Adapting Plagiarism Detection Techniques for Citation Identification in Legal Texts

Oleksandr Fetkovych[1], Peter Gurský[1], Dávid Varga[1] and Zoltán Szoplák[1]

[1]Institute of Computer Science, Faculty of Science, Pavol Jozef Šafárik University in Košice, Jesenná 5, 040 01 Košice, Slovakia

Abstract
Our work aims to create a tool for identifying citations among legal documents. In this paper, we present a method for detecting citations of sections of laws in court decisions using an adapted plagiarism detection technique. Our method uses the full-text database Elasticsearch to select the candidates. Since we are looking for citations of laws within court decisions, which might change over time, we must also consider the laws' amendments. We evaluated our approach on a sample of manually annotated court decisions.

Keywords
legal texts, anti-plagiarism system, citations, court decisions

1. Introduction

In our project, we aim to develop a system for analyzing relationships among legal texts. Recognizing references to other legal documents is essential for understanding connections within legal documentation. We aim to build a system that identifies which texts refer to other texts and, vice versa, which are derived from others, along with the nature and significance of these relationships. Currently, no available system records such relationships. Establishing one could improve legal analysis and research, making the legal system more efficient.

At the project's current stage, we focus on court decisions, which typically rely on the wording of legal paragraphs from laws and regulations, as well as previous decisions from other courts in similar or related matters. In their decisions, judges usually cite specific paragraph numbers of laws and case file numbers to which they refer. However, these references can sometimes be unreliable. Many cited laws have only a marginal connection to the text, and sometimes, sections of laws are cited without explicit reference to paragraph numbers. This paper introduces a method for detecting citations in the sense of a quoted text from another legal text. The cited legal texts have typically stronger connection with the original text. Therefore, quotations can be an essential part of relationship analysis.

The basic idea of our approach was to utilize a method from the well-researched area of plagiarism detection. However, plagiarism detection methods pursue slightly different goals. There are several important differences between detecting plagiarism and citations of legal texts:

- While a plagiarist tries to conceal their plagiarism by changing sentence formulations, using synonyms or other methods, a lawyer aims to quote another legal text as accurately as possible. Therefore, in our case, we can omit algorithms that detect word matches based on semantic similarity. On the other side, legal citations often contain typos, omissions of parts of sentences, or the insertion of phrases into the quoted texts.
- A standard text is usually considered plagiarized only when there is significant textual similarity over several sentences, typically spanning several pages or even paragraphs. On the other hand, legal text citations can be concise, sometimes limited to a single sentence from a law. Of course, this only holds true on certain occasions and cannot be applied as a general rule, especially in citations of lower court decisions in appellate decisions, which can involve large text sections.
- When citing laws in court decisions, it is essential to consider that laws are dynamic documents that change over time through amendments. A law cited in a decision typically refers to its most recent version relative to the decision's release date, although this may not always be the rule. The cited law text definitely does not come from a version that became effective after the decision date.

This paper presents our methods for detecting citations in legal documents. In addition to the specifics mentioned

ITAT 2024 Information Technologies – Applications and Theory 2024, September 20–24, 2024, Drienica, Slovakia

*Corresponding author.

†These authors contributed equally.

oleksandr.fetkovych@student.upjs.sk (O. Fetkovych); peter.gursky@upjs.sk (P. Gurský); david.varga@student.upjs.sk (D. Varga); zoltan.szoplak@student.upjs.sk (Z. Szoplák)
0009-0004-4672-4060 (O. Fetkovych); 0000-0002-4744-7390 (P. Gurský); 0000-0002-3176-8106 (D. Varga); 0000-0003-1823-0536 (Z. Szoplák)

above, our approach also takes into account the speed of citation detection. This paper is organized as follows:

- Section 2 introduces those anti-plagiarism systems that inspired the design of our method the most.
- Section 3 briefly presents our dataset.
- Section 4 details our citation detection methods.
- Section 5 describes the results of testing the effectiveness of our methods.
- Section 6 summarizes the results and the applicability of our methods but also discusses potential future research directions in the field of citation detection.

2. Related work

When defining our approach, we have taken inspiration from other works about creating anti-plagiarism systems. We are specifically referring to the particular intricacies of detecting and extracting legal citations mentioned in the previous section.

