilities, and potential targets, assisting in proactive threat intelligence;

– security risk assessment: actionable models can assess security risks by monitoring the roles and actions associated with different entities in a network. This can help organizations identify weak points and vulnerabilities.

The development of actionable models has the potential to provide significant benefits in both economic and cybersecurity contexts. These models can enhance decision-making, improve security, and enable proactive responses to changing conditions and threats.

Real world testing. Real-world testing is a crucial phase, as it allows us to assess how well the models work in practical, dynamic settings.

Real-world testing involves applying the actionable models to different scenarios and assessing their performance and usability [33]. These scenarios may include:

– economic analysis: testing the model's ability to provide investment recommendations or sentiment analysis in real-time as new financial data becomes available;

– cybersecurity defense: assessing the model's effectiveness in real-time threat detection and incident response within a live network environment;

– predictive capabilities: evaluating the model's predictive capabilities by monitoring its performance over an extended period to gauge its long-term accuracy;

– usability in practical tasks: testing the model's utility in practical cybersecurity tasks, such as security risk assessment, threat intelligence, and regulatory compliance monitoring.

Assessing model performance involves monitoring how well the actionable model performs in real-time. Key metrics and indicators should be observed, such as:

– accuracy: how well the model's predictions match real-world outcomes;

– precision and recall: the model's ability to correctly identify and act upon semantic roles in dynamic situations;

– latency: the time it takes for the model to process data and provide actionable insights.

The real-world testing phase provides an opportunity to evaluate the model's adaptability and scalability. Can it accommodate new data sources, handle unexpected scenarios, and scale to meet the demands of practical applications? These aspects are critical for assessing the model's long-term usability. Real-world testing often uncovers areas for improvement. Gathering feedback from users and stakeholders involved in the practical application of the model is invaluable. This feedback can guide iterative refinements and updates to enhance the model's performance and utility.

Usability testing focuses on how well the actionable model fits into the workflow of users. Is it user-friendly, easy to integrate, and does it meet the specific needs of the users in economic or cybersecurity roles? Real-world testing is the ultimate litmus test for the actionable models developed in our research. It's where the rubber meets the road, and the effectiveness of these models in practical economic and cybersecurity scenarios is demonstrated. The insights gained from this phase will provide valuable feedback for further refinement and real-world deployment, ultimately contributing to the real-world application of semantic role labeling and analysis. In the final chapters, we will explore the broader implications and contributions of our research in these critical domains.

Iterative refinement. While we have developed actionable models that are ready for real-world testing and deployment, it is important to focus on the importance of continuous refinement and adaptation. In dynamic domains, staying current and responsive to changes in linguistic patterns and domain-specific developments is essential.

The worlds of economics and cybersecurity are in a state of constant evolution. New terminologies, trends, and linguistic patterns emerge regularly. As a result, actionable models must be continuously refined to stay relevant and effective [20]. The need for iterative refinement is driven by several factors:

– dynamic language use: language use evolves over time, and new terms and phrases enter the lexicon. Iterative refinement allows models to adapt to these linguistic changes and remain accurate in their predictions [34];

– understanding machine learning: from theory to algorithms [30];

– changing threat landscape: in cybersecurity, the threat landscape is constantly changing as attackers develop new tactics and exploit novel vulnerabilities. Models must be updated to detect these emerging threats effectively;

– market dynamics: economic markets are influenced by various factors, including geopolitical events, economic policies, and global trends. Refinement is necessary to capture how these changes affect economic roles and actions;

– user feedback: feedback from users of the actionable models provides valuable insights into areas where improvement is needed. Iterative refinement is a response to this feedback.

Iterative refinement also involves adapting to domain-specific developments in economics and cybersecurity. Some ways to achieve this include:

– regular data updates: regularly updating the training data with the latest economic reports, financial news, cybersecurity incidents, and threat reports to ensure that the model reflects the current state of the domain;

– domain expertise: involving domain experts who can provide insights into the evolving landscape and guide model adjustments to account for new trends and terminologies;

– model re-training: re-training the model using up-to-date data and fine-tuning it to account for changes in the domain;

– incorporating external knowledge sources: integrating external sources of domain knowledge, such as industry reports or expert opinions, to inform model adjustments.

User feedback is a central component of iterative refinement [34]. Ensuring that the actionable models align with the needs and expectations of users is paramount. This may involve:

– user surveys and interviews: conducting surveys or interviews with users to gather their input on model performance, usability, and areas for improvement;

– user training: offering training sessions to users to familiarize them with the model and gather insights into how it can better fit into their workflow;

– user-driven feature requests: encouraging users to suggest new features or modifications that would enhance the model's usability [34].

Iterative refinement is an ongoing process, and continuous evaluation is crucial to assess the effectiveness of model updates. This evaluation should include the monitoring of key metrics, usability testing, and the measurement of model performance in real-world scenarios.

Deployment and practical use. The ultimate goal is to ensure that the model is capable of providing valuable insights and supports decision-making processes in these domains.

