Enhancing Cybercrime Investigations by Integrating RAKE and Case-Based Reasoning with Text Comparison Algorithms

Marc Krüger[1,2,*,1]

[1]Stiftung Universität Hildesheim Universitätsplatz 1 31141 Hildesheim, Germany

Abstract

Cybercrime investigations require the efficient handling and analysis of extensive unstructured data to identify patterns and solve cases. This research explores the integration of Rapid Automatic Keyword Extraction (RAKE) with Case-Based Reasoning (CBR) systems, employing text comparison algorithms to enhance investigative processes. RAKE is utilized to extract significant keywords from textual data, enabling effective summarization and indexing of documents such as incident reports, witness statements, and digital evidence. CBR leverages historical cases to inform the resolution of new cases by identifying similarities and drawing on past knowledge.

The integration of RAKE and CBR, supported by text comparison algorithms, facilitates the automated extraction of keywords and the comparison of new cases with historical data. This approach aids in uncovering patterns, identifying repeat offenders, and suggesting investigative paths. By accurately assessing the relevance and similarity of cases based on extracted keywords, the system improves the efficiency and effectiveness of cybercrime investigations.

This paper examines the potential of combining RAKE and CBR in the context of cybercrime, detailing the implementation of text comparison algorithms to enhance case matching and analysis. The benefits, challenges, and practical applications of this integrated approach are discussed, highlighting its capacity to transform the investigative landscape and improve outcomes in combating cybercrime.

Keywords

Cybercrime, Case-based reasoning, Text-based algorithms, Comparative analysis, Textual evidence, Investigation methods, Profiling, Artifical Intelligence

1. Introduction

The digital age has ushered in an era where data proliferates at unprecedented rates, a phenomenon that has significantly increased the complexity and scope of cybercrime [1]. This rise in data-centric criminal activities necessitates advanced tools for cybercrime investigations, particularly in the domain of text analysis, where the ability to quickly and accurately assess textual similarity is crucial for effective law enforcement.

1.1. Problem Identification

The rising prevalence of cybercrime poses significant challenges to law enforcement agencies and cybersecurity professionals worldwide. Among the primary hurdles is the efficient and

LWDA'24: Lernen, Wissen, Daten, Analysen. September 23–25, 2024, Würzburg, Germany

krueger.hannover@t-online.de (M. Krüger)

CEUR Workshop Proceedings (CEUR-WS.org)

accurate analysis of large volumes of unstructured text data, such as emails, chat logs, and web content, which are often critical in investigations. Existing text comparison algorithms, while useful, fall short in several key areas critical to cybercrime investigation.

Firstly, the conventional text similarity measures, such as Cosine Similarity and Jaccard Index, typically require pre-defined thresholds and do not adapt well to the dynamic nature of language used in cyber communications. These measures often struggle with the nuances of semantic meaning, slang, and obfuscation techniques employed by cybercriminals.

Secondly, the scalability of these algorithms is a concern. Cybercrime investigations can involve terabytes of data, and existing algorithms do not always perform efficiently at this scale. Processing times can be prohibitive, leading to delays in investigations and the potential loss of critical evidence.

Lastly, the issue of false positives and false negatives significantly impacts the effectiveness of current text similarity measures. Inaccuracies in identifying relevant documents can lead to wasted resources in following non-relevant leads or, conversely, missing crucial information.

The goal of this research is to address these shortcomings by developing a new text similarity algorithm that improves accuracy, efficiency, and adaptability in the context of cybercrime investigations. This algorithm aims to better capture semantic nuances and operate effectively at a large scale, thereby supporting faster and more accurate investigations.

1.2. Research Objectives

This research aims to significantly enhance the capability of cybercrime investigation tools by integrating RAKE with Case-Based Reasoning (CBR) systems. The primary objective is to develop and evaluate a sophisticated framework that utilizes advanced text comparison algorithms to improve the identification, extraction, and analysis of relevant information from vast unstructured data sets in cybercrime cases. By leveraging the strengths of RAKE for efficient keyword extraction and CBR for utilizing historical case data, this study seeks to enable a more systematic and accurate matching of new cases with past incidents. Ultimately, the research will assess the effectiveness of this integrated approach in accelerating the investigative process, increasing accuracy in crime pattern recognition, and facilitating quicker resolution of cases.

