Improving Search Engine Performance Through Dynamic Caching

1st Claudio Gutiérrez-Soto
Departamento de Sistemas de Información
Universidad del Bío-Bío
Concepción, Chile
cogutier@ubiobio.cl

2nd Marco A. Palomino
School of Engineering Computing and Mathematics
University of Plymouth
Plymouth, UK
marco.palomino@plymouth.ac.uk

3rd Ernesto Roa
Departamento de Sistemas de Información
Universidad del Bío-Bío
Concepción, Chile
erroa@alumnos.ubiobio.cl

4th Patricio Galdames
Departamento de Sistemas de Información
Universidad del Bío-Bío
Concepción, Chile
pgaldames@ubiobio.cl

***Abstract*—Web search engines process several millions of queries per second over several billions of documents. Without any optimization, this process can be very expensive in terms of processing times. In this regard, appropriate use of computing power is essential. One way to tackle this problem is through the use of caching mechanisms. Keep in mind, most research based on caching mechanisms uses repetitive queries–it means queries syntactically equals–to conform caches. Furthermore, the universe of repetitive queries is small in comparison with a set of similar semantically queries. This paper presents a dynamic cache that relies on an online algorithm, which performs a semantic match between the user's query and queries stored in the cache. Broadly speaking, the algorithm employs a priority queue, where popular queries are stored along with their relevant documents. Empirical results show that our proposed approach improves the response times and precision. Moreover, the use of semantically related keywords proves to be a key contribution that had been overlooked in previous research.**

***Index Terms*—Web search engines, dynamic caching, semantically related keywords.**

I. Introduction

Nowadays, it is well-known that social networks are highly used by million of users who share their concerns, hobbies, and feelings [1] [2]. Despite this, Web Search Engines (WSEs) follow being highly used to look for information from several heterogeneous sources and are also used to make decisions [3]. Millions of queries per second are submitted by users, who demand not only high quality in their answers but also these answers are timely. Furthermore, WSEs are constantly challenged since the number of pages increases exponentially day by day. From this baseline, a critical factor to consider is the running time to process the queries, notably in peak hours [4]. Accordingly, it is relevant to handle the computing power efficiently to avoid the server's overcrowded, considering at the same time that answers are being provided in acceptable response times.

A mechanism highly used to tackle this challenge is caching. A caching can be seen as a little mechanism of high performance (this can be both hardware or software), which aims to avoid accessing disk or to main memory. Three metrics are used to assess the caching policies' efficiency, cache hit, cache miss, and cache hit ratio. Thus, when the searched data is inside the cache, a hit takes place. Note that when a hit happens, it is unnecessary to access a disk or main memory. On the contrary, a cache miss occurs when data is not inside the cache. As regards a cache hit ratio, it is a proportion between a cache hit and cache miss. It should be noted that both cache hit and cache miss are just considered when an exact match between terms occurs. Most approaches that utilize caching to improve processing times are based on posting lists, which can be seen as skip list of terms, or the query results pages, which consider terms in queries.

Notably, in a pioneering approach presented by [5], the authors provide a caching that relies on a priority queue, which considers similar queries, using the relevant retrieved documents from past queries to respond to new queries. Consequently, it is feasible to improve not only the response time but also precision. Keep in mind, the similarity between queries and documents also considers exact terms.

According to our knowledge, no prior work in the literature deals with caching based on semantic similarity, which means that a hit can occur without an exact match between terms. Similar to the approach presented by [5], in this paper, we present an online algorithm that uses a priority queue, which is possible to improve both the response times and precision. One characteristic of this approach is that terms semantically related between past queries and relevant documents retrieved are obtained between periods where there are no updates inside the cache. Thus, the algorithm uses these terms to update caching and doing a match the new queries from a semantic point of view. By doing so, new relevant documents are incorporated as a result, which would not be considered part of the result without a semantic match.

