ANNOTATED SUFFIX TREE AS A WAY OF TEXT REPRESENTATION FOR INFORMATION RETRIEVAL IN TEXT COLLECTIONS

Dmitry S. FROLOV
Post-graduate student, Department of Data Analysis and Artificial Intelligence,
National Research University Higher School of Economics
Address: 20, Myasnitskaya Street, Moscow, 101000, Russian Federation
E-mail: dfrolov@hse.ru, dmitsf@gmail.com

A method for information retrieval based on annotated suffix trees (AST) is presented. The method is based on a string-to-document relevance score calculated using AST as well as fragment reverse indexing for improving performance. We developed a search engine based on the method. This engine is compared with some other popular text aggregating techniques: probabilistic latent semantic indexing (PLSA) and latent Dirichlet allocation (LDA).

We used real data for computation experiments: an online store's xml-catalogs and collections of web pages (both in Russian) and a real user's queries from the Yandex. Wordstat service. As quality metrics, we used point quality estimations and graphical representations. Our AST-based method generally leads to results that are similar to those obtained by the other methods. However, in the case of inaccurate queries, AST-based results are superior. The speed of the AST-based method is slightly worse than the speed of the PLSA/LDA-based methods. Due to the observed correlation between the average query performing time and the string lengths at the AST construction phase, one can improve the performance of the algorithm by dividing the texts into smaller fragments at the preprocessing stage. However, the quality of search may suffer if the fragments are too short. Therefore, the applicability of annotated suffix tree techniques for text retrieval problems is demonstrated. Moreover, the AST-based method has significant advantages in the case of fuzzy search.

Key words: text document retrieval, aggregate text representation, annotated suffix tree (AST), probabilistic latent semantic indexing (PLSI), latent Dirichlet allocation (LDA), fuzzy text search.

Citation: Frolov D.S. (2015) Annotated suffix tree as a way of text representation for information retrieval in text collections. *Business Informatics*, no. 4 (34), pp. 63–70. DOI:10.17323/1998-0663.2015.4.63.70.

Introduction

One of the key areas in data analysis is the processing of document collections: document rubrication, text similarity scoring, document search by given keywords and so on [4, 5, 9, 11]. The last problem forms a particular part of computer science, so-called information retrieval. Information retrieval problems are widely covered in books and research papers. A major reference is 'Introduction to Information Retrieval' by C.Manning et al. [11], which can be used as a textbook for information retrieval studies as well. In papers [4, 9], belonging to a rather popular scientific genre, one may find many methods and techniques which are applicable for design and implementation of search engines. In paper [4] information retrieval tools are considered, including retrieval techniques for specific documents like Internet forums pages at which most information is produced by users.

In principle, the problem of document retrieval for a given query can be considered as a task of full-text search or even as a basic string matching problem. There have been a number of methods developed for this problem including that emphasized in paper [15]. Applicable techniques aim at a reasonable balance between quality of obtained results and retrieval time.

A classic approach in information retrieval is indexing of document collections [5, 11]. However, there is a need to improve performance and accuracy, which leads to developing alternative approaches. One of these alternative methods is aggregate representation both for individual texts [3, 13, 14] and for collections as well [2, 7]. The Annotated suffix tree method (AST) was proposed in papers by B.G.Mirkin and E.L.Chernyak [3, 14]. The authors use AST for determining the so-called string-to-document relevance score and automated construction of taxonomy systems [3]. Regarding other approaches to aggregate document collection representations, let us indicate, first of all, probabilistic latent semantic indexing (PLSI) [7] and latent Dirichlet allocation (LDA) [2]. A recent survey of these and similar techniques can be found in A.Korshunov and A.Gomzin's paper [8]. Further modifications of classic methods in probabilistic topic modelling can be found in K.V.Vorontsov's papers, for example, in [16].

Note that both the probabilistic latent semantic indexing and latent Dirichlet allocation use feature-based text representation, whereas the annotated suffix tree approach uses fragment representation only [13]. This feature is very important for fuzzy text searching, for example, if the search query or documents in a collection contain mistakes. Feature-based methods require special adaptation for such situations [11]. The purpose of this work is to propose a technique for applying AST in information retrieval of texts or queries containing mistakes and to analyze its efficiency against the methods using probabilistic topic modelling.