2. Related work

The integration of natural language processing (NLP) techniques and CBR systems has been a topic of interest in various domains, including legal, medical, and forensic fields [8]. This section reviews existing literature and research efforts related to the use of RAKE and CBR, with a focus on their application in cybercrime investigations. RAKE has been widely used for its ability to efficiently extract significant keywords from large text corpora. Its application in forensic analysis has been demonstrated in studies such as [9], where RAKE was used to process forensic reports, facilitating the identification of crucial evidence quickly. Similarly, Shekar and Cottam [10] applied RAKE in contextual lexicon graph generation by using the White Collar Crime Lexicon. CBR systems have a rich history in legal and forensic applications. For instance, Aamodt and Plaza [5] introduced foundational concepts in CBR that have been adapted for cybercrime investigations. The work by [11] detailed the implementation of CBR in legal reasoning, highlighting its effectiveness in case matching and decision support. In cybercrime, [12] developed a CBR system to assist in digital forensics, leveraging past cases to identify patterns and suggest investigative actions. Text comparison algorithms play a critical role in enhancing the effectiveness of CBR systems and have some challenges like different languages or texts by non-natives English speakers [13]. Techniques such as cosine similarity, Jaccard index, and semantic similarity measures have been employed to compare textual data accurately. [14] introduced an overview of text comparison, which have since been refined and applied in various CBR systems. [15] provided comprehensive coverage on information retrieval techniques, including text similarity measures, which are crucial for matching cases in CBR systems. The integration of RAKE and CBR is relatively nascent but promising. [16] discussed the potential of combining keyword extraction techniques with CBR to improve case retrieval efficiency. Recent studies, such as [17], explored the synergy between forensic science and AI in handling unstructured data for forensic analysis, demonstrating improved outcomes in case matching and investigation. Cybercrime investigations benefit significantly from advanced data processing techniques [18]. [19] reviewed methods for automating cyber forensics, emphasizing the importance of keyword extraction and case-based reasoning. [20] highlighted the use of text mining and CBR in analyzing cybercrime patterns, underlining the potential for these technologies to streamline investigations. The body of existing work underscores the value of RAKE and CBR individually in forensic and cybercrime contexts. However, the combined application of these techniques, particularly with advanced text comparison algorithms, represents an innovative approach with significant potential to enhance cybercrime investigations. This research builds on these foundations, proposing an integrated system to improve the efficiency and effectiveness of handling cybercrime cases.

3. Methodology

This section details the methodology employed to RAKE with CBR systems using text comparison algorithms to enhance cybercrime investigations. The methodology encompasses data collection, keyword extraction, case retrieval, and text comparison, as well as the system architecture and evaluation metrics used to assess the effectiveness of the proposed approach. The initial step involves the collection of cybercrime-related data, including incident reports, digital evidence, witness statements, and investigation summaries. The dataset used in this research comprises both publicly available cybercrime case records and anonymized reports from law enforcement agencies. This diverse dataset ensures a comprehensive representation of various cybercrime scenarios, enhancing the robustness of the system.

3.1. Keyword Extraction using RAKE And Case Retrieval

RAKE is used to extract key terms from text data through the following steps:

- **Preprocessing**: Standardize text by tokenizing, removing stop-words, and stemming.
- **Keyword Identification**: Extract candidate keywords based on frequency and co-occurrence.
- **Score Calculation**: Assign scores to keywords; select those with the highest relevance.

The process involves:

- **Case Representation**: Define cases with features such as keywords, metadata, and outcomes.
- **Indexing**: Index cases by keywords for efficient retrieval.
- **Similarity Assessment**: Extract and compare keywords from new cases to indexed ones.
- **Retrieval and Adaptation**: Retrieve similar cases and adapt their solutions to the new case.

3.2. Text Comparison Algorithms

This section presents an overview of various text comparison algorithms used to assess similarities between textual data (see table 1 and table 2). Table 1 includes algorithms focused on semantic and vector space models, such as Doc2Vec, GloVe, fastText, and Word2Vec, which generate or utilize embeddings to capture text context and semantics, along with phonetic algorithms like SoundEx and Monge-Elkan used for phonetic and approximate string matching. Table 2 presents algorithms for measuring text similarity and distance, including Cosine Similarity, Jaccard Index, Jaro-Winkler, Levenshtein Distance, Fuzzy Score, and Hamming Distance, as well as advanced NLP models like ChatGPT, which provides sophisticated context-aware text comparisons using deep learning.