Incidentally, a traditional collection in IR is conformed by a documents' set, a queries' set, and a set of relevant documents for each query, whose relevance is determined by users' judgments according to the pertinence's documents regarding the query. In WSEs, documents can be seen as web pages and users' judgments as ranking functions. Most ranking functions use the "clickthrough rate" — among other parameters — associated with each page, with the aim to provide relevance. Nevertheless, this way to give relevance is not suitable when we wish to evaluate effectiveness over sets of queries semantically similar. As a consequence, traditional IR datasets and log file datasets are not convenient to evaluate this kind of approach. Indeed, this kind of approach requires two query's sets along with their respective sets of judgments' users. From this baseline, we have opted to simulate an IR system, where two semantically similar query sets along with their judgments' users can be properly evaluated. Several experimental scenarios were carried out comparing our approach with caching based on repetitive queries, and caching relies on similar queries as in [5]. Empirical results show that our approach improves both response times and precision.

The remainder of this paper is organized as follows: Section II reviews the body of research on related work. Section III presents the methodology carried out in our study. Section IV describes the experimental setup. Section V presents our results, and Section VI discusses them. Finally, Section VII summarizes our conclusions.

II. Related Work

The relevant literature can be largely divided in four categories: inverted indexes, results caching, hybrid approaches and priority queues. We elaborate on each of these categories in the following subsections.

A. Inverted indexes

Inverted indexes are the de facto data structure to support keyword search over large volumes of text [6]. An inverted index stores a mapping from content to locations, which, in the case of WSEs, generally involves a hashmap that cross-references Web pages to keywords. For each indexed term, the inverted index contains a posting list. Each posting is essentially a set of pointers marking all the term occurrences in the collection of indexed web pages [7]. By a large, the use of inverted files involves compressing and decompressing the posting lists to efficiently compute the results of the queries and reduce response times [8]–[12]. Incidentally, there is an additional cost because posting lists have to be crossed, which can be costly, especially for long posting lists.

In summary, the efficient use of inverted files underlying suitable management of compressing/decompressing indexes, expressed in terms of hit/miss ratio, without directly considering response times and precision.

B. Results caching

Given that query processing costs may differ significantly from one query to another, scientists have explored the idea of caching lists of results [13]–[16], or even storing the actual documents that constitute the results [17]. It is worth mentioning that all these approaches follow the same trend that Inverted files to assess the efficiency. Remarkable is the approach presented by [14], which deals with response times over log files. Unlike these approaches, we have moved one step further by maintaining queries with the highest popularity, relevant documents, and a set of semantically similar terms to these queries.

C. Hybrid approaches

Hybrid approaches combine the two previous categories. Examples of this are the work of Baeza *et al.* [18], Gil-Costa *et al.* [19] and Papadakis and Tzitzikas [20]. These authors devised caching strategies to reduce the number of computations required to retrieve results submitted to a WSE using pre-computed information. Again, the metric used to evaluated efficiency is the hit/miss ratio.

D. Priority queues

Recent research has implemented caching as a priority queue [21]–[23], whereby the priority of a query depends on its popularity or frequency—the higher the frequency, the higher the priority. Only queries that the queue cannot answer can eventually access a second-level caching, as in Fagni *et al.* [21], or for some other mechanism to compute the results.

Excepting Gan and Suel [24], who employed a large query log from a single WSE to evaluate their caching policies, the rest of the researchers cited in this section have tested their proposed methods using simulation instead of actual query logs. Indeed, Fagni *et al.* [21], Baeza-Yates *et al.* [18], Papadakis and Tzitzikas [20], and Ozcan *et al.* [16] used probability distributions to simulate query logs and carry out their experiments. Ayers et al. employed Intel's Pin Tool [25] to capture full instructions and data traces from a machine in a steady-state, which serves to real user queries. Li *et al.* [14] built a cache simulator based on the production logs of the Bing advertising system.

