ELEKTROTEHNIŠKI VESTNIK 92(3): 151-156, 2025
OVERVIEW PROFESSIONAL PAPER

Comparative Analysis of Text Similarity Algorithms and Their Practical Applications in Computer Science

Josip Poljak[1], Dražen Crčić[1], Tomislav Horvat[1]

[1] University North, Varaždin University Center, Jurja Križanića 31b, Croatia
E-mail: Drazen.Crcic@unin.hr

Abstract. In an era defined by vast volumes of digital text, the capacity to compare, interpret, and quantify textual similarity is a cornerstone of modern computational linguistics and natural language processing (NLP). Text similarity algorithms support critical applications in information retrieval, plagiarism detection, sentiment analysis, text summarization, and beyond. This paper provides a comprehensive survey and comparative analysis of established text similarity algorithms, including edit-distance-based metrics (Levenshtein and Damerau-Levenshtein), character-based measures (Jaro and Jaro-Winkler), local sequence alignment (Smith-Waterman), vector-based semantic measures (Cosine similarity), and methods reliant on subsequence statistics (N-gram similarity). Each algorithm is analyzed in terms of its underlying theoretical foundations, computational complexity, performance characteristics, and domain-specific suitability. While traditional approaches excel in correcting typographical errors or identifying subtle lexical variations, more robust methods handle semantically rich corpora, larger text bodies, and intricate linguistic phenomena. Moreover, potential avenues for improvement are explored, including hybridization of existing approaches and the integration of emerging machine learning and deep neural models. This holistic examination aims to inform the selection and development of text similarity measures for diverse real-world applications and to guide future research directions in computational linguistics.

Keywords: text similarity algorithms, natural language processing, computational linguistics

Primerjalna analiza algoritmov za podobnost besedil in njihove praktične uporabe v računalništvu

Primerjava in merjenje podobnosti med digitalnimi besedili sta ključna za računalniško lingvistiko in obdelavo naravnega jezika. Algoritmi za podobnost se uporabljajo pri iskanju informacij, zaznavanju plagiatorstva, analizi sentimenta in povzemanju besedil.

Prispevek predstavlja primerjalno analizo uveljavljenih metod, kot so Levenshteinova razdalja, Jaro-Winkler, Smith-Waterman, kosinusna podobnost in N-grami. Ocenjene so glede na teoretične osnove, računsko zahtevnost, učinkovitost in primernost za različna področja. Tradicionalne metode so učinkovite pri zaznavanju napak in leksikalnih razlik, naprednejše pa pri obravnavi semantično bogatih in daljših besedil. Raziskane so tudi možnosti izboljšav z združevanjem pristopov in uporabo metod strojnega učenja. Namen analize je usmerjati uporabo in nadaljnji razvoj teh algoritmov.

1 INTRODUCTION

Text constitutes a fundamental medium for communication, information dissemination, and knowledge representation in the digital age. As organizations and individuals produce and consume ever-increasing amounts of textual data—from academic literature, social media posts, and news articles to corporate documents—automated techniques for analyzing and comparing textual content have become indispensable [1, 2]. Text similarity algorithms measure how closely related two or more texts are, capturing lexical, syntactic, and sometimes semantic characteristics [2]. These measures underpin a range of applications. One of the applications is in information retrieval and search engines development, and retrieving documents based on user queries and ranking results by relevance [3]. Text similarity helps identify documents that share contents or topics. Another application is during plagiarism detection and identifying unauthorized reuse of the textual material by measuring the closeness of documents and detecting passages that show a suspicious overlap [4]. Text similarity algorithms could be used for text summarization and paraphrase detection, while determining whether one summary accurately represents another text or whether two sentences express similar meanings [5]. Sentiment and topic analysis is also one of the typical implementations, where algorithms group texts into similar thematic clusters or compare user-generated contents for a consistent emotional or topical alignment [6].

Spelling correction and OCR post-processing as a last example of the usage implies correcting typographical errors or character recognition mistakes in digitized texts by comparing candidate strings to known words [7].