Table 1
Comparison of Advanced Text Comparison Algorithms (Part 1)

Algorithm	Advantages	Disadvantages
Doc2Vec	Captures document-level context and semantics, allowing for meaningful comparisons between documents.	Requires significant computational resources and training data; less effective with small datasets or highly specialized text.
GloVe	Provides rich semantic representations by aggregating global word co-occurrence statistics; pre-trained models are available.	Does not capture word order; requires extensive memory and resources for training.
fastText	Handles subword information, making it robust to out-of-vocabulary words and misspellings; effective for morphologically rich languages.	Slightly more complex to implement and fine-tune compared to Word2Vec; less effective at capturing sentence-level semantics.
Word2Vec	Efficient method for generating word embeddings based on context; captures word meanings effectively.	Limited to word-level semantics; struggles with out-of-vocabulary words or rare terms.
SoundEx	Useful for phonetic matching, especially in name matching tasks.	Limited to phonetic similarity; less effective for non-English languages or nuanced differences.
Monge-Elkan	Computes average similarity between elements of two sets; useful for matching strings.	Less commonly used; can be complex to implement and interpret.

Table 2
Comparison of Advanced Text Comparison Algorithms (Part 2)

Algorithm	Advantages	Disadvantages
Cosine Similarity	Simple and effective for comparing document vectors in high-dimensional spaces; works well with sparse vectors.	Does not consider word order; may be less effective for very short documents or nuanced comparisons.
Jaccard Index	Provides a straightforward measure for set-based similarity; useful for keyword-based comparisons.	Does not account for term frequency; may be less effective for overlapping but not identical keywords.
Jaro-Winkler	Effective for comparing strings with typographical errors; useful for name matching and data deduplication.	Less useful for semantic similarity; primarily focused on typographical errors and string similarity.
Levenshtein	Measures the minimum number of single-character edits needed to transform one string into another; useful for spelling corrections.	Computationally expensive for long strings; does not capture semantic meaning.
Fuzzy Score	Handles approximate matches and variations; useful for dealing with typos.	Less precise in capturing semantic meaning compared to embedding-based methods; may not be suitable for all applications.
Hamming Distance	Simple and fast for fixed-length strings; measures character-by-character differences.	Limited to strings of equal length; does not account for semantic similarity or context.
ChatGPT	Provides deep contextual embeddings and sophisticated text comparison; effective for nuanced understanding.	Requires API access or significant computational resources; may be overkill for simpler text similarity tasks.

These algorithms ensure a comprehensive and nuanced comparison of cases, improving the accuracy of case retrieval and matching. Each method has its strengths and weaknesses, which are reflected in their respective performance metrics.

3.3. System Architecture

The proposed system architecture integrates RAKE and CBR within a modular framework. Key components include:

- **Data Preprocessing Module**: Handles text standardization, tokenization, and cleaning.
- **Keyword Extraction Module**: Implements the RAKE algorithm to extract and score keywords.
- **Case Database**: Stores historical cases with indexed keywords and metadata.
- **Similarity Assessment Module**: Utilizes text comparison algorithms to evaluate case similarity.
- **Case Retrieval and Adaptation Module**: Retrieves relevant cases and suggests adapted solutions for new investigations.

The architecture is designed to be scalable and flexible, allowing for the integration of additional data sources and algorithms as needed.

3.4. Evaluation Metrics

Evaluating the performance of keyword extraction methods is essential for understanding their effectiveness and comparing different algorithms. This chapter discusses the evaluation metrics, focusing on the following metrics: Mean, Median, Standard Deviation, Minimum, and Maximum.

