Jurnal Analisis Komputasi Digital

Vol.8 No.12, 2024
ISSN: 24455131

A HYBRID APPROACH COMBINING TRIE-BASED CACHING AND LEVENSHTEIN DISTANCE ALGORITHMS TO OPTIMIZE DOCUMENT SEARCH ENGINE PERFORMANCE AT PT CHEMCO HARAPAN NUSANTARA

Agung Brotokuncoro[1], Wiranto Herry Utomo[2]
President University
E-mail: agung.brotokuncoro@student.president.ac.id[1], wiranto.herry@president.ac.id[2]

Abstract

In the context of rapid data growth at PT Chemco Harapan Nusantara, optimizing the efficiency and accuracy of document search engines becomes very important. Current search engines face difficulties in accurately predicting user needs and take a long time to find documents. This study aims to overcome these challenges through a hybrid approach that combines Trie-based caching techniques and the Levenshtein Distance algorithm. Trie-based caching functions to significantly increase search speed by pre-indexing document keywords & autosuggestion. Meanwhile, the Levenshtein Distance algorithm improves the accuracy of the system in handling misspelled queries or searches with partial matches. The implementation of both algorithms significantly improved search performance by 43.23%, reducing the processing time to 44.88 ms, compared to 78.92 ms in the previous search engine that did not utilize caching. In addition, this system also achieved an increase in Precision from the previous 50.00% to 97.50%, Recall increased from 41.75% to 94.00%, and F1 Score also increased from 45% to 95%. These values indicate that this system is effective in finding relevant documents while reducing irrelevant search results. The combination of Trie-based caching and Levenshtein Distance not only increases search speed but also provides more accurate search results. Thus, this study successfully provides a solution to improve the performance of the document search engine at PT Chemco Harapan Nusantara, thereby supporting the company's operational efficiency amidst the challenges of increasingly complex data growth.

Keywords *— Trie-Based Caching ; Levenshtein Distance ; Search Engine Performance ; Misspelled Queries ; Auto Suggestion.*

1. INTRODUCTION

PT Chemco Harapan Nusantara is a company that manages a variety of operational and procedural documents that require rapid and precise access. Therefore, efficient document management is a critical component of its operations. These documents contain the necessary information for daily operations, reporting, and compliance with audit standards, including those from internal audits such as ISO and those from consumers. The performance of a document search system that is occasionally inaccurate can result in a variety of issues, including the inability to locate relevant documents and long search times. This ultimately affects negative assessments during audits from customers or during certification audits such as ISO. A frequent issue is the system's incapacity to adapt queries to variations in writing, such as typos or differences in capital and small letters. This can lead to search results that do not align with the user's requirements.

Consequently, a solution is required to enhance the efficiency of the document search engine, enabling more rapid and precise access to information.

In this context, the document search approach using the current search engine often faces limitations in handling increasing data volumes, variations in writing, and typos. Current search engines tend to be inefficient in providing relevant results if there is a slight error in writing a query from the user, such as different spellings or capitalization errors. This condition causes users to be dissatisfied because the search results do not match their needs. Difficulty in finding documents also affects the effectiveness of company operations because employees have to spend more time finding relevant documents.

To overcome these challenges, this study proposes the implementation of a hybrid approach that combines two techniques, namely Trie-Based Caching and the Levenshtein Distance algorithm. The Trie-Based Caching approach uses the Trie data structure to store and manage words based on prefixes, allowing fast and efficient searches for words or documents based on prefixes. Trie is very effective in implementing features such as auto-suggest or autocomplete, which make it easier for users to find relevant information faster. Through Trie, the system can build an efficient index for searching words based on prefixes, so that when users type the first few letters, the system can provide appropriate results without having to check the entire data thoroughly. This Trie can also help minimize search time because it will only check the relevant part of the data, not the entire database.