This paper surveys key algorithms for text similarity, examining their theoretical underpinnings, strengths, and weaknesses. Traditional edit-distance-based methods measure literal character-level differences [4, 5, 8], while more advanced vector-based techniques capture semantic

Received: 17 March 2025
Accepted: 28 April 2025

relationships [9, 10]. By analyzing a range of approaches, the paper highlights how these methods complement one another and can be combined or extended [11, 12]. Future research could explore emerging trends such as neural embeddings (e.g., Word2Vec) [13] and advanced machine learning-driven approaches to further enhance the accuracy and robustness of text similarity measures.

2 Related Research

Research on text similarity algorithms has evolved significantly in recent years, driven by the growing need for automated text analysis across various domains such as information retrieval, plagiarism detection, and natural language understanding. Traditional approaches focused on edit-distance metrics and vector-based models. However, more recent studies have explored advanced statistical techniques, neural embeddings, and hybrid models that combine different similarity measures.

[15] provides a comprehensive analysis of document similarity algorithms, including statistical models, neural network-based approaches, and corpus-based methods. Their evaluation highlights the effectiveness of neural embeddings like Word2Vec and BERT in capturing semantic relationships in large textual corpora, outperforming traditional edit-distance measures for longer texts. [16] focuses on case-based reasoning (CBR) to identify the most similar textual documents. The researchers compare multiple similarity measures, such as the TS-SS metric, Euclidean distance, and cosine similarity, demonstrating that CBR-based approaches can improve the accuracy of text similarity assessments. [17] proposes an innovative method for extracting knowledge from textual datasets using the Spearmans rank correlation coefficient. The approach proves effective for grouping similar documents and extracting common features from text collections, offering a new perspective on similarity analysis. The research in hybrid similarity measurement techniques has gained attention in recent years. [18] presents a hybrid algorithm that integrates the semantic term similarity with TF-IDF to improve the text similarity assessment, proving to be more effective than traditional TF-IDF-based approaches. Another significant study [19] explores methods for computing legal document similarity by comparing various approaches, including citation network analysis and textual content similarity measures. Their findings indicate that hybrid techniques can significantly improve the accuracy in legal text retrieval. [20] introduces an aspect-based document similarity approach for research papers. The method enhances traditional similarity detection by considering specific sections and aspects of a document rather than evaluating the entire text uniformly. The results show that aspect-based comparisons provide a more refined understanding of the textual similarity in academic research.

These advancements indicate a growing trend toward integrating traditional similarity measures with statistical inference and machine learning models. The adoption of neural embeddings and hybrid techniques offers a more nuanced understanding of the textual similarity, particularly for applications requiring deep semantic analysis.

3 Text Similarity Algorithms

This chapter describes five algorithms for measuring the text similarity used in the research. A detailed presentation is given of algorithms that use different approaches and techniques to solve the problem of determining the similarity of two texts.

3.1 Edit-Distance-Based Measures: Levenshtein and Damerau-Levenshtein

The Levenshtein distance quantifies the minimum number of the elementary operations—insertions, deletions, and substitutions—required to transform one string into another [4]. It can be defined using dynamic programming. For two strings a and b of lengths m and n, distance $lev_{a,b}(i,j)$ is:

$$lev_{a,b}(i,j) = \begin{cases} \max(i,j) & if\ \min(i,j) = 0, \\ min \begin{cases} lev_{a,b}(i-1,j) + 1 \\ lev_{a,b}(i,j-1) + 1 \\ lev_{a,b}(i-1,j-1) + 1_{(a_i \neq b_j)} \end{cases} \end{cases} \quad (1)$$

The Levenshtein distance is invaluable for detecting minor textual variations, typographical errors, and spelling corrections [4, 5]. However, its character-level orientation makes it less effective for capturing semantic relationships in longer, and more complex texts [3].