The Anti-plagiarism system Copyfind [1] employs a hashing function to encode all input documents, such that every word turns out to be a 32-bit hash code. This system makes pairwise comparisons between all documents, where cursors move over lists of their hash codes, searching for an identical pair. However, as in our case, working with many documents is very time-consuming. In addition to the aforementioned drawbacks, when working with long sentences or more structurally complicated texts, resolving which phrases were similar was problematic and did not fit our needs.

Stamatatos et al. [2] introduced a method for detecting plagiarism in document collections based on stop word n-grams. The authors assumed that stop word sequences reveal syntactic patterns in the document structure, which can be used to detect plagiarism. This makes it useful in cases where other methods based on context might fail to detect plagiarism. Additionally, the method can be executed quickly because it uses a low number of stop words, reducing processing time. However, such a method also has its disadvantages. It cannot detect multiple instances of plagiarism in a scrutinized document and particularly struggles with short matching fragments between the source and the inspected documents.

Abdi et al. [3] proposed a method based on syntactic and semantic features, improving accuracy and efficiency. The authors use the sentence approach for data preprocessing. They divide the texts into sentences and work with each sentence separately. However, this method may require more computing resources and processing time than other methods due to its reliance on syntactic and semantic features. Moreover, it might perform poorly when plagiarized fragments have similar syntactic structures, but different semantics.

The method proposed by Vani and Gupta [4] employs a vector space model (VSM) alongside syntactic feature extraction using shallow NLP techniques, such as Part-of-Speech (POS) tags, to represent documents as vectors. This approach enhances plagiarism detection by analyzing both syntactic and semantic properties of texts. The method classifies these features using algorithms such as Naïve Bayes, Support Vector Machine, and Decision Trees. However, since it focuses on whole documents rather than individual sentences, its applicability may be limited in certain scenarios.

The method of Altheneyan and Menai [5] includes several NLP features such as stop word removal, punctuation removal, and tokenization to prepare the text for comparison. The paragraph-level comparison step compares suspect and source documents at the paragraph level, while the sentence-level comparison step looks for common unigrams between sentences. The SVM classifier then checks detected instances of plagiarism, and consecutive sections are merged in a post-processing step. This method is, therefore, capable of detecting plagiarism in obfuscated text. The method relies heavily on the number of common unigrams between sentences and may not effectively reveal more complex forms of plagiarism that do not rely on word frequency.

Yalcin et al. [6] introduce an external plagiarism detection system using n-grams of POS tags and semantic vector representations of words. The preprocessing involves sentence segmentation, tokenization, and assigning one of 45 POS tags. The system generates POS n-grams (POSNG), indexed at the sentence level using the Lucene search engine [1], to reflect syntactic text properties. Using the full-text search engine for obtaining candidate documents was the main inspiration in our proposed method.

Matching POSNG tags between a suspicious document and the source indicates potential plagiarism. Searches for candidate sentences involve querying n-grams through Lucene, seeking the highest match scores. The method employs two decision techniques for identifying plagiarism: direct syntactic comparison (POSNG_{PD}) and syntactic plus semantic analysis ($\text{POSNG}_{\text{PD+SSBS}}$), where the latter assesses semantic similarities using the Word2Vec model [7].

The authors claim superior accuracy of their method, although acknowledging slower processing due to extensive data handling.

[1]Apache Lucene, a high-performance text search engine, available at https://lucene.apache. org/core/

3. Dataset

To check the correctness of our proposed methods, we used two data sets: a set of all laws in the Slovak Republic including its history and a randomly chosen subset of court decisions.

3.1. Law articles

The Slovak government publishes the collection of laws on the Slov-Lex portal [8] in HTML format both online and in a ZIP archive. Obtaining all laws with corresponding articles and historical changes was quite a complex process. First, we needed to distinguish between the original laws, their amendments and other kinds of documents in the archive. Typically, original laws contained amendments of other laws, therefore, we needed to identify which parts were relevant. After extracting necessary data, we converted them into the JSON format for structured storage and easier processing.

