Преглед НЦД 37 (2020), 46–51

Aleksandar Veljković
Faculty of Mathematics, University of Belgrade, Serbia

ALGORITHM FOR DOCUMENT AUTHORSHIP IDENTIFICATION AND PLAGIARISM EVALUATION BASED ON GENERALIZED SUFFIX TREE

Abstract. Identifying an author of an anonymous text document is an important problem when dealing with historical data. As authors have their own characteristic writing styles, expressed through specific phrases, sentence constructions or word choices, their text documents incorporate the style and create implicit connection with the author. This paper proposes an approach for identification of authors of the anonymous documents, based on generalized suffix tree data structure and defined similarity score, suitable for analysis of digitized historical text documents. The following method can also be used for detecting and evaluating plagiarism, where the document author is known, but the document shows a high similarity with documents from another author.

1. Introduction

When an anonymous historical text document is discovered and digitized, finding an author of the document is not a trivial task. For solving this problem, the base requirement is to provide a database of text documents from potential authors, against which the new document can be compared. Second requirement is defining a similarity scoring function which will evaluate similarity between the anonymous document and documents stored in the database. Another issue that may appear when analyzing digitized historical documents is that some parts of text may not be readable or simply missing so the similarity measure should be robust and account for the missing text parts. To successfully query a database of text documents, it is necessary to provide an efficient indexing data structure. The data structure proposed in this paper for indexing a large database of text documents is generalized suffix tree.

Linear space and time algorithms for suffix tree construction introduced efficient approaches for text data analysis in various research domains [1]. The suffix tree is an alphabet agnostic data structure, suitable for applications in the domain of natural language processing [2, 3], as it can execute pattern matching queries on multiple text documents in linear time [4]. Pattern matching queries on generalized suffix trees provide references to all documents where a specific pattern has been found, which is used when comparing the amount of text shared between a new text document and the database of text documents indexed with the tree. Ukkonen's algorithm [5] for construction of suffix trees has an on-line property that enables extensions of the existing trees with indices of new documents.

Suffix trees can be used for calculating similarity scores between text documents [6]. Similarity measure proposed in this paper is based on lengths of continuous exact text matches between the query document and indexed document. The similarity score is then used to determine which documents are the most similar to the anonymous query document. The author of the majority of the most similar documents is identified as the author of the anonymous document. It is important to note that the proposed similarity measure is asymmetric, due to differences in lengths of documents. The same measure can be used to evaluate plagiarism, as the plagiarism of a document with a known author is reflected in high similarity between documents with different authors. Special case of plagiarism is auto-plagiarism, where an author plagiarized his own texts.

Plagiarism in general is not easily detected as the high similarity can be achieved not only by copying exact parts of text from other documents, but the copied parts can be rephrased, arranged in different order or substituted with synonymous words, keeping the same meaning of text but not easily recognizable by the machine as a plagiarism [7]. There are approaches for evaluating plagiarism of natural language text on a semantic level using deep learning [8] but also using suffix trees for programming code plagiarism [9].

Algorithm proposed in this paper is based on generalized suffix trees and it is easily adaptable for any language. As the plagiarism doesn't often occur in the entire document, but only in specific sentences, proposed algorithm accounts for that by calculating similarity on unit level, where the unit can be any logically separated text block.

Complete authorship identification process consists of four steps. First step is data preprocessing, which prepares text documents for the analysis. Second step is construction of a generalized suffix tree, containing indices of all database documents. Third step is pattern matching and similarity scoring of anonymous text against database documents using the previously constructed suffix tree. The final, fourth step is authorship identification of the anonymous document based on computed similarity with database documents.

2. Data preprocessing

As digitized text documents often come in forms of scanned images of photographs, the initial preprocessing step is optical character recognition and transformation of documents into text format. Text documents are then split into smaller units, such as chapters, paragraphs or sentences. Every unit receives a unique identifier which contains reference to source document and unit order number. Units are then split into separate words, transformed to lowercase and cleaned from interpunction. To account for diversity of different word forms with the same stem, words are stemmed using language specific stemmer. Additionally, language specific stop words are removed. Units are represented as sequences over an infinite alphabet of word stems. To account for synonymous words, multiple stems are mapped to one stem that will represent a group of its synonyms. Defined preprocessing steps improve quality of plagiarism evaluation when dealing with translated parts of text from different languages.