The Damerau-Levenshtein distance extends the Levenshtein metric by adding transposition as a fourth allowed operation [5]. This addition is motivated by the fact that certain common typing errors involve swapping adjacent characters. As a result, the Damerau-Levenshtein distance can more closely align with the real-world human error patterns, reducing the cost when characters are simply reversed. Both metrics are widely used in contexts demanding fine-grained character-level comparisons [5].

The similarity score can be converted from the edit distance using:

$$Text\ similarity = 1 - \frac{Number\ of\ changes}{Longest\ word\ length} \quad (2)$$

3.2 Character-Based String Similarities: Jaro and Jaro-Winkler

The Jaro similarity focuses on the number of matching characters and the number of permissible transpositions within a certain window [6]. It is particularly effective for comparing shorter strings, such as personal names, product codes, or short terms, where slight reorderings can occur. The resulting similarity score ranges from 0 to 1, with 1 indicating identical strings.

The Jaro similarity measurement formula:

$$sim_j = \begin{cases} 0, & m = 0, \\ \frac{1}{3}\left(\frac{m}{|s_1|} + \frac{m}{|s_2|} + \frac{m-t}{m}\right), & m \neq 0 \end{cases} \quad (3)$$

The Jaro-Winkler similarity refines the Jaro measure by giving an increased weight to the matching prefixes. This adjustment recognizes that errors often occur toward the end of words and that strings sharing a common start are more likely to be related [7]. These measures are commonly applied in record linkage, data cleaning, and de-duplication tasks.

The Jaro-Winkler similarity measurement formula:

$$sim_w = sim_j + lp(1 - sim_j) \quad (4)$$

3.3 Local Sequence Alignment: Smith-Waterman Similarity

Originating from computational biology, the Smith-Waterman similarity identifies locally optimal alignments between two sequences, maximizing the scoring scheme that rewards matches and penalizes mismatches and gaps [8]. Unlike the global alignment methods, Smith-Waterman focuses on the most similar substring pairs within the larger sequences, making it suited for detecting highly similar segments in extensive texts.

The Smith-Waterman algorithm uses a scoring matrix $H(i,j)$:

$$H(i,j) = max\begin{cases} 0 \\ H(i-1,j-1) + match/mismatch\ score \\ H(i-1,j) + gap\ penalty \\ H(i,j-1) + gap\ penalty \end{cases} \quad (5)$$

However, the Smith-Waterman computational cost can be significant. Despite this overhead, it is considered exceptionally accurate for identifying similar text regions, making it useful in plagiarism detection, legal text analysis, and any domain where a local similarity is more relevant than an overall string resemblance [8].

3.4 Vector-Space Model and Cosine Similarity

The Cosine similarity represents documents as vectors in a high-dimensional feature space, where each dimension corresponds to a term weight (e.g., TF-IDF) [9]. By measuring the cosine of the angle between two vectors, it evaluates the similarity independently of the document length. The Cosine similarity is widely used in information retrieval, sentiment analysis, and text classification.

The Cosine similarity formula:

$$\text{Cosine Similarity} = \cos\theta = \frac{A \cdot B}{\|A\|\|B\|} \quad (6)$$

This vector-based approach transcends mere character overlaps, approximating the semantic relatedness when combined with appropriate feature extraction methods. Its computational complexity can vary, but with an efficient indexing and dimension reduction techniques, the cosine similarity remains scalable to very large text corpora [10].

3.5 N-gram Similarity

N-grams are consecutive sequences of n characters or words extracted from a text [11]. The N-gram similarity estimates how often two texts share these subsequences. For the character-level n-grams, the methods capture the orthographic patterns; for the word-level n-grams, they reflect the lexical and syntactic similarity.

The N-gram similarity formula:

$$\frac{2 \cdot |\ X \cap Y\ |}{|\ X\ | + |\ Y\ |} \quad (7)$$

This approach is frequently employed in the plagiarism detection, OCR output correction, machine translation evaluation (e.g., BLEU score), and author style analysis. By varying n, the N-gram similarity can be tuned to emphasize shorter lexical matches or longer phrase-level correspondences [12].

