# TB-Structure: Collective Intelligence for Exploratory Keyword Search

Vagan Terziyan[1], Mariia Golovianko[2] and Michael Cochez[3,4,1]

[1] Faculty of Information Technology, University of Jyvaskyla, Jyvaskyla, Finland
{vagan.terziyan, michael.cochez}@jyu.fi
[2] Department of Artificial Intelligence, Kharkiv National University of Radio Electronics, Kharkiv, Ukraine
mariia.golovianko@nure.ua
[3]Fraunhofer Institute for Applied Information Technology FIT, Sankt Augustin, Germany
[4]RWTH Aachen University, Informatik 5, Aachen, Germany

**Abstract.** In this paper we address an exploratory search challenge by presenting a new (structure-driven) collaborative filtering technique. The aim is to increase search effectiveness by predicting implicit seeker's intents at an early stage of the search process. This is achieved by uncovering behavioral patterns within large datasets of preserved collective search experience. We apply a specific tree-based data structure called a TB (There-and-Back) structure for compact storage of search history in the form of merged query trails – sequences of queries approaching iteratively a seeker's goal. The organization of TB-structures allows inferring new implicit trails for the prediction of a seeker's intents. We used experiments to demonstrate both: the storage compactness and inference potential of the proposed structure.

**Keywords:** keyword search, query trail, TB-structure, collective intelligence

## 1 Introduction

The collection of extremely large volumes of digital information having emerged during the last decades referred usually to as big data is highly promising: the expected impact of insights derived from large data sets is broadly recognized [1]. It has been shown that big data analytics gives deeper understanding of processes and helps recognizing hidden patterns exposing radically new knowledge which can be translated into significantly improved decisions and performance across industries. Thus big data is considered a new type of asset that brings a new culture of informed data-driven decision making.

Simultaneously, big data handling is exceptionally challenging due to its volume, velocity and variety [2, 3, 4]. This boosts the problem of big data accessibility implying therefore the need of new data models and techniques for big data storage and retrieval. Despite the fact that essential results have already been achieved to build search engines, there is still an obvious need for new approaches to large-scale search. Among others, Marz et al [5], Cambazoglu and Baeza-Yates [6] focus on the scalability problem related to the high computational costs of storing and processing large volumes of distributed data, Lewandowski [7] deals with providing fast access to

large amounts of information and effectiveness providing relevant search results and optimizing the search process for a user.

This paper represents a new collaborative filtering technique used for exploratory search in big data. It is based on the assumption that a search engine can recognize real intents and information needs of a new seeker faster or more accurately than can be done by the users themselves. Previous search experience of users in the form of users` queries organized in a special compact way will help to approach the intended search goal, with a minimal number of iterations. The main contribution of the research reported in this paper is the development of a tree-based data structure which is used for both compact storage of collective search experience and prediction of seekers' real information needs at an early stage of their search activity.

## 2 Related Work

The shift of the monopoly of the classical information retrieval to the concept of iterative and interactive search performed by search engines has been studied recently [8]. While information retrieval methods work well for closed world bases when the search output can be well predicted at the beginning of the search process, it is obviously insufficient when seekers have scarce knowledge of a topic, cannot specify a goal and a correct query to represent their information needs promptly. This is typical for search in open world systems, which are rapidly accumulating large-scale data and knowledge which is never complete. It was noticed that people's conceptions of their information problems change through the interactions with the dynamically changing environment containing other humans, sensors, and computational tools [9]. The search becomes more an *exploration* than a *retrieval*.

Traditional *Information retrieval* is best served by analytical strategies engaging in simple lookup activities: sending a carefully planned series of queries posed with precise syntax to a search engine and getting search results in the form of discrete and well-structured objects with minimal need for further examination and comparison. Effectiveness and efficiency of information retrieval is ensured by traditional techniques of crawling, indexing and ranking [10] engaging various combinations of syntactical, statistical and semantic analysis of either content itself or its structure and experimenting with various forms and models of page content and search query representation.

*Exploratory search* [11] aims at new information learning and investigation, i.e., new knowledge development. Such search involves multiple iterations and requires cognitive processing and interpretation of in-between search results before the search goal is achieved. A user iteratively applies a combination of querying and browsing strategies, makes on-the-fly selections and navigations, studies and assesses search results, compares and integrates the obtained information, and finally develops knowledge. In this view, search becomes a more user-dependent or user-driven process called *interactive information retrieval* [12], which is far more complicated than the usual matching of queries and documents and their further ranking. To ensure a precise query formulation a series of user-engine interactions is done: query formula-

tion, query modification, and inspection of the list of results. The overall process is usually referred to as *query expansion* [13], [14].

