INFORMATION SYSTEMS IN MANAGEMENT

Information Systems in Management (2017) Vol. 6 (3) 213–222

USING FREQUENT PATTERN MINING ALGORITHMS IN TEXT ANALYSIS

PIOTR OŻDŻYŃSKI, DANUTA ZAKRZEWSKA

Institute of Information Technology, Lodz University of Technology

In text mining, effectiveness of methods depends on document representations. The ones based on frequent word sequences are used in such tasks as categorization, clustering and topic modelling. In the paper a comparison of different algorithms for finding frequent word sequences is presented. There are considered techniques dedicated for market basket analysis such as GSP and PrefixSpan as well as a method based on a suffix array. The investigated techniques are compared with the new approach of searching maximum frequent word sequences in document sets. Performance of the algorithms is examined taking into account execution times for the considered test collections.

Keywords: GSP, SuffixArray, PrefixSpan, N-Gram, frequent sequence

1. Introduction

Nowadays text document analysis became a very important part of information retrieval process. One of the main issue connected with this task concerns the choice of document representation. As one of the most popular, there should be mentioned a bag-of-words representation, which was used in such algorithms as Rocchio [1], BM25 [2] or SVM [3]. However, there are several disadvantages of these techniques. The first one concerns polysemy property, which is connected with multi meanings of the same word. The next one is related to synonymy where multiple words have the same meaning [4]. To avoid arising problems, phrases instead of words may be used. Phrases seem to be more

intuitive, less ambiguous and more discriminative. However, on the other hand phrases have low frequency and some of them are redundant and meaningless.

To recognize meaning of phrases in the text a complete set of information about all their subsequences together with information concerning their connections may be useful. Therefore, the structure in the form of graph, with frequent sequences represented by nodes can be used for text representation building. Such structure for a single n-gram node is presented in Fig. 1. Each node holds information about words in a sequence and positions of each appearance of this sequence. Additionally, references to shorter subsequences are stored. On the other side, there are used two lists with references to longer sequences. The first list holds links to all sequences that start with the considered sequence and the second one contains references to sequences which end with this sequence.

In the paper algorithms for finding frequent sequences of words are examined. There is compared the performance of such algorithms as GSP, PrefixSpan and SuffixArray as well as of the new approach for finding maximum frequent word sequences called SequenceJoining. Additionally, the last algorithm enables to build described above node structure.

The remainder of the paper is organized as follows. In the next section, related work concerning finding frequent sequences in text documents is depicted. Next, all the examined algorithms are shortly described. In the following section the experiments, which aim at comparing the performance of algorithms are presented and their results are discussed. Finally, some concluding remarks are depicted.

2. Related work

Frequent pattern mining algorithms have been widely used in many real life problems. Broad review of the techniques and their applications is presented in [5]. Researchers have developed some of the frequent pattern mining algorithms to be used in text mining area. Garcia-Hernández et al. indicated that main difference between searching for frequent patterns in texts and in transactions concerns the ratio of numbers of transactions and attributes. In text mining there may occur a small number of items with big number of documents, and algorithms based on finding all possible candidates may not be efficient enough [6]. The authors proposed the algorithm, that use the pattern-growth strategy which process the documents in an incremental way. The algorithm produces an array, where each node holds identifier of a word pair, frequency of the pair and the list of positions where the pair appears [6].

Zhong et al. introduced a pattern discovery technique, which uses two processes: pattern deploying and pattern evolving, to refine the discovered patterns in text documents. The proposed approach allows to overcome the low-frequency and misinterpretation problems for text mining [4]. An automatic method for

discovering textual patterns is described in [7]. The method is extended to find generalized sequences in documents with additional annotations for each word.

