Research on Stemming and Relevance Ranking Optimization for Retrieval Services (Postprint)

Authors: Zhu Yan, Zhang Jingwei, Yang Qing, Hu Xiaoli, Shan Meijing, Shan Meijing

Date: 2022-10-26T00:00:00+00:00

Abstract

The emergence of new-generation information technologies and the rapid development of the Internet industry have resulted in an explosive growth in data volume. To meet the needs of billions of users to rapidly retrieve effective information from massive datasets, enhancing the retrieval quality and query efficiency of search engines is of significant importance, yet also presents challenges. On the one hand, user queries are becoming increasingly complex, and the morphological variation characteristics of linguistic vocabulary lead to diversified search terms, while existing stemming algorithms commonly suffer from insufficient stemming and low stemming accuracy; on the other hand, retrieving document results that satisfy user query requirements from massive data is a highly time-consuming task, and existing approaches that distribute documents across multiple servers to handle query latency frequently encounter tail latency issues. To address these issues, in the text preprocessing stage, we designed a lexical normalization algorithm APS that re-encodes rule functions to optimize feature term extraction; in the relevance ranking stage, we designed an anytime ranking algorithm SAR based on the term-at-a-time query processing strategy, which can terminate the query process early after processing a specified number of inverted segments within a given time budget, substantially reducing query evaluation time. Experiments on multiple real-world datasets validated the effectiveness of the APS algorithm in improving stemming accuracy and the effectiveness of the SAR algorithm in controlling query latency.

Full Text

Title and Authorship

Research on Stemming and Related Ranking Optimization for Retrieval Services

ZHANG Jingwei[1], HU Xiaoli[1], SHAN Meijing[2]
[1]Guangxi Key Laboratory of Trusted Software, Guilin University of Electronic Technology, Guilin 541004, China
[2]Guangxi Key Laboratory of Automatic Detection Technology and Instruments, Guilin University of Electronic Technology, Guilin 541004, China
[3]School of Criminal Law, East China University of Political Science and Law, Shanghai 201620, China

Abstract

The rise of new-generation information technologies and the rapid development of the Internet industry have led to explosive growth in data volume. To meet the needs of billions of users to quickly obtain effective information from massive datasets, improving the retrieval quality and query efficiency of search engines is of great significance but also presents challenges. The morphological variation characteristics of user query language lead to diversified search terms, while existing stemming algorithms generally suffer from under-stemming and low accuracy. Retrieving documents that meet user query requirements from massive data is an extremely time-consuming task, and existing methods that partition documents across multiple servers often suffer from tail latency problems. To address these issues, we designed a morphological normalization algorithm (Advanced Porter Stemmer, APS) in the text preprocessing stage, which recodes rule functions and optimizes feature word extraction. In the relevance ranking stage, we designed an anytime ranking algorithm based on the Score-at-a-Time (SAAT) query processing strategy, which can terminate the query process early after processing a specified number of inverted segments within a given time budget, significantly reducing query evaluation time. Experiments on multiple real datasets verify the effectiveness of the APS algorithm in improving stemming accuracy and the authenticity of the SAR algorithm in controlling query delay.

Keywords: stemming algorithms; anytime ranking; text preprocessing; SAAT; related ranking

1. Introduction

The emergence of new technologies such as online self-media and big data, coupled with the vigorous development of Internet industries including e-commerce, entertainment, and communications, has caused information volume to grow exponentially. According to statistics [1], the global annual data generation reaches zettabytes, with non-paper information accounting for 99.7%. Despite advances in big data technology and deep learning/neural network computing accelerating information processing capabilities, the mitigation of information

overload remains minimal. How to obtain effective information from exponentially growing data in a short time has become an urgent problem, and search engines are one of the primary means for people to access information.

Research shows [2] that as the Internet industry develops rapidly, search users' information needs become increasingly complex, and search terms gradually diversify. If retrieval terms are not normalized morphologically, multiple documents with different word forms may appear in search results, affecting the accuracy of corpus learning. In information retrieval and text mining research, stemming is one of the core technologies for morphological normalization. By optimizing stemming algorithms to normalize words, the number of terms can be reduced, text vector space dimensionality can be lowered, and retrieval accuracy and efficiency can be effectively improved.

However, existing stemming algorithms generally suffer from under-stemming and low accuracy, creating a contradiction between missing keyword features and high computational complexity with low query efficiency. Moreover, for large-scale distributed systems, the tail latency phenomenon is more prevalent and can severely affect overall service performance. Research indicates [3-4] that excessively long response times directly impact user experience, causing significant potential profit losses. While current approaches to query latency mostly partition document collections across multiple servers to share time delays, this method still suffers from tail latency problems [5-11].

