Implementation of Cosine Similarity Algorithm on Omnibus Law Drafting

Aristoteles*, Muhammad Umaruddin Syam, Tristiyanto, Bambang Hermanto
Department of Computer Science, Faculty of Mathematics and Natural Sciences, Lampung University, Indonesia

***Abstract*—Drafting of Omnibus Laws presents a complex challenge in legal governance, often involving the integration and consolidation of disparate legal provisions into a unified framework. In this context, the application of advanced computational techniques becomes crucial for streamlining the drafting process and ensuring coherence across the law's various components. Cosine similarity, a widely used measure in natural language processing and document analysis, offers a quantitative means to assess the similarity between different sections or articles within the Omnibus Law draft. By representing legal texts as high-dimensional vectors in a vector space model, cosine similarity enables the comparison of textual similarity based on the cosine of the angle between these vectors. Implementing cosine similarity in the context of omnibus law using FastAPI and Laravel can be a valuable tool for analyzing similarity between legal documents, especially in the context of omnibus law. Legal practitioners and researchers can use the cosine similarity measure to compare the textual content of different legal documents and identify similarities. This can aid in tasks such as legal document retrieval, clustering similar provisions, and detecting potential inconsistencies. The combination of FastAPI and Laravel provides a potent and efficient way to develop and deploy this functionality, contributing to the advancement of legal informatics and analysis. The dataset used is Undang-Undang (UU) which used Bahasa from 1945 to 2022, comprising a total of 1705 UU. The implemented cosine similarity yielded a recall rate of 90.10% on the law.**

***Keywords*—Cosine similarity; FastAPI; Laravel; Omnibus Law**

I. Introduction

Indonesia is a country of law with numerous regulations. President Jokowi has acknowledged that there are too many regulations in Indonesia and that it is not ideal for the country to be known as a 'country of regulations'. President Jokowi has acknowledged that there are too many regulations in Indonesia and that it is not ideal for the country to be known as a 'country of regulations" [1]. According to the website peraturan.go.id, as of May 19, 2022, Indonesia has 50,538 regulations. When considering the median growth for the top five hierarchies outlined in Article 7, Paragraph 1 of Undang-Undang (UU) Number 12 of 2011, the Constitution has a median of 0, the MPR Tap has a median of 0, the UU has a median of 16, the PP has a median of 55, and the Perpres has a median of 0. Regulations are implemented to organize and regulate people's lives within a nation or state. The large number of regulations can have a negative impact, which is grouped into four categories: (1) Regulatory conflict, which occurs when some regulations or articles conflict with existing regulations, (2) Inconsistent regulatory consistency, which occurs when there are inconsistent regulations or provisions in one legislation and its derivatives, (3) Diverse regulatory interpretations, which occurs when the objectives and subjects of regulation are unclear. This results in unclear language that is difficult to understand and an inappropriate system. Additionally, there may be non-operational regulations, which are regulations that are inconsistent with one law and its derivatives or have no effect, but are still valid or lack enforcement regulations [2].

The government aims to reduce the growth of regulations, particularly UU, by simplifying or shortening them. This is achieved through the implementation of the omnibus law approach, which is an appropriate way to develop a legal framework for licensing Indonesia's business processes. This method allows the creation of regulations that cover several substantive materials or several smaller ones in one rule, promoting order, legal certainty, and expediency [3]. The omnibus law is a method or concept of rule-making that aims to solve problems related to licensing by creating a new UU that revises articles in several existing UU [4].

According to Article 64 paragraph 1b of UU No. 13 of 2022, one of the omnibus methods is to revoke laws and regulations of the same type and hierarchy. To avoid overlap between rules, legal drafters must identify the source of the revoked rules. This requires an understanding of similar regulations and their hierarchy. A database of UU, including articles and paragraphs, can aid legal drafters. Additionally, it is important to check for similarities between norms and UU to minimize language differences.

Algorithms that can be used to check for similarity include cosine similarity. Cosine similarity measures vector similarity by calculating the angle between them [5]. Calculating similarity using two angles has the advantage of being language-independent. It can be used even when no corpus is available. Unfortunately, cosine similarity is not sensitive to differences in size, which means that two patterns with very different attribute values can have high similarity scores [5]. Using Term Frequency and Inverse Document Frequency (TF-IDF) can solve the problem of cosine similarity, which is insensitive to vector magnitude [6].

