2019 Iran Workshop on Communication and Information Theory (IWCIT)

THE STATISTICAL DICTIONARY-BASED STRING MATCHING PROBLEM

M. Suri and S. Rini

Electrical and Computer Engineering Department,
National Chiao Tung University (NCTU), Hsinchu, Taiwan
{meersuri.eic06g, stefano} @nctu.edu.tw

ABSTRACT

In the Dictionary-based String Matching (DSM) problem, an Information Retrieval (IR) system has access to a source sequence and stores the position of a certain number of strings in a posting table. When a user inquires the position of a string, the IR system, instead of searching in the source sequence directly, relies on the the posting table to answer the query more efficiently. In this paper, the Statistical DSM problem is proposed as a statistical and information-theoretic formulation of the classic DSM problem in which both the source and the query have a statistical description while the strings stored in the posting sequence are described as a code. Through this formulation, we define the communication efficiency of the IR system as the average cost in retrieving the entries of the posting list from the posting table, in the limit of an infinitely long source sequence. This formulation is used to study the communication efficiency for the case in which the dictionary is composed of (i) all the strings of a given length, referred to as k-grams , and (ii) run-length codes.

Index Terms— Dictionary-based string matching, Content based retrieval, Indexing database, Information retrieval, Phrase searching

1. INTRODUCTION

Let us define a Dictionary-based String Matching (DSM) problem as follows. The *IR system* has access to a *source sequence* and constructs a *posting table* in which it stores the positions of a set of source substrings, referred to as a *posting code*. More precisely, each row in the posting table contains the *posting list* corresponding to a given codeword, consisting of a list of positions in which the codeword appears in the source sequence, as selected by the *posting function*. At a later time, a user submits a string to the IR system, termed a *query*, and the IR system is tasked with providing the positions of the query in the source sequence, called *matches*. If the query does not appear in the source sequence, a empty message is returned to the user. If the IR system is not able to retrieve some of the matches, an error message is returned. The performance of an IR system can be evaluated through a number of parameters: the computational complexity in answering a query, the average system memory requirements or the average communication rate between the posting table and the IR system.

In this paper, we consider a variation of the classic DSM in which we assume that (i) the source sequence and the query have a statistical description and that (ii) the cost of a query is proportional to the product of the length of the entries in the posting table visited by the IR system in answering a query. Under these two assumptions, we consider the problem of designing the posting code that minimizes the expected communication cost between the posting table and the IR system when retrieving the positions of a query in the limit of an infinitely long source sequence. As this paper introduces a novel problem formulation, it represents a stepping stone toward the development of a more complete and detailed understanding of the communication cost in retrieving a query from a posting list. Variations of this problem in which computation and storing costs are minimized are left for future research.

Literature Review: The DSM problem has been studied in a number of context and modeled through various assumptions so that a vast literature is available on the topic. See [7] and [9] for an extensive review of the topic. To the best of our knowledge, no formulation has explicitly considered either the distribution of source and queries, or the performance in the limit of large source length. In the information retrieval context, the DSM problem is referred to as "inverted index" problem and the concern is with the respect to the memory required to store the entries of the posting table [5], efficient retrieval and relevance ranking of results. The distribution of the queries is used in [6] to design a three level memory organization for a search engine inverted file index. In computation lingustics and natural language processing, the DSM problem has been studied to determine robust retrieval methods [4, 10] to search a text affected by errors. In the context of genomics, bioinformatics and computational biology, the

The work of the authors is funded my the Ministry Of Science and Technology (MOST) under the grant number 107-2221-E-009 -032 -MY3.

978-1-7281-0584-0/19/$31.00 ©2019 IEEE

problem is sometimes referred to as off-line or indexed pattern matching: here the focus is on the retrieval of sequences that approximatively match the given query [3, 11]. More generally, the source distribution implicitly appears in the literature concerned with the compression of the entries of the posting table, such as [2], [12] and [8].

Contributions & Organization: The remainder of the paper is organized as follows:

• **Sec. 2- Problem formulation:** We propose formulation of the SDSM problem which accounts for source and query along with the cost of accessing the entries in the posting table. We define the communication efficiency of an IR system as the expected cost of communicating the information from the posting table to the IR system in the limit of large source length. Through this performance measure, we formulate an optimization problem that helps us determine the code with the minimal expected communication cost.