Currently there are several implementations of the AST method. Using annotated suffix arrays (ASA) and tree constructing algorithms for substring search proposed in [6] significantly reduces the time complexity of text aggregation and provides an appreciable saving of memory resources.

1. Methods

In this work we compare aggregate text representations of whole document collections (PLSI, LDA) and individual texts (AST) as ways to solve the information retrieval problem for a given query.

In PLSA and LDA information retrieval, the problem is reduced to a problem of detecting documents which are similar to the given one (search query). Therefore the search query is transformed into a feature vector form used in these models [11]. After that, one has to carry out procedures of calculating the similarity score between the query and representations of all documents in a collection [17], sorting by obtained values and choosing documents gaining the greatest scores. Of course, different parameters of the models (in particular, the number of detected themes and algorithm parameters) may lead to different results of these procedures. Note that there are many implementations of PLSA and LDA models; the main differences between these implementations are development tools. For our experiments, we used implementations from the free distributed framework Gensim (*https://radimrehurek.com/gensim*) for Python, with significant modifications and special adjustments for our aims. In particular, the feature processing module was appended, and we also changed document storage formats for more convenient work with the databases we used.

Let us describe a method we developed for full-text search based on AST representations of the documents. Here we give a classical AST constructing algorithm for a given document [3]. AST is a weighted root tree, which is used as a data structure for storing and processing text, which stores fragments and corresponding frequencies. One of the tree nodes is a root of the tree; an empty tree contains the root only. Other nodes contain text symbols and corresponding frequencies (so-called annotations).

To use AST, we have to split document into strings – symbol sequences. As a rule, one string is formed by 2-4 sequential words from a text. A k-suffix of a string $x = x_1 x_2 \dots x_N$ of length N is a substring $x_k = x_{N-k+1} x_{N-k+2} \dots x_N$.

For example, 3-suffix of a string *INFORNATION* is a substring *ION*. Note that an N-suffix of any string is a whole string. An algorithm for constructing AST for a given string x is described below:

1. Initialize an empty AST T;

2. Find all suffixes of a given string $\{x_k : k = 1, 2, \dots N\}$;

3. For each suffix x_k find maximal overlap (path from the root) in T: x_k^{max}. For all nodes from x_k^{max} increase annotations by 1. If the length of x_k^{max} is less than k, one creates new nodes from the remaining part of this suffix; annotations of all the new nodes are equal to 1.

An example of AST for a string *ABCBA* is shown in *Fig. 1.*

To construct AST for 2 or more strings, one has to perform sequential addition of these strings into a common tree. Hence, if we represent a document as a set of strings, we can construct an AST for the document.

The AST constructing algorithm, which is given above, is quite costly both for the time and space complexity. Currently there are methods based on classical string algorithms, for example, the Ukkonen algorithm

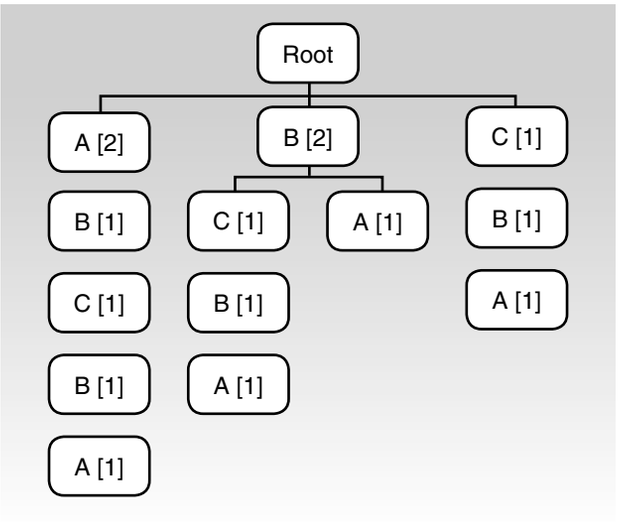

Fig 1. An example of AST for a string ABCBA

for substring matching [6]. This makes it possible to reduce time complexity for AST construction. Another way to improve the algorithm is to reduce memory consumption. This goal can be achieved using special data structures for AST storage, for example, special arrays.