Figure 1 below illustrates the JSON object representation for Section 113 of the Criminal Code. This object comprises several key attributes:

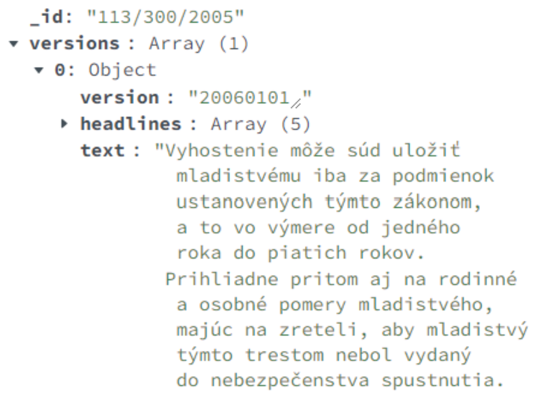
- `_id`: A unique identifier for a law section, which combines the articles section number, in our case 113, with an identifier from the Collection of Laws, e.g., 300/2005 for the Criminal Code.
- `versions`: Contains an array of objects representing all historical versions of the article.
- `version`: Indicates the effective date from which the law's current version applies.
- `text`: Provides the actual text of the law's paragraph for the pertinent version.
- `headlines`: These are headings within the structure of the entire law leading to the specific section. However, this attribute was not utilized for our analysis.

3.2. Court decisions

This work analyzes a subset of court decisions published on the Open Data website of the Ministry of Justice of the Slovak Republic [9].

The court decisions are structured as JSON objects that include details like the court type, the court name, the judge's name, and the legal domain. Each object also features a `document_fulltext` attribute, which holds the anonymized text of the court decision on which we focused primarily.

Also, the attribute `decision_issue_date` signifies when the court decision was issued. This attribute is significant because it indicates the specific date when the judge wrote the decision. With this date, we can avoid reviewing citations of laws enacted after the judgment since the judge would not have been able to reference laws that did not exist then.



```
  _id: "113/300/2005"
▾ versions : Array (1)
  ▾ 0: Object
      version : "20060101"
    ▸ headlines : Array (5)
      text : "Vyhostenie môže súd uložiť
              mladistvému iba za podmienok
              ustanovených týmto zákonom,
              a to vo výmere od jedného
              roka do piatich rokov.
             Prihliadne pritom aj na rodinné
              a osobné pomery mladistvého,
              majúc na zreteli, aby mladistvý
              týmto trestom nebol vydaný
              do nebezpečenstva spustnutia.
```

Figure 1: Example of an object (article §113 of the Criminal Law).

4. Methods

Our initial approach to detecting citations involved a comparison the text from court decisions against all legal paragraphs to identify citations. Given the extensive dataset and the complexity of comparison algorithms, this process proved to be very time-consuming.

Consequently, we transitioned to a more sophisticated solution utilizing Elasticsearch [10]. Elasticsearch employs advanced techniques for rapid searching through extensive text datasets. Its key advantages include storing data as JSON objects, scalability, and full-text search capabilities. Additionally, Elasticsearch leverages an inverted index, enhancing the efficiency of search operations. Its flexible and customizable text analysis methods convert text into structured data optimized for effective storage and retrieval, thus significantly improving the system's performance and responsiveness.

This new approach consists of several integral stages, each designed to optimize the processing and analysis of legal texts. Here are the main components of our new method, which we will discuss in detail in upcoming sections:

- **Data Indexing:** We initiate the process by indexing the data of legal paragraphs into an Elasticsearch index. This step organizes the data efficiently, setting the foundation for rapid retrieval and detailed analysis.
- **Finding candidate documents:** A specific query is crafted for Elasticsearch to sift through the indexed data and extract candidate legal paragraphs. Finding candidates for deeper inspection narrows down the scope significantly.

- **Texts matches:** We employ a custom algorithm to search common parts of the original and candidate document. This phase analyses the presence of citations within the court decisions and evaluates their relevance.
- **Decision-Making:** After verifying the citations, we initiate a decision-making process. This step determines whether the legal paragraphs were cited accurately in the texts under review.

We validated this methodology by conducting reviews and analyses of its performance on a real court decision dataset, as will be discussed in Section 5. The results confirm the efficacy and efficiency of our approach in handling complex legal texts.

4.1. The process of data indexing

Due to constant law amendments, most legal paragraphs have multiple versions over time, as we demonstrated in Section 3.1.

If we index each paragraph with multiple versions as a single document, then during the search for similar law paragraphs for a court decision, Elasticsearch will compare the text of the court decision with the texts of all versions of the paragraph, leading to incorrect results. The paragraph with more versions can be incorrectly returned because of many common texts, which are in fact the same texts. Furthermore, the JSON document returned by Elasticsearch can contain matching texts in irrelevant versions while there is no match in the relevant one. To avoid this, we used a simple script to split each law paragraph into separate documents, each containing only one unique version of the paragraph while retaining the original paragraph ID.