In this research we propose a new exploratory search approach which takes advantage of a new query expansion technique working with a tree-based data structure for query log storage. Our idea is to go further than simple query reformulation and to use hidden patterns in users' querying history for more effective organization of big data search (e.g., web pages). Recognition of hidden models or patterns in users' behavior can not only provide a basis for multipurpose analytics, e.g., extracting the exact semantics intended by the user [15], but also contribute to essential optimization of data processing and more accurate and fast information provisioning to end users.

## 3 Smart Data Structure for Search Trails Organization

### 3.1 TB-structures and Their Application for Search Trails Storage

Users' collective behavior keeps hidden patterns that can be used by a search engine for prediction of new users search intents at early stages of search. In our research we focus on the sequential and interactive nature of exploratory search which allows approaching a search goal by mutual co-evolution of three components: a user's knowledge and intents, queries and query answers. It is done by a series of iterations called seeking episodes – interactions of a seeker, a search engine and the content provided by the engine addressing a single search goal. A single seeking episode lasts from the first recorded time stamp to the last recorded time stamp on the search engine server from a particular searcher during a particular search period [16]. A 30-minutes inactivity timeout or the termination of the browser or the tab is usually used to demarcate seeking episodes in a web log.

A seeking episode is a set of triples $(S_i, Q_i, R_i)$, where $Q_i$ is a query sent by a seeker to a search engine; $R_i$ is a set of web pages generated by a search engine in response to the query; $S_i$ is a state of a user's knowledge after the study of $R_i$ and processing of a subset $R_i^{`}$ containing the content considered relevant by a seeker. The search process is iteratively continued implying a change of seeker`s knowledge and thus further query expansion. Three types of search sequences or trails show comprehensively the search dynamics: a queries trail, a response trail, and a knowledge (mind). The last one trail shows how a user's understanding of the problem has changed along with the query expansion. Analysis of all S, Q, R trails can be used for search optimization.

This paper covers the first stage of the study: query trails processing and organization for further search optimization. We assume that a seeker can benefit from collective search experience represented as organized query trails leading to satisfaction of users information needs. We call the field of computational learning dealing with algorithms of query trail organization, pattern recognition and a classifier induction "query trail learning" (QTL) and use a predictive model based on a data structure invented by Lovitskii and Terziyan [17] called a "there-and-back" structure or simply a TB-structure. In their paper a TB structure is defined as a merge of a tree forest and an inverted tree forest. A TB structure was initially applied to combinatorial pattern matching for indexing and generating string data and proved to be useful for fast full

text searches. A words space modeled with TB-structures is constructed from trees merged by the common prefixes and suffixes containing nodes denoted with alphabet symbols. A TB-structure is promising due to the possibility of its self-growth as a result of automatic generation of new links and new subtrees constructions.

We use a TB-structure for query trails storage and generation. It is a group of nodes where each node has a value in form of a keyword query: a word or a string of words sent by a seeker to a search engine, and a set of references to other nodes indicating possible transfers to next queries during search. A root layer contains nodes denoted by a set of possible initial queries, the leaves layer keeps terminal queries – the ones leading to either satisfaction (the search goal achievement) or disappointment (the search termination because of inability to reach the goal), and internal nodes are organized as intersected subtrees. Roots and leaves layers are formed by sets of non-repeating elements.

A QTB-structure allows mapping observations about a user's initiated query expansion to a target value, i.e., a query formulated in a way, so that an answer from a search engine would satisfy real information needs of a user. The structure has the power to generate new implicit relationships, thus new query trails can be automatically inferred over the basic structure. Prefixes of QTB-structures help clustering people by their initial intents and similar knowledge while suffixes define seekers` real information needs.

### 3.2 Algorithm of a QTB-Structure Feeding

Construction of a QTB-structure also called QTB feeding is an iterative process of incorporation of new query trails into an existing structure. Query sets are stored in form of strings $Trail_i$: $\{Q_{i1}, Q_{i2}, \dots, Q_{it}\}$ in batches $Batch_k\{\text{Trail}_i: \{Q_{i1}, Q_{i2}, \dots, Q_{it}\}\}$, where $Q_{\text{i1}}$ is the first query in the $i$-search session and $Q_{it}$ is the terminal one. A batch contains a set of all non-repeating trails captured by a search engine. The order of query trails influences a resulting QTB-structure view and its compactness.