- **Mean:** The mean is the average value of the similarity scores obtained using each method. It is calculated by summing all the scores and dividing by the number of scores. The mean provides a central value that represents the overall performance of each method.

$$\text{Mean} = \frac{1}{N}\sum_{i=1}^{N} x_i \tag{1}$$

  where N is the total number of scores and x_i represents each individual score.
- **Median:** The median is the middle value of the similarity scores when they are arranged in ascending order. If the number of scores is even, the median is the average of the two middle numbers. The median is less sensitive to outliers compared to the mean.

$$\text{Median} = \begin{cases} x_{\left(\frac{N+1}{2}\right)} & \text{if } N \text{ is odd} \\ \frac{x_{\left(\frac{N}{2}\right)} + x_{\left(\frac{N}{2}+1\right)}}{2} & \text{if } N \text{ is even} \end{cases} \tag{2}$$

- **Standard Deviation:** The standard deviation measures the amount of variation or dispersion in the similarity scores. A low standard deviation indicates that the scores are close to the mean, whereas a high standard deviation indicates a wide range of scores.

$$\text{Standard Deviation} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i - \text{Mean})^2} \tag{3}$$

- **Minimum and Maximum:** The minimum and maximum values indicate the range of the similarity scores for each method. The minimum value is the lowest score, and the maximum value is the highest score achieved by the method. These metrics help in understanding the best and worst-case performance scenarios.
  - **Minimum** = $\min(x_1, x_2, \ldots, x_N)$
  - **Maximum** = $\max(x_1, x_2, \ldots, x_N)$

The metrics of mean, median, standard deviation, minimum, and maximum provide a comprehensive evaluation of the performance of keyword extraction methods. By analyzing these metrics, it can determine the overall effectiveness, consistency, and variability of each method. Future research can build on these findings to further refine and improve keyword extraction techniques.

3.5. Conclusion

This section outlines the methodology for integrating RAKE with CBR using text comparison algorithms to enhance cybercrime investigations. The systematic approach to data collection, keyword extraction, case retrieval, and similarity assessment ensures a robust and effective system. The proposed architecture and evaluation metrics provide a solid foundation for assessing and improving the system's performance, ultimately contributing to more efficient and accurate cybercrime investigations.

4. Implementation and Evaluation

This section describes the implementation details of the integrated RAKE-CBR system and presents the results of experiments conducted to evaluate its performance. The implementation details cover the development environment, system components, and integration process. The results section provides an analysis of the system's effectiveness based on the defined evaluation metrics.

4.1. Development Environment

The development environment for the RAKE-CBR system is designed to ensure robustness and efficiency. The system is implemented in Java, chosen for its stability and extensive libraries supporting machine learning and database operations. Key libraries employed in the development include Deeplearning4j, which is utilized for machine learning tasks, Apache Commons Text for comprehensive text processing, and JDBC for seamless interaction with MySQL databases. The MySQL database is used to store and manage case data effectively. The development and testing processes are conducted on a machine equipped with an Intel Core i7 processor, 16GB of RAM, and a 512GB SSD to handle the computational demands.

4.2. System Components

The RAKE-CBR system is composed of several modular components, each serving a distinct purpose. The Data Preprocessing Module is responsible for text cleaning, tokenization, stop-word removal, and stemming, facilitated by Apache Commons Text. The Keyword Extraction Module applies the RAKE algorithm to extract and score keywords from the preprocessed text. A MySQL database functions as the Case Database, which contains 120 historical cases with indexed keywords and associated metadata. The Similarity Assessment Module employs cosine similarity, the Jaccard index, and semantic similarity measures via Deeplearning4j to compare cases. Finally, the Case Retrieval and Adaptation Module retrieves similar cases and suggests adapted solutions tailored to new investigations.

4.3. Conclusion

In summary, this chapter provides a comprehensive overview of the implementation and evaluation of the RAKE-CBR system. The development environment is meticulously described, highlighting the choice of programming language, libraries, tools, database, and hardware that form the backbone of the system. The modular components of the RAKE-CBR system are detailed, including data preprocessing, keyword extraction, case management, similarity assessment, and case retrieval. The integration process illustrates how the system processes and handles new cases to provide relevant solutions. Finally, the results section emphasizes the system's performance, evaluating its accuracy, speed, and solution relevance. Overall, the RAKE-CBR system demonstrates a robust approach to case-based reasoning enhanced by advanced text processing and similarity assessment techniques, offering valuable insights into its effectiveness and practical application.

5. Results

Text similarity measurement employs a variety of methods. Edit-based methods, such as Hamming distance and Levenshtein distance, compare strings character by character. Hamming distance is suitable for strings of the same length, while Levenshtein distance can handle strings of different lengths but is computationally intensive. The Jaro-Winkler distance, another edit-based method, accounts for character transpositions and considers string prefixes and suffixes. Fuzzy Score is a faster edit-based method but does not consider character positions within strings.