Based on these considerations, this paper conducts in-depth research in two aspects: text preprocessing and related ranking. In the text preprocessing stage, we designed the APS algorithm (Advanced Porter Stemmer), which adjusts rule function definitions based on inflectional and derivational morphology to address under-stemming and low accuracy issues in the Porter Stemmer algorithm. In the related ranking stage, we designed the SAR algorithm (SAAT-anytime ranking) based on the Score-at-a-Time query processing strategy, which can terminate the retrieval process early within a given time budget or after processing a specified number of inverted segments, thereby controlling query delay. The anytime ranking algorithm can significantly reduce system resource consumption and maintenance costs, solving the prevalent high-percentage tail latency problem [12] to adapt to Service Level Agreements (SLAs).

---

2. Related Work

2.1 Stemming Algorithms

According to the implementation principles of stemming methods, they can be categorized into: rule-based affix removal methods [13-17], dictionary-based lookup methods [18], statistical methods based on word distribution patterns [19-21], and hybrid methods [22-24].

Dictionary-based methods rely on authoritative dictionaries, yielding more ac-

curate results but requiring time-consuming dictionary traversal and having dictionary dependency issues. Statistical methods identify patterns through word frequency statistics but produce large errors and low accuracy, making them more suitable for small language word stemming. Hybrid methods combine multiple approaches for higher accuracy but have complex algorithmic processes requiring extensive background knowledge support.

Rule-based affix removal methods utilize the inherent patterns of inflectional and derivational morphology in words, offering wide applicability and fast processing of regular words. Representative algorithms include Lovins [13], Dawson [14], Lancaster (Paice/Husk) [15], and Porter [16-17] stemmers. However, these algorithms still suffer from under-stemming and low accuracy, requiring further optimization. This paper focuses on improving and optimizing the rule-based affix removal method.

2.2 Query Optimization Techniques

Top-k query ranking is a widely applied optimization technique in information retrieval. Current mainstream query efficiency optimization technologies include pruning algorithms, selective search, and anytime ranking algorithms.

Dynamic pruning algorithms aim to process as few relevant documents as possible, using skip-list access to inverted lists to reduce processing of irrelevant or low-relevance documents. Common dynamic pruning algorithms include MaxScore [27-28] and WAND [29]. However, studies show [30] that tail queries processed by pruning algorithms can take several orders of magnitude longer than average query latency.

Selective search partitions document collections by topic during index construction, with each shard containing thematically related documents [31-32]. Queries are predicted and routed to relevant shards, reducing workload and improving efficiency. However, results may deviate from exhaustive search due to processing only partial shards.

Anytime ranking algorithms implement impact-ordered indexes, processing document priorities based on influence scores [32-33]. The SAAT (Score-at-a-Time) query processing strategy can output relatively accurate ranking results without traversing all documents, aligning with anytime ranking goals. When response time is predetermined by SLA, the query processing must support interruption, and anytime ranking algorithms provide solutions for such scenarios.

---

3. Methods

3.1 APS Algorithm Design

The APS algorithm makes the following optimizations to address Porter Stemmer limitations:

1. **Supplementation of irregular verb conjugations and plural exceptions**: The Porter Stemmer ignores certain irregular forms. For example, verbs like "buy" → "bought" are handled through enumeration of irregular verb forms.
2. **Optimization of -ing suffix handling**: For verbs ending in -ing, the APS algorithm first converts participles to root forms before applying deletion rules, avoiding incorrect transformations like "try" → "tri" .
3. **Optimization of double consonant endings**: For words ending in double consonants (non- 'l' , non- 's'), the algorithm avoids incorrect removal of one consonant, preventing errors like "ebb" → "eb" .
4. **Composite suffix handling**: The algorithm recodes mapping rules from composite to non-composite suffixes, addressing issues with words like "generalization" → "generalize" → "general" .
5. **Supplementation of additional suffixes**: Added processing for suffixes like -tor, -sory, -ship, -ate, -ative, -atic, etc.

Definitions: - **Vowel**: a, e, i, o, u - **Consonant**: All letters except vowels - **Inflexion**: Morphological changes like plurals and tense forms (e.g., "looked") - **Morphological Derivation**: Forms derived by adding substantial affixes (e.g., "condition" → "conditional") - **Double**: Forms with double letters (tt, mm, nn) like "illegal" , "irregular" - **Suffix**: Simple non-composite suffixes (e.g., -ation in "generalization")

The overall stemming flow of the APS algorithm is shown in [Figure 1: see original paper].