The AST constructed for the document allows us to solve a problem of similarity scoring between a given string and the document. This problem is called string-to-document relevance scoring. Let us describe a way to solve this problem proposed in [3].

Consider the conditional probability of node u in given AST T with the root R. Denote annotation of node u as $f(u)$. The conditional probability of node u is:

$$p(u) = \frac{f(u)}{\sum_{v \in T: ancestor(v) = ancestor(u)} f(v)}, \quad (1)$$

where $ancestor(u)$ – is an ancestor of node u in the AST T.

For each suffix x_k of the string x, the relevance score is defined as the sum of conditional probabilities of all nodes which belong to maximal overlap x_k^{max}:

$$s(x_k, T) = \frac{1}{k_{max}} \sum_{i=0}^{k_{max}} p(x_i^k). \quad (2)$$

Finally, the relevance score of the string x for AST T is expressed by the following formula:

$$S(x, T) = \frac{1}{N} \sum_{k=0}^{N} s(x_k, T). \quad (3)$$

For example, the relevance score of a substring *BAC* and AST constructed for a string *ABCBA* is 0.35.

Hence, one can formulate the following strategy for retrieval documents for a given query. If we have an AST constructed for each document in a collection, then one can calculate string-to-document relevance scores, sort them and obtain the most relevant documents. Obviously, iteration of the procedure for string-to-document relevance scoring makes the speed of this method very slow. We propose to compute relevance scores not for a whole collection but for documents which have a high probability to gain a high score. To detect these documents, we use an approach based on an inverted index [5, 11] of ancillary features (fragments). We use a special data structure, which is exactly a hash-table as the following:

$$\begin{cases} f_1 : \left[n_{11}, \ldots, n_{1m_1} \right] \\ f_2 : \left[n_{21}, \ldots, n_{2m_2} \right] \\ \ldots \\ f_K : \left[n_{K1}, \ldots, n_{Km_K} \right] \end{cases}, \quad (4)$$

where f_i – document fragments;

n_{ij} – indexes of corresponding documents (or links to documents) which contain these fragments;

K – count of fragments.

We can consider any document feature as a fragment. In our implementation, we use 3- and 4-grams (sequential symbols of a text). One can use individual words or bigrams; this approach gives us a classical inverted index. If a collection is very large, one can consider not all fragments but fragments having a frequency which is more than the given one. Thus, query processing before string-to-document relevance scoring includes the selection of «candidate documents».

The final search algorithm consists of the following steps:

1. Split a query into substrings and select corresponding candidate documents from an inverted index;

2. Calculate the string-to-document relevance score for these candidate documents using AST;

3. Sort the documents by relevance score.

These steps are the simplest indexing algorithm before using AST. One can improve the hash-table using fragments with the numbers of their occurrences in documents, of some transformations (for example, tf-idf [12]). In principle, it is possible to use ranking functions (BM25 and others) with a hash-table and following comparison with the results obtained with AST. A particular issue to investigate is a situation if there are no ancillary features in a search query. In our algorithm, we include a whole collection in a list of candidate documents.

2. Experiments and results

Let us describe an experimental approval of the proposed method. In our comparison, we have considered the AST-based method and methods based on PLSI and LDA (for the last one we implemented a modification used bigrams [2]). We used a laptop with a 2.0 GHz dual-core processor and 4 GB RAM, running under the Ubuntu 12.04 operating system for our aims. Test document collections were stored in the document-oriented database MongoDB (*http://mongodb.org*). The full-text search engine of this database was also included in our comparison. We used 2 document collections for experimental approval.

The first (Collection #1) was produced from the xml-catalogue of fantasy books in Russian in the Ozon.ru online superstore (*http://www.ozon.ru*). The catalogue contains about 90,000 documents, formatted to contain the title, author, description and a set of parameters (such as current price, discount, number of pages, publisher, etc.). To use these documents for our purposes, we remove all except the title, author and description and transform documents into text form.