• **Sec. 3- Relevant examples:** To validate the proposed model, we study in detail the performance of two IR systems: one storing (i) k-grams, all possible source sequences of length k and (ii) run-length-coded information. For illustrative purposes, we consider the simple case of binary i.i.d. source and query distributions. These are two simple yet powerful classes of posting codes which provide relevant insight on the optimal solution for the SDSM problem for more complex set of sources, queries and codes.

• **Sec. 4- Numerical Evaluations:** we numerically investigate the design of the optimal code for the case of binary i.i.d. source and query distributions.

Notation: With $\mathbf{x} = [x_1, \dots, x_N] \subseteq \mathcal{X}^N$ we indicate a sequence of elements from $\mathcal{X}$ with length N. The notation $\mathbf{x}_i^j$ indicates the substring $[x_i, \dots, x_j]$. Given the sequence $\mathbf{x}$, $l(\mathbf{x})$ indicates the length of the sequence $\mathbf{x}$, $w(\mathbf{x})$ indicates the Hamming weight, respectively. The notation $\mathbf{a}.\mathbf{b}$ indicates the vector concatenation operation. The notation $\mathbf{a} \preceq \mathbf{b}$ indicates that $\mathbf{a}$ is a substring of $\mathbf{b}$. Let $\mathcal{P}(\mathcal{X})$ indicate the power set of $\mathcal{X}$. Finally, define $\overline{x} = 1 - x$.

2. PROBLEM FORMULATION

The Statistical Dictionary-based String Matching (SDSM) problem is comprised of a *source sequence*, a *query* and an *IR system*.

• The source sequence is defined as the sequence X^n obtained as a subsequence from the stationary ergodic process $\{X_i\}_{i\in\mathbb{N}}$ over the space of infinite sequences with support $\mathcal{X}$.

• A *query* $Q = Q^L$ is defined as the of random vector of random length L with support $\mathcal{X}^L$ and distribution $P_Q(\mathbf{q}) = \sum_{j\in\mathbb{N}} P_L(j) P_{Q^j|L}(q^j)$, where $P_L(j)$ is the probability distribution of the query length and $P_{Q^j|L}(q^j)$ is the distribution of the query random vector given its length is j and support $\mathcal{X}^j$

• The IR system is comprised of a *posting code*, a *storing algorithm*, *posting lists* and an *IR algorithm*. A posting code of size M is defined as the set $\mathcal{C} = \{\mathbf{c}_i\}_{i=1}^M$ with $\mathbf{c}_i \in \mathcal{P}(\mathcal{X}^n)$: $\mathbf{c}_i$ is referred here to as a posting codeword. For each $\mathbf{c}_i$, define the set $\mathcal{I}(\mathbf{c}_i, X^n) = \{W_m(i)\}_{m\in\mathbb{N}}$ as the set of positions $W_m(i)$ in the source sequence in which the codeword $\mathbf{c}_i$ appears, i.e.

$$X_{W_m(i)}^{W_m(i)+l(\mathbf{c}_i)} = \mathbf{c}_i, \quad \forall\, i \in [1:M],\ m \in \mathbb{N}. \tag{1}$$

The *storing algorithm* is the mapping $f_S(\cdot)$ which produces the *posting list* $\mathcal{T}(\mathbf{c}_i, X^n)$ from the set $\mathcal{I}(\mathbf{c}_i, X^n)$, i.e.

$$f_S:\ \mathcal{I}(\mathbf{c}_i, X^n) \to \mathcal{T}(\mathbf{c}_i, X^n) \subseteq \mathcal{I}(\mathbf{c}_i, X^n). \tag{2}$$

In other words, the storing function determines which elements of $\mathcal{I}(\mathbf{c}_i, X^n)$ are present in $\mathcal{T}(\mathbf{c}_i, X^n)$. The set of posting lists is referred to as a posting table, i.e. $\mathcal{T}(\mathcal{C}, X^n) = \{\mathcal{T}(\mathbf{c}_i, X^n)\}_{i=1}^M$. Given a query $Q = \mathbf{q}$, the *IR algorithm* produces a *covering* of size V, of the whole query, defined as the tuple $\mathcal{S}(\mathbf{q}) = \{\mathbf{n}^{(V)}, \mathbf{c}^{(V)}\}$ such that

$$\mathbf{q}_{n_{(i)}}^{l(\mathbf{c}_{(i)})} = \mathbf{c}_{(i)}, \quad \forall\, i \in [1:V]. \tag{3}$$