3.2 SAR Algorithm Design

Retrieving documents that meet user query requirements from massive data is extremely time-consuming. Studies show [36] that artificially extending Google search query time by 100-400ms reduces daily search volume by 0.2%-0.6%. Existing methods partition documents across multiple servers, but query latency remains unpredictable.

The SAR algorithm is designed based on the SAAT query processing strategy. It can terminate the query process early after processing a specified number of inverted segments or within a given time budget, significantly reducing query evaluation latency. The algorithm returns relatively accurate retrieval results while sacrificing an acceptable range of search quality, solving the prevalent tail latency problem.

Impact Score Calculation: Given query q and document d, the impact score is:

$$score(d, q) = \sum_{t \in q \cap d} w_{t,d} \cdot w_{t,q}$$

where $w_{t,d}$ is the weight of term t for document d (quantized to b bytes during indexing), and $w_{t,q}$ is the weight of term t for query q.

Index Organization: The index consists of inverted lists where each term points to segments. Each inverted entry is a quadruple {score, start, end, num} representing a segment, where: - score: Impact score value - start: Pointer to segment data start - end: Pointer to segment data end - num: Number of documents in segment

Segments are sorted by document ID ascending within each segment, and segments are sorted by impact score descending across the index.

3.3 SAAT Query Processing Strategy

The SAR algorithm implements three processing modes:

1. **OR Mode**: All documents are assigned score accumulators and scored. This is the initial processing mode.
2. **AND Mode**: New documents beyond the current accumulator set are no longer assigned accumulators. Only documents already assigned accumulators are scored.
3. **REFINE Mode**: When the top-k documents are determined, query processing terminates and results are returned in descending order of accumulator values.

The algorithm maintains a heap of accumulators to track the highest-scoring documents in real-time. Before processing each segment, the accumulated impact score is compared against threshold parameter . If greater than , processing continues; otherwise, the loop terminates.

The core algorithm pseudocode is shown in Algorithm 1.

---

4. Experiments

4.1 Experimental Environment and Datasets

Software Environment: Intel Xeon E3-1226 v3 @ 3.30GHz, Linux Enterprise

Datasets: 1. **Word Paice**: Compiled from library/information science word lists 2. **Word Paice Scrabble**: Contains approximately 60,000 real dataset vocabulary samples from Scrabble word checkers 3. **ClueWeb09b** and **ClueWeb12-B13**: Large-scale web collections with TREC topics 51-200 and 201-300

All documents were preprocessed by converting invalid characters to spaces, separating alphabetic and numeric characters, and removing tag markers, with UTF-8 encoding.

4.2 APS Algorithm Evaluation

The APS algorithm was compared against existing stemming algorithms (Lovins, Paice/Husk) and the original Porter Stemmer using three metrics on the Word Paice dataset: - Under-stemming index (UI) - Over-stemming index (OI) - Relative truncation error rate (ERRT)

Results: - APS reduced under-stemming index by 48.4% compared to Porter Stemmer - Relative truncation error rate improved by 30.2% - APS outperformed Lovins and Paice/Husk algorithms in overall accuracy - The UI improvement indicates APS can merge more related words into the same stem group (e.g., “ability” and “able”)

4.3 SAR Algorithm Evaluation

The SAR algorithm was evaluated on ClueWeb09b and ClueWeb12-B13 datasets using: - Retrieval quality: nDCG@10 - Efficiency: Query latency (excluding index loading and output writing)

Parameter Tuning: The number of processed inverted items was varied to determine optimal values. Setting to 0.1% of dataset size achieved good balance between quality and efficiency.

Time Budget Experiments: Time budgets were set at 50ms, 100ms, 150ms, and 200ms. Results show: - Query latency is effectively controlled with early termination - Retrieval quality decreases slightly but within acceptable range - Quality improves proportionally with increased time budget - Larger datasets (ClueWeb12-B13) show more quality degradation than smaller ones (ClueWeb09b) at same time budgets

Early Termination: The algorithm significantly reduces processed inverted segments, with [Figure 8: see original paper] and [Figure 9: see original paper] showing the ratio of early-terminated segments to total segments on both datasets.

---

5. Discussion and Future Work

While the SAR algorithm shows slight delays in special cases, it effectively controls query latency overall. The experiments validate its effectiveness in controlling tail latency, reducing computational resource consumption, and improving user experience. Retrieval quality improves proportionally with budget time, though with some acceptable quality trade-off.