Cosine similarity is a technique used in legal document retrieval to assess the similarity of several legal documents. To guarantee that cases with comparable circumstances are handled uniformly in each instance, it is employed to obtain previous case records of a particular case [7]. It has been demonstrated through experimentation that cosine similarity and the doc2vec approach may automatically obtain court rulings of comparable legal issues [8]. When it comes to legal aid, cosine similarity is used to compare legal document

*Corresponding Author.

embeddings, which are texts represented in a semantic vector space [9]. Cosine similarity is another method that uses unsupervised learning techniques like Word2Vec CBoW, Word2Vec Skip-gram, and TF-IDF to determine how similar court documents are to one another [10]. Cosine similarity is employed to group lawful [11].

This research aims to confidently assist lawmakers and legal writers in thoroughly searching for keywords (norms) in UU using Bahasa and implementing them in lawmaking with the aid of a search engine, this approach will significantly save time in understanding each existing law.

II. METHODS

The research was conducted using the waterfall model, one of the software development models. Waterfall model is divided into four stages, namely analysis, design, implementation and testing. Fig. 1 illustrates the research design using the waterfall model.

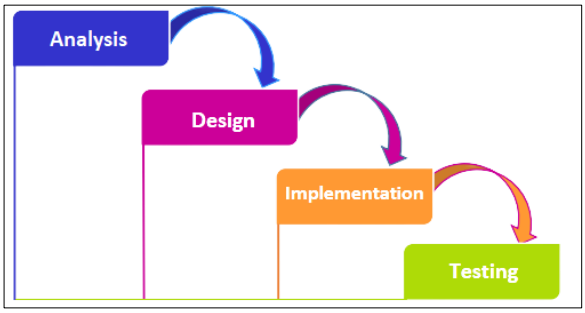


Fig. 1. Research design.

A. Analysis Phase

The analysis stage, also known as software requirements specification (SRS), is a thorough and comprehensive description of the software to be developed. Analysis includes system and business analysis to define both functional and non-functional requirements. The task for the analysis stage is to identify functional and data requirements for the software. Requirements gathering can be done by conducting observations, interviews, and document analysis. Function and data requirements that are complete can facilitate the next stage in the waterfall model. Raw data collection is transformed or pre-processed into mature, processable data. Preprocessing is the stage of converting data into a form that can be understood by machines [12]. Preprocessing aims to standardize data to facilitate processing [13]. This process involves four steps.

1) Case folding: Case folding is the process of equalizing cases in a document [14]. Phase of case folding aims to make all characters in the law lower case, remove numbers, remove punctuation marks, and remove spaces.

2) Tokenizing: Tokenization is the process of dividing a document into smaller parts, known as tokens [14].

3) Filtering: Filtering involves removing words deemed unnecessary or unimportant, based on stop words [15].

4) Stemming: Stemming is the process of removing all of the affixes and suffixes from the words, and this process is used to filter the results [9].

B. Design Phase

The design stage involves planning and problem-solving for a software solution. It requires the software developer to define the plan of the solution, including algorithm design, software architecture design, database conceptual schema and logic diagram design, concept design, graphical interface, and data structure definition.

Algorithm and architecture design can be aided by image processing software such as Paint, Adobe Photoshop, or Corel Draw. Conceptual schemes and data structure definitions can be created using ERD diagrams or other charting software. For interface design, Balsamiq or Figma can be used.

C. Implementation Phase

The implementation stage is when business requirements and design specifications are transformed into executable programs, databases, websites, or software components through programming. During the implementation phase, real code is written and compiled into a working application where the database and text have been created; in other words, the implementation phase is the process of converting all the requirements and blue ink into a production environment.

The research implementation utilizes information systems to enhance user comprehension. Information systems are organizational systems that summarize management functions and daily transaction processing requirements. These systems support the strategic activities of the organization in order to provide certain external parties with the information needed to facilitate decision making [16]. The information system is referred to as Omnibus Law Information System for the purpose of this research. The architecture is structured using FastAPI, Laravel, and MariaDB. FastAPI was chosen due to its superior performance compared to NodeJS and Go [17].

D. Testing Phase

The testing phase is commonly referred to as verification and validation. Verification is the process of verifying that the software meets the requirements and specifications to achieve the goals of software development. The purpose of verification is to evaluate the software to determine whether the product of the development phase meets the conditions imposed at the beginning of the stages of the waterfall model. Validation is the process of evaluating software during or at the end of the development process to determine if it meets the specified requirements.