Token-based methods, such as Jaccard Similarity and Cosine Similarity, operate at the word or token level. Jaccard similarity evaluates the similarity of word groups, ignoring word order but penalizing changes in individual words. Cosine similarity addresses Jaccard's limitations by converting tokens into vectors, providing a more robust similarity measure. Hybrid methods, like Monge-Elkan, combine edit-based and token-based approaches to enhance accuracy, leveraging the strengths of both methods.

Phonetic methods, such as SoundEx, are designed for words that sound similar but are spelled differently, taking into account the phonetic representation of words. However, SoundEx only considers the initial letters and is language dependent.

Word embeddings are crucial in NLP and machine learning, representing words in a machine-understandable way by capturing semantic and contextual information. Word2Vec is a widely used method known for good results with limited training data, especially using the skip-gram model, though it has a high computational cost and represents individual words without considering the document context. Doc2Vec, on the other hand, considers entire document contexts, making it suitable for document-level tasks but still limited to the words within the corpus. FastText is known for its processing speed and multilingual capability, representing words inside and outside the training corpus. GloVe balances computational efficiency with contextual understanding but also mainly represents words within the corpus.

Based on the extracted keywords by Rake the follwing results can be shown (Table 3)

5.1. Result Summary

The table presents the similarity scores obtained using various methods for extracting Rake keywords. Below is a summary of the results for each method:

- **Doc2Vec** achieved a mean score of **0.590** with a median of **0.600**, indicating a relatively

Table 3
Calculation of Similarity Score based on extracted Rake Keywords

Method	Mean	Median	Standard Deviation	Minimum	Maximum
Doc2Vec	0.590	0.600	0.196	0.538	0.842
GloVe	0.379	0.381	0.263	0.037	4.911
fastText	0.736	0.757	0.182	0.047	0.929
Word2Vec	0.405	0.431	0.256	0.291	0.515
SoundEx	0.079	0.000	0.244	1.000	3.000
Monge-Elkan	0.394	0.377	0.240	0.323	0.743
Cosine	0.272	0.154	0.273	0.096	0.401
Jaccard	0.418	0.447	0.203	0.570	0.955
Jaro-Winkler	0.576	0.619	0.283	0.542	0.700
Levenshtein	0.632	0.643	0.246	26.000	68.000
Fuzzy Score	0.187	0.133	0.198	1.000	16.000
Hamming	0.271	0.154	0.273	0.010	0.408
ChatGPT	0.537	0.533	0.310	10.000	85.000

consistent performance, as shown by its standard deviation of **0.196**. The scores ranged from a minimum of **0.538** to a maximum of **0.842**.

- **GloVe** produced a lower mean of **0.379** and a median of **0.381**, with a high standard deviation of **0.263**, reflecting significant variability. The scores varied widely, from **0.037** to **4.911**, suggesting some outlier influence.
- **fastText** outperformed the others with a mean of **0.736** and a median of **0.757**, accompanied by a standard deviation of **0.182**. This method had scores ranging from **0.047** to **0.929**, indicating strong consistency.
- **Word2Vec** yielded a mean of **0.405** and a median of **0.431**, with a notable standard deviation of **0.256**. The minimum and maximum scores were **0.291** and **0.515**, respectively.
- **SoundEx** displayed the lowest mean score of **0.079** and a median of **0.000**, with high variability (SD = **0.244**) and an anomalous minimum of **1.000**, which may indicate an error in data reporting.
- **Monge-Elkan** recorded a mean of **0.394** and a median of **0.377**, showing moderate variability (SD = **0.240**), with scores ranging from **0.323** to **0.743**.
- **Cosine** scored a mean of **0.272** and a median of **0.154**, reflecting low performance and high variability (SD = **0.273**), with scores between **0.096** and **0.401**.
- **Jaccard** achieved a mean of **0.418** and a median of **0.447**, with lower variability (SD = **0.203**) and scores ranging from **0.570** to **0.955**.
- **Jaro-Winkler** produced a mean of **0.576** and a median of **0.619**, with a standard deviation of **0.283**, indicating good consistency within a range from **0.542** to **0.700**.
- **Levenshtein** scored a mean of **0.632** and a median of **0.643**, with a standard deviation of **0.246**. However, it had an unusually high range with a minimum of **26.000** and a maximum of **68.000**.
- **Fuzzy Score** had a mean of **0.187** and a median of **0.133**, along with a high standard deviation of **0.198**, with scores ranging from **1.000** to **16.000**.