We used real user queries obtained from the Yandex. Wordstat online service (*http://wordstat.yandex.ru*). This service is used for efficient website promotion in the Yandex search engine and SEO-optimization. It allows us to get frequency statistics for a given query (weekly, monthly etc.) and to obtain a wide list of associated queries (which are similar by form or by search results). We used the last feature. To obtain a set of queries from this system, we make requests for some chosen test query q and obtain a list of real user queries $S(q) = \{q_1, ..., q_N\}$. We formed 2 sets of test queries. The first one contained titles of subcategories in the original xml-catalog; the second one consisted of words and phrases from documents. From these sets of queries was formed 3 groups of real users' queries. Group #1 consisted of queries from lists $S(q)$ for the first set. We called this group «Titles of subcategory». Groups #2 and #3 - «Transparent queries» and «Inaccurate queries» consisting of user queries obtained from lists for the second set. But for Group #3, we selected inaccurate and damaged queries in $S(q)$ lists. Altogether we considered 90 queries. Note that we used special scripts in Python to automate processing of the results of our experiments.

To measure quality metrics of the methods under consideration, precision at level of N documents R_N and recall R were calculated. These metrics are calculated for each query using cardinality of the relevant document set and set of documents retrieved by the system. Note that one has to choose N documents (which has the largest weight assigned by the system) as a second set to calculate N document level precision. The standard formulas are as follows:

$$R = \frac{\left|X_{relevant} \cap X_{obtained}\right|}{\left|X_{relevant}\right|}, \quad (5)$$

$$P_N = \frac{\left|X_{relevant} \cap X^N_{obtained}\right|}{\left|X^N_{obtained}\right|}, \quad (6)$$

where $X_{relevant}$ – a set of documents which are relevant for a given query;

$X_{obtained}$ – set of documents retrieved by the system;

$X^N_{obtained}$ – set of N documents which gained the largest relevance score in the search system for a given query.

Average precision of the methods is shown in *Table 1* and *2* for the level of documents 5 and 10 respectively.

Table 1.

5-document level precision

Number of group of queries	AST	PLSI	LDA	LDA with bigrams	MongoDB fulltext search
1	**0.87**	0.71	0.85	**0.87**	0.55
2	0.84	0.64	0.82	**0.86**	0.57
3	**0.78**	0.43	0.45	0.59	0.28
Average	**0.83**	0.59	0.71	0.77	0.47

Table 2.

10-document level precision

Number of group of queries	AST	PLSI	LDA	LDA with bigrams	MongoDB fulltext search
1	0.85	0.70	0.85	**0.86**	0.5
2	0.84	0.68	0.81	**0.86**	0.5
3	**0.79**	0.41	0.43	0.55	0.2
Average	**0.82**	0.56	0.70	0.76	0.4

Note that precision levels of the AST-based method for the 3rd group of queries («Inaccurate queries») are almost the same as for the 1st and 2nd groups. This should be attributed to the fact that AST uses text fragments, not features.

To show the level of recall and see the interplay between the precision and recall, let us use 11-point precision-recall TREC curves [1]. Such a curve is one of the graphical representations of the quality level of a search engine. It shows the precision level for a given recall level at the same search engine. Usually one uses 11 levels of