In addition, By computing the minimum number of modifications (insertions, deletions, or character substitutions) necessary to transform one string into another, the Levenshtein Distance algorithm is employed to evaluate the similarity between two strings. This algorithm is very useful in the context of document search, because it is able to overcome the problem of spelling variations caused by typos or minor differences in spelling. By implementing this algorithm, the document search system can be more tolerant of user input errors, and can still provide relevant and appropriate results. The combination of Trie and the Levenshtein Distance algorithm provides flexibility in dealing with input variations, while increasing search speed.

The application of this hybrid approach not only aims to improve search efficiency, but also to provide more accurate and relevant results for users. Trie-based caching allows the system to quickly find documents that match the user's input prefix, while the Levenshtein Distance algorithm ensures that spelling variations and typos will not hinder the search. In this way, the search engine can display results that are more in line with user expectations and reduce search errors.

In terms of search speed and result accuracy, it is anticipated that this hybrid approach will enhance the performance of the document search engine at PT Chemco Harapan Nusantara. Implementation of Trie-based Caching with Least Recently Used (LRU) policy will help in efficient cache management, where the cache will store the words that are most frequently accessed by the user, and automatically remove the words that are rarely accessed. This aims to improve search efficiency and memory usage. Meanwhile, the Levenshtein Distance algorithm will ensure the accuracy of the search results, even in situations where the user makes typos or spelling variations. Trie-based caching allows the search system to store and manage document indexes based on frequently used prefixes.

2. METHODOLOGY

This study uses quantitative and experimental approaches to evaluate the performance of document search engines. The dataset is taken from the current document

search database of PT Chemco Harapan Nusantara. The methodology includes data collecting, data preprocessing, model implementation, and performance evaluation using metrics such as query latency, Precision, Recall and F1 Score. The flowchart visually outlines the process from data collection to model evaluation, providing a clear overview of the methodological framework.

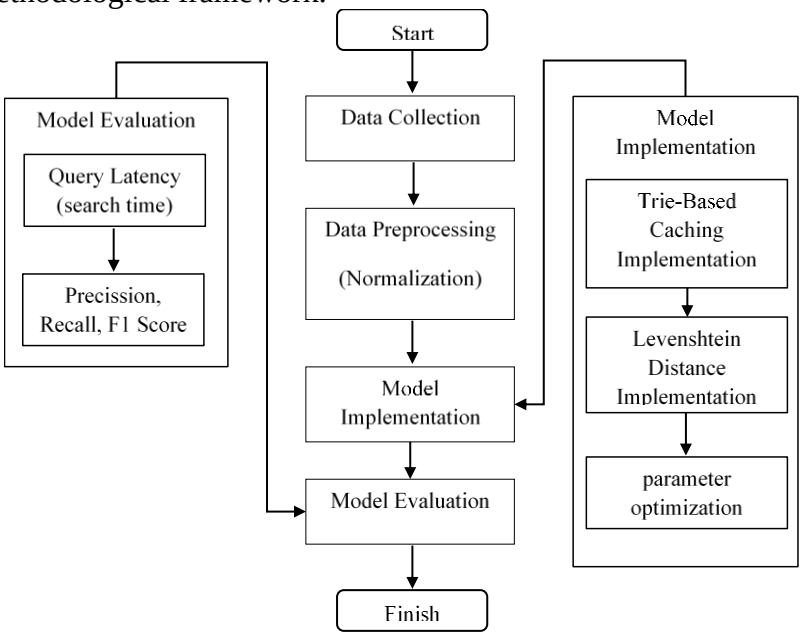

Figure 1 Research Design Flowchart.

3. RESULTS AND DISCUSSION

In this section, we will analyze and discuss the test results that have been carried out on the Trie-based caching and hybrid approaches. This analysis includes the advantages and limitations of implementing Trie-based caching in improving search performance.