4 Web Application For Calculating The Textual Similarity

In order to apply the textual similarity algorithms practically, a custom web application is developed, allowing users to compare texts using various algorithms. The application is designed to be user-friendly yet flexible enough to meet the needs of researchers and practitioners in Natural Language Processing (NLP). This custom developed application enables a real-time text comparison and a support for texts of varying lengths.

4.1 Application Architecture

The architecture of the web application is designed to enable modularity, scalability, and ease of maintenance. The application consists of three main parts: the user interface (Frontend Module), the server-side component (Backend Module), and the functionality of the text similarity algorithms (Algorithm Module). The user interface allows for an intuitive application management, while the server-side component processes requests and runs similarity algorithms. The implemented algorithms offer different methods for calculating the text similarity, providing users with flexibility in choosing the most suitable algorithm for their needs. The combination of these elements ensures a high efficiency and reliability of the application.

4.1.1 Frontend Module

The user interface is the first point of contact for the user with the application. It is designed to be intuitive and easy to use, allowing users to easily find the required functionalities. The main feature of the user interface is enabling the input of texts to be compared. The text input

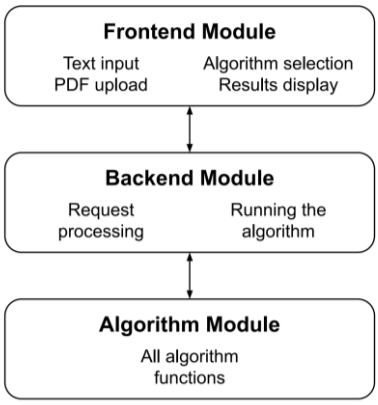

Figure 1. Application architecture diagram.

can be done manually in text fields or by uploading PDF files, where users can upload two PDF files, and the text will be automatically extracted. After the text input, the user selects an algorithm for a comparison, where one of the available similarity measurement algorithms can be chosen, or all algorithms at once. Finally, the comparison results are displayed through the user interface, either individually or for all algorithms in a table format, allowing users to easily analyze the similarity.

The user interface is developed using the HTML technology for the web application structure, CSS for styling, and JavaScript and jQuery for a dynamic content management and communication with the server-side system via AJAX requests. Bootstrap is used to ensure the responsiveness of the interface.

4.1.2 Backend Module

After the user enters all the input parameters, the system initiates the server-side, the backend module of the application. The initial request processing performs the input data validation and triggers the appropriate similarity measurement algorithms. The server-side component executes the implemented algorithms to calculate the similarity between two texts or two extracted textual contents from PDF files. After data processing, the server-side returns the results to the user interface in the form of a JSON response, which is then displayed in a visual format within the user interface.

The server-side component of the application is implemented using the Flask web framework for the Python programming language. Flask provides an easy way to manage the HTTP requests, handle different routes, and integrate with other services. The server-side also uses PyMuPDF for text extraction from PDF files and Psutil for measuring the resource consumption, such as the memory usage and processor time[14].

4.1.3 Algorithm Module

The third module of the application, which contains the functionality of the text similarity algorithms, uses the implemented algorithms in the programming language and performs an actual comparison. Each algorithm is implemented as a separate function within the algorithm module, enabling an easy integration and maintenance. A total of seven algorithmic function combinations have been developed, based on the five algorithms described in Chapter 3. The functions used are:

- Levenshtein Distance - measures the number of operations (insertions, deletions, substitutions) required to transform one string into another.
- Damerau-Levenshtein Distance - an extension of the Levenshtein distance that includes transpositions (swapping adjacent characters).
- Jaro Similarity - focuses on the match of characters and the number of transpositions between two strings.
- Jaro-Winkler Similarity - an extension of the Jaro similarity that gives more weight to matching the initial segments of a string.
- Smith-Waterman Similarity - an algorithm for local sequence alignment used to identify similar segments within longer texts.
- Cosine Similarity - measures the similarity between two non-zero vectors in a multidimensional space based on the cosine of the angle between them.
- N-gram Similarity - analyzes textual data based on overlapping n-grams (sequences of n consecutive elements).