The feeding procedure uses an incremental strategy which takes query trails out of a batch one by one and insert them into a QTB-structure $QTB$. $QTB$ is traversed first top-down and then bottom-up with the purpose of finding the longest common "top-down" path $Trail_i^{td}$ from the root node that matches a prefix of the trail:

$Trail_i^{td}$: $\{Seq[1, \dots m]\}$, where $Seq[1, \dots m] \subseteq \{Trail_i\}$ and $Seq[1, \dots m] \subseteq QTB$,

and the longest common bottom-up path $Trail_i^{bu}$ from a leaf that matches a suffix of a trail for a query trail and a QTB-structure:

$Trail_i^{bu}$: $\{Seq[\text{l}, \dots n]\}$, where $Seq[\text{l}, \dots n] \subseteq \{Trail_i - Trail_i^{td}\}$ and $Seq[\text{l}, \dots n] \subseteq QTB$.

$Seq_{ij}[1]$ is always a root node while $Seq_{ij}[\text{n}]$ is a leaf. A node traversed at the $j$-level in a QTB must also be at the $j$-position in the added trail. For example, see Figure 2, in case of the QTB-structure containing one trail $Trail_1$: $\{Q_1, Q_7, Q_9, Q_{12}, Q_{18}\}$

and a new trail $Trail_2$: $\{Q_1, Q_7, Q_{10}, Q_{13}, Q_{19}\}$ the longest common top-down path is $Trail_2^{td}$: $\{Q_1, Q_7\}$ and the longest common bottom-up path is $Trail_2^{bu}$: $\{\emptyset\}$.

After $Trail_i^{td}$ and $Trail_i^{bu}$ are defined, new nodes $Trail_i^{new}$ to be inserted into the QTB-structure can be obtained:

$$Trail_i^{new}: Trail_i - Trail_i^{td} - Trail_i^{bu} \text{ or } Seq_{ij}[\mathrm{m}+1, \ldots, l-1].$$

$Trail_i^{new}$ needs to be added to $QTB$. We insert $Trail_i^{new}$ by linking the last node of $Trail_i^{td}$ to the first node of the sequence $Trail_i^{new}$ and the last node of $Trail_i^{new}$ to the first node of $Trail_i^{bu}$ (see figure 1).

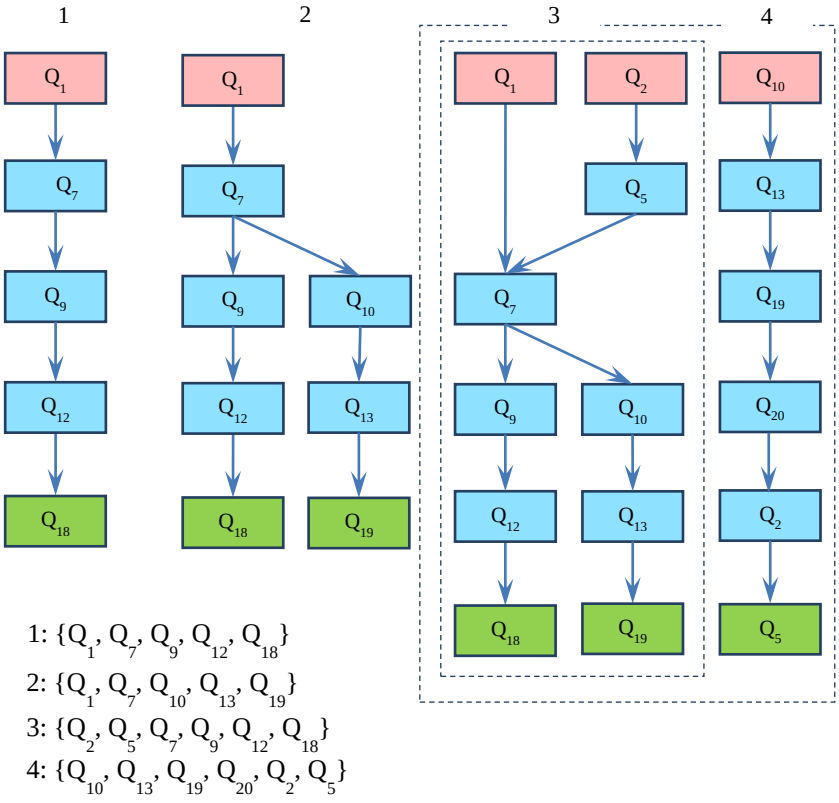


**Fig. 1.** An example of a QTB-structure feeding

Imagine that a new trail $Trail_4$: $\{Q_{10}, Q_{13}, Q_{19}, Q_{20}, Q_2, Q_5\}$ has to be incorporated into the structure from figure 1. The algorithm will reveal that $Trail_4^{td}$ and $Trail_4^{bu}$ are empty and will add a new independent trail into the structure. It makes a structure less compact and causes some duplication but eliminates the situation when cycles