Performance Analysis of Trie-based Caching

Based on results testing, the implementation of Trie-based caching shows improvement significant in performance search:

1. Advantages of Trie-based Caching

Trie-based caching has demonstrated the ability to improve average search response speed by up to 43.23% compared to searches conducted without caching. This significant improvement is achieved through the caching mechanism, which stores previously searched results. When identical or similar queries are entered, the Trie can directly access the cached results without needing to repeat the search across the database or document. This faster search performance is crucial for applications with high query loads or those requiring real-time search results, ensuring both efficiency and responsiveness.

2. Limitations of Trie-based Caching

1. Requires Additional Memory For Trie Structure
   - Even though Trie has efficiency in storing strings with the same prefix, the use of Trie-based caching is still consume lots of memory.
2. Performance Decreases on Very Complex Queries
   - Trie-based caching is very effective for frequent queries repeated or own pattern general, but when the query becomes more complex or very specific, performance can decrease.
   - Complex queries involving long words or rarely cached maybe no get profit from caching, so that time response Can approach search without cache.
   - Trie performance can also be influenced If there is too many nodes to be checked for queries that are not general, so that time required for Trie traversing

increases.

3. Conclusion of Trie-based Caching Performance Analysis

In general overall, Trie-based caching is proven capable give improvement significant performance with excess main in the form of speed high response. However, the use of Trie-based caching remains need consideration related allocation memory addition as well as possible performance decrease for very complex queries.

Trie-based caching is ideal for applications that require search with frequent patterns repeated, especially when speed and consistency results search become priority. However, for handle more scenarios area, for example search complex or with large data scale, approach This Possible need combined with method others, so that performance system can remain optimal in various condition search.

Analysis Levenshtein Distance

This section serve evaluation algorithm Levenshtein Distance in context his ability handle variation error Typing so that recall increases. Analysis This focus on recall as metric main, supported with threshold optimization for increase search engine effectiveness in tolerate error typing and differences small on query. Algorithm This tested on a database covering 1,000 variations error Typing in a way random and evaluated for his ability identify document relevant to various threshold setting.

1. Levenstein's advantages

The recall result of 76% shows that algorithm Levenshtein Distance is very effective in handle variation error typing. The majority document relevant succeed found, so that show robust algorithm This in face difference small on query.

Threshold on the algorithm Levenshtein Distance is used For determine level tolerance to error in query writing. Higher threshold tall apply more matching tight, while the threshold is lower low allow more tolerance big to difference between queries and indexed documents.

For determine the optimal threshold, algorithm tested using 1000 errors Typing random with threshold variation start from 0.5 to 0.9. Test results summarized in table 4.3, table 4.4 and table 4.5.

Based on results evaluation, concluded that the optimal threshold for algorithm is 0.6. This threshold setting allow algorithm tolerate up to 3 errors typing, with a recall of 76%.

2. Limitations Levenstein's

Following is limitation Algorithm Levenshtein:

1. Dependence on Optimal Threshold

One of limitations main algorithm Levenshtein Distance is its dependence on the threshold setting. This study show that threshold value 0.6 give results best with tolerance until three typo, resulting in a recall of 76%. However, this threshold setting Possible No can applied in a way common in datasets with level error or different noise distributions. If the threshold is set too tight (for example, 0.8 or 0.9), algorithm fail identify document with error small, which causes decrease in recall. On the other hand, a threshold that is too loose can cause results that are not relevant, so that lower precision. Dependence This make algorithm not enough adaptive against dynamic datasets or heterogeneous.

2. Precision Limited to High Error Cases

Although algorithm Levenshtein Distance works Good For recognize document with error small, precise tend decrease in case with noise level or high error. For example, on a larger dataset big or with Lots typo random, algorithm Possible match documents that are not relevant only Because pattern the text similar with queries. Trade-off between recall and precision This the more clear when the threshold is relaxed, which can reduce quality results search.

3. Sensitivity to Relevance Contextual

Algorithm This only focuses on the similarity of text strings without consider relevance semantics or context from results matching. As a result, the algorithm can be

misordered document with difference typography small more tall than documents that have conformity more contextual meaningful. For example, a document that has term relevant in a way semantics but different in a way phonetics can ignored fully Because high edit distance.