To sum up, none of the previously mentioned works deal with semantic similarity. Furthermore, most of them evaluated precision (it is also well-known as effectiveness) without use neither users' judgments nor ranking functions; precision is mainly obtained under the clickthrough rate concept. Besides, most of them do not use response time as a metric to assess efficiency.
We will now explain in detail how our simulation is set up.

III. Methodology

As Gutiérrez-Soto *et al.* proposed in [5], we carried out the simulation in two stages. In the first stage, we create the terms, documents and queries. We used two distributions to create the documents: Heaps' distribution, which is used to represent the number of terms that a document contains, and the exponential distribution, which is used to choose terms from topics. It should be noted that neither stop words nor stemming were employed.

The second stage involves creating queries from documents simulating users' judgments. *Bradford's law* [26] was applied over retrieved documents to specify how relevant they are. We describe these procedures in depth below.

A. Simulating Documents

The fundamental elements to simulate a document are the letters of the English alphabet. From these elements, it is possible to create a *term*—i.e., a *word*.

To create a term, the English alphabet's letters are chosen by using the uniform distribution. In this way, a set of terms forms a topic. The main idea of a topic is to represent a particular issue, for example, "computer science" or "biology". It is important to point out that a term does not belong to two topics. Consequently, the intersection between topics is empty.

A document is formed by terms belonging to different topics, such as in real documents. However, a document should contain more terms of a particular topic in comparison with others. To choose the topic, we employ the exponential distribution. Subsequently, the uniform distribution is applied to select a term over the chosen topic. A document of size $O(t)$, where t is the number of terms, and whose vocabulary size is (O^{β}), where $0 < \beta < 1$, is represented using Heaps' law [27]. Therefore, a simulated document expresses a document $O(t^2)$—i.e., the total number of terms including the repeated terms [28].

B. Simulating Queries

Documents are used to create past queries. For each past query —a past query corresponds to a query previously submitted in the WSE—, one document is selected using a Uniform distribution. The query's terms are acquired from the selected document, and these terms also are chosen using the Uniform distribution. Note that the intersection between the set of past queries is empty—i.e., there are no common terms among them.

To create a new query — a new query corresponds to a query, which the WSE has not processed —, a past query is chosen, and one term is changed or added. Therefore, the most similar query for a new query is its corresponding past query.

C. Relevance judgment simulation

Precision is reached as follows. There are two lists of recovered documents for each query (i.e., a list of documents corresponds to a set of documents in decreasing order, ordered according to its similarity regarding the query). Cosine distance is used to provide the similarity between the query and documents. Thereafter, Bradford's law is applied over each list with the aim to acquire the relevance of each document (i.e., a document is classified as relevant or non-relevant). Aiming to shed light, we provide the following example. Let $\mathcal{A}$ be a list, which has the documents $\{d_3, d_{10}, d_{11}, d_{12}, d_{17}, d_{20}\}$ and let $\mathcal{B}$ be a list that contains the next documents $\{d_1, d_5, d_7, d_{11}, d_{12}, d_{17}\}$. Firstly, an intersection between both lists is determined, which is composed by $\{d_{11}, d_{12}, d_{17}\}$. Subsequently, Bradford's law is applied over this intersection obtaining $\{d_{11}(r), d_{12}(r), d_{17}(n)\}$ (i.e., such as $d_i(r)$ means that d_i is a relevant document, by contrast $d_i(n)$ implies that the document d_i is non-relevant). In the final step, Bradford's law is applied over each list without overwriting the intersection between them. In this way, the final result for list $\mathcal{A}$ is given as $\{d_3(n), d_{10}(r), d_{11}(r), d_{12}(r), d_{17}(n), d_{20}(n)\}$, meanwhile the list $\mathcal{B}$ is given as $\{d_1(r), d_5(r), d_7(n), d_{11}(r), d_{12}(r), d_{17}(n)\}$. Consequently, both lists are formed by relevant and non-relevant documents. Hence, both lists have different precision (in our case P@10).