This approach allows us to filter relevant paragraph versions based on the creation date of court decisions. After preparing the dataset for indexing, we created an appropriate mapping [11] for an efficient data indexing process. The mapping in Elasticsearch defines how each field in the document will be indexed, including text analysis rules, data types, and storage options. This mapping is crucial as it ensures that Elasticsearch can efficiently store and retrieve data, optimize search performance, and correctly handle our documents' nested structures.

The mapping begins by establishing a custom analyzer that simplifies and standardizes text. This text is then uniformly processed to ensure consistency across all documents, e.g., lowercasing and ascii folding. The mapping specifies different types of data fields, including identifiers for precise searches and versioning information to track document updates. Each document version is analyzed using the same method to maintain uniformity in data handling. The result of the indexing is an internal Elasticsearch structure.

4.2. Finding candidate documents

This step reduces the number of source documents (legal paragraphs) that are then compared in detail with the text of the court decision. Using the Elasticsearch index, we created a query based on the full text of the court decision. Elasticsearch searches for documents with similar text and returns an ordered list of source documents ranked by relevance.

Key features of our query include:

- The similarity query contains the whole text of the decision.
- The query will return the top 30 most similar documents to streamline further processing.
- Comparison is then performed based on the court decision text.
- Each of the 30 documents will have a unique law article ID and contain only one version, valid at the time of the court decision.

Since the query is extensive and complex, we omit its detailed description in this paper. As a result, we will get 30 candidate documents, which we will use to search for specific quotes. The number 30, a heuristic parameter, is chosen to provide us with sufficient results. It is important to note, however, that this parameter is not fixed and can be adjusted to suit the needs of our research.

4.3. Searching for common matching texts

Elasticsearch does not return positions of matching texts. Since it is important to us what texts match and where the matches are located, we need to find the matching places in both documents.

4.3.1. Finding common sequences using the Needleman-Wunsch algorithm

To find matching text sequences, we utilized the well-known Needleman-Wunsch algorithm (NW) of dynamic programming [12] for the longest common subsequence search. The core principle of the algorithm is as follows:

- **Text preprocessing:** Texts are stripped of punctuation and short words, simplifying subsequent analysis and reducing noise.
- **Matrix initialization:** A two-dimensional array M(dp matrix) is created where each element $M(i,$ $j)$ stores the length of the longest common subsequence between the first i words of text1 and the first j words of text2.

Part of the text from the legal paragraph with word offset 10: "...v jeho prospech odvolanie, poškodený, zúčastnená osoba, ako aj prokurátor sa môžu výslovným vyhlásením vzdať ... Osoba, ktorá je oprávnená podať ..."
Part of the text from the court decision with word offset 100 : "...v jeho prospech obvolanie, poškodený, ako aj prokurátor sa môžu výslovným vyhlásením vzdať ... osobe, ktorá je oprávnená podať ... ako prokurátor môže ..."

jeho$_{10}$ prospech$_{11}$ odvolanie$_{12}$ poskodeny$_{13}$ ... $\boxed{\text{ako}_{16}\text{ prokurator}_{17}\text{ mozu}_{18}}$ vyslovnym$_{19}$ vyhlasenim$_{20}$ vzdat$_{21}$ ...
osoba$_{35}$ ktora$_{36}$ opravnena$_{37}$ podat$_{38}$ ...

jeho$_{100}$ prospech$_{101}$ obvolanie$_{102}$ poskodeny$_{103}$ ako$_{104}$ prokurator$_{105}$ mozu$_{106}$ vyslovnym$_{107}$ vyhlasenim$_{108}$ vzdat$_{109}$ ...
osobe$_{120}$ ktora$_{121}$ opravnena$_{122}$ podat$_{123}$ ... $\boxed{\text{ako}_{250}\text{ prokurator}_{251}\text{ moze}_{252}}$ ...