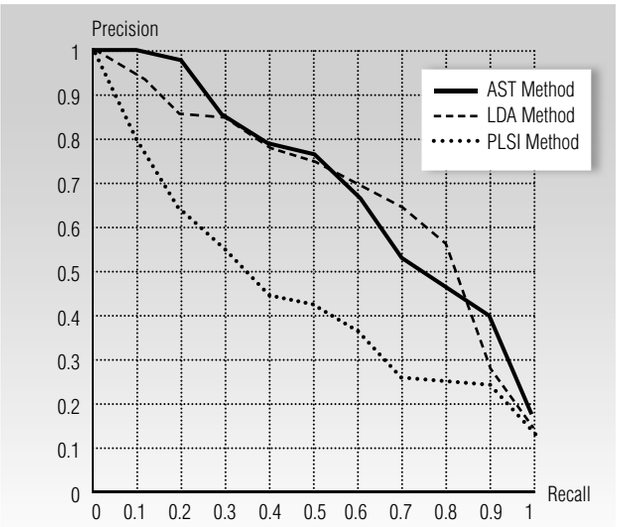

Fig. 2. Precision-recall curves for different methods, group of queries #1

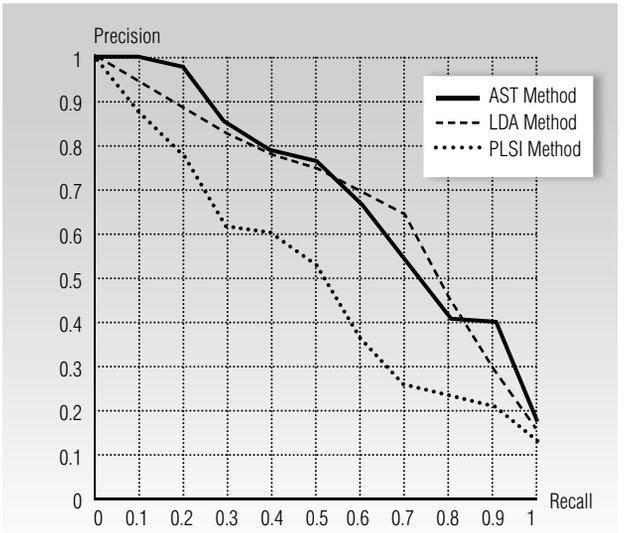

Fig. 3. Precision-recall curves for different methods, group of queries #2

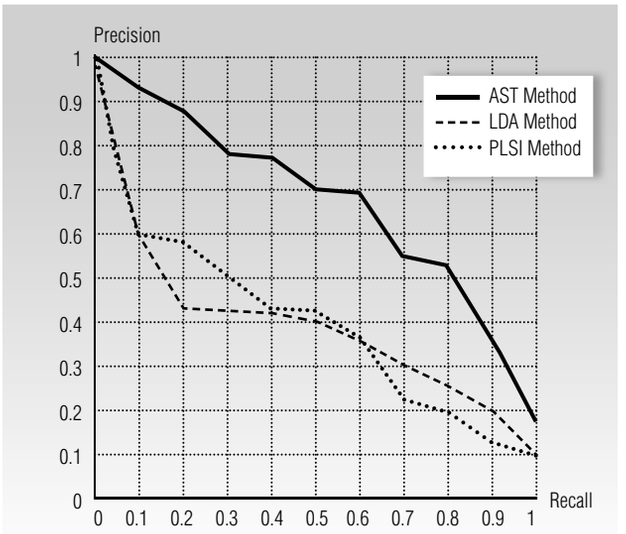

Fig. 4. Precision-recall curves for different methods, group of queries #3

recall (from 0.0 to 1.0 with a step equal to 0.1). It gives more complete information than one point characteristic. Les us describe a procedure of constructing the precision-recall curve.

1. Fix 11 recall values from 0.0 to 1,0 with the step 0.1;
2. Choose and fix the set of queries;
3. For each query perform the precision interpolation procedure for given recall values R_i, $i = 1, ..., 11$:
   - carry out the search procedure and obtain a list of results sorted by descending relevance score;
   - in this list note the number of the document for which the value of recall is equal to $R_i - K$, and calculate the corresponding precision value (for level of K documents) - P_K;
4. Find the averages for obtained values;
5. Get the pairs of values (precision, recall), draw a curve.

Precision-recall curves for different search methods obtained for different group of queries (1, 2, 3) are shown by *Fig. 2-4* below.

For the first two groups, the AST-based method leads to results that are similar to those obtained by the other methods. However, for the 3rd group of queries («Inaccurate queries») the AST method results are superior.

Average query processing times are shown in *Table 3*. So, the AST-based method yields to PLSA and LDA but nevertheless it is significantly faster than MongoDB full-text search. Note that the performance of the AST method is a really difficult problem.

Table 3.