Once a covering is produced, the IR algorithm fetches the position of the codewords in the covering from the posting lists. Finally, the IR algorithm applies the mapping f_R to identify the contiguous positions in the posting lists of each element in the covering, that is

$$f_R:\ \{\mathcal{T}(\mathbf{c}_{(i)}, X^n),\ \mathbf{c}_{(i)} \in \mathcal{S}(\mathbf{q})\}_{i=1}^V \to \mathbf{m} \subset \mathbb{N}^n, \tag{4}$$

where $\mathbf{m}$ is such that $X_{m_i}^{l(\mathbf{q})} = \mathbf{q}$ for all $m_i \in \mathbf{m}$.

Next, we detail the measures needed to identify the cost of the query. The average size of the posting list and the average size of the posting table are defined, respectively, as

$$\mathbb{E}\left[|\mathcal{T}(\mathbf{c}_i, X^n)|\right] = \mathbb{E}[l(\mathcal{T}(\mathbf{c}_i, X^n))]$$
$$\mathbb{E}\left[|\mathcal{T}(\mathcal{C}, X^n)|\right] = \sum_{i=1}^{M} \mathbb{E}\left[|\mathcal{T}(\mathbf{c}_i, X^n)|\right].$$

If a covering for the query exists, then the cost of a covering $\mathcal{S}(\mathbf{q})$ is defined as the function $c\left(\mathcal{S}(\mathbf{q})\right)$. The *minimum expected cost* for a given query $Q = \mathbf{q}$, $c(\mathbf{q})$, is defined as

$$\mathbb{E}[c^*(\mathbf{q})] = \min_{\mathcal{S}(\mathbf{q})} \mathbb{E}\left[c\left(\mathcal{S}(\mathbf{q})\right)\right]. \tag{5}$$

For given source, query distributions, and the size of the posting code, the maximal efficiency in the SDSM problem for the cost function c is defined as

$$\eta_c = \min \lim_{n\to\infty} \mathbb{E}[c^*(\mathbf{q})], \tag{6}$$

where the minimum is over $\mathcal{C},\ f_R(\cdot)$ and $f_S(\cdot)$.

2.1. Cost function

The cost function c can be chosen so as to reflect a number of concerns in the design, construction and operation of the IR system. As an example, the cost function

$$c\left(\mathcal{S}(\mathbf{q})\right) = \frac{\kappa V}{n} \min_i \left|\mathcal{T}(\mathbf{c}_{(i)}, X^n)\right| \cdot \log\left(\frac{\max_i \left|\mathcal{T}(\mathbf{c}_{(i)}, X^n)\right|}{\min_i \left|\mathcal{T}(\mathbf{c}_{(i)}, X^n)\right|} + 1\right) \tag{7}$$

for some $\kappa > 0$, reflects the complexity cost of the phrase search algorithm implemented with the galloping search for identifying the query matches from the posting list entries [7, Ch. 2.1]. On the other hand, the cost function

$$c\left(\mathcal{S}(\mathbf{q})\right) = \frac{\kappa}{n} \sum_{i \in \mathcal{S}(\mathbf{q})} \left|\mathcal{T}(\mathbf{c}_{(i)}, X^n)\right| \tag{8}$$

reflects the system main memory requirements for processing the query.