The deployment of actionable models in real-world contexts involves several considerations, including:

– infrastructure and scalability: ensuring that the necessary computational infrastructure is in place to support the model's real-time processing needs and scalability to handle increasing data volumes;

– data integration: integrating the model with data sources, ensuring that it can access relevant economic and cybersecurity texts or data feeds;

– user training: providing training and onboarding for users who will interact with the model to ensure they understand how to utilize it effectively;

– security and compliance: ensuring that the deployment adheres to security standards and regulatory compliance, especially in the cybersecurity domain where sensitive information is involved.

For practical use, the model should be designed to facilitate user interaction and interpretation of results. This may include creating user-friendly interfaces, dashboards, or reports that allow users to make informed decisions based on the insights provided by the model. Deployment is not the endpoint but rather the beginning of an ongoing cycle of monitoring and continuous improvement. Regularly evaluating the model's performance, gathering user feedback, and making necessary refinements is essential to ensure that the actionable model remains effective in the long term.

4.4 METHODOLOGICAL FOUNDATIONS FOR DEVELOPING A SEMANTIC ROLE CLASSIFIER FOR CYBER THREAT ANALYSIS AND ECONOMICAL SPHERE USING ARTIFICIAL NEURAL NETWORKS

ANNs have been applied in the study of language in various ways. The multi-layered perceptron (MLP) is the most practical ANN architecture for statistical modeling. MLPs have been expanded to represent both sequential and structured data and can be utilized for feature induction and probability estimation. Language modeling and parsing have been their most effective uses in NLP. MLPs can be reinterpreted as approximate versions of latent variable models, and they have served as an inspiration for much of the recent research in machine learning techniques.

Artificial neural networks, or simply "neural networks", are a general term for a group of computational models that have some characteristics in common with the networks of neurons present in the brain. They are often built to be educated using data and comprise of a distributed network of simple processing units. These were some of the earliest machine learning techniques in the field of artificial intelligence (AI), and they have had a significant impact on various areas of machine learning research. The majority of ANN research within AI no longer has any neurological underpinnings and is now mostly driven by engineering concerns [34]. Research in NLP has been primarily driven by its applicability for engineering solutions.

Unsupervised representation induction during learning is another characteristic that is frequently associated with ANNs. Some of the artificial neural networks (ANN) processing units do not have predefined meanings; instead, they develop them during training. In other instances, such as the unsupervised clustering of self-organizing maps, these units serve as the output of the

artificial neural network (ANN). In other instances, these units serve as an intermediary representation between the input and output of the ANN. These units are known as "hidden units"; they are comparable to latent variables. The multilayer perceptron (MLP) and its recurrent variations have been the most frequently used types of artificial neural networks (ANNs). MLPs are used for sequence modeling, categorization, and function approximation.

Another trait that is usually linked to ANNs is unsupervised representation induction during learning. Some of the processing units of an artificial neural network (ANN) gain their meanings during training rather than having them predefined. These units are used as the artificial neural network (ANN) output in other situations, such as the unsupervised clustering of self-organizing maps [24]. In other cases, these units act as a representational bridge between the ANN's input and output. They are referred to as "hidden units" and are similar to latent variables. The most popular varieties of artificial neural networks (ANNs) have been the multilayer perceptron (MLP) and its recurrent versions. Sequence modeling, categorization, and function approximation are all performed using MLPs.

In response to the argument that the perceptron algorithm could only learn a very small class of problems, multi-layered perceptrons (MLPs) were created as it is possible to see in **Fig. 4.5**. The perceptron algorithm learns to distinguish between output classes based on a linear combination of its input features. Because of its linearity, a perceptron can only solve problems that can be divided into classes of outputs by a line (or, more generally, a hyperplane) that can be drawn in the input space. The XOR function is a clear example of a problem that cannot be linearly separated because there is no line that can divide the zero cases (0, 0, 1, 1) from the one cases (0, 1, 1, 0) [35].

The input layer consists of neurons (also called nodes) that represent the features or input data for your problem. Each neuron corresponds to a specific feature, and these neurons pass their values to the neurons in the first hidden layer. An MLP can have one or more hidden layers, each consisting of multiple neurons. The number of hidden layers and the number of neurons in each layer are hyperparameters that can be adjusted based on the complexity of the problem. Each connection between two neurons (either between the input and hidden layers or between hidden layers) has an associated weight. These weights are the parameters that the network learns during training. The weights determine the strength of the connection and play a crucial role in shaping the network's behavior. Activation functions are applied to the weighted sum of inputs at each neuron to introduce non-linearity into the network. Each neuron in the hidden and output layers typically has an associated bias term. The bias allows the network to model shifts in the data that are not accounted for by the weights and is added to the weighted sum of inputs before the activation function is applied.

The middle layers of MLPs contain processing units whose outputs are a continuous non-linear function of their inputs. This addresses the restriction by incorporating multiple layers of units. An MLP can transform the input space into a new set of features, where the output classes become linearly separable due to the presence of these middle layers, also referred to as hidden layers. In reality, MLPs can approximate any arbitrary function due to the non-linearity of

the hidden units [34]. Backpropagation is a straightforward learning procedure for MLPs because the hidden unit functions are continuous [34].

The output of each neuron in a hidden layer is determined by the weighted sum of its inputs, including the bias term, passed through the activation function. This output is then used as input to neurons in subsequent layers. The output layer consists of neurons that provide the final output of the network. The number of neurons in this layer depends on the specific problem you are trying to solve. For classification tasks, you might have one neuron per class for softmax classification, while for regression tasks, there might be a single output neuron. The loss function measures the difference between the network's predictions and the true target values. The choice of the loss function depends on the type of problem, e.g., mean squared error for regression or cross-entropy for classification.