IV. Experimental Setup

In the following paragraph, we define the experimental environment used in this paper:

- Each term is formed between 3 and 7 alphabet English letters.
- The total number of terms is 3500.
- The number of topics corresponds to 7, which has formed by 500 terms each one.
- Each document covers a range between 15 and 30 terms.
- The number of documents used in each experiment covers 700, 1400, 2100, 2800, and 3500.
- The number of queries in each experiment involves 30, 60, 90, 120, 150, 180, and 210.
- Each query has between 3 and 8 terms.

In the semantic approach, 10% of terms by topic contain similarly semantic terms. The terms having again semantic terms are selected using a Uniform distribution, and these can involve between 1 and 3 terms. Three scenarios with different types of queries are analyzed. The first scenario implies repetitive queries (e.g., given a query q' previously submitted in the WSE, this query q' is newly submitted in the WSE); for instance, when the number of queries is 30, 15 queries have a repetitive query. The same rule applies to all queries in each experiment. The second scenario involves similar queries (i.e., two queries are similar if they have at least one term in common, note that if all terms are equal, then the query is repetitive, and the similarity is 1), in the same way, that in the first scenario, each query has only one similar query. Finally, semantic queries are evaluated; similarly to the previous, each query has a similar semantic query.

Aiming to build the queue, Exponential distribution (with parameters $\lambda = 1.0$ and $\lambda = 1.5$) was applied over each set of queries. Thereafter, the top K-queries (i.e., in all experiments K was instantiated with the value 30) are submitted in the system with the aim to evaluate the ten most similar documents (i.e., according to the cosine measure) for each query. To simulate the judgments of users over the retrieved documents, Zeta distribution was used on the first ten retrieved documents (with the aim to assess P@10) for each query (the parameters used in each experiment were $S = 2$, $S = 3$ y $S = 4$). In this way, it is possible to put the top K-queries along with their documents in the queue. In order to simulate the dynamic environment, one hundred selections of queries are carried out for each set of queries (e.g., 100 selections are executed over the set of 30 queries). By doing this, it is feasible to simulate the arrival of new queries submitted by users in the WSE, checking first if repetitive, similar, or semantic queries are inside the query; when this occurs, a hit is done in the cache. On the contrary, the documents that are retrieved from the system increased the cache miss (i.e., a miss indicates that the query result was not in the queue).

Simulations were implemented on C language and running on Processor Intel Core i5-2410M de 2.30 GHz, 4 GB RAM memory DDR3; Linux Operating System Ubuntu 16.04.2 LT and compiler gcc 5.4.0.

V. Results

From Tables 1 and 2, we can observe that precision increases in all cases regarding the cosine distance (It is important to point out that precisions are calculated before putting the queries along with their documents in the queue). The best results are provided by repetitive queries, this occurs because when a query is submitted again in the system, the relevant documents are available to answer this query. In this way, the relevant documents are shifted at the beginning of the result list, which allows to improve substantially precision. Nevertheless, according to Teevan *et al.* [29], the number of query repetitive on the web is very low (i.e., this is around 7%). This is in line with the results shown in Tables 3 and 4, where the lowest number of hits is produced by repetitive queries. Remember that a repetitive query corresponds to a previously submitted query in the WSE. The second best precision is presented by semantic queries in Tables 1 and 2. Although precision decreases in Table 2, This is consistent with the hits in Tables 3 and 4. On the other hand, the lowest results of precision are presented by similar queries. However, these results correspond to improvement regarding the cosine distance. Roughly, from Tables 1 and 2, every time that S increases (i.e., this can be seen as follows, when S rises, a more demanding query takes place since there are fewer documents relevant for it) precision decreases.

Regarding hits, the best results are provided by semantic queries followed by similar queries. Both cases provide better hits than repetitive queries. As mentioned in the previous paragraph, it is remarkable due to repetitive queries are rare.