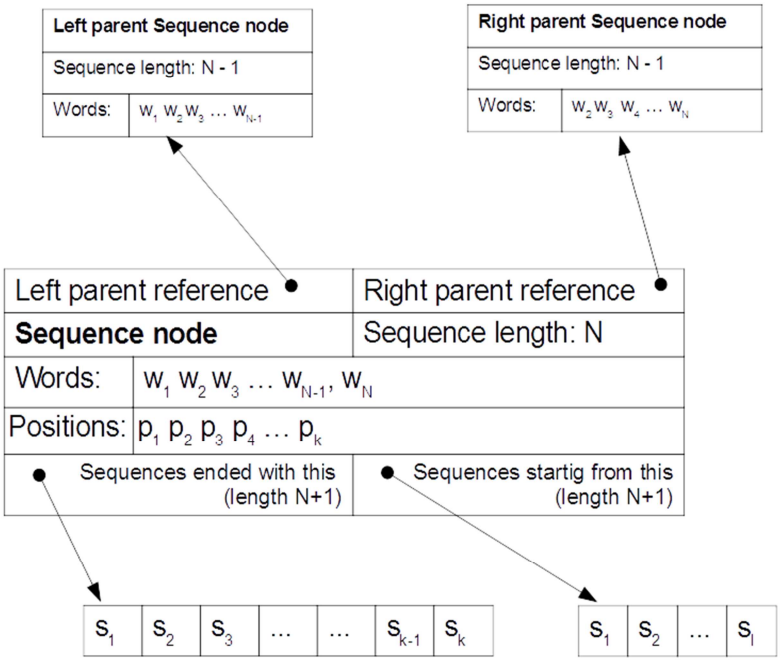

Figure 1. Single n-gram node

3. Frequent pattern mining algorithms

In [8] frequent pattern mining algorithm has been used for building frequent sequences graph in topic modeling approach. For building the required structure of frequent N-grams the technique based on apriori observation [9] has been considered. In the current research the performance of methods of finding frequent sequences has been compared taking into account their applications to topic modeling. However to achieve that, the structure in the form of graph with frequent sequences represented by nodes should be built, and hence connections between parent subsequences and child nodes should be generated. In order to attain this goal NGramLinking algorithm for frequent sequences is proposed. It is proceeded in the step following finding sequences.

For the comparison purpose the algorithms, which gather information concerning frequent sequences and their positions have been chosen. Required information is further used to add links between sequences. Such approach can be

applied in frequent itemset mining algorithms: PrefixSpan and GSP; as well as SuffixArrays dedicated to text datasets. The performance of all the techniques will be compared to SequenceJoining algorithm, which is designed to use in topic modeling tasks. All the mentioned algorithms are shortly described in the following subsections.

3.1. GSP algorithm

The GSP (Generalised Sequential Patterns) algorithm [10] has been designed for transactional data. The technique discovers generalized sequential patterns in the form of taxonomy, where each sequence represents a list of transactions and items are included across all levels of a hierarchy. The pseudocode of the algorithm is presented in Fig. 2.

In the first step, having a set of k-length sequences, all new candidate sequences of length k+1 are generated by joining the existing ones. In the second step the generated set of sequences is pruned and sequences of less than required support value are removed. The steps are executed till the set of frequent sequences is empty.

```
✓ Obtain a sequence in form of <x> as length-1 candidates
✓ find F1 (the set of length-1 sequential patterns), after a unique
  scan of database
✓ Let k=1
While Fk is not empty do
    - Form Ck+1, the set of length-(k+1) candidates from Fk;
    - If Ck+1 is not empty, unique database scan,
       find Fk+1 (the set of length(k+1) sequential patterns)
    Let k=k+1;
End While
```

Figure 2. Pseudocode of GSP algorithm [11]

3.2. PrefixSpan algorithm

PrefixSpan (Prefix-Projected Sequential Pattern Mining) algorithm is a "projection-based, sequential pattern-growth approach for sequential pattern mining. In this approach, a sequence database is recursively projected into a set of smaller projected databases, and sequential patterns are grown in each projected database by exploring only locally frequent fragments" [12].

The algorithm finds the complete set of sequential patterns and reduces the number of operations necessary to generate a candidate subsequence. As PrefixSpan based its ordered growth on prefix-ordered expansion, reduced number of "growth points" is used and projected databases are of smaller sizes. Fig. 3 presents a pseudocode of PrefixSpan algorithm.

```
Input: A sequence database S, and the minimum support
threshold min_sup
Output: The complete set of sequential patterns
Method: Call PrefixSpan(‹›, 0, S)
Subroutine PrefixSpan(α, l, S|α)
Parameters: α: a sequential pattern; l the length of α;
S|α: α-projected database, α ≠ ‹›; otherwise,
the sequence database S.
Method:
     1. Scan S|α once, find the set of frequent items b such that
       (a) b can be assembled to the last element of α to
       form a sequential pattern; or
       (b) ‹b› can be appended to α to form a sequential
        pattern.
     2. For each frequent item b , append it to α to form
        a sequential pattern α' , and output α';
     3. For each α', construct α'-projected database S|α', and call
PrefixSpan(α', l+1, S|α')
```

Figure 3. PrefixSpan pseudocode [12]

3.3. SuffixArrays algorithm

Suffix Arrays algorithm has been developed for string searching [13]. Main idea of the algorithm consists in using each word of a document set as a first word of a sequence. Then all the indicated sequences are organized alphabetically. In fact, only an array of pointers to first words of each sequences is sorted. Then sequences are compared and grouped taking into account number of subsequences of equal prefix. An example of this approach is presented in Fig. 4.