During training, the network adjusts its weights and biases to minimize the loss function. This is typically done using optimization algorithms like stochastic gradient descent (SGD) or its variants. Backpropagation is a key technique for computing gradients and updating the weights. To prevent overfitting, techniques like dropout, weight decay (L1 or L2 regularization), and early stopping can be employed to make the network generalize better to unseen data. An MLP with multiple hidden layers and appropriate activation functions can approximate complex functions and is widely used in various machine learning tasks. The design of the network, including the number of layers, neurons per layer, activation functions, and other hyperparameters, is highly dependent on the specific problem being addressed.

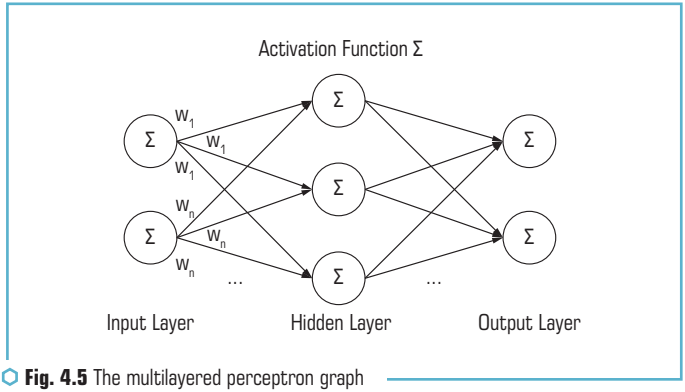

Fig. 4.5 The multilayered perceptron graph

The pursuit of understanding and representing words in a manner that accurately captures their semantic and syntactic connections has been a fundamental focus of research in Natural Language Processing (NLP). We came across word vectors during our journey, and they have revolutionized NLP applications. To grasp the fundamental principles of word embeddings, it is essential to comprehend the connection between word vectors, artificial neural networks, and the multi-layered perceptron (MLP).

Words in a continuous vector space are represented numerically as word vectors, also referred to as word embeddings. These embeddings provide syntactic and semantic connections between words, enabling natural language processing (NLP) systems to comprehend meaning and context. These compact numerical representations of words have opened new opportunities for tasks such as sentiment analysis, machine translation, and document classification. At the heart of this revolution lies the interaction between artificial neural networks and multi-layered perceptrons, which collaborate to capture word semantics and predict their context.

Word2Vec and FastText are two well-known models that have dominated the development of word vectors (**Fig. 4.6**). Both of these models train and generate word embeddings using artificial neural networks, specifically the multi-layer perceptron. The Word2Vec model for word embeddings was first presented by Mikolov. The Continuous Bag of Words (CBOW) and Skip-gram architectures are the two main architectures used by Word2Vec. Continuous Bag of Words (CBOW) predicts the target word by using the words in its context. The goal is to increase the likelihood of the target word given the context using a shallow neural network with one hidden layer. On the other hand, skip-gram guesses context words from a target word. Similar to CBOW, but with the aim reversed. In both situations, the word vectors are adjusted to capture linguistic correlations after the training process has fine-tuned the neural network weights [36].

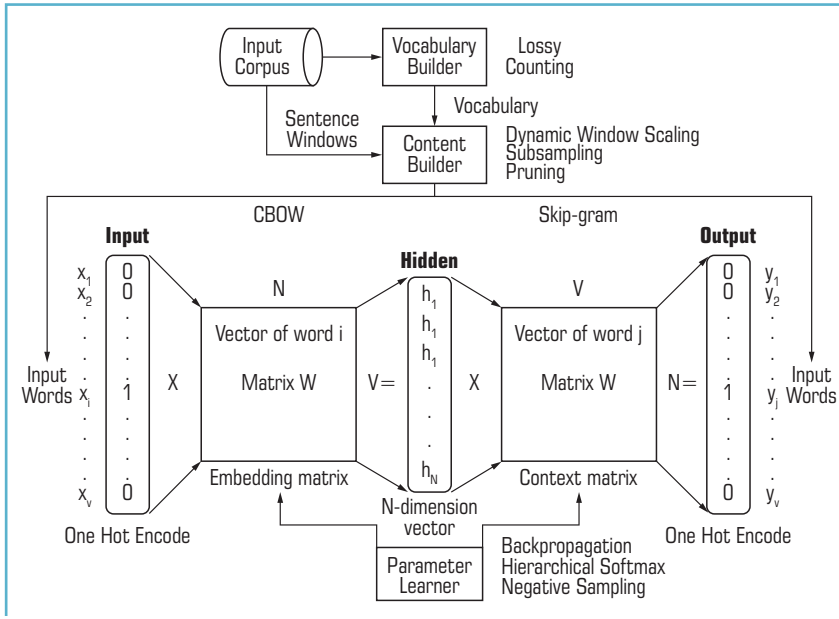

Fig. 4.6 The architecture of Word2Vec model

With a predefined vocabulary and a substantial text dataset as an initial input, Word2Vec can commence text processing. Although the vocabulary is typically broad, it can be condensed to include only common words. As depicted in **Fig. 4.7**, each word in the dictionary is assigned a vector representation using Word2Vec. Using a distributional similarity objective, Word2Vec aims to learn word vectors from a text corpus. Predicting which words will appear in the context of other words is necessary for this challenge. Operationally, Word2Vec utilizes the concepts of context words (*O*) and center words (*C*). While context words are the terms that appear around the center word in the text, center words are the words that are being considered.