Many large scale distributed database systems hold posting tables that occupy terabytes of disk space. In such systems, the communication rate over the bus connecting the storage system and the query processing units naturally arises as the bottleneck in the query processed speed. In the following, we choose the cost function as the logarithm of the product of posting list lengths as this is proportional to number of bits needed to communicate the indices of the posting table in a distributed system, where all the clients have a copy of the posting table, i.e.

$$c\left(\mathcal{S}(\mathbf{q})\right) = \frac{\kappa}{\log n} \log \prod_{i \in \mathcal{S}(\mathbf{q})} |T(c_{(i)}, X^n)| \tag{9a}$$

$$= \frac{\kappa}{\log(n)} \sum_{i \in \mathcal{S}(\mathbf{q})} \log \left|\mathcal{T}(\mathbf{c}_{(i)}, X^n)\right|. \tag{9b}$$

2.2. The Pre-fix free coded, Complete and Parsed (PCP) SDSM problem

In the above formulation, the SDSM problem is presented in the greatest possible generality. In the following, we focus on a specific formulation of the SDSM problem, the Prefix free coded, Complete and Parsed (PCP) SDSM problem, which can be more readily analyzed. In particular, we consider the case in which (i) the posting code is a complete pre-fix free code (see App. A), (ii) the posting table stores all the matches, (iii) queries are covered by non-overlapping codewords. While property (i) and (ii) are straightforward, for (iii), we resort to the following definition. A covering $\mathcal{S}(\mathbf{q}) = \{\mathbf{n}^{(V)}, \mathbf{c}^{(V)}\}$ is defined as a *parsing* if there exists a K such that $n_j + l(\mathbf{c}_{(j)}) = n_{j+1}$ for $j \in [1 : K-1]$, while

$$n_j = n_K, \;\; n_j + l(\mathbf{c}_j) > l(q), \tag{10}$$

for $j \in [K : V]$ and no codeword outside the set $\{\mathbf{c}_j\}_K^V$ satisfies (10). In other words, a parsing of a query is a covering with no overlapping of codewords, apart from the tail of the query. In the tail of the query, codewords start from the same position n_K and overflow the end of the query sequence. The parsing between K and V contains all codewords that contain $Q_{n_K}^{l(Q)}$ as a prefix. The string $Q_{n_K}^{l(Q)}$ is referred to as the tail of the query. Our interest in the PCP-SDSM problem is motivated by the next theorem.

Theorem 2.1. *In the PCP-SDSM problem, the following holds:* ***(i)*** *no retrieval error occurs,* ***(ii)*** *the minimum covering cost is always finite,* ***(iii)*** *there exists only one parsing of any query, thus this parsing is the optimal covering, and* ***(iv)*** *the number of entries in the posting table is always equal to n.*

Proof. See App. A. □

3. RELEVANT EXAMPLES

In the remainder of the paper, we evaluate the efficiency for two example codes. In both cases, we consider the scenarios of binary i.i.d. sources and queries distribution. In particular, the source distribution is obtained as

$$X^n \sim \;\; i.i.d. \;\; \mathcal{B}(p)^n. \tag{11}$$

PCP-SDSM problem with a k-gram code: Perhaps the simplest choice of posting code for the binary i.i.d. setting is the case in which $\mathcal{C}$ contains all possible binary sequences of length k such that $M = 2^k$. Such a posting code is usually referred to as k-gram code and is typically employed in genomic research for indexing DNA sequences, such as in the well-known BLAST algorithm [1].

In the regime of large blocklength, the length of the posting table is obtained by constructing a Markov chain with M states, each corresponding to a possible k-gram. Refer to Appendix A for the details. By considering the structure of the Markov chain and transition probability matrix, we obtain the steady state distribution $\pi_i = \overline{p}^{k-w(\mathbf{c}_i)} p^{w(\mathbf{c}_i)}$ for $i = 1, 2 \ldots, M$ and the average time spent in the state $K_i = \mathbf{c}_i$. Let us next consider the communication cost of each query: let us assume that the queries are obtained as $\sum_l P_L(l) P_{Q|L=l}$ and that

$$Q|l \sim \;\; i.i.d. \;\; \mathcal{B}(q)^l, \tag{12}$$

that is, given that the query length is l, the query is an i.i.d. sequence of Bernoulli distribution with parameter q of length l. The RV determining the length of the query can always be expressed as quotient and remainder of the division by k, i.e.

$$L = kZ + R, \tag{13}$$

where $Z \in \mathbb{N}$ and $R = [0 : k-1]$. The minimum expected communication cost of the query is then

$$\mathbb{E}\left[c^*\left(\mathcal{S}(\mathbf{q})\right)\right] = \mathbb{E}[c(Q_1^k)] \left(Z + 1_{\{R>0\}} 2^{k-R}\right), \tag{14}$$

that is, the cost of the query is the cost of parsing the query with Z k-grams along with covering the tail and accounting for its cost. If the tail has length zero, then the cost of the tail is zero, otherwise the tail of the query is composed of all 2^{k-R} codewords with prefix Q_{kZ}^{kZ+R}.