4.2 Application Functionalities

The text similarity application is designed to provide an intuitive user experience and powerful analytical capabilities. The application user interface allows users to input textual data in two ways. The first method involves a direct manual entry of two texts. Users can directly enter two texts to be compared for the similarity into text fields within the application. This option is useful for a quick analysis of shorter texts or specific excerpts. The second method is uploading PDF files. Users can upload two PDF files through the user interface. The application uses the PyMuPDF library to extract a textual content from the PDF files, enabling the analysis of longer and more complex documents. The extracted texts are automatically populated into the corresponding text fields.

After entering or uploading the texts, users can select one of the available algorithms for a text comparison through a dropdown menu. The available algorithms include the Levenshtein distance, Damerau-Levenshtein distance, Jaro similarity, Jaro-Winkler similarity, Smith-Waterman similarity, Cosine similarity, and N-gram similarity. This functionality provides users with the flexibility to choose the most suitable algorithm for their specific analysis needs.

The initial user interface of the application is shown in Figure 2. After the user enters the texts and selects an algorithm, the application performs the similarity calculation between the two texts. The application works by having the user interface send all the necessary parameters to the backend module. The backend module triggers the corresponding algorithm for calculating the similarity, processes the texts, and returns the result

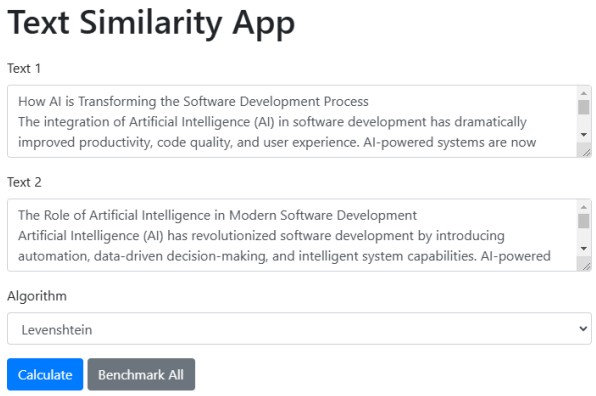

Figure 1. User interface of the application.

as a percentage (%) that is immediately displayed to the user within the application as a degree of the similarity between the two texts. This functionality allows users to quickly and accurately calculate the text similarity using different methods and algorithms.

The application also provides the functionality for measuring the performance of all available algorithms to compare their relative ratios. This feature allows users to analyze the efficiency of different algorithms in terms of the execution time and memory usage. In this process, the backend module runs all available similarity calculation algorithms. Each algorithm is executed separately, measuring the execution time and memory load. The performance results are displayed to the user in a table format, enabling an easy comparison of the performance of all available algorithms. This functionality offers users a detailed insight into the performance of different algorithms, helping them choose the most suitable algorithm for their needs.

The results of the similarity and performance calculations are displayed in a table format so that users can easily compare the performance of different algorithms (see Fig. 3). The results display the similarity scores and additionally, performance testing results, i.e., "benchmark" results. The similarity results refer to the similarity calculation for the selected algorithm from the dropdown menu. The result is shown as a percentage (%), allowing users to immediately see the degree of the similarity between the two texts. The additional performance testing results, i.e., "benchmark" results for all algorithms, display information about the algorithm name, similarity (%), execution time (seconds), and memory load (bytes). The table format allows users to easily compare the performance of different algorithms and make informed decisions about which algorithm to choose for specific analysis needs.

4.3 Application Testing

Testing is a crucial step in the software development to ensure the accuracy, reliability, and performance of the application. The application is tested using a variety of test cases that include diverse textual data. Each test case is designed to check specific functionalities of the application and the performance of the algorithms. The test cases cover examples of simple texts, similar texts with different formatting, texts with grammatical errors, completely different texts, and very long texts.