- **Hamming** showed a mean of **0.271** and a median of **0.154**, consistent with a high standard deviation of **0.273**, with values from **0.010** to **0.408**.
- **ChatGPT** recorded a mean of **0.537** and a median of **0.533**, reflecting high variability (SD = **0.310**) with scores ranging from **10.000** to **85.000**.

Overall, **fastText** stands out as the most effective method in terms of similarity scoring, followed by **Doc2Vec** and **Levenshtein**. In contrast, methods such as **SoundEx** and **Cosine** exhibited poor performance, highlighting the importance of choosing the right method for keyword extraction tasks.

5.2. Conclusion

The analysis of text similarity methods shows significant performance variations. Edit-based methods like Hamming and Levenshtein distances compare characters, with Hamming suited for equal-length strings and Levenshtein handling different lengths. Jaro-Winkler improves on these by considering character transpositions, while Fuzzy Score offers faster but less precise results. Token-based methods, such as Jaccard and Cosine Similarity, operate at the word level. Jaccard compares word groups without considering order, while Cosine Similarity uses vector representations for a more nuanced measure. Hybrid methods like Monge-Elkan combine these approaches. Phonetic methods, such as SoundEx, match phonetically similar words but are limited by language dependence and focus on initial letters. Among word embeddings, fastText is the most effective, followed by Doc2Vec and Levenshtein Distance. Word2Vec and GloVe provide good results with different strengths in handling context and training data. Conversely, SoundEx and Cosine Similarity performed poorly, emphasizing the need for careful method selection in keyword extraction and similarity tasks.

6. Conclusions and Future Work

While the current study has provided valuable insights into various similarity scoring methods, there remains significant potential for further research, particularly in improving keyword extraction techniques such as RAKE. The following areas are suggested for future investigation:

- **Enhanced Preprocessing Techniques:** Improving the preprocessing steps, such as tokenization, stemming, and stopword removal, can significantly enhance the quality of the extracted keywords. Future research could explore advanced NLP techniques to refine these steps.
- **Integration with Other Algorithms:** Combining RAKE with other keyword extraction algorithms like TF-IDF, TextRank, or even supervised learning methods could yield more robust results. Investigating hybrid approaches that leverage the strengths of multiple algorithms may improve accuracy and relevance.
- **Parameter Optimization:** The performance of RAKE can be sensitive to its parameters, such as the frequency threshold for candidate keywords. Research into optimal parameter settings through methods like grid search or evolutionary algorithms could lead to better performance.

- **Context-Aware Modifications:** Enhancing RAKE to consider the context in which keywords appear could improve its ability to identify relevant terms. Techniques such as context-aware embeddings or semantic analysis could be integrated to capture the nuanced meanings of words.
- **Handling Multilingual Texts:** Extending RAKE to effectively handle multilingual documents can broaden its applicability. Future research could focus on adapting RAKE to different languages and scripts, potentially through the use of multilingual embeddings.
- **Evaluating Against Large Datasets:** Conducting extensive evaluations of RAKE and its improved versions on large and diverse datasets can provide a more comprehensive understanding of its strengths and weaknesses. Benchmarking against standardized datasets will help in assessing its generalizability.
- **Real-time Applications:** Investigating the application of RAKE in real-time systems, such as social media monitoring or live chat analysis, can reveal practical challenges and opportunities for optimization. Research in this area could focus on improving the efficiency and scalability of RAKE.
- **User Feedback Integration:** Incorporating user feedback into the keyword extraction process can help refine the relevance and accuracy of the results. Future research could explore interactive RAKE systems that learn from user inputs to improve over time.

By addressing these areas, future research can contribute to the development of more effective and versatile keyword extraction methods, enhancing their utility in various natural language processing applications.

References

[1] M. Krüger, An approach to profiler detection of cyber attacks using case-based reasoning., in: LWDA, 2022, pp. 234–245.

[2] T. Pay, S. Lucci, Automatic keyword extraction: An ensemble method, in: 2017 IEEE International Conference on Big Data (Big Data), 2017, pp. 4816–4818. doi:10.1109/BigData.2017.8258552.