III. RESULT AND DISCUSSION

The results include the research process on the case study that is the Omnibus Law and the data which are the UU. The Discussion includes the implementation process of each research methodology in the research, which consists of analysis, design, implementation, and testing.

A. Research Results

The Omnibus Law is a means of reducing the number of laws in Indonesia, which is considered an effective way to consolidate numerous regulations into a single unit [18]. Another potential benefit of the Omnibus Law is the ability to modify laws that have overlapping or other issues. However,

the challenge of the Omnibus Law is to modify the content without creating problems such as overlap or multiple interpretations. This study aimed to address the challenge of identifying similarities between legal content and the normative vocabulary to be established.

This research utilizes cosine similarity through the Python programming language and the FastAPI framework as the backend for calculations. The frontend uses the Laravel framework with PHP as the primary language. The FastAPI framework is employed to implement the cosine similarity model, and the final result is the API requested by the Laravel framework.

B. Discussion

The discussion is structured according to the research design presented in Fig. 1. The research process begins with the analysis stage, followed by the design, implementation, and testing stages.

1) Analysis phase: The author designs the function and data requirements based on the analysis and observation of the Law. The following are the function and data requirements.

a) Function requirements: Functional requirements define the software's necessary functions. The intended user is a legal drafter or lawmaker.

- Users can calculate the similarity of the vocabulary (norms) with the law as a whole.
- Users can perform vocabulary (norm) search on any article of the law.
- Users can see the highlight of the vocabulary (norm) in each article.
- Users can select the article to be printed in pdf or docx.

b) Data requirements: The study's data requirements are focused on preprocessed UU data. Preprocessing is accomplished using regular expressions and natural language toolkits (NLTK) in the Python programming language. The preprocessing process generated 60,186 data points, which is 15,595 fewer than the extraction process.

2) Design phase: The design stage is the implementation phase of the analysis stage. The draft module development design phase includes algorithm design, use case diagram, and user interface design.

a) Algorithm design: The drafting module was developed using the cosine similarity algorithm to calculate the distance between two angles. Converting words into vectors is essential for accurately measuring the distance between two vectors projected onto the X and Y axes. The angle is obtained from the TF-IDF results formula can be seen at Eq. (1) and Eq. (2), which are calculated using the cosine similarity formula [19]. The cosine similarity, as a method of measuring the similarity between the documents, varies between positive values from 0 to 1 [20]. Two texts are virtually identical, as evidenced by a cosine similarity close to 1, indicating a small angle between them. Cosine similarity method enables us to identify similarities between vectors as long as these words are present in the document. The formula can be seen in Eq. (3) and an overview of the cosine similarity algorithm process is described in Fig. 2.

$$idf_i \tag{1}$$

$$w_{i,j} = tf_{i,j} \times idf_i \tag{2}$$

$$cos(A,B) = \frac{A \cdot B}{|A||B|} = \frac{\sum_{i=1}^{n} A_i \, x \, B_i}{\sqrt{\sum_{i=1}^{n}(A_i)^2} \, x \, \sqrt{\sum_{i=1}^{n}(AB_i)^2}} \tag{3}$$

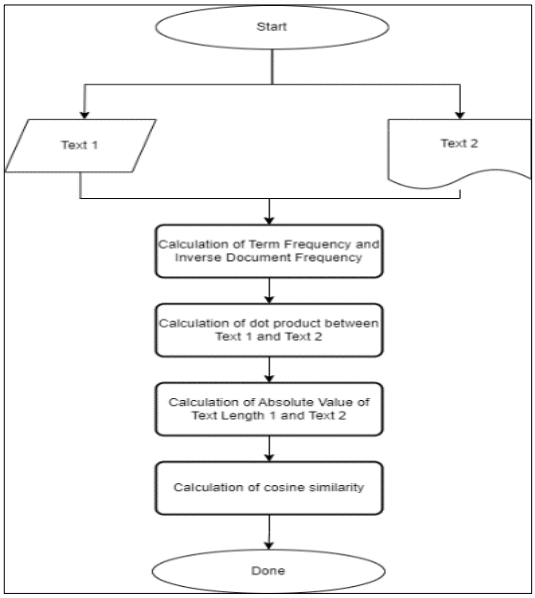


Fig. 2. Cosine similarity process.

b) Use case diagram: The Use Case diagram is a design tool used to address business problems in research. Fig. 3 displays the Use Case Diagram for the drafting module in the Omnibus Law Information System.