Lemma 3.1. *For the PCP-SDSM problem with a k-gram posting code and source and query distribution in* (11) *and* (12)*, the efficiency is obtained as*

$$\eta = \left(1 + k \log\left(\overline{pq} + pq\right) \mathcal{O}(\log(n)^{-1})\right) \cdot \left(\mathbb{E}[Z] + \mathbb{E}\left[2^{k-R} \middle| R > 0\right]\right),$$

for Z and R are defined as in (13)*.*

Proof. Proof is provided in App. B. □

PCP-SDSM problem for Run-Length Encoding (RLE): RLE is very simple form of lossless data compression to encode binary data in which one symbol occurs with much higher frequency than the other. This coding is useful, for instance, when encoding line drawings, as the black pixels are sparse. For such a setting, we consider the problem of identifying a specific binary pattern that can itself be described as a set of B run lengths. For this reason, we consider a posting code of the form

$$\mathbf{c}_i = \begin{cases} 1 & i = 1 \\ 0.\mathbf{c}_{i-1} & i \in [2 : M-1] \\ \mathbf{0}(M-1) & i = M, \end{cases} \tag{15}$$

where $\mathbf{0}(x)$ is the vector of all zeros of length x, so that $l(\mathbf{c}_i) = i$ for $i \in [1 : M-1]$ and $l(\mathbf{c}_M) = M-1$. In other words, the IR system stores the successive occurrences of zeros before a one appears, up to length $M-1$. As argued for the case of k-grams, the length of each entry in the posting table can obtained from the average time spent in the state $K_i = \mathbf{c}_i$ in the Markov chain corresponding to the windowing of the source sequence. Accordingly, in the regime of sufficiently large n, the length of each entry in the posting sequence converges to

$$\lim_{n\to\infty} \frac{|\mathcal{T}(\mathbf{c}_i, X^n)|}{n} = \begin{cases} p\overline{p}^{i-1} & 1 \le i < M \\ \overline{p}^{M-1} & i = M, \end{cases}$$

since each codeword apart from $\mathbf{c}_M$ has unitary weight. Let us next define the distribution queries: queries are of the form $S_1.S_2 \dots S_B$, where S_j is a run length of length k_j and B is the number of run lengths. The distribution of the S_j is obtained

$$\mathbb{P}[S_j = \mathbf{0}(k_j - 1).1] = q\overline{q}^{k_j - 1}. \tag{16}$$

Similar to (13), the length of success-run S_j can be expressed as quotient and remainder of the division by $M-1$, i.e.

$$l(S_j) = Z_j(M-1) + R_j, \tag{17}$$

so that the communication cost of the query is Z_j-times the cost of the all zero codeword plus the cost of the codeword equal to R_j.

Lemma 3.2. *For the PCP-SDSM problem with a RLE code and source and query distribution in* (12) *and* (16)*, the efficiency is obtained as*

$$\eta = \mathbb{E}[B]\left(\mathbb{E}[Z] + 1 + \frac{\log p\overline{p}^{\mathbb{E}[Z](M-1)+\mathbb{E}[R]-1}}{\log(n)}\right), \tag{18}$$

where B is the number of run lengths in the query.

The proof is omitted for brevity.

4. NUMERICAL EVALUATIONS

We performed two of sets of preliminary, small-scale simulations to evaluate the the proposed model and to gain some insight into the performance of the codes discussed in the previous section. For the first simulation, prefix-free codes with maximum codeword lengths of 8 were generated by probabilistic splitting of nodes in the code trees. The efficiency of these randomly generated prefix-free codes was compared with that of k-gram codes, with a maximum k of 8 and two sets of p and q. We observe that values for M in the neighborhood of 128, the prefix-free codes have a better efficiency than the k gram for $k = 7$ as the minimum expected communication costs are lower in this region. We also see that, as M approaches 2^8, the efficiency of the prefix-free codes approaches that of the k-gram for $k = 8$. In the case of run-length codes, from Lem. 3.2 we see that as the source sequence grows large, the efficiency approaches $\mathbb{E}[B](\mathbb{E}[Z] + 1)$. This behaviour can be observed in Fig. 2. Queries with up to 4 run-lengths ($B = 4$), were simulated and the distribution of B was chosen to be geometric with a success probability of 0.8.