appear. We are planning to solve the problem of the possible ambiguity of TB-structures` traversal with swarm intelligence, ant-algorithms particularly, which allow choosing the best alternatives according to collective experience.

## 4 Experiments

The idea behind the first round of the experiments was (1) to reveal all possible cases causing controversial or ambiguous situations to ensure the applicability of the algorithm to real keyword queries and (2) evaluate the generative power of TB-structures: an ability to infer new trails, implicit for initial trails collections. Therefore an artificial data set was created to ensure the reveal of all possible trails.

Experiments revealed two types of specific cases which could have a stand-alone effect on TB-structures construction: a TB-structure with several equal nodes at the same level of the structure implying ambiguity in its traversal and a one node trail. There are several ways to deal with such cases: either eliminate trails violating the algorithm requirements or extend a basic TB-structure and generate a structure which we called *a fail-over TB-structure* representing a list of TB-structures. The algorithm of a fail-over TB-structure feeding uses all trails including those violating constraints: if a trail is not accepted by a structure, an attempt is made to add it to the second one, if that one does not accept it either, the algorithm continues to search until a structure accepting the trail is found. If all TB-structures are exhausted and none accepts the trail, one more (empty) structure is added to the list and the trail is inserted into it. The *automatic generation of trails* was performed in two different ways: 1. With a uniform generator, which generated trails by choosing symbols uniformly at random and constructing trails of the length chosen randomly between $l_{min}$ and $l_{max}$; 2. With a random traversal generator which created a graph of the symbols by placing each symbol in a node and adding a directed edge from each node to a fixed number of randomly chosen other nodes. A trail was generated by a random graph traversal from a randomly chosen starting node.

We automatically generated 5390 restricted TB-structures and 5390 fail-over structures with permutations of the following settings. We varied the minimal trail length $l_{min}$ from 5 to 55 nodes and the maximal length $l_{max}$ from 5 to 60 nodes; the step size for the length was 5 nodes. The number of symbols in the alphabet was varied from 20 to 1280 (step $10*2^n$) and the number of initial trails from 100 to 51200 (step $100*2^n$).

The experiments showed a non-linear, exponential-like dependency between the number of initial trails and newly generated ones. The maximum number of generated trails was achieved when two conditions were satisfied: a big difference between the values of the minimal and the maximal possible length of trails ($l_{min} = 5;\ l_{max} = 60$) and the smallest possible alphabet ($alphabet = 20$). The biggest explosion of new trails was observed in a fail-over structure, which is not surprising given the fact that the fail-over structure contains at least all trails in a normal TB structure. In many cases the number of generated trails was so large that the number could not be counted within reasonable time.

The combinatorial explosion is not critical in our case because usually real search tasks imply shorter query trails than the ones we used in the experiments, and larger alphabets.

## 5 Conclusions and future work

A web seeker traditionally uses keywords to formulate search objectives to a search engine. Typical IR systems answer a query literally and return a set of results accordingly. Quite often "human-machine misunderstanding" or "misinterpretation" leads to the need of multiple iterations approaching a query which is understood similarly by a user and the search engine. Current techniques of search optimization use so called automatic query expansion (AQE) are based on semantic or syntactic similarity of terms used in queries. However open-world systems operating with big data require new search techniques due to data volumes, variety and velocity. Engines searching in open world settings should be proactive and predict hidden information needs of a seeker; becoming rather a navigator towards a real information need, not only an IR system.