Documents:
a b a b c
b c b a b

Subsequences	Sorted subsequences	Prefixes	
		Sequence	Occurences
a b a b c	a b	a	3
b a b c	a b a b c	a b	3
a b c	a b c	b	5
b c	b	b a	2
c	b a b	b a b	2
b c b a b	b a b c	b c	2
c b a b	b c	c	2
b a b	b c b a b		
a b	c		
b	c b a b		

Figure 4. Sequences sorted by SuffixArray

3.4. SequenceJoining algorithm

SequenceJoining algorithm, similarly to GSP, is based on candidate generation and their testing approach. However, the proposed algorithm builds data structure which is a graph of connected nodes representing all frequent sequences. The algorithm starts by building a reverse bigram index as it is presented in Fig. 5.

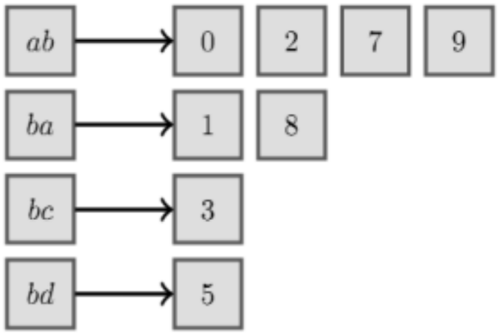

Figure 5. Reverted bigram index [14]

All pairs of words are the key for a list of positions. Each position is a number which indicates the document in the set and the offset of the first word of a bigram. Sequences of length $k + 1$ are created by joining two sequences of length k. All sequences of the length k are stored in a hash map. The keys are built from $k - 1$ beginning words and are connected to list of n-grams starting with this key. For each sequence the key from all words except the first one is created. All sequences from the hash map linked with this key are selected. New N-gram of length $k+1$ is created by joining pairs with matching keys. The support is calculated by using lists of positions of both joined subsequences.

Since the lists are in ascending order, finding positions of consecutive subsequences can be realized with the complexity of $O(m + n)$ where m and n mean lengths of respective position lists. In the proposed implementation binary search on both lists is used alternately. Such approach may significantly increase the efficiency of the whole algorithm. The complexity depends on the size of the result set. Only N-grams of satisfying support are selected as frequent. These steps are repeated for each length until the result is not empty.

After creating a sequence of length k both joined sequences are linked as the left and right parent. Furthermore the sequence is added as a child to lists of children in both parent sequences. Thus the required structure in the form of graph with frequent sequences represented by nodes is built.

3.5. NGramLinking algorithm

Except SequenceJoining algorithm all the considered ones do not generate connections between parent subsequences and child nodes. Therefore, after finding all frequent sequences additional step should be executed. All sequences with the same length are stored in a hash map. For all the sequences of the length *k* first and last subsequences are searched in a previously prepared map. These sequences are stored as the left and the right parents. The current sequence is added as a child to both parent sequences. Finally, the expected structure is built.

4. Experiment results

The experiments aimed at comparing the performance of GSP, PrefixSpan, SuffixArray and FrequenceJoining algorithms taking into account execution time for different amount of text documents. There were used two document datasets:

- The OHSUMED test collection [15], which contains 20,000 first records of documents from MEDLINE.
- The 20Newsgroups data set [16] - approximately 20,000 newsgroup documents.

During experiments there were used implementations of GSP and PrefixSpan algorithms in SPMF, which is an open-source data mining library written in Java, dedicated to pattern mining [17]. As original implementations are prepared for searching patterns of itemsets, the modifications have been done and finally itemsets were represented by words. The remainder of algorithms were also implemented in Java software. All the tests were done on PC with a processor Intel®Core™ i3-540, (4M Cache, 3.06 GHz).