4. Inability Handle Variation Complex Linguistics

Although algorithm Levenshtein Distance effective For error Typing simple, algorithm This difficulty handle variation more linguistics complex like synonyms, abbreviations, or multi-word phrases. As example, algorithm Possible fail matching queries like "quality assurance" with document that only mention "QA" because his inability recognize equality linguistics. Limitations This limit the application in scenarios that require understanding more languages advanced.

3. Conclusion

Algorithm Levenshtein Distance, though effective in handle error Typing simple with a recall of 76% at threshold 0.6, has a number of limitations like dependence on threshold settings, precision limited to cases error height, and inability handle relevance semantics as well as variation complex linguistics.

Analysis Hybrid Approach

This section will discuss analysis about a hybrid approach that combines Trie-based caching and algorithms Levenshtein Distance in system search. Combination second method This done For maximize speed and Recall, especially in facing a query that has variation error Typing (typo). The hybrid approach is designed For overcome limitations of each method If used in a way separate and utilize superiority both of them in different contexts.

Hybrid approach works create good synergy between Trie-based caching and Levenshtein Distance, where each method give different contributions However each other complete.

Trie-based caching is highly effective in improving search speed, particularly for queries that are frequently repeated or share similar patterns. By leveraging caching, previously performed search results can be stored and quickly accessed without the need for recalculating or traversing the Trie from the beginning. The use of a Trie allows search results to be stored based on prefixes, enabling queries with similar beginnings or patterns to be served with pre-existing results. This approach significantly reduces response time, making it especially beneficial for optimizing search performance in systems with high query repetition or pattern similarity.

The Levenshtein Distance algorithm improves recall by adding a layer of tolerance for typographical errors in queries that are not found in the cache, effectively handling common variations in misspellings. By calculating the edit distance between the query and stored keywords, Levenshtein Distance allows the search engine to provide relevant results even when typos are present in the query. This algorithm works optimally in a hybrid approach, as it is only utilized during cache misses, ensuring computational load remains low while maintaining search accuracy.

The combination of Trie-based caching and Levenshtein Distance delivers optimal performance by effectively addressing different query scenarios. Trie-based caching handles frequently used queries with speed and efficiency, while Levenshtein Distance ensures accuracy for queries not found in the cache, especially those containing typographical errors. These two methods complement each other, with Trie-based caching enhancing efficiency for repeated queries and Levenshtein Distance improving accuracy for more complex or error-prone queries. This synergy creates a balance between speed and recall, making the hybrid approach ideal for search systems that require flexibility and reliability in managing diverse queries.

Recommendations Development

Development system learning for dynamic threshold in algorithm Levenshtein

Distance can increase accuracy search with adjust the threshold automatic based on user query patterns, context search, or data characteristics. With use machine learning approach, systems can determine the optimal threshold adaptive based on behavior users and historical data, so that capable handle error Typing with more accurate and relevant. Future research can integrate more techniques forward, like search model semantics or method based on machine learning, for increase accuracy and adaptability system, making it more strong and suitable for diverse and large-scale datasets big

4. CONCLUSION

Based on the research and implementation of a hybrid approach combining Trie-based caching and Levenshtein Distance algorithms to optimize document search engine performance at PT Chemco Harapan Nusantara, several key conclusions can be drawn:

The hybrid search engine achieved significant performance improvements over the current search engine, with query latency reduced from 78.92ms to 44.88ms, precision increased from 50.00% to 97.50%, recall improved from 41.75% to 94.00%, and F1 Score enhanced from 45.00% to 95.00%. These substantial enhancements highlight the effectiveness of combining both algorithms to optimize search performance, delivering faster, more accurate, and more comprehensive search results.