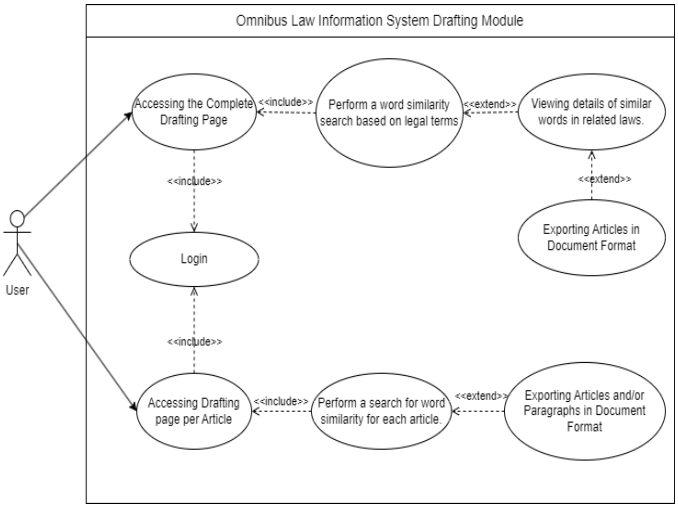


Fig. 3. Use case diagram of the drafting module of the omnibus law information system.

Legal drafters are the users of the Omnibus Law Information System. Explanation of Fig. 3 regarding use of use cases can be shown in Table I.

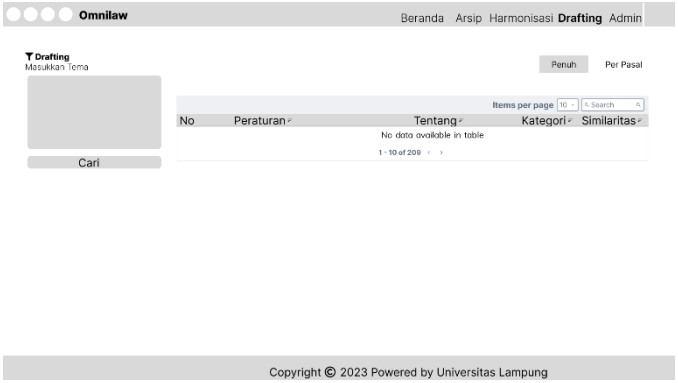
TABLE I. DESCRIPTION OF USE CASE DIAGRAM

Actor	Use Case	Description
User	Accessing the complete drafting page	User can view the full design page
	Perform a word similarity search base on legal terms	Users can search for similarities between words based on legal terms.
	Viewing details of similar word in related laws.	Users can view details of word similarities in related laws.
	Exporting articles in document format	Users can export articles in document format
	Accessing drafting page per article	Users can view the draft page per article
	Perform a search for word similarity for each article	Users can search for word similarities per article
	Exporting articles and/or paragraphs in document format	Users can export articles and/or paragraphs in document format

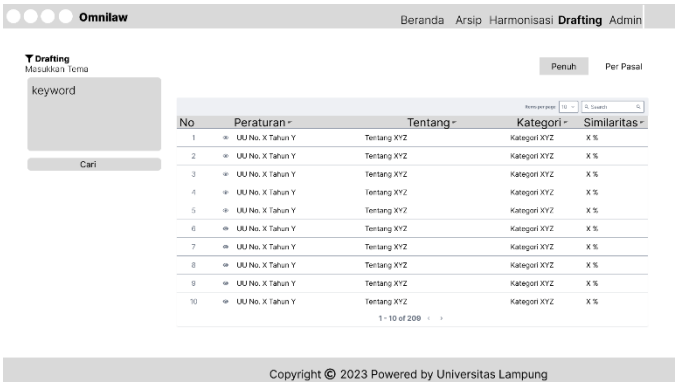
c) User interface design: User interface design is carried out as a reference for system display design. Fig. 4 shows the display design consists of five display pages that follow the design of each display.

3) Implementation phase: The implementation phase is the phase in which the results of the analysis and design are applied. This stage involves implementing algorithm, activity diagram and user interface design.