Algorithm	Similarity	Time (s)	Memory (bytes)
Levenshtein	30.78%	0.6519	0
Damerau-Levenshtein	30.96%	2.2222	1654784
Jaro	80.09%	0.0507	0
Jaro-Winkler	80.09%	0.0450	0
Smith-Waterman	12.14%	38.3630	2592768
Cosine	57.11%	0.0000	0
N-gram	65.30%	0.0021	0

Figure 2. Example overview of results within the application.

The testing results are compared with the expected outcomes to assess the accuracy and performance of the application. For each test case, the results are analyzed and compared with the predicted outcomes. The comparison of the expected and actual results is shown in Table 1.

Table 1. Comparison of the expected and actual results

Test case	Expected result	Actual result	Comment
simple texts	high similarity	93%	aligned with expectations
similar texts with different formatting	high similarity	89%	aligned with expectations
texts with grammatical errors	medium similarity	76%	aligned with expectations
completely different texts	low similarity	12%	aligned with expectations
very long texts	variable similarity	40%	depending on the specific content of the texts

The testing of the text similarity application is conducted systematically to ensure its accuracy, reliability, and performance. The test cases cover various scenarios, from simple texts to complex and very long documents. The comparison of the expected and actual results demonstrates a high accuracy of the algorithms.

5 CONCLUSION

The paper provides a comprehensive analysis of the text similarity algorithms, covering traditional edit-distance approaches, character-based similarity measures, vector-space models, and hybrid methods. By integrating these methodologies, their theoretical foundations, computational efficiency, and application domains are explored.

A custom web application is developed to implement and compare multiple text similarity algorithms in a real-world environment. The application enables users to upload texts, select appropriate algorithms, and visualize results, thus providing valuable insights into the effectiveness and efficiency of different approaches. The performance testing demonstrates that while the classical methods like the Levenshtein distance are effective for minor text variations, more advanced vector-based and machine learning-driven approaches offer a superior semantic understanding and scalability.

Our future research will focus on enhancing hybrid approaches that combine character-level corrections with deep-learning models for the semantic similarity. Optimizing the algorithmic performance for large-scale datasets and exploring domain-specific adaptations can further improve the text comparison methodologies. Advancement of the text similarity techniques supports applications in natural language processing, information retrieval, and artificial intelligence-driven decision-making, thus fostering a more accurate and meaningful text analysis.

REFERENCES

[1] Manning, C. D., Schütze, H. "Foundations of Statistical Natural Language Processing", *The MIT Press,* 1999.

[2] Jurafsky, D., Martin, J. H. "Speech and Language Processing", 2nd edition, *Pearson Prentice Hall*, ISBN 978-0-13-187321-6, 2008.

[3] Navarro, G. "A Guided Tour to Approximate String Matching." *ACM Computing Surveys,* vol. 33, no. 1, 2001, pp. 31-88.

[4] Levenshtein, V.I. "Binary codes capable of correcting deletions, insertions, and reversals." *Soviet Physics Doklady,* vol. 10, no. 8, 1966, pp. 707-710.

[5] Damerau, F. J. "A technique for computer detection and correction of spelling errors." *Communications of the ACM,* vol. 7, no. 3, 1964, pp. 171–176.

[6] Jaro, M. A. "Advances in record-linkage methodology as applied to matching the 1985 census of Tampa, Florida." *Journal of the American Statistical Association,* vol. 84, no. 406, 1989, pp. 414-420.

[7] Winkler, W. E. "String comparator metrics and enhanced decision rules in the Fellegi-Sunter model of record linkage." *Proceedings of the Section on Survey Research Methods,* American Statistical Association, 1990, pp. 354-359.

[8] Smith, T. F., Waterman, M.S. "Identification of common molecular subsequences." *Journal of Molecular Biology,* vol. 147, no. 1, 1981, pp. 195-197.