Finally, the response times are presented in Table 5, the lowest times are presented by repetitive queries, followed by similar queries. The worst response times in the queue are presented by semantic queries. It is because, in addition to accessing the queue, a semantic search for each query keyword is carried out. The latter adds additional time, which can be seen in Table 5. Nevertheless, these times are much less than direct access to the system (the system column in Table 5), where query's terms are contrasted with the documents' terms, the retrieval is carried out providing a list of ordered documents according to their similarity with the query, such as occurs in a WSE.

TABLE I
Percentage increase in precision over traditional information retrieval system, using Exponential distribution $\lambda = 1.0$, and Zeta distribution with parameters S.

S	$Precision$		
	$Similar$	$Semantic$	$Repetitive$
2	6.649%	20.995%	27.755%
3	5.455%	18.868%	26.609%
4	0.727%	15.146%	23.190%
Average	4.277%	18.338%	25.851%

TABLE II
Percentage increase in precision over traditional information retrieval system, using Exponential distribution $\lambda = 1.5$, and Zeta distribution with parameters S.

S	$Precision$		
	$Similar$	$Semantic$	$Repetitive$
2	9.583%	16.706%	29.196%
3	8.597%	15.272%	27.880%
4	2.288%	10.531%	22.435%
Average	6.822%	14.170%	26.503%

TABLE III
The average number of hit in the queue, using Exponential distribution $\lambda = 1.0$.

# of q	Hit		
	$Similar$	$Semantic$	$Repetitive$
30	98.989	98.989	92.776
60	90.007	83.135	80.137
90	80.007	82.537	73.125
120	76.156	78.145	72.005
150	82.876	79.978	75.899
180	73.007	81.012	72.899
210	75.976	80.342	72.125
Average	82.431	83.448	76.995

TABLE IV
The average number of hit in the queue, using Exponential distribution $\lambda = 1.5$.

# of q	Hit		
	$Similar$	$Semantic$	$Repetitive$
30	97.989	97.989	97.007
60	78.007	70.005	74.784
90	80.007	80.007	77.137
120	76.007	73.125	74.007
150	78.007	85.125	75.125
180	85.007	85.007	83.989
210	74.007	77.989	68.125
Average	81.290	81.321	78.596

TABLE V
AVERAGE RESPONSE TIMES IN MILLISECONDS, USING EXPONENTIAL DISTRIBUTION λ.

λ	*Response Times*			
	Similar	*Semantic*	*Repetitive*	*System*
1.0	0.0000055	0.0000069	0.0000053	0.00000191
1.5	0.0000052	0.0000061	0.0000049	0.00000188
Average	0.0000054	0.0000065	0.0000051	0.00000190

VI. DISCUSSION

As we mentioned earlier, works based on log files or classical IR datasets do not provide adequate user judgments to assess approaches based on similar and semantic queries. On the one hand, when precision is assessed using datasets based on log files, clickthrough is used to relevance the documents. Incidentally, when precision is evaluated using classical IR datasets, the terms from retrieved documents are used to give relevance when documents or queries are new. In this manner, a document can be considered relevant with a few terms without it and vice versa. Strictly speaking, there should be two sets of user judgments, one for each set of queries (i.e., the set of queries Q has a set of judgments of user JU. Following this reasoning, for a set of semantically similar queries to Q (Q'), a corresponding set of judgments of user JU' for Q') exist. Therefore, the best way to represent this situation is using simulation. It is worth mentioning that the use of synthetic datasets is not new; in [30], and [31] (to mention a few), documents, queries, and judgments' users are simulated to evaluate precision using past queries and probabilistic algorithms, respectively.

It is essential to point out that we assume a set of terms semantically similar to queries' terms in the priority queue. Several techniques can be used to mining this set, to mention two, deductive logic and query-document clustering. Both techniques can operate over terms inside the queue (it includes queries and relevant documents retrieved) to limit the processing time. To exemplify this, in Gutiérrez-Soto *et al.* [32] a measure to capture the context in which the queries are submitted is presented. Note that this measure incorporates complimentary terms to the submitted queries and their relevant documents, which perfectly can be used to link semantically related terms. Indeed, most complexities times for clustering algorithms are between $O(n^2)$ and $O(n^2 log_2 n)$; therefore, for a small set of queries and documents (inside of queue), this cost is not considerable and can be fully executed before doing an update.