Average time of query processing for different methods (s)

AST	PLSI	LDA	LDA with bigrams	MongoDB fulltext search
0.43	**0.23**	0.25	0.29	8.55

The second collection of documents was obtained from a set of Habrahabr.ru (*http://habrahabr.ru*) web pages. We downloaded several thousand web pages from this site and chose the 2,000 largest pages from this set. After that we removed the HTML markdowns from the web pages and saved the results as raw texts. The average length of these texts was significantly greater than of those used in the first experiment.

We use this collection of documents in order to test the performance of the search engines under consideration. The queries here were chosen as slightly modified strings of the documents. The AST-based retrieval method appears to be slower than those based on PLSI and LDA, but it is significantly faster than the MongoDB embedded full-text search engine.

Table 4.

Average time of query processing for different methods (s)

AST	PLSI	LDA	LDA with bigrams	MongoDB fulltext search
0.15	**0.04**	0.05	0.08	3.09

The correlation between the time for query processing and length of strings used for AST constructing also was investigated. As a rule, one uses 2-3 sequential words as a string for tree constructing. We implement algorithm modifications using 1 word and 5-6 words as a string. The experiment results are presented in *Table 5*. From the results one can conclude that string «extending» increases the query processing time.

Table 5.

Influence of string length used in AST and query processing time (s)

AST with 1 word per string	AST with 2-3 words per string	AST with 5-6 words per string
0.12	0.15	0.21

One can suppose we can improve the performance of the algorithm by using the minimal possible length of AST strings. However, in such a case we can lose helpful conjunctions between words and impair the quality metrics of the algorithms. We are going to investigate this in further work.

Conclusion

We described and verified the method of applying AST technique for information retrieval in document collections. Experiments using real data show us that the AST-based method does not yield to classical PLSI and LDA methods, and, furthermore, has some advantages, for example, in the case of inaccurate queries. We can conclude that it is necessary to make further investigations of the applicability of AST techniques in information retrieval. Our future plans include developing effective text retrieval methods based on the AST techniques. This includes the following subtasks:

1. Improvement of the AST-based method for document retrieval with the help of both index-based principles and approaches using no indexing;
2. Developing a method for document retrieval using aggregate representation of a whole collection or representation of document groups;
3. Adaptation of the methods for distributed document storing and computation;
4. Adaptation of the methods to cases of dynamically changing document collections;
5. Conducting experimental computations for comparison of the developed methods.

I wish to thank Prof. B. Mirkin for his advice and valuable comments. ■

References

1. Ageev M., Kuralenok I., Nekresyanov I. (2004) Prilozhenie A. Oficial'nye metriki ROMIP'2004 [ROMIP'2004 official metrics]. Proceedings of the *ROMIP 2004 Seminar, October 1, 2004, Saint Petersburg, Russia*. Saint Petersburg: SPbSU Institute of Chemistry, pp. 142–150 (in Russian).
2. Blei D.M., Ng A.Y., Jordan M.I. (2003) Latent Dirichlet allocation. *Journal of Machine Learning Research*, no. 3, pp. 993–1022.
3. Chernjak E.L., Mirkin B.G. (2013) Ispol'zovanie resursov Interneta dlja postroenija taksonomii [Using the Internet for Taxonomy Constructing]. Proceedings of the *AIST 2013 Conference, April 4-6, 2013, Ekaterinburg, Russia*. National Open University INTUIT, pp. 36–48 (in Russian).
4. Croft W.B., Metzler D., Strohman T. (2010) *Search engines: Information retrieval in practice*, Boston: Addison-Wesley.
5. Cutting D., Pedersen J. (1989) Optimization for dynamic inverted index maintenance. Proceedings of the *13th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGR 1989), June 25-28, 1989, Cambridge, MA, USA*. NY: ACM Press, pp. 405–411.
6. Dubov M., Chernjak E. (2013) Annotirovannye suffiksnye derev'ja: Osobennosti realizacii [Annotated suffix trees: Implementation features]. Proceedings of the A*IST 2013 Conference, April 4-6, 2013, Ekaterinburg, Russia*. National Open University INTUIT, pp. 49–57 (in Russian).
7. Hofmann T. (1999) Probabilistic latent semantic indexing. Proceedings of the *22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 1999). August 15-19, 1999, Berkeley, CA, USA*. NY: ACM Press, pp. 50–57.
8. Korshunov A., Gomzin A. (2012) Tematicheskoe modelirovanie tekstov na estestvennom jazyke [Topic modeling for texts in natural language]. *Proceedings of the Institute for System Programming of the RAS*, no. 23, pp. 215–238 (in Russian).