Arrays of matching word 3-grams:
[10, 11, 16, 16, 17, 18, 19, 35, 36]

[100, 101, 104, 250, 105, 106, 107, 120, 121]

Arrays of matching sequencies:
[[10, 11], [16, 17, 18, 19], [16], [35, 36]]

[[100, 101], [104, 105, 106, 107], [250], [120, 121]]

Figure 2: This is an example of how the original texts are preprocessed and matched. The sequences of word 3-grams highlighted with the same color or surrounded by a frame have a significant similarity (and thus considered equal), even with a typo in the word 'ob(d)volanie' and a declension in word 'osoba'. The indices of first words of matching 3-grams of words are stored in two arrays. The upper one corresponds to the 3-grams of the law article and the bottom one corresponds to the 3-grams of the court decision. Finally, the continuous sequences of word 3-gram indices are merged together, resulting in arrays of arrays.

- **Matrix filling:** Using dynamic programming, the matrix is filled with values based on word comparisons. If the words match, the value increases by one compared to the previous words. Otherwise, the maximum value from adjacent cells is selected.
- **Subsequence reconstruction:** Starting from the bottom-right element of the matrix, the subsequence itself is reconstructed by following the path that led to the maximum length. This is achieved by comparing the values in the matrix and selecting the path with the maximum value.

Since the sequence assembly is performed from the end, the results (arrays of indices) are reversed to be presented in the correct order.

4.3.2. Finding common sequences using 3-grams of words

Although the algorithm in Section 4.3.1 is fast, we found over time that it had trouble identifying matches in case of typos. This algorithm cannot return multiple matches of same texts, resulting in overlooking potentially longer citations that have words added or removed somewhere in the middle. Note that the problem of adding or removing words inside the citations is covered in Section 4.4. Therefore, we created an alternative approach that would eliminate these shortcomings.

The approach involves systematically comparing the text of a judicial decision with candidate paragraphs of laws. The process begins by taking both decision and paragraph texts and processing them to remove punctuation marks, numbers, and words shorter than three characters. This step helps to ensure that only meaningful content is considered, as shorter words typically lack semantic significance and can easily be replaced by synonyms. Then, we divide each text into 3-grams of words and calculate the Levenshtein [13] similarity coefficient for each pair of compared word 3-grams.

We chose the Levenshtein method for its efficiency in handling typos. Typos are often found in court decisions, but almost never in laws. By calculating the Levenshtein distance, we can quickly and accurately identify matches even in texts containing such errors.

The similarity coefficient (normalized insertion-deletion similarity) between two 3-grams of words is calculated using the Levenshtein method described in [14] according to the following formula:

$$\text{Similarity} = 1 - \frac{\text{Distance}}{\text{Length}_1 + \text{Length}_2}$$

Where:

- Similarity: similarity coefficient.
- Distance: Levenshtein distance between two 3-grams.
- Length_1: length of the first 3-gram.
- Length_2: length of the second 3-gram.

This coefficient reflects the normalized similarity between two 3-grams of words on a scale from 0 to 1, where 0 means complete dissimilarity and 1 means complete match. If similarity of compared 3-grams exceeds a threshold of 0.9, we consider the 3-grams to be equal. The indices of their positions in original texts are recorded in arrays: one for the 3-gram indices of the court decision and one for the law paragraph.

Next, the algorithm searches for increasing continuous sequences of indices, resulting in all common subsequences with at least 3 words between the two texts. The result is represented as an array of arrays of word 3-gram positions, as depicted in Figure 2.

4.4. Inserted and missed words in citations

The previous two methods are designed to find continuous sequences within analyzed texts. Sometimes, judges use extra words or miss some words from cited laws and thus do not form precisely continuous citations. This step takes into account these kinds of citations and merges them. However, we cannot merge similar 3-grams of words if they are too far apart, so we merge only those sequences that are at most ten words apart. Finally, the process returns two arrays that store sequences that are merged if possible, one associated with the law article and the other with the decision.

Below we provide an example, where we continue our example from Figure 2. Each input array contains subarrays with strictly increasing subsequences by 1. In Figure 3, the green color marks the last and first elements of adjacent arrays whose difference is less than ten positions in both texts; these will be merged into a single sub-array, as seen in Figure 4.

The red color marks the last and first elements of adjacent arrays whose difference is greater than ten words, indicating they will not be merged in the array for a court decision nor corresponding sub-arrays for the law article. We can see that the pair of subarrays <[16],[250]> do not merge with <[10,11], [100,101]>, because the distance between 101 and 250 is too big.

```
[[ 10,  11], [ 16,  17,  18,  19], [ 35,  36], [ 16]]

[[100, 101], [104, 105, 106, 107], [120, 121], [250]]
```

Figure 3: Indices of similar word 3-grams with connectivity conditions.

```
[[ 10,  11,  16,  17,  18,  19], [ 16], [ 35,  36]]

[[100, 101, 104, 105, 106, 107], [250] [120, 121]]
```

Figure 4: Merged sequences after applying connectivity condition.