The implementation of trie-based caching demonstrated a 43.23% improvement in response time by effectively reducing the need for repeated full database searches. This caching mechanism proved particularly beneficial for frequently repeated queries, queries with common prefixes, and auto-suggestion functionality, highlighting its ability to optimize query handling and enhance overall system performance.

The application of the Levenshtein Distance algorithm, with an optimal threshold setting of 0.6, significantly improves the system's ability to handle up to three character variations in search terms, case-sensitivity variations, and common typos. This approach improves recall by 76% for queries with spelling errors, providing effective error tolerance while maintaining search accuracy and ensuring robust handling of various user input variations.

Future Work

In the ever-evolving search technology landscape, optimizing search performance and user experience requires an innovative approach that balances accuracy, speed, and adaptability. Modern systems increasingly incorporate sophisticated algorithms and intelligent mechanisms to handle different types of queries, user behaviors, and data complexities. By leveraging machine learning, natural language processing, and adaptive algorithms, search engines are becoming not only more accurate but also more responsive to user needs.

Dynamic threshold optimization was achieved through the development of machine learning models for automatic threshold adjustment in the Levenshtein Distance algorithm, allowing the system to adapt more effectively to varying query demands. By integrating user behavior patterns, the threshold settings were optimized to align with real-world usage, while context-aware adjustments based on query types further enhanced performance. Additionally, research into adaptive threshold mechanisms tailored for different document categories ensured a more precise and efficient search experience across diverse datasets.

Advanced search capabilities are enhanced through the integration of semantic search functions, which enable better understanding of context and more accurate results. Natural language processing capabilities are developed to enable more intuitive and user-friendly searches, while the implementation of multilingual support with special

algorithmic adjustments expands accessibility across a wide range of user groups.

5. REFERENCES

S. Ibrihich, A. Oussous, O. Ibrihich, and M. Esghir, “A Review on recent research in information retrieval,” Procedia Comput. Sci., vol. 201, no. C, pp. 777–782, 2022, doi: 10.1016/j.procs.2022.03.106.

S. Sivarajkumar et al., “Clinical Information Retrieval: A Literature Review,” J. Healthc. Informatics Res., pp. 1–31, 2024, doi: 10.1007/s41666-024-00159-4.

K. A. Hambarde and H. Proenca, “Information Retrieval: Recent Advances and beyond,” IEEE Access, vol. 11, pp. 76581–76604, 2023, doi: 10.1109/ACCESS.2023.3295776.

R. Nadia and A. A. Nababan, “Fitur Autocomplete Menggunakan Algoritma Knuth-Morris-Prat (KMP) Pada Pencarian Istilah Komputer,” J. Ilmu Komput. dan Sist. Inf., vol. 4, no. 1, pp. 1–5, 2021.

M. Işik and H. Dağ, “The impact of text preprocessing on the prediction of review ratings,” Turkish J. Electr. Eng. Comput. Sci., vol. 28, no. 3, pp. 1405–1421, 2020, doi: 10.3906/elk-1907-46.

A. Tabassum and R. R. Patil, “A Survey on Text Pre-Processing & Feature Extraction Techniques in Natural Language Processing,” Int. Res. J. Eng. Technol., no. June, pp. 4864–4867, 2020, [Online]. Available: www.irjet.net

E. Mourão, J. F. Pimentel, L. Murta, M. Kalinowski, E. Mendes, and C. Wohlin, “On the performance of hybrid search strategies for systematic literature reviews in software engineering,” Inf. Softw. Technol., vol. 123, no. July 2019, 2020, doi: 10.1016/j.infsof.2020.106294.

A. F. Wicaksono and A. Moffat, “Metrics, User Models, and Satisfaction,” pp. 654–662, 2020.

G. A. Sharma P, Agrawal A, Alai L, “Challenges and techniques in preprocessing for twitter dataNo Title,” Int. J. Eng. Sci. Comput., pp. 7 (4): 6611-6613., 2017.

P. M. Camacho-Collados J, “On the role of text preprocessing in neural network architectures,” Proc. 2018 EMNLP Work. BlackboxNLP Anal. Interpret. Neural Networks NLP; Brussels, Belgium, pp. 40–46, 2018.