9. Langville A. N., Meyer C. D. (2011) *Google's PageRank and beyond: The science of search engine rankings*, Princeton: Princeton University Press.
10. Malkov Yu., Ponomarenko A., Logvinov A., Krylov V. (2014) Approximate nearest neighbor algorithm based on navigable small world graphs. *Information Systems*, no. 45, pp. 61–68.
11. Manning K., Raghavan P., Shjutce H. (2011) *Vvedenie v informacionnyj poisk* [Introduction to information retrieval]. Moscow: Williams (in Russian).
12. Mirkin B. (2011) *Core concepts in data analysis: Summarization, correlation and visualization*, London: Springer.
13. Mirkin B.G. (2011) *Metody klaster-analiza dlya podderzhki prinyatiya reshenii: obzor* [Cluster analysis for decision making: Review]. Working paper WP7/2011/03. Moscow: HSE (in Russian).
14. Mirkin B.G., Chernjak E.L., Chugunova O.N. (2012) Metod annotirovannogo suffiksnogo dereva dlja ocenki stepeni vhozhdenija strok v tekstovye dokumenty [Annotated suffix tree as a way of string-to-document score evaluating]. *Business Informatics*, no. 3 (21), pp. 31–41 (in Russian).
15. Natarajan J., Shaw S., Bruchez R., Coles M. (2012) *Pro T-SQL 2012 Programmer's Guide*, NY: Apress.
16. Vorontsov K.V., Potapenko A.A. (2011) Modifikacii EM-algoritma dlja verojatnostnogo tematicheskogo modelirovanija [EM algorithm modifications for probabilistic topic modeling]. *Journal of Mathematical Physics*, no. 52, pp. 1–21 (in Russian).
17. Wei X., Croft W.B. (2006) LDA-based document models for ad-hoc retrieval. Proceedings of the *29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGR 2006), August 6-11, 2006, Seattle, WA*, USA. NY: ACM Press, pp. 178–185.

ПРИМЕНЕНИЕ МЕТОДА АННОТИРОВАННОГО СУФФИКСНОГО ДЕРЕВА В ЗАДАЧАХ ПОИСКА В КОЛЛЕКЦИЯХ ТЕКСТОВЫХ ДОКУМЕНТОВ

Д.С. ФРОЛОВ
аспирант, департамент анализа данных и искусственного интеллекта,
Национальный исследовательский университет «Высшая школа экономики»
Адрес: 101000, г. Москва, ул. Мясницкая, д. 20
E-mail: dfrolov@hse.ru, dmitsf@gmail.com

В работе представлен метод информационного поиска в коллекциях текстовых документов, основанный на аннотированных суффиксных деревьях (АСД). В методе используется определение степени вхождения строки в АСД, полученные для документов, а также обратный индекс, построенный по фрагментам документов (с целью улучшения производительности). На основе представленного метода реализована поисковая система и произведено ее сравнение с алгоритмами поиска, использующими другие способы агрегированного представления текстов (всей коллекции целиком) – вероятностным латентно-семантическим индексированием (PLSI) и скрытым размещением Дирихле (LDA).

Для проведения вычислительных экспериментов использованы реальные данные: коллекция xml-каталогов онлайн-магазина и коллекция веб-страниц (обе – на русском языке), а также пользовательские поисковые запросы, полученные с помощью сервиса Yandex.Wordstat. Исследованы качественные метрики рассматриваемых систем: получены точечные оценки и графические характеристики. Метод поиска, основанный на АСД, в целом показывает результаты, сравнимые с другими алгоритмами, однако, на неточных запросах существенно превосходит их. Была исследована производительность сравниваемых поисковых систем, в результате отмечено, что метод на основе АСД несколько уступает другим по скорости поиска. Также изучена зависимость между временем выполнения запроса и длиной строк текста, используемых для построения АСД: для улучшения производительности необходимо выбирать минимально возможную длину строк, принимая во внимание тот факт, что слишком короткие строки могут ухудшить качественные характеристики метода. Отдельно отмечен факт применимости метода на основе АСД к задачам нечеткого поиска, что должно стать предметом будущих исследований.