5. CONCLUSION

In this paper we propose a statistical and information-theoretic formulation of the dictionary-based string matching (SDSM) problem. In the SDSM problem, an IR system has access to a source sequence and it stores the position of a certain number of strings, in a table called the posting table. Upon receiving a query from a user, the IR system accesses the entries in the table to efficiently determine the position of the matches in the source sequence. For this problem, we assume that source and query distributions are described as random processes and we propose a communication cost function for the query retrieval. This cost approximates the cost of transmitting the source indexes in the posting table required to answer the user's query. Through this formulation, we are able to define an optimal posting code as the code which attains the smallest expected communication cost in retrieving a query. For the proposed model, we provide some relevant examples and preliminary numerical evaluations.

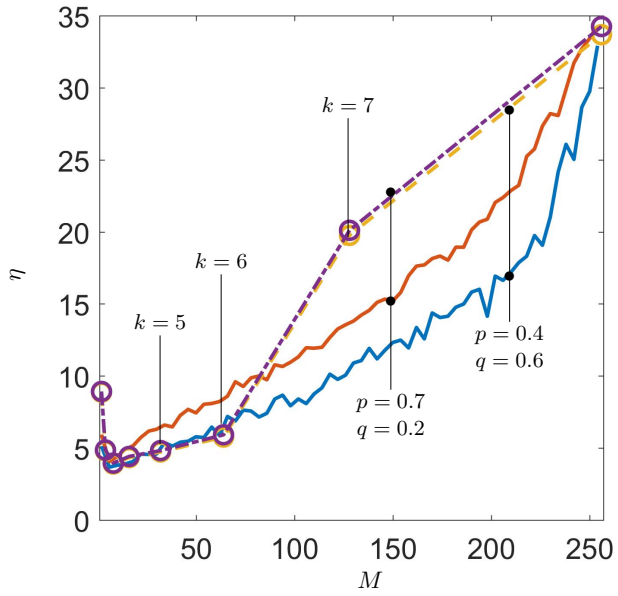

Fig. 1. Comparison of efficiency of k-gram codes (dotted lines) and randomly generated prefix-free codes (solid lines). M is the number of posting codewords. The codes were generated for two different i.i.d. source and query Bernoulli parameters p and q

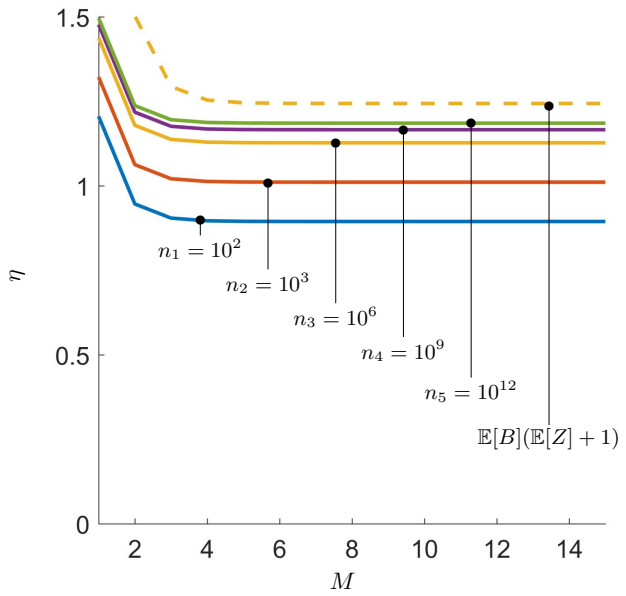

Fig. 2. Convergence of run-length code efficiency to its asymptotic efficiency (dotted line) $\mathbb{E}[B](\mathbb{E}[Z]+1)$ according to Eq. (18).