L. Hickman, S. Thapa, L. Tay, M. Cao, and P. Srinivasan, “Text Preprocessing for Text Mining in Organizational Research: Review and Recommendations,” Organ. Res. Methods, vol. 25, no. 1, pp. 114–146, 2022, doi: 10.1177/1094428120971683.

D. N. Kobayashi, V. B., Mol, S. T., Berkers, H. A., Kismih´ok, G., & Den Hartog, “Text mining in organizational research,” Organ. Res. Methods, pp. 21(3), 733–765, 2018, [Online]. Available: https://doi.org/10.1177/ 1094428117722619

Y. HaCohen-Kerner, D. Miller, and Y. Yigal, “The influence of preprocessing on text classification using a bag-of-words representation,” PLoS One, vol. 15, no. 5, pp. 1–22, 2020, doi: 10.1371/journal.pone.0232525.

M. A. Rosid, A. S. Fitrani, I. R. I. Astutik, N. I. Mulloh, and H. A. Gozali, “Improving Text Preprocessing for Student Complaint Document Classification Using Sastrawi,” IOP Conf. Ser. Mater. Sci. Eng., vol. 874, no. 1, 2020, doi: 10.1088/1757-899X/874/1/012017.

T. Sahu, “Autosuggestion services in web search,” pp. 1–19, 2022.

K. Rinartha, L. Gede, and S. Kartika, “Penerapan LSA dan Query Suggestion untuk Pencarian Judul Artikel Menggunakan Framework FLASK LSA and Query Suggestion for Article Searching with FLASK Framework,” Cogito Smart J., vol. 8, no. 1, pp. 183–193, 2022.

H. T. Sadiah, M. Saad Nurul Ishlah, and N. Najwa Rokhmah, “Query Suggestion on Drugs e-Dictionary Using the Levenshtein Distance Algorithm,” Lontar Komput. J. Ilm. Teknol. Inf., vol. 10, no. 3, p. 193, 2019, doi: 10.24843/lkjiti.2019.v10.i03.p07.

M. A. Raza, R. Mokhtar, N. Ahmad, M. Pasha, and U. Pasha, “A Taxonomy and Survey of Semantic Approaches for Query Expansion,” IEEE Access, vol. 7, pp. 17823–17833, 2019, doi: 10.1109/ACCESS.2019.2894679.

Y. Wang, H. Huang, and C. Feng, “Query Expansion with Local Conceptual Word Embeddings in Microblog Retrieval,” IEEE Trans. Knowl. Data Eng., vol. 33, no. 4, pp. 1737–1749, 2021,