On the other hand, regarding the time complexity of our caching, we have adapted a heap sort over which updates and searches take place. Each time that a new query is submitted, it should be compared searching for each term of the new query a semantic term (i.e., the semantic dictionary implies a search of $O(m^2)$, where m is the number of terms) in a queue node to obtain the maximum similarity. In this way, the time complexity to calculate the maximum similarity of a new query q' with one node in the queue (q) is $O(tm^2(sim(q', q)))$ such as t is the number of terms for the new query, which can be changed by terms of the dictionary —It is important to mention that to calculate the cosine distance between two semantic queries, at least should have one term in common between queries—. According to Vandic *et al.* [33], the calculation of cosine distance can be executed in linear time. Thus, assuming that t is the greater number of terms between both queries, it implies $O(t^2m^2)$ for each new query. Overall, the time complexity implies $O(|\widehat{Q}|t^2m^2log_2K)$, where $|\widehat{Q}|$ corresponds to the set of new queries.

On the other hand, we're aware that our synthetic dataset is small; however, we have not found evidence to suggest that the size of data over a synthetic dataset directly impacts the final results. By the way, evaluating the caching's size scapes from our concern; nonetheless, much research points out that there is a direct relation regarding the caching's size.

Finally, according to Beitzel *et al.* [34], in a WSE the queries tend to be more similar at peak hours; therefore, we believe that our approach is a good start since it provides more hits when the queries are similar (at least have a one-term in common) and semantically similar (these could not have common terms).

VII. CONCLUSION AND FUTURE WORK

Every day, WSEs index thousands of documents, so finding relevant information (i.e., relevant documents) for users becomes a difficult task. Web search engines deal with two important metrics on which underlying their efficiency. The first metric aims to the query answer being suitable; in simple words, it implies improving precision. The second metric intends to provide the query answers as soon as possible in terms of processing time. Thus, this paper addressed both metrics using a priority queue (which stores the popular queries along with their relevant documents). Specifically, we evaluated three variants of queries inside the queue. The first involves repetitive queries (these are unique inside the queue). The second metric deals with similar queries in the queue. The third covers semantic queries (each query has a unique, similar query from a semantic point of view). The experimental results show that semantic queries provide better hit (i.e., it means that it is possible to find more answers in the queue) than the others. However, considering precision, better results are presented by repetitive queries in the queue. Final experimental results show that it is feasible to improve the processing times. Ideas for future research directions in this area are the following: comparing our approach with other caching techniques considering different peak hours.

VIII. ACKNOWLEDGEMENTS

This research was supported by Universidad del Bío-Bío, Chile, under Grants No. DIUBB GI 195212/EF and DIUBB

2130253 IF/R. Marco Palomino acknowledges the funding provided by the Interreg 2 Seas Mers Zeeën AGE'IN project (2S05-014).

References

[1] A. Fogli and L. Veldkamp, "Germs, Social Networks, and Growth," *The Review of Economic Studies*, vol. 88, no. 3, pp. 1074–1100, 04 2021. [Online]. Available: https://doi.org/10.1093/restud/rdab008

[2] S. Samanta, V. K. Dubey, and B. Sarkar, "Measure of influences in social networks," *Applied Soft Computing*, vol. 99, p. 106858, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1568494620307961

[3] E. Turner and L. Rainie, "Most Americans Rely on their Own Research to Make Big Decisions, and that often Means Online Searches," *Pew Research Center: Internet, Science & Tech (blog)*, 2020.