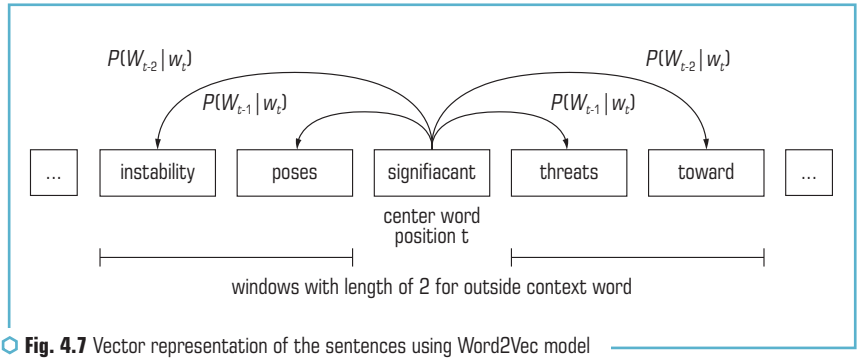

Fig. 4.7 Vector representation of the sentences using Word2Vec model

With a predefined Word2Vec uses the current word vectors to calculate the probability of a context word occurring, given the center word, based on the model. The goal is to adjust the word vectors to maximize the probability assigned to words that actually occur in the context of the center word. This adjustment is performed iteratively as the model processes the text. The central component of Word2Vec is the objective function, which is a cost or loss function that requires optimization. The objective function aims to maximize the likelihood of the context words surrounding the center words. and a substantial text dataset as an initial input, Word2Vec can commence text processing. Although the vocabulary is typically broad, it can be condensed to include only common words. Each word in the dictionary is assigned a vector representation using Word2Vec. Using a distributional similarity objective, Word2Vec aims to learn word vectors from a text corpus. Predicting which words will appear in the context of other words is necessary for this challenge. Operationally, Word2Vec utilizes the concepts of context words (*O*) and center words (*C*). While context words are the terms that appear around the center word in the text, center words are the words that are being considered [37].

The likelihood is formally defined as the product of the probabilities of predicting context words for each center word. However, for the sake of computational simplicity, Word2Vec converts products into sums and utilizes log likelihood instead. Word2Vec works with the average log likelihood. The sum of log likelihoods is divided by the number of words in the corpus. Word2Vec employs

a minimization objective function (denoted as $J(\theta)$) rather than maximizing it. By minimizing this objective function, the model aims to maximize its predictive accuracy:

$$L(\theta)\prod_{t=1}^{T}\prod_{\substack{-m\le j\le m\\ j\neq 0}} P\left(w_{t+1}\mid w_t;\theta\right); \tag{4.11}$$

$$J(\theta)=-\frac{1}{T}\log L(\theta)=-\frac{1}{T}\sum_{t=1}^{T}\sum_{\substack{-m\le j\le m\\ j\neq 0}}\log P\left(w_{t+1}\mid w_t;\theta\right). \tag{4.12}$$

Each word in the lexicon has a vector representation thanks to Word2Vec. The context and meaning of words are captured by these vectors.

Each word is associated with two-word vectors: one for its use as the center word and another for its use as a context word. The mathematical calculations and optimization procedure are made easier by this method. Although it could appear a little strange, it is a sensible decision that will help with word vector creation.

Word2Vec uses a specific equation to determine the likelihood of a context word arising given the center word. The description refers to "the expression in the middle bottom of my slide", indicating that the precise equation is shown visually rather than in the text [37]. However, the exact equation is not provided in the text:

$$P\left(p\mid c\right)=\frac{\exp\left(u_0^T v_c\right)}{\sum_{w\in v}\exp\left(u_w^T v_c\right)}. \tag{4.13}$$

By adding the vector representations of the two words, $U(O)$ and $V(C)$, the probability of a context word (O) occurring given a center word (C) is calculated. The dot product compares the similarity between the vectors representing the letters "o" and "c". A larger dot product indicates a higher degree of similarity and a greater likelihood. Exponentiation transforms everything into positive values, while probability distribution is created by normalizing the entire lexicon through summation. Similarity is indicated by positive values, while dissimilarity is indicated by negative numbers.

The softmax function is used to transform these dot product values into a probability distribution. The dot product is transformed by the softmax function, which guarantees that all values are positive. The outcome is then normalized to create a probability distribution. More related terms are given higher odds in this distribution.