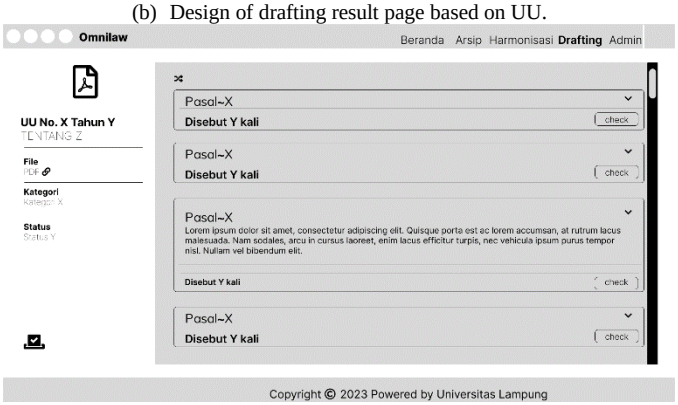
a) Algorithm: Cosine similarity is the algorithm used in the development of the Drafting module. Cosine similarity uses the cosine as the basis for calculation, which means that if this means that if two words or documents have a calculation result with an angle of 0 degrees, they have 100% similarity. FastAPI is used to implement the algorithm. Construction of FastAPI uses standards from OpenAPI known as Swagger and JSON Schema [21]. Interactive documentation can be used to test the response of any feature provided by Swagger UI. Fig. 5 shows the documentation page provided by Swagger UI.



(a) Design of drafting home page based on UU.

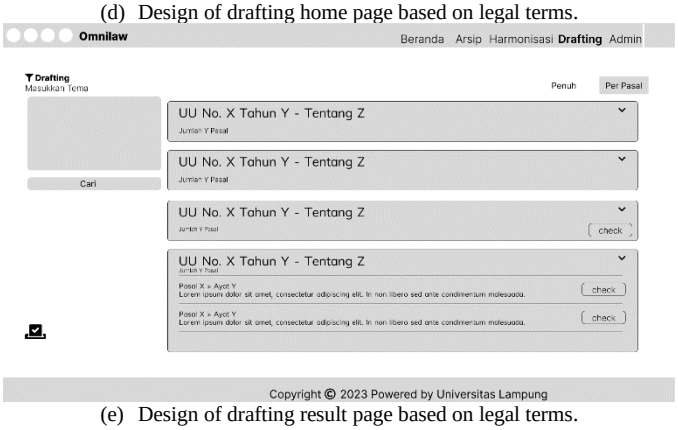


(b) Design of drafting result page based on UU.



(c) Design of drafting result detail page based on UU.

(d) Design of drafting home page based on legal terms.



(e) Design of drafting result page based on legal terms.

Fig. 4. User interface design (a) Drafting home page base on UU (b) Result drafting page based on UU (c) Result detail page base on UU (d) Drafting home page on legal terms (e) Design of drafting result page based on legal terms.

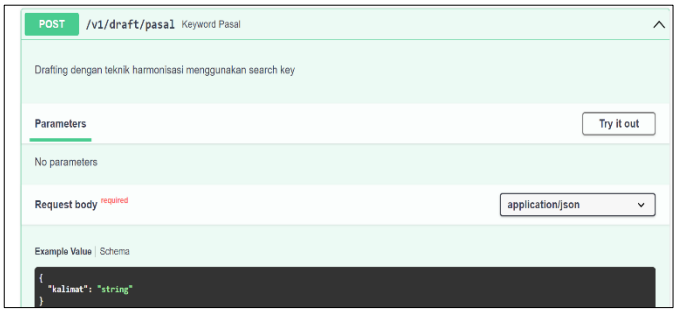
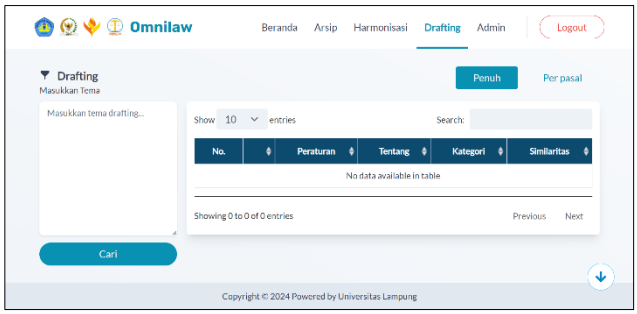
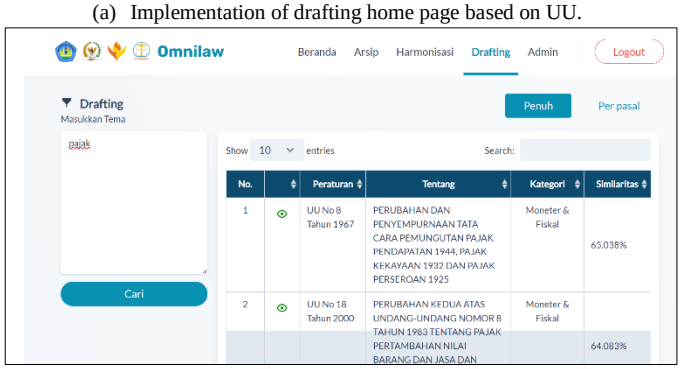