[4] M. Zahedi, B. Mansouri, S. Moradkhani, M. Farhoodi, and F. Oroumchian, "How questions are posed to a search engine? an empiricial analysis of question queries in a large scale persian search engine log," *2017 3th International Conference on Web Research (ICWR)*, pp. 84–89, 2017.

[5] C. Gutiérrez-Soto, A. C. Díaz, and P. Galdames, "Exploiting the use of similar past search results through a dynamic cache," in *2019 IEEE CHILEAN Conference on Electrical, Electronics Engineering, Information and Communication Technologies (CHILECON)*, 2019, pp. 1–5.

[6] S. Büttcher, C. L. Clarke, and G. V. Cormack, *Information Retrieval: Implementing and Evaluating Search Engines*. MIT Press, 2016.

[7] P. Baldi, P. Frasconi, and P. Smyth, *Modeling the Internet and the Web*. Wiley Online Library, 2003.

[8] M. Catena, C. Macdonald, and I. Ounis, "On Inverted Index Compression for Search Engine Efficiency," in *European Conference on Information Retrieval*. Springer, 2014, pp. 359–371.

[9] D. Arroyuelo, M. Oyarzún, S. González, and V. Sepulveda, "Hybrid Compression of Inverted Lists for Reordered Document Collections," *Information Processing & Management*, vol. 54, no. 6, pp. 1308–1324, 2018.

[10] Z. Dai and J. Callan, "Inverted List Caching for Topical Index Shards," in *European Conference on Information Retrieval*. Springer, 2018, pp. 577–583.

[11] S. Roy, A. Banerjee, P. Ghosh, A. Chatterjee, and S. Sen, "Intelligent Web Service Searching Using Inverted Index," in *Contemporary Advances in Innovative and Applicable Information Technology*. Springer, 2019, pp. 13–21.

[12] G. Tolosa, E. Feuerstein, L. Becchetti, and A. Marchetti-Spaccamela, "Performance Improvements for Search Systems Using an Integrated Cache of Lists + Intersections," *Information Retrieval Journal*, vol. 20, no. 3, pp. 172–198, 2017.

[13] G. Ayers, J. H. Ahn, C. Kozyrakis, and P. Ranganathan, "Memory Hierarchy for Web Search," in *2018 IEEE International Symposium on High Performance Computer Architecture (HPCA)*. IEEE, 2018, pp. 643–656.

[14] C. Li, D. G. Andersen, Q. Fu, S. Elnikety, and Y. He, "Better Caching in Search Advertising Systems with Rapid Refresh Predictions," in *Proceedings of the 2018 World Wide Web Conference*, 2018, pp. 1875–1884.

[15] X. Qiao, P. Ren, J. Chen, W. Tan, M. B. Blake, and W. Xu, "Session persistence for dynamic web applications in Named Data Networking," *Journal of Network and Computer Applications*, vol. 125, pp. 220–235, 2019.

[16] R. Ozcan, I. S. Altingovde, and Ö. Ulusoy, "Cost-Aware Strategies for Query Result Caching in Web Search Engines," *ACM Transactions on the Web (TWEB)*, vol. 5, no. 2, pp. 1–25, 2011.

[17] T. Strohman and W. B. Croft, "Efficient Document Retrieval in Main Memory," in *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, 2007, pp. 175–182.

[18] R. Baeza-Yates, A. Gionis, F. Junqueira, V. Murdock, V. Plachouras, and F. Silvestri, "The Impact of Caching on Search Engines," in *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, Amsterdam, The Netherlands, 2007, pp. 183–190.

[19] V. Gil-Costa, M. Marin, C. Bonacic, and R. Solar, "A Graph-Based Cache for Large-Scale Similarity Search Engines," *The Journal of Supercomputing*, vol. 74, no. 5, pp. 2006–2034, 2018.

[20] M. Papadakis and Y. Tzitzikas, "Answering Keyword Queries through Cached Subqueries in Best Match Retrieval Models," *Journal of Intelligent Information Systems*, vol. 44, no. 1, pp. 67–106, 2015.