The objective is to minimize the average negative logarithm probability of the model's predictions, or the objective function (J of theta). Optimizing the model's parameters, which in the case of Word2Vec are the word vectors, is necessary to minimize this function. Two vectors – the context vector and the center vector – are included in the parameters for each word:

$$R^n\rightarrow(0,1)^n,\ soft\ \max\left(x_i\right)=\frac{\exp\left(x_i\right)}{\sum_{j=1}^{n}\exp\left(x_i\right)}=p_i. \tag{4.14}$$

Calculus is used to modify the word vectors and reduce the loss function. The gradients, which indicate the direction of the steepest descent, are computed by the model. The word vectors are iteratively updated using these gradients in gradient descent algorithms to improve their ability to anticipate context words. The text explains how to utilize the chain rule to unravel intricate expressions and simplifies the process of calculating derivatives by dividing it into multiple steps.

An expectation, which is an average across all the context vectors weighted by their probabilities, is the result of the derivative calculation. The derivative in softmax-style models is calculated by subtracting the expected value from the observed value. If the model predicts word vectors that are similar to the observed context words, it is considered effective:

$$\theta^{new} = \theta^{old} - \alpha\Delta_{\theta}J(\theta). \tag{4.15}$$

There are various approaches to visualizing word vectors. One common approach is to utilize dimensionality reduction techniques, such as t-SNE, to project the high-dimensional word vectors into a lower-dimensional space that is more easily visualized [38]. This enables the exploration of clusters and patterns within the word vector space. Another approach involves visualizing word vectors in a semantic space, where words with similar meanings are positioned close to each other. This can be achieved by plotting the word vectors on a semantic map or by employing interactive visualization tools [38]. These visualizations aid researchers and practitioners in understanding the semantic relationships between words and exploring the structure of the word vector space. In addition to visualizing word vectors, there are methods available for enriching word vectors with subword information. By considering subword units, such as character n-grams, in addition to whole words, the word vectors can capture more detailed information about word morphology and compositionality. This is particularly useful for languages with rich morphology or for tasks that require an understanding of word formation and derivation. Furthermore, word vectors can be adapted and fine-tuned using various techniques. For instance, one study proposed adapting word vectors using a tree structure to incorporate visual semantics [39]. By combining visual features with word vectors, the resulting representations can capture both the visual and semantic aspects of words. This can be advantageous for tasks such as image captioning or visual question answering.

Tools and Software. For this research, we utilized the GloVe (Global Vectors for Word Representation) pre-trained Word2Vec model. The selected dataset, glove.6B.100d.word2vec, has a vector dimensionality of 100 and includes a large vocabulary size. To process and analyze the data, we utilized well-known Natural Language Processing (NLP) libraries. These included NLTK for tokenization and stemming, Gensim for integrating the Word2Vec model, and scikit-learn for constructing and training the classifiers. The machine learning framework used in this study was scikit-learn, which is a robust and versatile library for machine learning tasks. Deep learning techniques were implemented using the Keras library.

Hardware. Our experiments were conducted on a machine with an Intel Core i7 processor, 16 GB of RAM, and SSD storage. As well as on a high-performance computing cluster equipped with multiple CPUs, a dedicated GPU, ample RAM, and ample storage. This infrastructure facilitated the efficient processing of the large-scale dataset and the training of complex neural network architectures.

Data. The dataset used in this research comprises economic and cybersecurity text data collected from reputable sources, such as financial reports, research papers, and cybersecurity incident reports. The dataset is extensive, containing a diverse range of texts, which enables a comprehensive analysis of semantic roles in various contexts.

Preprocessing. Prior to analysis, the data underwent rigorous preprocessing steps. This involved tokenization, which breaks down the text into individual words, stemming to reduce words to their root forms, and domain-specific transformations to enhance the relevance of the data in economic and cybersecurity contexts.

Feature extraction. Word2Vec embeddings were used to convert words into numerical vectors. The Word2Vec model used a window size of 5 and employed negative sampling to generate precise word representations. These embeddings served as crucial features for subsequent semantic role labeling and analysis tasks.

Neural Network Architecture. Word2vec uses neural networks for training. TWord2vec uses neural networks for training. The following layers are presented:

– as many neurons as there are words in the training vocabulary make up one input layer;

– the dimensionality of the generated word vectors determines the size of the hidden layer, which is the second layer, in terms of neurons;

– the output layer, which has the same number of neurons as the input layer, is the third and final layer.

One-Hot Encoding is the simplest method of converting words into vectors. The size of this vector is determined by the number of words in the vocabulary, and each word has its own vector. The word representing itself is encoded at position 1, and all additional locations are encoded at position 0. The input layer receives the one-hot encoding of the center word. We train our neural network by randomly initializing the weights. Here, the neural network updates its weights using the backpropagation approach. The central word receives our input and produces a specific outcome. The softmax classifier is processing the output. Because the softmax classifier can convert the output into a probability, it is utilized. This vector indicates which terms in the lexicon are most likely to be associated with the input word.

Every time a relevant word's one-hot encoding is inputted, the weights between the hidden and output layers and the input layer are adjusted to ensure that the output corresponds to the paired word. The weights are adjusted based on the calculated difference between the current output and the predicted input. The vectors we need to locate are the weights between the input layer and the hidden layer. The fact that each of these words has a context and association makes them significant. Words with similar vectors or closer spacing are used in the same context.

Input parameters. The input parameters for the neural network included the Word2Vec embeddings that represented the words in the text data. These embeddings, combined with additional domain-specific features, provide comprehensive input for the semantic role labeling classifier, ensuring a nuanced analysis of the text.