Future Research Directions: 1. **Index Compression**: Implement data compression algorithms for inverted indexes to reduce disk space and I/O time, potentially using SIMD-based decoders [38-39]. 2. **Language Model Smoothing**: Apply smoothing methods like Dirichlet or Jelinek-Mercer [40] to

improve modeling of rare terms and adapt to uncommon query words. 3. **User Interface Optimization**: Develop more interactive and visual user interfaces beyond current text-based systems to enhance information presentation and user experience.

6. Conclusion

This paper addresses the challenges of morphological variation in query terms and tail latency in distributed retrieval systems. In the text preprocessing stage, the APS algorithm optimizes stemming by adjusting rule functions based on derivational morphology, supplementing irregular verbs, and improving suffix handling. In the relevance ranking stage, the SAR algorithm based on SAAT strategy enables early termination within time budgets or after processing specified inverted segments, effectively controlling query latency. Experiments on multiple real datasets validate the effectiveness of APS in improving stemming accuracy and SAR in controlling tail latency, while maintaining acceptable retrieval quality. Future work will explore index compression, language model improvements, and enhanced user interfaces.

References

[1] ADADI A, BERRADA M. Peeking inside the black-box: a survey on explainable artificial intelligence[J]. IEEE Access, 2018, 6: 52138-52160.

[2] BRUTLAG J. Speed matters for Google web search[EB/OL]. (2009-06-23)[2012-02-15]. https://venturebeat.com/wp-content/uploads/2009/11/delayexp.pdf.

[3] FRIED J, RUAN Z, OUSTERHOUT K. Caladan: mitigating interference at microsecond timescales[C]//Proceedings of the 14th USENIX Symposium on Operating Systems Design and Implementation (OSDI 20). Berkeley, CA: USENIX Press, 2020: 281-297.

[4] ARAPAKIS I, BAI X, CAMBAZOGLU B. Impact of response latency on user behavior in web search[C]//Proceedings of the 37th International ACM SIGIR Conference on Research and Development in Information Retrieval. New York, NY: ACM Press, 2014: 103-112.

[5] KOHAVI R, DENG A, FRASCA B. Online controlled experiments at large scale[C]//Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. New York, NY: ACM Press, 2013: 1168-1176.

[6] HWANG W, KIM M, JEON M. Predictability of search query latencies for query acceleration[J]. ACM Transactions on the Web, 2016, 10(3): 1-28.

[7] JEON M, KIM S, HWANG W. Predictive early termination for efficient search[C]//Proceedings of the 37th International ACM SIGIR Conference on Research and Development in Information Retrieval. New York, NY: ACM Press, 2014: 253-262.

[8] MACKENZIE J, SCHOLER F, CULPEPPER S. Early termination heuristics for score-at-a-time index traversal[C]//Proceedings of the 39th Australasian Computer Science Conference. New York, NY: ACM Press, 2017: 1-8.

[9] YUN H, KIM J, HWANG W. Optimal aggregation policy for reducing tail latency in distributed search[C]//Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval. New York, NY: ACM Press, 2015: 301-310.

[10] KIM J, KULKARNI A, CALLAN J. Selective search: efficient and effective search of large textual collections[J]. ACM Transactions on Information Systems, 2015, 33(4): 1-33.

[11] KIM J, CULPEPPER S, CRANE M. Delayed-Dynamic-Selective (DDS): a real-time constraint propagation method for reducing search tail latency[C]//Proceedings of the 24th ACM International Conference on Information and Knowledge Management. New York, NY: ACM Press, 2015: 7-16.

[12] TROTMAN A, CRANE M, CULPEPPER S. Anytime ranking for search[J]. Software: Practice and Experience, 2019, 49(5): 942-950.

[13] LOVINS J. Development of a stemming algorithm[J]. Mechanical Translation and Computational Linguistics, 1968, 11(1/2): 22-31.

[14] DAWSON J. Suffix removal for word conflation[J]. Bulletin of the Association for Literary and Linguistic Computing, 1974, 2(3): 33-46.

[15] PAICE C. An evaluation method for stemming algorithms[C]//Proceedings of the 17th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval. New York, NY: ACM Press, 1994: 42-50.

[16] PORTER M. An algorithm for suffix stripping[J]. Program, 1980, 14(3): 130-137.

[17] PORTER M. Snowball: a language for stemming algorithms[EB/OL]. (2001-07-16)[2022-02-15]. http://snowball.tartarus.org/texts/introduction.html.

[18] KROVETZ R. Viewing morphology as an inference process[C]//Proceedings of the 16th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval. New York, NY: ACM Press, 1993: 191-202.