Fig. 5. Swagger UI documentation.

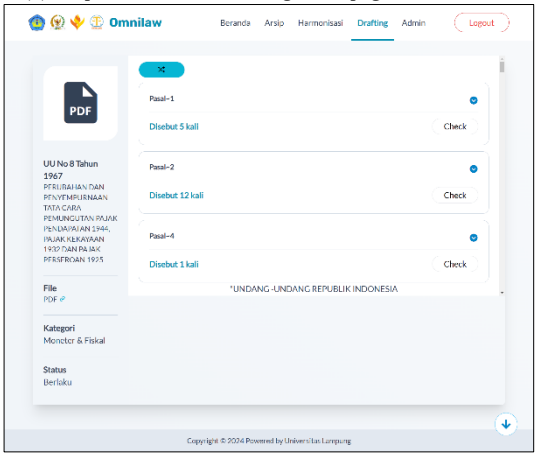
b) Laravel: Laravel is a PHP framework used as a front end that adheres to the Model - View - Controller (MVC) concept [22]. The functions included in Laravel are the implementation of use cases, activity diagrams and user interfaces. There are seven use cases implemented with Laravel as shown in Fig. 6.



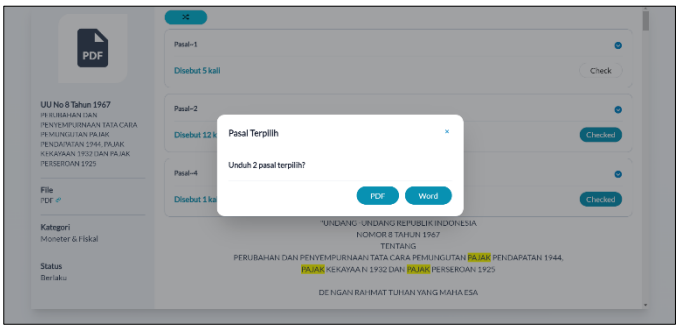
(a) Implementation of drafting home page based on UU.



(b) Implementation of drafting result page based on UU.



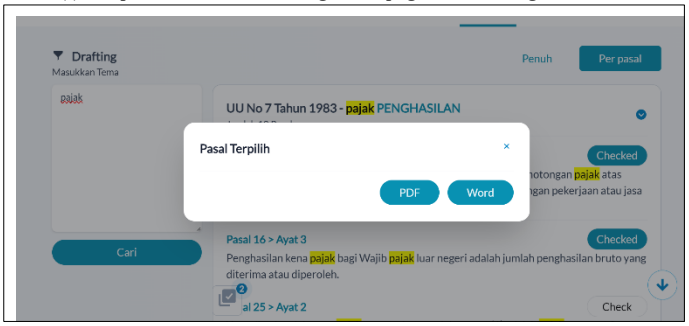
(c) Implementation of drafting result detail based on UU.



(d) Implementation of exporting result in document format based on UU.

(e) Implementation of drafting home page based on legal terms.

(f) Implementation of drafting result page based on legal terms.



(g) Implementation of exporting result in document format based on legal terms.

Fig. 6. Implementation of (a) Drafting home page based on UU (b) Drafting result page based on UU (c) Drafting result detail based on UU (d) Exporting result based on UU (e) Drafting home page based on legal terms (f) Drafting result page based on legal terms (g) Exporting result in based on legal terms.

c) Information systems architecture: The Omnibus Law information system architecture uses a full stack architecture. Full stack architecture divides an information system into two parts, namely the front-end or user display and the back-end or screen behind the process. Fig. 7 shows the omnibus law information system architecture.