The experiments were carried out for different number of documents taking into account different required support values. Moreover for each algorithm its performance was checked on raw as well as preprocessed with stop-list and stemming data.

PrefixSpan and SuffixArray algorithms had similar run times. Additional step used by NGramLinking occurred to be insignificant. The proposed SequenceJoining algorithm was faster for smaller text sets (1,5 million and less words). Run time of SequenceJoining was significantly reduced for result sets of smaller sizes. Such a fact took place for higher support thresholds. It means that an execution time of this algorithm depends on numbers of sequences with the successive lengths. The same dependence have not been noticed for the other algorithms. In all the cases, execution time for GSP considerably exceeded run times of the other algorithms. The results for all the algorithms except GSP and different parameter values are presented in Fig.6 till 10.

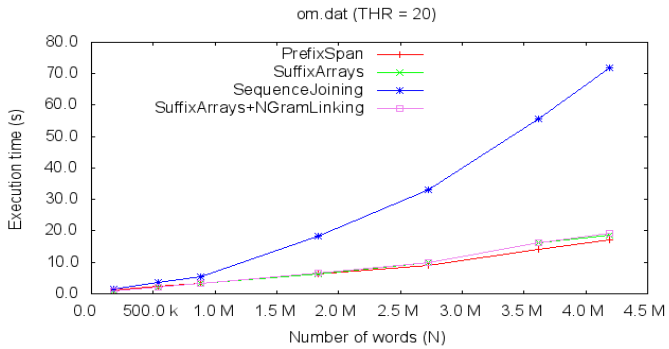

Figure 6. Execution time for OHSUMED with threshold 20

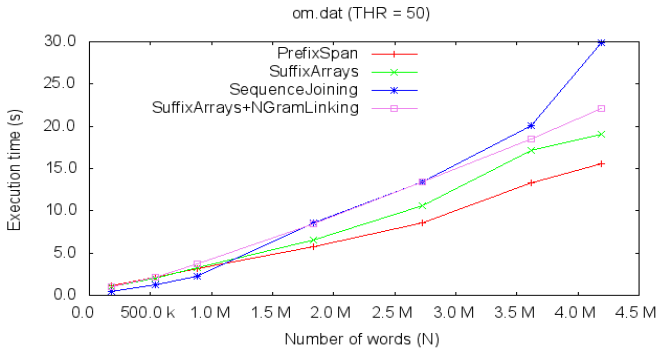

Figure 7. Execution time for OHSUMED with threshold 50

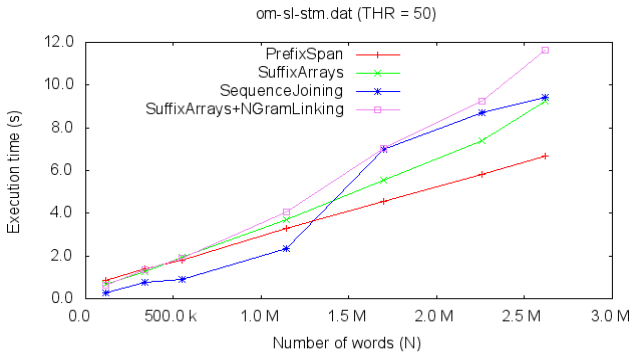

Figure 8. Run time OHSUMED (threshold 50) without stop-words after stemming

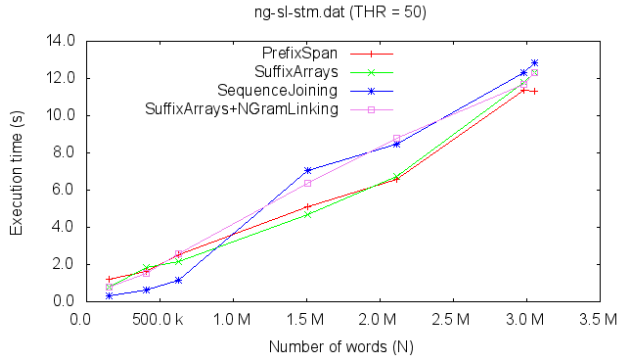

Figure 9. Run time 20Newsgroups (threshold 50) without stop-words after stemming

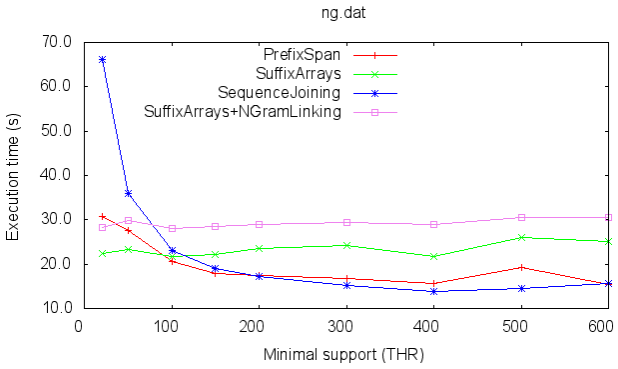

Figure 10. Time of execution for 20Newsgroups depending on threshold

5. Concluding remarks

In the paper the performance of frequent pattern mining algorithms GSP, PrefixSpan, SuffixArray and the new approach SequenceJoining were considered. In the case of the first three algorithms additional step of building graph structure has been proposed. Experiments have shown that run time of all the algorithms except GSP is of similar range. Execution time is reduced for higher support thresholds, when the result sets are smaller. The SequenceJoining gave the best results for small document datasets. This feature has not been observed for the other examined algorithms.

REFERENCES

[1] Manning Ch. D., Raghavan P, Schütze H. (2008) *An Introduction to Information Retrieval*, Cambridge University Press.

[2] Robertson S., Zaragoza H. (2009) *The Probabilistic Relevance Framework: BM25 and Beyond*, Found. Trends Inf. Retr, 3(4), 333–389.