[21] T. Fagni, R. Perego, F. Silvestri, and S. Orlando, "Boosting the Performance of Web Search Engines: Caching and Prefetching Query Results by Exploiting Historical Usage Data," *ACM Transactions on Information Systems (TOIS)*, vol. 24, no. 1, pp. 51–78, 2006.

[22] A. Cidon, A. Eisenman, M. Alizadeh, and S. Katti, "Cliffhanger: Scaling Performance Cliffs in Web Memory Caches," in *13th Symposium on Networked Systems Design and Implementation*, 2016, pp. 379–392.

[23] R. Solar, V. Gil-Costa, and M. Marín, "Evaluation of Static/Dynamic Cache for Similarity Search Engines," in *International Conference on Current Trends in Theory and Practice of Informatics*. Springer, 2016, pp. 615–627.

[24] Q. Gan and T. Suel, "Improved Techniques for Result Caching in Web Search Engines," in *Proceedings of the 18th International Conference on World Wide Web*, Madrid, Spain, 2009, pp. 431–440.

[25] C.-K. Luk, R. Cohn, R. Muth, H. Patil, A. Klauser, G. Lowney, S. Wallace, V. J. Reddi, and K. Hazelwood, "Pin: Building Customized Program Analysis Tools with Dynamic Instrumentation," *ACM SIGPLAN Notices*, vol. 40, no. 6, pp. 190–200, 2005.

[26] E. Garfield, "Bradford's Law and Related Statistical Patterns," *Essays of an Information Scientist*, vol. 4, no. 19, pp. 476–483, May 1980. [Online]. Available: http://www.garfield.library.upenn.edu/essays/v4p476y1979-80.pdf

[27] H. S. Heaps, *Information Retrieval: Computational and Theoretical Aspects*. Orlando, FL, USA: Academic Press, Inc., 1978.

[28] G. Navarro, E. S. De Moura, M. Neubert, N. Ziviani, and R. Baeza-Yates, "Adding compression to block addressing inverted indexes," *Inf. Retr.*, vol. 3, no. 1, pp. 49–77, Jul. 2000.

[29] J. Teevan, E. Adar, R. Jones, and M. Potts, "History repeats itself: Repeat queries in yahoo's logs," in *Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, ser. SIGIR '06. New York, NY, USA: ACM, 2006, pp. 703–704. [Online]. Available: http://doi.acm.org/10.1145/1148170.1148326

[30] C. Gutiérrez-Soto and G. Hubert, "Evaluating the interest of revamping past search results," in *Database and Expert Systems Applications*, H. Decker, L. Lhotská, S. Link, J. Basl, and A. M. Tjoa, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 73–80.

[31] C. Gutierrez-Soto and G. Hubert, "On the reuse of past searches in information retrieval: Study of two probabilistic algorithms," *Int. J. Inf. Syst. Model. Des.*, vol. 6, pp. 72–92, 2015.

[32] Claudio Gutierrez-Soto, Marco Palomino, Arturo Curiel, Hector Riquelme Cerda, Fernando Bejar Rain, "Evaluating the Effectiveness of Query-Document Clustering Using the QDSM Measure," *Advances in Science, Technology and Engineering Systems Journal*, vol. 5, no. 6, pp. 883–893, 2020.

[33] D. Vandic, F. Frasincar, and F. Hogenboom, "Scaling pair-wise similarity-based algorithms in tagging spaces," in *Web Engineering*, M. Brambilla, T. Tokuda, and R. Tolksdorf, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 46–60.

[34] S. M. Beitzel, E. C. Jensen, A. Chowdhury, D. Grossman, and O. Frieder, "Hourly analysis of a very large topically categorized web query log," in *Proceedings of the 27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, ser. SIGIR '04. New York, NY, USA: ACM, 2004, pp. 321–328. [Online]. Available: http://doi.acm.org/10.1145/1008992.1009048