3. Tree construction and pattern matching

Let *S*be a sequence of length *n* symbols from alphabet *W*. Suffix *Sk* with length *k* of sequence *S* is defined as a subsequence of consecutive elements from position *n-k* to *n*. All sequence suffixes are used to construct suffix tree data structure that contains all sequence suffixes as unique paths from suffix tree root to leaves. To address the problem of ambiguity that appears when one suffix is a prefix of another suffix, all suffixes are terminated using the alphabet character unique to every suffix. Suffixes of multiple sequences can be joined into a single generalized suffix tree, having each suffix information extended with source sequence reference. All text document units are indexed in generalized suffix tree structure.

Every path from tree root contains information about unit identifiers and suffix position within the unit. Exact pattern matching using suffix trees is done by matching a pattern with prefixes of indexed suffixes starting from the root node. Once the pattern is matched, references stored with every matched suffix, give the information about the units that contain the matched pattern. When using word stems as alphabet symbols, equality between two symbols assumes that the entire sequences of stem characters from both symbols are matched. When querying a suffix tree of document units, it is required to preprocess the query sequence in the same way as the units are preprocessed for the tree construction.

4. Similarity measure

Similarity between documents can be calculated by computing a number of shared words between documents. For that purpose, Sørensen–Dice coefficient [10, 11] can be used, as the documents are represented as word sets. [12] This approach would result in the same similarity score independent from the ordering of words, which may not reflect semantic similarity. Proposed solution to this problem is a similarity measure that results with a score proportional to the length of the sequences matched between documents, consisting of potentially multiple words in the correct order.

For example, let the query text be a sentence “The given tree is constructed using Ukkonen’s algorithm”. Let the database contain sentences `DB1`: “Suffix trees are constructed according to Ukkonen’s algorithm” and `DB2`: "Given algorithm construction is based on suffix tree matching algorithm". When the units are preprocessed by the methods defined in section 2., they are transformed into:

```
Query: ["given", "tree", "construct", "ukkonen", "algorithm"]

DB1:   ["suffix", "tree", "construct", "accord", "ukkonen",
        "algorithm"]

DB2:   ["given", "algorithm", "construct", "bas", "suffix",
"tree",
        "match"]
```

Similarity between query and database sentences, using Sorensen-Dice coefficient, will result in the same score between query sentence and database sentence 1 (`DB1`) as well as between query and database sentence 2 (`DB2`). The query sentence shares exactly 4 stems with each database sentence. Although scores are the same, the meanings of database sentences are different and the correct score should reflect those differences.

Adding a weight function, with values proportional to the number of consecutively matched words in a sentence, results in a higher score for longer sequences of consecutive matches. In a simplest form, adding weight function *w,* with values equal to the number of words in consecutive matches, database sentence 1 will result in a greater score than database sentence 2, reflecting the fact that unit 1 is more similar in meaning with the query sentence than it is the case with database sentence 2. Weightcan be defined as any monotonically increasing function of consecutive match length, such are certain polynomials, logarithms or exponentials.

Similarity between any two text units should be interpretable and comparable, achieving the highest value when all the symbols from one unit are identical to and in the same order as the symbols in the other unit. To achieve those properties, all match scores between two units are summarized and normalized by dividing with the value of the selected weighting function for the input parameter being the length of the query unit.

$$similarity(a,b) = \frac{\sum_{i=0}^{k} matchScore_i(a,b)}{w(|a|)}$$

Normalized similarity values between two units are now in range *[0,1]*. One unit can have matches with multiple units. Aggregated similarity score for all matches can be calculated as an average or maximum over all similarity score values with other units and it is labeled as unit similarity.

$$unitSimilarity(u) = max_i\ similarity(u\ , du_i)$$

or

$$unitSimilarity(u) = \frac{1}{n}\sum_{i=0}^{n} similarity(u, du_i)$$

Finally, the document similarity score is calculated as the sum of unit scores for all of its uncited units, divided by the number of uncited units in the document. Document similarity scores are also in range *[0,1]*, with value 0 being associated with a document with no similarities with any database document and 1 representing score of a document whose units are all stored in the database.