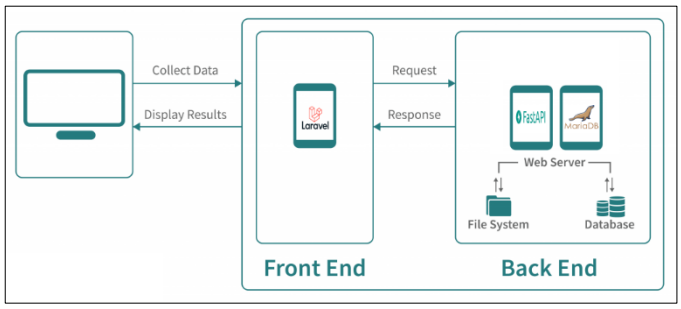


Fig. 7. Information system architecture.

4) Testing phase: Testing is the stage of validation and verification of each function included in the information system to ensure that it works correctly. Testing the algorithm using the confusion matrix. Confusion Matrix is used to see the predictive ability of the algorithm. Confusion matrix is a predictive analysis tool that compares actual values with predicted model values [23]. From the results of the confusion matrix calculation performed, a recall of 90.10%, accuracy of 56%, precision of 30% and F1 score of 45% was obtained.

IV. Conclusion

Based on the results of research on the implementation of cosine similarity in the Omnibus Law Information System, the authors draw the following conclusions: (1) The degree of similarity between keywords for drafting laws and existing laws can be measured using the cosine similarity algorithm and TF-IDF, with recall 90.10%. (2) Implementation of the similarity model between keywords and existing UU so that it can be used by users (legal drafters) using FastAPI as a back-end and Laravel as a front-end, so that it speeds up user work in seeing the similarity between keywords and existing UU.

For future works, Inclusion of datasets containing regulations with a different hierarchy from UU, such as Perpres and others, would be more beneficial for lawmakers. This would aid in determining which parts are attached to the new laws, without causing any overlap in Indonesian regulations.

References

[1] W. Setiadi, "Simplifikasi Regulasi dengan Menggunakan Metode Pendekatan Omnibus Law," J. Rechts Vinding Media Pembin. Huk. Nas., vol. 9, no. 1, p. 39, 2020.

[2] D. Sadiawati, Strategi Nasional Reformasi Regulasi: Mewujudkan Regulasi yang Tertib dan Sederhana. Jakarta: Kementerian Perencanaan dan Pembangunan Nasional/ Bappenas, 2015.

[3] I. Mayasari, "Kebijakan Reformasi Regulasi Melalui Implementasi Omnibus Law Di Indonesia," Ima Mayasari, vol. 9, no. 1, 2020.

[4] M. Azhar, "Omnibus Law sebagai Solusi Hiperregulasi Menuju Sonkronisasi Peraturan Per-Undang-undangan di Indonesia," Adm. Law Gov. J., vol. 2, no. 1, pp. 170–178, 2019, doi: 10.14710/ihis.v%vi%i.6671.

[5] P. Xia, L. Zhang, and F. Li, "Learning similarity with cosine similarity ensemble," Inf. Sci. (Ny)., vol. 307, pp. 39–52, 2015, doi: 10.1016/j.ins.2015.02.024.

[6] A. R. Lahitani, A. E. Permanasari, and N. A. Setiawan, "Cosine similarity to determine similarity measure: Study case in online essay assessment," Proc. 2016 4th Int. Conf. Cyber IT Serv. Manag. CITSM 2016, 2016, doi: 10.1109/CITSM.2016.7577578.

[7] D. Thenmozhi, K. Kannan, and C. Aravindan, "A Text Similarity Approach for Precedence Retrieval from Legal Documents," CEUR Workshop Proc., vol. 2036, no. December, pp. 90–91, 2017, [Online]. Available: https://ceur-ws.org/Vol-2036/T3-9.pdf

[8] T. Novotná, "Document Similarity of Czech Supreme Court Decisions," Masaryk Univ. J. Law Technol., vol. 14, no. 1, pp. 105–122, 2020, doi: 10.5817/MUJLT2020-1-5.