Ключевые слова: информационный поиск в коллекциях текстов, агрегированное представление текстов, аннотированное суффиксное дерево (АСД), вероятностное латентно-семантическое индексирование (PLSI), скрытое размещение Дирихле (LDA), нечеткий текстовый поиск.

Цитирование: Frolov D.S. Annotated suffix tree as a way of text representation for information retrieval in text collections // Business Informatics. 2015. No. 4 (34). P. 63–70. DOI: 10.17323/1998-0663.2015.4.63.70.

Литература

1. Агеев М., Кураленок И., Некрестьянов И. Приложение А. Официальные метрики РОМИП'2004 // Труды второго российского семинара по оценке методов информационного поиска, 1 октября 2004 г., г. Санкт-Петербург; под ред. И.С.Некрестьянова. СПб.: Институт Химии СПбГУ, 2004. С. 142–150.
2. Blei D.M., Ng A.Y., Jordan M.I. Latent Dirichlet allocation // Journal of Machine Learning Research. 2003. No. 3. P. 993–1022.
3. Черняк Е.Л., Миркин Б.Г. Использование ресурсов Интернета для построения таксономии // Доклады всероссийской научной конференции АИСТ 2013, 4-6 апреля 2013 г., г. Екатеринбург. Национальный открытый университет ИНТУИТ, 2013. С. 36–48.
4. Croft W.B., Metzler D., Strohman T. Search engines: Information retrieval in practice. Boston: Addison-Wesley, 2010. 520 p.
5. Cutting D., Pedersen J. Optimization for dynamic inverted index maintenance // Proceedings of the 13th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGR 1989), June 25-28, 1989, Cambridge, MA, USA. NY: ACM Press, 1989. P. 405–411.
6. Дубов М., Черняк Е. Аннотированные суффиксные деревья: особенности реализации // Доклады всероссийской научной конференции АИСТ 2013, 4-6 апреля 2013 г., г. Екатеринбург. Национальный открытый университет ИНТУИТ, 2013. С. 49–57.
7. Hofmann T. Probabilistic latent semantic indexing // Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 1999). August 15-19, 1999, Berkeley, CA, USA. NY: ACM Press, 1999. P. 50–57.
8. Коршунов А., Гомзин А. Тематическое моделирование текстов на естественном языке // Труды Института системного программирования РАН. 2012. № 23. С. 215–238.
9. Langville A.N., Meyer C.D. Google's PageRank and beyond: The science of search engine rankings. Princeton: Princeton University Press, 2011. 240 p.
10. Approximate nearest neighbor algorithm based on navigable small world graphs / Malkov Yu. [et al.] // Information Systems. 2014. No. 45. P. 61–68.
11. Маннинг К., Рагхаван П., Шютце Х. Введение в информационный поиск. М.: Вильямс, 2011. 528 с.
12. Mirkin B. Core concepts in data analysis: Summarization, correlation and visualization. London: Springer, 2011. 390 p.
13. Миркин Б.Г. Методы кластер-анализа для поддержки принятия решений: обзор. Препринт WP7/2011/03. М.: НИУ ВШЭ, 2011. 39 с.
14. Миркин Б.Г., Черняк Е.Л., Чугунова О.Н. Метод аннотированного суффиксного дерева для оценки степени вхождения строк в текстовые документы // Бизнес-информатика. 2012. № 3 (21). С. 31–41.
15. Pro T-SQL 2012 Programmer's Guide / M.Coles [et al.]. NY: Apress, 2012. 679 p.
16. Воронцов К.В., Потапенко А.А. Модификации EM-алгоритма для вероятностного тематического моделирования // Journal of Mathematical Physics. 2011. № 52. С. 1–21.
17. Wei X., Croft W.B. LDA-based document models for ad-hoc retrieval // Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGR 2006), August 6-11, 2006, Seattle, WA, USA. NY: ACM Press, 2006. P. 178–185.