[19] MAYFIELD J, MCNAMEE P. Single n-gram stemming[C]//Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval. New York, NY: ACM Press, 2003: 415-416.

[20] MELUCCI M, ORIO N. A novel method for stemmer generation based on hidden Markov models[C]//Proceedings of the Twelfth International Conference on Information and Knowledge Management. New York, NY: ACM Press, 2003: 131-138.

[21] MAJUMDER P, MITRA M, PARUI S. YASS: yet another suffix stripper[J]. ACM Transactions on Information Systems, 2007, 25(4): 18-24.

[22] BONUS P, SILVA G, OLIVEIRA A. A lexicon-based stemming algorithm for the Portuguese language[C]//Proceedings of the 2003 Symposium on Natural Language Processing. Manila: DLSU Press, 2003: 272-274.

[23] AL-SHAMMARI T, LIN J. Towards error-free Arabic stemming[C]//Proceedings of the 2006 International Conference on Information and Knowledge Management. New York, NY: ACM Press, 2006: 439-448.

[24] STROHMAN T, TURTLE H, CROFT B. Optimization strategies for complex queries[C]//Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval. New York, NY: ACM Press, 2006: 219-226.

[25] BRODER A, CARMEL D, HERSCOVICI M. Efficient evaluation of complex queries using two-level retrieval[C]//Proceedings of the 12th International Conference on Information and Knowledge Management. New York, NY: ACM Press, 2003: 426-434.

[26] DING S, SUEL T. Faster top-k document retrieval using block-max indexes[C]//Proceedings of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval. New York, NY: ACM Press, 2011: 993-1002.

[27] DIMOPOULOS C, NEPOMNYACHIY S, SUEL T. Optimizing top-k document retrieval strategies for block-max indexes[C]//Proceedings of the Sixth International Conference on Web Search and Data Mining. New York, NY: ACM Press, 2013: 113-122.

[28] MALLIA A, OTTAVIANO G, PORCIANI M. Faster BlockMax with variable-sized blocks[C]//Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval. New York, NY: ACM Press, 2017: 605-614.

[29] CRANE M, CULPEPPER S, LIN J. A comparison of document-at-a-time and score-at-a-time evaluation strategies[C]//Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval. New York, NY: ACM Press, 2017: 201-210.

[30] GOODWIN B, TROTMAN A. Anytime ranking for search[J]. Software: Practice and Experience, 2019, 49(5): 942-950.

[31] KULKARNI A, CALLAN J. Selective search: efficient and effective search of large textual collections[J]. ACM Transactions on Information Systems, 2015,

33(4): 1-33.

[32] LIN J, CALLAN J. Pipelined feedback for selective search[C]//Proceedings of the 39th International ACM SIGIR Conference on Research and Development in Information Retrieval. New York, NY: ACM Press, 2016: 401-410.

[33] PEDERSEN T. A comparative study of stemming algorithms for English information retrieval[J]. Journal of the American Society for Information Science, 1996, 47(8): 627-643.

[34] ANH V, KRETSER O, MOFFAT A. Vector-space ranking with effective early termination[C]//Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval. New York, NY: ACM Press, 2001: 35-42.

[35] PIBIRI E, VENTURINI R. Techniques for inverted index compression[J]. ACM Computing Surveys, 2020, 53(6): 1-36.

[36] TROTMAN J. Query understanding at Bing[EB/OL]. (2012-03-15)[2022-02-15]. https://dl.acm.org/doi/pdf/10.1145/2009916.2010190.

[37] ZHAI C, LAFFERTY J. A study of smoothing methods for language models applied to information retrieval[J]. ACM Transactions on Information Systems, 2004, 22(2): 179-214.

[38] TROTMAN J, TROTMAN A. Anytime signatures for search[C]//Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval. New York, NY: ACM Press, 2015: 301-304.

[39] KIM J, CRANE M, JIA M. Towards efficient search engine evaluation[C]//Proceedings of the 39th International ACM SIGIR Conference on Research and Development in Information Retrieval. New York, NY: ACM Press, 2016: 401-410.

[40] ZHAI C, LAFFERTY J. A study of smoothing methods for language models applied to information retrieval[J]. ACM Transactions on Information Systems, 2004, 22(2): 179-214.

Note: Figure and table references in the original text are preserved as [FIGURE:N] and [TABLE:N] markers, though the actual figures/tables are not included in this translation. Mathematical expressions and algorithm pseudocode have been preserved in their original form as requested.

Note: Figure translations are in progress. See original paper for figures.