doi: 10.1109/TKDE.2019.2945764.
R. Nogueira, W. Yang, J. Lin, and K. Cho, “Document Expansion by Query Prediction,” pp. 1–7, 2019, [Online]. Available: http://arxiv.org/abs/1904.08375
Z. Zheng, K. Hui, B. He, X. Han, L. Sun, and A. Yates, “BERT-QE: Contextualized query expansion for document re-ranking,” Find. Assoc. Comput. Linguist. Find. ACL EMNLP 2020, pp. 4718–4728, 2020, doi: 10.18653/v1/2020.findings-emnlp.424.
S. Tahery and S. Farzi, “Customized query auto-completion and suggestion — A review,” Inf. Syst., vol. 87, p. 101415, 2020, doi: 10.1016/j.is.2019.101415.
N. Agrawal, “Auto Complete Using Graph Mining : A Different Approach,” pp. 268–271, 2011.
S. Gog, G. E. Pibiri, and R. Venturini, “Efficient and Effective Query Auto-Completion,” SIGIR 2020 - Proc. 43rd Int. ACM SIGIR Conf. Res. Dev. Inf. Retr., pp. 2271–2280, 2020, doi: 10.1145/3397271.3401432.
D. Hiemstra, “Reducing Misinformation in Query Autocompletions,” 2020, [Online]. Available: http://arxiv.org/abs/2007.02620
S. S. AMAN, B. G. N’GUESSAN, D. D. A. AGBO, and T. KONE, “Search engine Performance optimization: methods and techniques,” F1000Research, vol. 12, pp. 1–31, 2024, doi: 10.12688/f1000research.140393.3.
S. Tahery and S. Farzi, “TIPS: Time-aware Personalised Semantic-based query auto-completion,” J. Inf. Sci., vol. 48, no. 4, pp. 524–543, 2022, doi: 10.1177/0165551520968690.
A. K. Mishra and D. K. Jha, “For Web Data Mining an Improved FP-Tree Algorithm,” vol. 5, no. 6, pp. 147–152, 2018.
M. Engineering, “Predictive auto-completion for query in search engine Vinay Singh and Dheeraj Kumar Purohit Vimal Kumar, Pratima Verma and Ankita Malviya,” vol. 28, no. 3, 2018.
V. Touzeau, C. Maïza, D. Monniaux, and J. Reineke, “Fast and exact analysis for LRU caches,” Proc. ACM Program. Lang., vol. 3, no. POPL, 2019, doi: 10.1145/3290367.
G. Hasslinger, K. Ntougias, F. Hasslinger, and O. Hohlfeld, “Comparing Web Cache Implementations for Fast O(1) Updates Based on LRU, LFU and Score Gated Strategies,” IEEE Int. Work. Comput. Aided Model. Des. Commun. Links Networks, CAMAD, vol. 2018-September, no. 1, pp. 1–7, 2018, doi: 10.1109/CAMAD.2018.8514951.
D. Chen, C. Ding, F. Liu, B. Reber, W. Smith, and P. Li, “Uniform lease vs. LRU cache: Analysis and evaluation,” ISMM 2021 - Proc. 2021 ACM SIGPLAN Int. Symp. Mem. Manag. co-located with PLDI 2021, pp. 15–27, 2021, doi: 10.1145/3459898.3463908.
I. Mele, N. Tonellotto, O. Frieder, and R. Perego, “Topical result caching in web search engines,” Inf. Process. Manag., vol. 57, no. 3, p. 102193, 2020, doi: 10.1016/j.ipm.2019.102193.
B. Berger, M. S. Waterman, and Y. W. Yu, “Levenshtein Distance, Sequence Comparison and Biological Database Search,” IEEE Trans. Inf. Theory, vol. 67, no. 6, pp. 3287–3294, 2021, doi: 10.1109/TIT.2020.2996543.
M. M. Yulianto, R. Arifudin, and A. Alamsyah, “Autocomplete and Spell Checking Levenshtein Distance Algorithm To Getting Text Suggest Error Data Searching In Library,” Sci. J. Informatics, vol. 5, no. 1, p. 75, 2018, doi: 10.15294/sji.v5i1.14148.
R. Umar, Y. Hendriana, and E. Budiyono, “Implementation of Levenshtein Distance Algorithm for ECommerce of Bravoisitees Distro,” Int. J. Comput. Trends Technol., vol. 27, no. 3, pp. 131–136, 2015, doi: 10.14445/22312803/ijctt-v27p123.
Y. H. Agustin, Y. Septiana, and A. Y. Aspahany, “Search Optimization of PIP Scholarship Recipients In Web-Based Student Data Application Using The Levenshtein Distance Algorithm,” Sinkron, vol. 8, no. 4, pp. 2069–2081, 2023, doi: 10.33395/sinkron.v8i4.12898.
T. Anjali, T. R. Krishnaprasad, and P. Jayakumar, “A Novel Sentiment Classification of Product Reviews using Levenshtein Distance,” Proc. 2020 IEEE Int. Conf. Commun. Signal Process. ICCSP 2020, pp. 507–511, 2020, doi: 10.1109/ICCSP48568.2020.9182198.