4.5. Decision making

After the entire process, we obtain a set of arrays containing sequences of indices of words found in both texts. Next, we need to decide, which matching texts are real citations. Many times, even if there are matched texts, it is just a coincidence, not a real citation.

Actually, it is difficult to correctly identify real citations. Judges often do not put quoted text in quotation marks. Sometimes, even a relatively short text is a real citation and at the same time a longer text does not have to be a citation. Therefore, the following two approaches should be considered heuristics rather than informed decisions.

The decision-making process results in whether or not the law article is cited in the judicial decision. We decide whether a citation is present in the judicial decision based on the longest sequence length in input arrays.

We have tested the following two conditions

- The longest citation contains at least 7 words
- The longest citation covers at least 5% of the original law article

The first approach was chosen to minimize the risk of false positives arising from random or insignificant matches of short text segments. Such short citations are often too general to identify a specific legal provision uniquely and could lead to incorrect conclusions. This threshold allows us to increase the accuracy and reliability of the method.

The second approach prefers citations that cite a significant part of the law article. This approach suppresses the occurrence of false positives but, on the other hand, citations within large law articles can be skipped.

As a result of the last step, the identifiers `_id` and `version` of the legal articles, together with the cited positions, are returned.

5. Evaluation

We created two methods for citation search and two methods for decision-making. We used two decision-making

Table 1
Evaluation of methods with combination of decision making approach. We present true positives as TP, false negatives as FN, false positives as FP, precision, recall and an F1 Score metric.

Methods	TP	FN	FP	Precision	Recall	F1 Score
NW, absolute length	179	0	145	0.55	1.00	0.71
NW, percentage	73	106	0	1.00	0.40	0.58
3-grams, absolute length	174	5	37	0.82	0.97	0.89
3-grams, percentage	174	5	45	0.79	0.97	0.87

methods for each citation search method, resulting in four distinct methods in total. The implementation can be found on our GitHub repository [11]. Due to the many versions of laws, we were unable to compare our approach with other plagiarism detection systems.

We tested our methods on randomly chosen 100 court decisions and all laws of the Slovak Republic and summarized the results in Table 1.

In our study, we used the F1 Score to evaluate our method for identifying law citations in court decisions. We chose this metric because it balances precision and recall, making it a reliable measure of our method's accuracy. The F1 Score helps ensure that we correctly identify real citations while reducing mistakes, which is crucial for trustworthy legal analysis.

While achieving perfect recall, the *NW, absolute length* method suffers from low precision, resulting in numerous false positives and an F1 Score of 0.71.

The *NW, percentage* method offers perfect precision but low recall, leading to an F1 Score of 0.58. It identifies citations accurately but misses a large number of citations.

The *3-grams, absolute length* method shows a balanced performance with both high precision and recall, yielding an F1 Score of 0.89, indicating effective citation detection.

The *3-grams, percentage* method also performs well, with slightly lower precision and an F1 Score of 0.87 compared to its absolute length counterpart. Overall, the 3-gram methods outperform the NW methods in balancing precision and recall, suggesting they are more suitable for nuanced detection of legal citations in court decisions.

When we compare these results, we can see, that Needleman-Wunsch algorithm can only search for exact matches; it often fails to detect long citations and typically only detects smaller parts of citations. As a result, it often happens that the longest citation does not reach 5% of the content of the paragraph of the law. On the other hand, NW method does not have the restriction of at least three consecutive matching words forming a 3-gram. The consequence is that the method for dealing with inserted and missing words (Section 4.4) may mistakenly evaluate as citations close matches of unigrams and bigrams up to a total length exceeding the threshold of 7 words. Therefore the *NW, absolute length* method identifies many false positives.

6. Conclusion

In this article, we presented our methods for searching for citations in legal texts. Although at first glance it looks like a classic plagiarism detection task, citations in legal texts have their specifics, which we listed in Section 1. We focused on finding citations of laws in court decisions. We presented a total of 4 methods to find citations and compared them on a dataset of 100 random judicial decisions.