[9] S. Renjit and S. M. Idicula, "CUSAT NLP@AILA-FIRE2019: Similarity in legal texts using document level embeddings," CEUR Workshop Proc., vol. 2517, no. December, pp. 25–30, 2019.

[10] N. Tang and A. Engelbrecht, Data Clustering. London: Intech, 2022. [Online]. Available: http://dx.doi.org/10.1039/C7RA00172J%0Ahttps://www.intechopen.com/books/advanced-biometric-technologies/liveness-detection-in-biometrics%0Ahttp://dx.doi.org/10.1016/j.colsurfa.2011.12.014

[11] B. K. Triwijoyo and K. Kartarina, "Analysis of Document Clustering based on Cosine Similarity and K-Main Algorithms," J. Inf. Syst. Informatics, vol. 1, no. 2, pp. 164–177, 2019, doi: 10.33557/journalisi.v1i2.18.

[12] F. R. Lumbanraja, R. E. Pramswary, and Aristoteles, "Classification of Cracked Concrete Images Using Convolutional Neural Algorithm," AIP Conf. Proc., vol. 2563, no. October, 2022, doi: 10.1063/5.0103114.

[13] M. Silvi Lydia, S. Dara Fadillah, and M. Huda, "Perbandingan Metode Klaster dan Preprocessing Untuk Dokumen Berbahasa Indonesia," pdfs.semanticscholar.org, 2018, doi: 10.17529/jre.v14i1.9027.

[14] D. Alita and A. Rahman, "Pendeteksian Sarkasme pada Proses Analisis Sentimen Menggunakan Random Forest Classifier," jurnal.fmipa.unila.ac.id, vol. 8, 2020, Accessed: May 27, 2022. [Online]. Available: https://jurnal.fmipa.unila.ac.id/komputasi/article/view/2615

[15] L. Hermawan, M. B. Ismiati, J. Bangau, N. 60, and M. Charitas, "Pembelajaran text preprocessing berbasis simulator untuk mata kuliah information retrieval," 156.67.218.228, vol. 17, no. 2, pp. 188–199, 2020, Accessed: May 27, 2022. [Online]. Available: https://156.67.218.228/index.php/transformatika/article/view/1705

[16] E. Y. Anggraeni and R. Irviani, Pengantar Sistem Informasi, 1st ed. Yogyakarta: CV. Andi Offset, 2017. Accessed: May 24, 2022. [Online]. Available: https://books.google.co.id/books?id=8VNLDwAAQBAJ

[17] C. Samuel Rajiv and P. K, "A Platform to Help in Generating Code for Machine Learning and Data Science Projects," SSRN Electron. J., Feb. 2022, doi: 10.2139/ssrn.4033072.

[18] Rudy, Aristoteles, and R. C. Kurniawan, Model Omnilaw: Solusi Pemecahan Masalah Penyederhanaan Legislasi dalam Rangka Pembangunan Hukum. Bandarlampung: Pusaka Media, 2021.

[19] P. K. Hima Pandalu, "Pencarian dan Perankingan Obat Tradisional Berdasarkan Gejala Penyakit Menggunakan Metode Cosine Similarity," Universitas Islam Negeri Maulana Malik Ibrahim Malang, 2014.

[20] Y. Januzaj and A. Luma, "Cosine Similarity – A Computing Approach to Match Similarity Between Higher Education Programs and Job Market Demands Based on Maximum Number of Common Words," Int. J. Emerg. Technol. Learn., vol. 17, no. 12, pp. 258–268, 2022, doi: 10.3991/ijet.v17i12.30375.

[21] M. Malhotra, "Internship Report At Paxcom India Pvt. Ltd.," Himachal Pradesh, 2022.

[22] S. Dwiyatno, E. Rachmat, A. P. Sari, and O. Gustiawan, "Implementasi Virtualisasi Server Berbasis Docker Container," PROSISKO J. Pengemb. Ris. dan Obs. Sist. Komput., vol. 7, no. 2, pp. 165–175, 2020, doi: 10.30656/prosisko.v7i2.2520.

[23] Aristoteles, A. Syarif, Sutyarso, F. R. Lumbanraja, and A. Hidayatullah, "Identification of Human Sperm based on Morphology Using the You Only Look Once Version 4 Algorithm," Int. J. Adv. Comput. Sci. Appl., vol. 13, no. 7, pp. 424–431, 2022, doi: 10.14569/IJACSA.2022.0130752.