[3] Burges Ch. J. C. (1998) *A Tutorial on Support Vector Machines for Pattern Recognition*, Data Mining and Knowledge Discovery, 2, 121–167.

[4] Zhong N., Li Y., Wu Sh.-T. (2012) *Effective Pattern Discovery for Text Mining*, IEEE Transactions on Data Engineering, 24(1), 30-44.

[5] Aggarwal Ch. C., Han J. [eds] (2014) *Frequent Pattern Mining*, Springer International Publishing Switzerland.

[6] Garcia-Hernández R. A., Martínez-Trinidad J.F., Carrasco-Ochoa J.A. (2010) *Finding maximal sequential patterns in text document collections and single documents*, Informatica, 34, 93–101.

[7] Ahonen-Myka H. (2002) *Discovery of frequent word sequences in text*, Proc. the ESF Exploratory Workshop on Pattern Detection and Discovery, London, UK, 180–189.

[8] Ożdżyński P., Zakrzewska D. (2017) *Topic Modeling Based on Frequent Sequences Graphs*, Świątek J., Tomczak J.M. (eds.), *Advances in Systems Science*, Advances in Intelligent Systems and Computing 539, Springer International Publishing, 86-97.

[9] Agrawal, R., Srikant R. (1994) *Fast algorithms for mining association rules in large databases*, Proc. the 20th International Conference on Very Large Data Bases, VLDB, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 487-499.

[10] Agrawal R., Srikant R. (1995) *Mining sequential patterns*, Proc. 1995 Int. Conf. Data Engineering (ICDE'95), 3–14

[11] Slimani T., Lazzez A., (2013) *Sequential Mining: Patterns and Algorithms Analysis*, International Journal of Computer and Electronics Research, 2 (5), 639-647.

[12] Pei J, Han J., Mortazavi-Asl J., Pinto H., Chen Q., Dayal U., Hsu M. (2001) *PrefixSpan: Mining Sequential Patterns Efficiently by Prefix-Projected Pattern Growth*, Proc. 2001 Int. Conf. Data Engineering (ICDE '01), 215-224.

[13] Manber U., Myers G. (1989) *Suffix arrays: A new method for on-line string searches*, SODA '90 Proc. the first ACM-SIAM symposium on Discrete algorithms, 319-327.

[14] Ożdżyński P. (2014) *Text Document Categorization Based on Word Frequent Sequence Mining*, Information Systems Architecture and Technology, Contemporary Approaches to Design and Evaluation of Information Systems, 129-138.

[15] ftp://medir.ohsu.edu/pub/ohsumed

[16] http://www.ai.mit.edu/people/jrennie/20Newsgroups/

[17] Fournier-Viger, P., Lin, C.W., Gomariz, A., Gueniche, T., Soltani, A., Deng, Z., Lam, H. T. (2016). *The SPMF Open-Source Data Mining Library Version 2*. Proc. PKDD 2016 Part III, Springer LNCS 9853, 36-40.