[9] Salton, G., Wong, A., Yang, C.S. "A vector space model for automatic indexing." *Communications of the ACM,* vol. 18, no. 11, 1975, pp. 613-620.

[10] Singh, J., Singh, L. "Cosine Similarity Algorithm for Natural Language Processing." *International Journal of Advanced Research in Computer Science and Software Engineering,* vol. 7, no. 7, 2017, pp. 1-5.

[11] Brown, T.G., Smith, J.L. "Analysis of Edit Distance Algorithms in Text Processing." *Journal of Computer Science and Technology,* vol. 36, no. 4, 2021, pp. 755-772.

[12] Nguyen, T., Zhang, Y. "N-gram Based Text Analysis: A Comparative Study." *Linguistics and Computational Models,* vol. 8, 2023, pp. 210-230.

[13] Mikolov, T., Chen, K., Corrado, G., Dean, J. "Efficient Estimation of Word Representations in Vector Space." Proceedings of ICLR, 2013.

[14] Grinberg, M. "Flask Web Development: Developing Web Applications with Python", *O'Reilly Media*, 2018.

[15] Nicholas Gahman, Vinayak Elangovan "A Comparison of Document Similarity Algorithms", *International Journal of Artificial Intelligence and Applications (IJAIA),* Vol.14, No.2, 2023.

[16] Marko Mihajlovic, Ning Xiong "Finding the Most Similar Textual Documents Using Case-Based Reasoning." *arXiv preprint*, 2019. (https://arxiv.org/abs/1911.00262)

[17] Nino Arsov, Milan Dukovski, Blagoja Evkoski, Stefan Cvetkovski "A Measure of Similarity in Textual Data Using Spearman's Rank Correlation Coefficient." *arXiv preprint*, 2019. (https://arxiv.org/abs/1911.11750)

[18] Huang, C.-H., Yin, J., & Hou, F. "Research on Text Similarity Measurement Hybrid Algorithm with Term Semantic Information and TF-IDF Method" *International Journal of Intelligent Systems*, 37(5), 1505-1521, 2022.

[19] Bhattacharya, P., Ghosh, K., Pal, A., & Ghosh, S. "Methods for Computing Legal Document Similarity: A Comparative Study". *International Journal of Legal and Information Technology*, 28(3), 289-305, 2020.

[20] Ostendorff, M., Ruas, T., Blume, T., Gipp, B., & Rehm, G. "Aspect-based Document Similarity for Research Papers", *Journal of Information Science*, 46(4), 544-560, 2020.

Josip Poljak is currently working as a Solution Architect and Developer specializing in software development, system architecture and the implementation of advanced technological solutions. He holds a Bachelor's degree in Computing and Informatics from the University North, Croatia. With over ten years of experience, he has been actively involved in designing and developing software solutions, leading development teams, and optimizing the system performance. His primary interests are in the implementation and application of artificial intelligence (AI) across various technologies and industries, with a strong focus on natural language processing (NLP), intelligent automation, and AI-driven decision-making systems. Through his work, he aims to bridge the gap between the traditional software engineering practices and the emerging AI capabilities, enabling more efficient and intelligent solutions in the digital world.

Dražen Crčić is currently working as a lecturer at the University North, Croatia. He graduated from the Faculty of Electrical Engineering and Computing, Zagreb. He has over 20 years of experience in the practical implementation of information and communication technologies in telecommunications, software development and data center infrastructure deployment, both as an IT specialist and manager in enterprise environments. His research interests are mainly in the field of web technologies, information security and the application of artificial intelligence tools.

Tomislav Horvat is currently working as an assistant professor at the University North, Croatia. He received his Ph.D. degree from the Faculty of Electrical Engineering, Computer Science, and Information Technology, Osijek. His research interests are predominantly in the application of artificial intelligence tools, particularly in predictive modeling and outcome predicting across various domains. With a strong background in the AI-driven methodologies, he has contributed to several academic projects and publications that explore innovative uses of machine learning and data analytics to address complex problems.