We offer a technique for search optimization benefiting from collective search experience captured and processed by a search engine. Users' collective behavior contains hidden patterns that can be used by a search engine for the prediction of new users' search intents at early stages of the search process. This paper demonstrates how new forms of tree-based structures' organization can contribute to effective storage and operation over big data sequences, e.g., query trails, describing the search experience of multiple users. We build our theory on top of the idea of the sequential nature of the search process implying that a search goal is approached by mutual co-evolution of three components: person's knowledge and intents, queries and query results. This research is the first step towards comprehensive optimization of exploratory search. A tree-based TB-structure described in the paper is chosen as a smart data model used for compact storage of explicit query trails and inference of implicit trails useful for new users' intents prediction. Recognition of hidden patterns by analyzing explicit query trails and inferring new ones can be applied to the process of knowledge discovery, e.g., to support ontology learning process. And vice versa, semantic conceptualization of keyword queries based on ontologies can enhance the described search algorithm and provide additional semantics to TB-structures.

The structure itself is promising and gives wide possibilities of applications besides keyword search. It can be applied for various tasks implying sequential processes and configurations in biology, medicine, industry, academic field, for logistics and planning, etc. Experiments show that generative power of the proposed data structures is very high, in some cases we experience explosion of new implicit knowledge emergence. This is both an opportunity (due to learning capabilities of the systems built over TB-structures) and a challenge (because the bigger volumes of information cause processing complexity). In our ongoing research we are addressing this problem by application of swarm intelligence for TB-structures self-organization and prediction of a user's real information needs.

**Acknowledgements.** This article is based upon work from COST Action KEYSTONE IC1302, supported by COST (European Cooperation in Science and Technology).

## 6 References

1. Mayer-Schönberger, V., & Cukier, K. (2013). *Big data: A revolution that will transform how we live, work, and think*. Houghton Mifflin Harcourt.
2. McAfee, A., Brynjolfsson, E., Davenport, T. H., Patil, D. J., & Barton, D. (2012). Big data. *The management revolution. Harvard Bus Rev*, *90*(10), 61-67.
3. Chen, H., Chiang, R. H., & Storey, V. C. (2012). Business Intelligence and Analytics: From Big Data to Big Impact. *MIS quarterly*, *36*(4), 1165-1188.
4. Chen, M., Mao, S., & Liu, Y. (2014). Big data: A survey. Mobile Networks and Applications, 19(2), 171-209.
5. Marz, N., & Warren, J. (2015). Big Data: Principles and best practices of scalable realtime data systems. Manning Publications Co..
6. Cambazoglu, B. B., & Baeza-Yates, R. (2015). Scalability Challenges in Web Search Engines. Synthesis Lectures on Information Concept, Retrieval, and Services, 7(6), 1-138.
7. Lewandowski, D. (2015). Evaluating the retrieval effectiveness of web search engines using a representative query sample. Journal of the Association for Information Science and Technology, 66(9), 1763-1775.
8. Bao, Z., Zeng, Y., Jagadish, H. V., & Ling, T. W. (2015, May). Exploratory Keyword Search with Interactive Input. In *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data* (pp. 871-876). ACM.
9. Belkin, N. J., Cool, C., Stein, A., & Thiel, U. (1995). Cases, scripts, and information-seeking strategies: On the design of interactive information retrieval systems. Expert systems with applications, 9(3), 379-395.
10. Brin, S., & Page, L. (2012). Reprint of: The anatomy of a large-scale hypertextual web search engine. Computer networks, 56(18), 3825-3833.
11. Marchionini, G. (2006). Exploratory search: from finding to understanding.Communications of the ACM, 49(4), 41-46.
12. Bao, Z., Zeng, Y., Jagadish, H. V., & Ling, T. W. (2015, May). Exploratory Keyword Search with Interactive Input. In *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data* (pp. 871-876). ACM.
13. Efthimiadis, E. N. (2000). Interactive query expansion: A user-based evaluation in a relevance feedback environment. Journal of the American Society for Information Science, 51(11), 989-1003.
14. Fattahi, R., Parirokh, M., Dayyani, M. H., Khosravi, A., & Zareivenovel, M. (2016). Effectiveness of Google keyword suggestion on users' relevance judgment: A mixed method approach to query expansion. The Electronic Library,34(2), 302-314.
15. Bobed, C., Trillo, R., Mena, E., & Ilarri, S. (2010, December). From keywords to queries: Discovering the user's intended meaning. In International Conference on Web Information Systems Engineering (pp. 190-203). Springer Berlin Heidelberg.
16. Jansen, B. J., & Spink, A. (2006). How are we searching the World Wide Web? A comparison of nine search engine transaction logs. Information Processing & Management, 42(1), 248-263.
17. Lovitskii, V. A., Terziyan, V. (1981). Words' Coding in TB-Structure. *Problemy Bioniki*, 26, 60-68. (*In Russian*).