$$documentSimilarity = \frac{1}{|u|}\sum_{i=0}^{|u|} unitSimilarity(u_i)$$

5. Implementation and Verification

The algorithm has been implemented using Python programming language and has been tested on a small dataset of digitized documents from eLibrary hosted by the Faculty of Mathematics. Stemmer described in [13], adapted for the Python programming language in [14], was used for stemming words of Serbian language. The preliminary results encourage further testing on a larger dataset. Testing should be done on a corpus of at least 100000 sentences by 5 different authors indexed using the generalized suffix tree. Furthermore, selected 1000 sentences, by the same authors, that are not indexed, should be used for authorship identification. Different weighting functions will be tested. For getting the source code of the algorithm, please contact the author.

6. Conclusion

Presented algorithm and similarity measure utilizes the benefits of suffix tree data structure with its fast construction and pattern matching algorithms and demonstrates an efficient approach for authorship identification, similarity and plagiarism evaluation, applied on digitized historical text documents. The presented algorithm can be adapted for any natural language. Further improvements may include alphabet symbol comparison based on different alignment scores, instead of equality comparison, and assignment of specific weights for matches of specific symbols. Testing phase will give a clearer picture on potential bottlenecks of the algorithm and display more potential directions for further improvements.

References

[1] Grossi, Roberto, "Suffix Trees and their Applications in String Algorithms", *https://www.researchgate.net/publication/2457466_Suffix_Trees_and_their_Applications_in_String_Algorithms* (1997).
[2] Kennington, Annemarie, "Suffix Trees as Language Models.", *Proceedings of the Eighth International Conference on Language Resources and Evaluation* (LREC2012), European Language Resources Association (ELRA), (2012): 446–453
[3] Shareghi, Trevor. "Compact, Efficient and Unlimited Capacity: Language Modeling with Compressed Suffix Trees." *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, (2015):2409–2418.
[4] Weiner, Peter "Linear Pattern Matching Algorithm". *Proc 14th IEEE Symp Switching and Automata Theory*. 1-11. 10.1109/SWAT.1973.13., (1973)
[5] Ukkonen, E., "On-line construction of suffix trees." *Algorithmica* 14 (2005): 249–260.
[6] Chim, Hung & Deng, Xiaotie, "A new suffix tree similarity measure for document clustering." *16th International World Wide Web Conference,* WWW2007, (2007):121–130, doi:10.1145/1242572.1242590.
[7] Maurer, Hermann A., Frank Kappe, Bilal Zaka, "Plagiarism-A survey." *J. UCS* 12.8 (2006): 1050–1084.
[8] D. Suleiman, A. Awajan, and N. Al-Madi, "Deep Learning Based Technique for Plagiarism Detection in Arabic Texts.", *2017 International Conference on New Trends in Computing Sciences (ICTCS)*, (2017):216–222
[9] Joy, M. and Michael Luck, "Plagiarism in programming assignments", *IEEE Transactions on Education* 42 (1999): 129–133.
[10] Sørensen, T., "A method of establishing groups of equal amplitude in plant sociology based on similarity of species and its application to analyses of the vegetation on Danish commons", *Kongelige Danske VidenskabernesSelskab*. 5 (4), (1948): 1–34.
[11] Dice, Lee R, "Measures of the Amount of Ecologic Association Between Species", *Ecology* 26, no. 3, (1945):297–302, doi:10.2307/1932409.
[12] Taerungruang, Supawat, and Wirote, Aroonmanakun, "Constructing an Academic Thai Plagiarism Corpus for Benchmarking Plagiarism Detection Systems", *GEMA Online Journal of Language Studies* 18, (2018): 186–202.
[13] Vuk Batanović, Boško Nikolić, Milan Milosavljević, "Reliable Baselines for Sentiment Analysis in Resource-Limited Languages: The Serbian Movie Review Dataset",

Proceedings of the 10th International Conference on Language Resources and Evaluation (LREC 2016), Portorož, Slovenia (2016):2688–2696.
[14] Milosevic, N. "Stemmer for Serbian language", *arXiv preprint*, arXiv (2012):1209.4471

aleksandar@matf.bg.ac.rs