6. REFERENCES

[1] Stephen F Altschul, Warren Gish, Webb Miller, Eugene W Myers, and David J Lipman. Basic local alignment search tool. *Journal of molecular biology*, 215(3):403–410, 1990.

[2] Vo Ngoc Anh and Alistair Moffat. Inverted index compression using word-aligned binary codes. *Information Retrieval*, 8(1):151–166, 2005.

[3] Galil Apostolico. *Pattern matching algorithms*. Oxford University Press on Demand, 1997.

[4] Ricardo Baeza-Yates and Gonzalo Navarro. Fast approximate string matching in a dictionary. In *Proc. International Symposium on String Processing and Information Retrieval (SPIRE)*, page 0014. IEEE, 1998.

[5] Ricardo Baeza-Yates, Berthier Ribeiro-Neto, et al. *Modern information retrieval*, volume 463. ACM press New York, 1999.

[6] Ricardo Baeza-Yates and Felipe Saint-Jean. A three level search engine index based in query log distribution. In *Proc. International Symposium on String Processing and Information Retrieval (SPIRE)*, pages 56–65. Springer, 2003.

[7] Stefan Büttcher, Charles LA Clarke, and Gordon V Cormack. *Information retrieval: Implementing and evaluating search engines*. MIT Press, 2016.

[8] David M Chen, Sam S Tsai, Vijay Chandrasekhar, Gabriel Takacs, Ramakrishna Vedantham, Radek Grzeszczuk, and Bernd Girod. Inverted index compression for scalable image matching. In *Proc. Data Compression Conference (DCC)*, page 525, 2010.

[9] William Bruce Frakes and Ricardo Baeza-Yates. *Information retrieval: Data structures & algorithms*, volume 331. Prentice Hall Englewood Cliffs, NJ, 1992.

[10] Stoyan Mihov and Klaus U Schulz. Fast approximate search in large dictionaries. *Computational Linguistics*, 30(4):451–477, 2004.

[11] Gonzalo Navarro, Ricardo A. Baeza-Yates, Erkki Sutinen, and Jorma Tarhio. Indexing methods for approximate string matching. *IEEE Data Eng. Bull.*, 24(4):19–27, 2001.

[12] Hao Yan, Shuai Ding, and Torsten Suel. Inverted index compression and query processing with optimized document ordering. In *Proc. International Conference on World Wide Web (WWW)*, pages 401–410. ACM, 2009.

Appendix A: Proof of Th. 2.1

Any posting code can be represented by a $|\mathcal{X}|$-ary tree in which codewords are represented as nodes in the tree and each branch outgoing from an edge is labeled with one of the elements in $\mathcal{X}$. The codeword associated with each node is obtained as the sequence of labels visited in the path from the root of the tree to the node. A prefix-free code is represented by a tree in which all codewords are leaves. A code is complete if all the leaves in the tree representing the code are codewords. Let us prove each of the properties of the PCP SDSM in Th. 2.1. (i) **no retrieval error occurs:** since the posting code is complete, any sequence can be parsed using such code. This follows because, starting from the beginning of the source sequence, the first codeword ends in a leaf of the tree. The next symbol in the source sequence will, consequently, start from the root of the coding tree and the second codeword will again end in a leaf. By repeating this argument, the desired property is shown. (ii) **the minimum covering cost is always finite:** since queries are parsed with posting codewords and given that the posting code is complete, it follows that a parsing of a query always exists (iii) **there exists only one parsing of any query, thus this parsing is the optimal covering:** from the completeness of the posting code, it follows that there exists a unique parsing of any codeword. (iv) **the number of entries in the posting table is always equal to n:** at each position in the source sequence, a codeword exists. Following from the completeness of the storing function, such codeword is stored in the posting table.