Training and Evaluation. The LSTM network was trained using the Adam optimizer with a suitable learning rate. The dataset was divided into training, validation, and test sets to facilitate model

training and evaluation. Performance was assessed using evaluation metrics such as F1-score, accuracy, precision, and recall, providing a comprehensive view of the classifier's effectiveness in capturing semantic roles in economic and cybersecurity texts.

Limitations and Assumptions. Several limitations were identified during the research, including potential biases in the data sources and the inherent limitations of the Word2Vec model in capturing highly nuanced semantic relationships. Additionally, assumptions were made regarding the relevance and accuracy of the selected features, recognizing the necessity for additional research to improve the model and overcome these limitations.

Output Data. The analysis yielded valuable insights into the semantic roles present in economic and cybersecurity texts. The output data includes detailed semantic role annotations, which highlight key relationships within the texts. Visualizations and graphs were generated to present the findings, providing a clear and concise representation of the semantic structures in the analyzed texts (**Fig. 4.8**). Additionally, the research provided valuable implications for economic analysts and cybersecurity experts, enabling them to enhance their understanding of textual data in their respective fields.

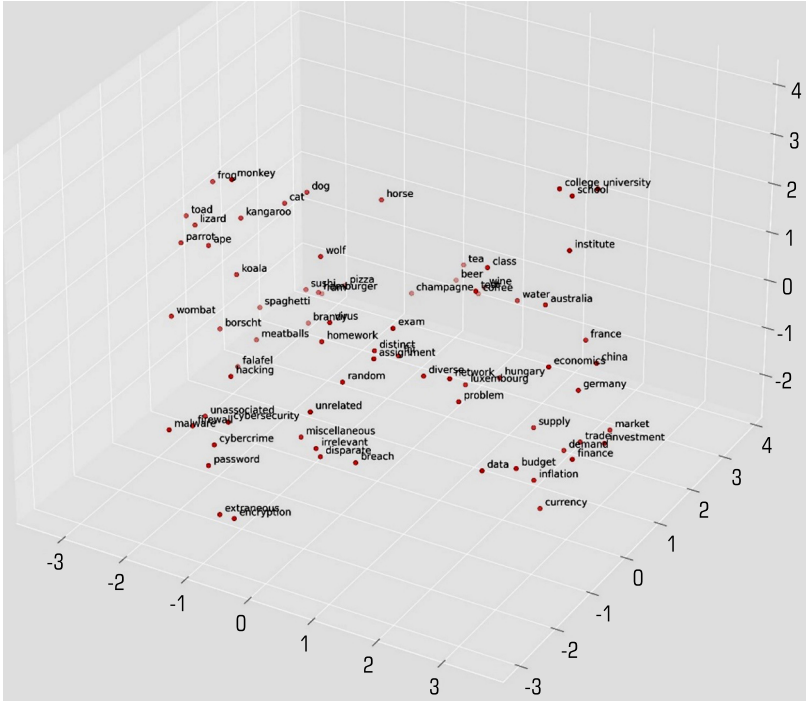

Fig. 4.8 Visualization of economical, cybersecurity and other non-connected words clustering

The three-dimensional PCA (Principal Component Analysis) visualization of words from diverse categories, including economic, cybersecurity, and non-connected words, provides valuable insights into the semantic relationships and similarities among these words. Through clustering, words within the same domain tend to group together, indicating their semantic proximity. Economic terms, such as "economics", "market", and "investment", form a distinct cluster, while cybersecurity-related words like "cybersecurity", "hacking", and "firewall" create another cohesive cluster. Non-connected words appear scattered, highlighting their lack of semantic correlation.

The clustered words can be leveraged for textual analysis tasks. Understanding the semantic context of words within specific domains can aid in sentiment analysis, topic modeling, and document classification, especially in economic and cybersecurity-related texts. Semantic clustering can enhance content recommendation systems. By identifying related terms, businesses can recommend relevant articles, research papers, or products to users based on their interests in economics or cybersecurity. The identified semantic relationships can contribute to constructing knowledge graphs, connecting economic concepts, cybersecurity terms, and other domains. This interconnected knowledge can be valuable for educational purposes or data-driven decision-making.

CONCLUSIONS

Semantic role labeling (SRL) plays a critical role in extracting important information from text, particularly in the fields of cybersecurity and economics. When it comes to interpreting financial reports, news stories, and economic literature, SRL is crucial. It assists in identifying important actors, events, and objects, which ultimately enhances decision-making and market analysis. This emphasizes the usefulness of deriving meaningful insights from textual data. In the field of cybersecurity, SRL (Semantic Role Labeling) is essential for understanding and processing text data that is related to security. It automates the analysis of large volumes of text, enabling faster responses to security concerns. In order to organize unstructured text data, evaluate risks, and make informed decisions in response to evolving security issues, NLP classifiers and machine learning models utilize SRL (Semantic Role Labeling).

The visualization provides a snapshot of word relationships but lacks contextual depth. Understanding the nuances of word meanings within sentences and paragraphs is crucial for conducting more precise semantic analysis. The quality and bias of the underlying data used to train the Word2Vec model can impact the clustering results. Biased data may result in distorted word representations, which can impact the accuracy of semantic relationships. Future research can explore advanced word embedding models, such as contextual embeddings (e.g., BERT), that capture word meanings based on surrounding context. These models provide more nuanced semantic representations, which can enhance the accuracy of clustering. Integrating textual data with other modalities, such as images, audio, or video, can enhance semantic analysis. Multimodal approaches enable a more comprehensive understanding of concepts, particularly in domains where visual or auditory cues play a significant role.