In the future, we would like to explore other approaches to the decision-making process, for example, involving the semantic proximity of the court decision and the cited law.

Another goal is to examine the search for citations among judicial decisions. We will experiment with replacing the Needleman-Wunsh algorithm with the Smith–Waterman [15] one as it should be more suitable for finding local alignments such as law citations within larger, potentially dissimilar texts. For the purposes of faster detection for real-time tasks, we also plan to test the suitability of the BLAST algorithm [16] applied to such a task with natural language text instead of biological sequence identification.

Building on our previous research [17], which involved extracting references to laws, we aim to refine how relationships are weighted within legal texts. When a law is both referenced and cited within a ruling, it underscores its substantial influence. We currently use references to law paragraphs to extract keyphrases. In the future, we plan to explore assigning greater weight to phrases from highly valued law paragraphs to enhance our keyphrase extraction method. In our future work, we also aim to test whether removing stop words instead of short words improves our performance.

Judges sometimes omit specific letters, sections, or even laws' names, yet may still cite the text directly. Identifying these citations helps accurately pinpoint the relevant law paragraph, further enhancing the precision of our law reference extraction.

Acknowledgments

The Slovak Research and Development Agency supported this work under contract No. APVV-21-0336 Analysis of Court Decisions by Methods of Artificial Intelligence. Pavol Jozef Šafárik University in Košice supported this work with the internal project at vvgs-2023-2547 Legal Text Analysis Using Computer Linguistics. This article was also supported by the Scientific Grant Agency of the Ministry of Education, Science, Research and Sport of the Slovak Republic under contract VEGA 1/0645/22 entitled Proposal of Novel Methods in the Field of Formal Concept Analysis and Their Application.

References

[1] L. Bloomfield, WCopyfind, University of Virginia. Available at URL: http://plagiarism. bloomfieldmedia. com/wordpress/software/wcopyfind/. (2016).

[2] E. Stamatatos, Plagiarism detection using stopword n-grams, Journal of the American Society for Information Science and Technology 62 (2011) 2512–2527.

[3] A. Abdi, N. Idris, R. M. Alguliyev, R. M. Aliguliyev, Pdlk: Plagiarism detection using linguistic knowledge, Expert Systems with Applications 42 (2015) 8936–8946.

[4] K. Vani, D. Gupta, Text plagiarism classification using syntax based linguistic features, Expert Systems with Applications 88 (2017) 448–464.

[5] A. S. Altheneyan, M. E. B. Menai, Automatic plagiarism detection in obfuscated text, Pattern Analysis and Applications 23 (2020) 1627–1650.

[6] K. Yalcin, I. Cicekli, G. Ercan, An external plagiarism detection system based on part-of-speech (POS) tag n-grams and word embedding, Expert Systems with Applications 197 (2022) 116677.

[7] T. Mikolov, K. Chen, G. Corrado, J. Dean, Efficient Estimation of Word Representations in Vector Space, 2013. arXiv:1301.3781.

[8] Slov-Lex, 2022. URL: https://www.slov-lex.sk/vyhladavanie-pravnych-predpisov.

[9] Rozvoj elektronických služieb súdnictva (RESS) (2016). URL: https://obcan.justice.sk/.

[10] Elastic, Elasticsearch, 2024. URL: https://www.elastic.co/elasticsearch/.

[11] O. Fetkovych, P. Gurský, Revealing implicit legal phrases in court decisions, https://github.com/vargadavid304/legal_citations, 2024. GitHub repository.

[12] S. B. Needleman, C. D. Wunsch, A general method applicable to the search for similarities in the amino acid sequence of two proteins, Journal of molecular biology 48 (1970) 443–453.

[13] V. I. Levenshtein, et al., Binary codes capable of correcting deletions, insertions, and reversals, in: Soviet physics doklady, volume 10, Soviet Union, 1966, pp. 707–710.

[14] M. Bachmann, python-levenshtein, 2021. URL: https://rapidfuzz.github.io/Levenshtein.

[15] R. Mott, Smith–Waterman Algorithm, 2005. doi:10.1038/npg.els.0005263.

[16] S. Altschul, BLAST Algorithm, 2005. doi:10.1038/npg.els.0005253.

[17] D. Varga, M. Gojdič, Z. Szoplák, P. Gurský, Horvát, S. Krajči, L. Antoni, Extraction of legal references from court decisions., in: ITAT, 2023, pp. 89–95.