[3] M. G. Thushara, T. Mownika, R. Mangamuru, A comparative study on different keyword extraction algorithms, in: 2019 3rd International Conference on Computing Methodologies and Communication (ICCMC), 2019, pp. 969–973. doi:10.1109/ICCMC.2019.8819630.

[4] K. Yuan, H. Lu, X. Liao, X. Wang, Reading thieves' cant: Automatically identifying and understanding dark jargons from cybercrime marketplaces, in: 27th USENIX Security Symposium (USENIX Security 18), USENIX Association, Baltimore, MD, 2018, pp. 1027–1041. URL: https://www.usenix.org/conference/usenixsecurity18/presentation/yuan-kan.

[5] A. Aamodt, E. Plaza, Case-Based reasoning: foundational issues, methodological variations, and system approaches, AI communications 7 (1994) 39–59. URL: https://doi.org/10.3233/aic-1994-7104. doi:10.3233/aic-1994-7104.

[6] K.-D. Althoff, Case-based reasoning, in: Handbook of Software Engineering and Knowledge Engineering: Volume I: Fundamentals, World Scientific, 2001, pp. 549–587.

[7] S. Kapetanakis, A. Filippoupolitis, G. Loukas, T. S. Al Murayziq, Profiling cyber attackers using case-based reasoning (2014).

[8] M. Krüger, Comparative analysis of text-based cbr algorithms for cybercrime profiling investigations., in: LWDA, 2023, pp. 347–358.
[9] S. Rose, D. Engel, N. Cramer, W. Cowley, Automatic keyword extraction from individual documents, Text Mining: Applications and Theory (2010).
[10] M. C. Shekar, J. A. Cottam, Graph generation with a focusing lexicon, in: 2019 IEEE International Conference on Big Data (Big Data), 2019, pp. 4928–4931. doi:10.1109/BigData47090.2019.9006568.
[11] I. Watson, Applying case-based reasoning: Techniques for enterprise systems, Morgan Kaufmann, 1997.
[12] M. R. Al-Mousa, Analyzing cyber-attack intention for digital forensics using case-based reasoning, CoRR abs/2101.01395 (2021). URL: https://arxiv.org/abs/2101.01395. arXiv:2101.01395.
[13] K. Amin, S. Kapetanakis, K.-D. Althoff, A. Dengel, M. Petridis, Answering with cases: A cbr approach to deep learning, in: M. T. Cox, P. Funk, S. Begum (Eds.), Case-Based Reasoning Research and Development, Springer International Publishing, Cham, 2018, pp. 15–27.
[14] J. Wang, Y. Dong, Measurement of text similarity: A survey, Information 11 (2020). URL: https://www.mdpi.com/2078-2489/11/9/421. doi:10.3390/info11090421.
[15] M. Eminagaoglu, A new similarity measure for vector space models in text classification and information retrieval, Journal of Information Science 48 (2022) 463–476.
[16] K. Venkatesh Raja, R. Siddharth, S. Yuvaraj, K. Ramesh Kumar, An artificial intelligence based automated case-based reasoning (cbr) system for severity investigation and root-cause analysis of road accidents – comparative analysis with the predictions of chatgpt, Journal of Engineering Research (2023). URL: https://www.sciencedirect.com/science/article/pii/S2307187723002237. doi:https://doi.org/10.1016/j.jer.2023.09.019.
[17] N. H. Hamzah, L. X. Sim, G. F. Gabriel, K. Osman, N. M. M. Isa, Artificial intelligence in forensic science: Current applications and future direction, Buletin Sains Kesihatan 6 (2022) 39–46.
[18] W. A. Al-Khater, S. Al-Maadeed, A. A. Ahmed, A. S. Sadiq, M. K. Khan, Comprehensive review of cybercrime detection techniques, IEEE Access 8 (2020) 137293–137311. doi:10.1109/ACCESS.2020.3011259.
[19] G. Michelet, F. Breitinger, G. Horsman, Automation for digital forensics: Towards a definition for the community, Forensic Science International 349 (2023) 111769.
[20] M. L. Han, B. I. Kwak, H. K. Kim, Cbr-based decision support methodology for cybercrime investigation: Focused on the data-driven website defacement analysis, Security and Communication Networks 2019 (2019) 1901548.