Overall, the findings suggest that self-regulated learning (SRL) can play a vital role in economic analysis and cybersecurity risk management. By accurately identifying and classifying semantic roles, NLP classifiers can facilitate the extraction of valuable information from textual data, thereby enabling more informed decision-making processes in economic contexts. In the cybersecurity domain, Security Risk Management (SRL) can aid in the detection and prevention of cyber threats, thereby enhancing the overall security posture of organizations and critical infrastructure. Future research can explore advanced word embedding models, such as contextual embeddings (e.g., BERT), which capture word meanings based on surrounding context. These models provide more nuanced semantic representations and can improve the accuracy of clustering.

However, further research is needed to address the challenges associated with SRL, such as the scarcity of annotated data for specific economic and cybersecurity domains. Additionally, the development of more robust natural language processing (NLP) classifiers and the integration of semantic role labeling (SRL) with other NLP techniques hold promise for advancing the application of SRL in these contexts. It highlights the potential of SRL in improving economic analysis and enhancing cybersecurity measures. by utilizing NLP techniques to extract valuable insights from textual data and mitigate risks in these domains. Future research should focus on addressing the challenges and further advancing the application of self-regulated learning (SRL) in economic and cybersecurity contexts.

CONFLICT OF INTEREST

The authors declare that they have no conflict of interest in relation to this research, whether financial, personal, authorship or otherwise, that could affect the research and its results presented in this paper.

REFERENCES

1. Täckström, O., Ganchev, K., Das, D. (2015). Efficient Inference and Structured Learning for Semantic Role Labeling. Transactions of the Association for Computational Linguistics, 3, 29–41. doi: https://doi.org/10.1162/tacl_a_00120
2. Màrquez, L., Carreras, X., Litkowski, K. C., Stevenson, S. (2008). Semantic Role Labeling: An Introduction to the Special Issue. Computational Linguistics, 34 (2), 145–159. doi: https://doi.org/10.1162/coli.2008.34.2.145
3. Lang, J., Lapata, M. (2014). Similarity-Driven Semantic Role Induction via Graph Partitioning. Computational Linguistics, 40 (3), 633–669. doi: https://doi.org/10.1162/coli_a_00195
4. Lample, G., Ballesteros, M., Subramanian, S., Kawakami, K., Dyer, C. (2016). Neural Architectures for Named Entity Recognition. Proceedings of the 2016 Conference of the North

American Chapter of the Association for Computational Linguistics: Human Language Technologies. doi: https://doi.org/10.18653/v1/n16-1030
5. Manning, C., Schutze, H. (1999). Foundations of statistical natural language processing. MIT press.
6. Chomsky, N. (1964). Aspects of the theory of syntax. doi: https://doi.org/10.21236/ad0616323
7. Laukaitis, A., Ostašius, E., Plikynas, D. (2021). Deep Semantic Parsing with Upper Ontologies. Applied Sciences, 11 (20), 9423. doi: https://doi.org/10.3390/app11209423
8. Cherifi, H. (2019). The Essential Contributions of Corpora in Language Research. NETSOL: New Trends in Social and Liberal Sciences, 4 (2), 62–73. doi: https://doi.org/10.24819/netsol2019.08
9. Dupre, G. (2021). Empiricism, syntax, and ontogeny. Philosophical Psychology, 34 (7), 1011–1046. doi: https://doi.org/10.1080/09515089.2021.1937591
10. Bolt, J., Coecke, B., Genovese, F., Lewis, M., Marsden, D., Piedeleu, R. (2019). Interacting conceptual spaces i: grammatical composition of concepts. Conceptual Spaces: Elaborations and Applications, 151–181. doi: https://doi.org/10.1007/978-3-030-12800-5_9
11. Hare, A., Chen, Y., Liu, Y., Liu, Z., Brinton, C. G. (2020). On Extending NLP Techniques from the Categorical to the Latent Space: KL Divergence, Zipf's Law, and Similarity Search. doi: https://doi.org/10.48550/arxiv.2012.01941
12. Jurafsky, D., Martin, J. H. (2018). Speech and language processing. Prentice Hall.
13. Clark, A., Lappin, S.; Clark, A., Fox, C., Lappin, S. (Eds.) (2010). Unsupervised Learning and Grammar Induction. The Handbook of Computational Linguistics and Natural Language Processing, 197–220. doi: https://doi.org/10.1002/9781444324044.ch8
14. Corral, Á., Boleda, G., Ferrer-i-Cancho, R. (2015). Zipf's Law for Word Frequencies: Word Forms versus Lemmas in Long Texts. PLOS ONE, 10 (7), e0129031. doi: https://doi.org/10.1371/journal.pone.0129031
15. Goldsmith, J., van Riemsdijk, H., Williams, E. (1989). Introduction to the Theory of Grammar. Language, 65 (1), 150. doi: https://doi.org/10.2307/414851
16. Hopcroft, J. E., Motwani, R., Ullman, J. D. (2001). Introduction to automata theory, languages, and computation, 2nd edition. ACM SIGACT News, 32 (1), 60–65. doi: https://doi.org/10.1145/568438.568455
17. Zeroual, I., Lakhouaja, A. (2018). Data science in light of natural language processing: An overview. Procedia Computer Science, 127, 82–91. doi: https://doi.org/10.1016/j.procs.2018.01.101
18. Hrkút, P., Toth, Š., Ďuračík, M., Meško, M., Kršák, E., Mikušová, M. (2020). Data Collection for Natural Language Processing Systems. Intelligent Information and Database Systems, 60–70. doi: https://doi.org/10.1007/978-981-15-3380-8_6
19. Manning, C. D. (2009). An introduction to information retrieval. Cambridge university press.
20. Pang, B., Lee, L. (2008). Opinion Mining and Sentiment Analysis. Foundations and Trends® in Information Retrieval, 2 (1-2), 1–135. doi: https://doi.org/10.1561/1500000011