Appendix B: Proof of Lem. 3.1

To construct the postings table, the source sequence X^N is parsed into overlapping k-grams such that each k-gram has an overlap of $k-1$ bits with its adjacent k-grams and the positions of each k-gram in the sequence are recorded in postings lists. We first derive the average length of each posting list: this can be determined by observing that the transition from a k-gram to its adjacent overlapping k-gram can be described through a Markov chain. Consider the Markov chain with M states, each corresponding to a possible k-gram, and each labelled with the decimal representation of the corresponding k-gram. The transition between two states corresponds to the sliding of the k-gram of a position forward, i.e. $K_i = X_i^{i+k}$ to $K_{i+1} = X_{i+1}^{i+k+1}$. The transition matrix can be constructed by observing that

$$P_{ij} = \begin{cases} \overline{p}, & \text{if } i \leq 2^{k-1} \text{ and } j = 2i-1 \\ p, & \text{if } i \leq 2^{k-1} \text{ and } j = 2i \\ \overline{p}, & \text{if } i > 2^{k-1} \text{ and } j = 2(i-2^{k-1})-1 \\ p, & \text{if } i > 2^{k-1} \text{ and } j = 2(i-2^{k-1}) \\ 0, & \text{otherwise ,} \end{cases}$$

since the sliding removes the most significant bit in the k-grams and introduces a least significant bit. For a Markov chain with transition matrix P and initial state i, the steady state distribution is denoted by π, and it satisfies the equation $\pi^T P = \pi^T$. The number of visits to state i before time n is

$$V_i(n) = \sum_{t=0}^{n-1} 1\{X_t = i\}, \tag{19}$$

Under suitable conditions, we can determine the average time spent in a state using the following result

$$\mathbb{P}\left[\lim_{n\to\infty} \frac{V_i(n)}{n} = \pi_i\right] = 1, \tag{20}$$

and so the average length of the postings list of codeword i is given by $\mathbb{E}[|\mathcal{T}(\mathbf{c}_i, X^N)|] = N\pi_i$. The steady-state distribution is obtained as the left eigenvector corresponding the eigenvalue at one. We denote component i of the eigenvector u by u_i. This eigenvector has an eigenvalue of 1. Based on the structure of the transition matrix P, we guess the following

$$u_i = \left(\frac{\overline{p}}{p}\right)^{k-w_i} \qquad i = 1, 2, \ldots, M \tag{21}$$

where w_i is the Hamming weight of the k-gram w_i. If u is the eigenvector corresponding to eigenvalue 1, it will satisfy $u^T P = u^T$. If this holds for the choice in (21), this must indeed be the eigenvector as eigenvectors are unique. The eigenvector equation $u^T P = u^T$ can be expressed component wise as

$$u_j = \sum_{i=1}^{M} u_i P_{ij} \quad j = 1, 2, \ldots, M \tag{22}$$

There are only two non-zero entries in every column of P and that these entries are always separated by 2^{k-1} rows. This implies that $w_{i+2^{k-1}} = w_i + 1$. Using this observation, the RHS of the preceding equation for $j = 0, 2, \ldots, M$ can be written as

$$(22)_{\text{RHS}} = \overline{p}\left(\frac{\overline{p}}{p}\right)^{k-w_{\frac{j+1}{2}}} + \overline{p}\left(\frac{\overline{p}}{p}\right)^{k-w_{\frac{j+1}{2}}+2^{k-1}} \tag{23a}$$

$$= \left(\frac{\overline{p}}{p}\right)^{k-w_{\frac{j+1}{2}}} = u_{\frac{j+1}{2}} = u_j \tag{23b}$$

When $j = 1, 3, \ldots, M-1$, the multiplication factor $\overline{p}$ is replaced with p.

$$(22)_{\text{RHS}} = (p)\left(\frac{\overline{p}}{p}\right)^{k-w_{\frac{j-1}{2}}} + (p)\left(\frac{\overline{p}}{p}\right)^{k-w_{\frac{j-1}{2}}-1} \tag{24a}$$

$$= \left(\frac{\overline{p}}{p}\right)^{k-w_{\frac{j-1}{2}}-1} = u_{\frac{j-1}{2}+2^{k-1}} = u_j \tag{24b}$$

For the last step, note that $w_{\frac{j-1}{2}} = w_j - 1$ and so $w_{\frac{j-1}{2}+2^{k-1}} = w_j$. Finally, normalize u by dividing it by $\sum_i u_i$ which can be shown to be $1/p^k$, to obtain the steady the steady-state distribution $\pi_i = \overline{p}^{k-w_i} p^{w_i}$. The remainder of the proof is omitted for brevity.