21. Brown, P. F., Della Pietra, S. A., Della Pietra, V. J., Mercer, R. L. (1993). The mathematics of statistical machine translation: Parameter estimation. Computational Linguistics, 19 (2), 263–311.
22. Palmer, M., Gildea, D., Kingsbury, P. (2005). The Proposition Bank: An Annotated Corpus of Semantic Roles. Computational Linguistics, 31 (1), 71–106. doi: https://doi.org/10.1162/0891201053630264
23. Satyapanich, T., Ferraro, F., Finin, T. (2020). CASIE: Extracting Cybersecurity Event Information from Text. Proceedings of the AAAI Conference on Artificial Intelligence, 34 (5), 8749–8757. doi: https://doi.org/10.1609/aaai.v34i05.6401
24. Peters, M., Neumann, M., Zettlemoyer, L., Yih, W. (2018). Dissecting Contextual Word Embeddings: Architecture and Representation. Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing. doi: https://doi.org/10.18653/v1/d18-1179
25. Roth, M., Lapata, M. (2016). Neural Semantic Role Labeling with Dependency Path Embeddings. Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). doi: https://doi.org/10.18653/v1/p16-1113
26. Marcheggiani, D., Titov, I. (2017). Encoding Sentences with Graph Convolutional Networks for Semantic Role Labeling. Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing. doi: https://doi.org/10.18653/v1/d17-1159
27. Howard, J., Ruder, S. (2018). Universal Language Model Fine-tuning for Text Classification. Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). doi: https://doi.org/10.18653/v1/p18-1031
28. Iter, D., Grangier, D. (2021). On the Complementarity of Data Selection and Fine Tuning for Domain Adaptation. Computation and Language. doi: https://doi.org/10.48550/arXiv.2109.07591
29. Devlin, J., Chang, M. W., Lee, K., Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint. doi: https://doi.org/10.48550/arXiv.1810.04805
30. Collobert, R., Weston, J., Bottou, L., Karlen, M., Kavukcuoglu, K., Kuksa, P. (2011). Natural language processing (Almost) from scratch. Journal of machine learning research, 12, 2493–2537.
31. Schütze, H. (1998). Automatic word sense discrimination. Computational linguistics, 24 (1), 97–123.
32. Pennington, J., Socher, R., Manning, C. D. (2014). Glove: Global vectors for word representation. Proceedings of the 2014 conference on empirical methods in natural language processing, 1532–1543. doi: https://doi.org/10.3115/v1/d14-1162
33. Surdeanu, M., Johansson, R., Meyers, A., Màrquez, L., Nivre, J. (2008). The CoNLL 2008 shared task on joint parsing of syntactic and semantic dependencies. CoNLL 2008: Proceedings of the Twelfth Conference on Computational Natural Language Learning, 159–177. doi: https://doi.org/10.3115/1596324.1596352

34. Shalev-Shwartz, S., Ben-David, S. (2014). Understanding machine learning: From theory to algorithms. Cambridge university press. doi: https://doi.org/10.1017/cbo9781107298019
35. Zhang, W., Yin, Z., Sheng, Z., Li, Y., Ouyang, W., Li, X. et al. (2022). Graph Attention Multi-Layer Perceptron. Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining. doi: https://doi.org/10.1145/3534678.3539121
36. Sandhu, A., Edara, A., Narayan, V., Wajid, F., Agrawala, A. (2022). Temporal Analysis on Topics Using Word2Vec. doi: https://doi.org/10.48550/arXiv.2209.11717
37. Heimerl, F., Gleicher, M. (2018). Interactive Analysis of Word Vector Embeddings. Computer Graphics Forum, 37 (3), 253–265. doi: https://doi.org/10.1111/cgf.13417
38. Bojanowski, P., Grave, E., Joulin, A., Mikolov, T. (2017). Enriching Word Vectors with Subword Information. Transactions of the Association for Computational Linguistics, 5, 135–146. doi: https://doi.org/10.1162/tacl_a_00051
39. Inoue, N., Shinoda, K. (2016). Adaptation of Word Vectors using Tree Structure for Visual Semantics. Proceedings of the 24th ACM International Conference on Multimedia. doi: https://doi.org/10.1145/2964284.2967226