IJIRSET
International Journal of Innovative Research in
SCIENCE | ENGINEERING | TECHNOLOGY

e-ISSN: 2319-8753 | p-ISSN: 2320-6710

INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH

IN SCIENCE | ENGINEERING | TECHNOLOGY

Volume 9, Issue 9, September 2020

ISSN INTERNATIONAL STANDARD SERIAL NUMBER INDIA

Impact Factor: 7.512

9940 572 462 | 6381 907 438 | ijirset@gmail.com | www.ijirset.com

International Journal of Innovative Research in Science, Engineering and Technology (IJIRSET)

| e-ISSN: 2319-8753, p-ISSN: 2320-6710| www.ijirset.com | Impact Factor: 7.512|

|| Volume 9, Issue 9, September 2020 ||

Efficient Top k Output for String Transformation

Leena Kale, Prof. Shyam S Gupta

Department of Computer Engineering, Siddhant College of Engineering, Pune, India

ABSTRACT: Top-k approximate querying on string collections is an important data analysis tool for many applications. However, the string generation algorithm based on pruning is guaranteed to generate the optimal top k candidates. This paper proposed for explore the efficient top k string problem The approach include method for training the model, a general framework for approximate top-k string search with q-gram, Several efficient algorithm are introduced, such as length aware and adaptive q-gram selection.

KEYWORDS*:* String transformation, Q-grams, Top k string, Rule extraction, Count filtering, and Length filtering

I. INTRODUCTION

This paper addresses top k string for transformation of collection of string. String transformation is defined as given an input string and a set of operators; we can transform the input string to the k most output strings by applying a number of operators. Here the strings can be strings of words, characters, or any type of tokens. Each operator is a transformation rule that defines the replacement of a substring with another substring. The likelihood of transformation can represent similarity, relevance, and association between two strings in a specific application.

From a given collection of strings, such queries ask for strings that are similar to a given string, or those from another collection of strings. While the applications based on similar string querying are not new, and there are numerous works supporting these queries efficiently, such as the scale of the problem has dramatically increased because of the prevalence of the Web.

We introduce several efficient strategies for top-k approximate string search. The count filtering and length-aware mechanism are adapted to tackle the top-k similar search issue, while the adaptive q-gram selection is employed to improve further the performance of frequency counting.

The real data sets and evaluate the querying efficiency of the proposed approaches in terms of several aspects, such as the dataset size, value of k, and q-gram dictionary selection.

II. LITERATURE SURVEY

2.1 Q-Grams

Given a string s and a positive integer q, a positional q-gram of s is a pair (i,g), where g is the q-gram of s starting at the i-th character, that is g = s[i,i + q − 1]. The set of positional q-grams of s, denoted by G(s,q), is obtained by sliding a window of length q over the characters of string s. There are |s| − q + 1 positional q-grams in G(s,q). For instance, suppose q = 3, and s = university, then G(s,q) = {(1,uni), (2,niv), (3,ive), (4,ver), (5,ers), (6,rsi), (7,sit), (8,ity)}. We use a sliding window of size q on the new string to generate positional q-grams. For simplicity, in our notations we omit positional information, which is assumed implicitly to be attached to each gram.

2.1 Top-k Approximate String Queries

The edit distance between two strings s1 and s2 is the minimum number of edit operations of single characters needed to transform s1 to s2. Edit operations include insertion, deletion, and substitution. We denote the edit distance between s1 and s2 as ed(s1, s2). For example, ed(“blank”, “blunt”) = 2. In this paper, we consider the top- k approximate string query on a given collection of strings S. Formally, for a query string Q, finding a set of k strings R in S most similar to Q, that is, $\forall r \in R$ and $\forall s \in (S - R)$ will yield $ed(Q, r) \leq ed(Q, s)$.

To extract the top-k similar strings efficiently, we apply the q-gram strategy with deliberate optimization, as will be discussed shortly. The q-gram similarity of two strings is the number of q-grams shared by the strings.

2.3 Framework for Top-k Similar String Query Processing

We will propose several efficient strategies and introduce our optimal algorithm for the top-k similar string search. The main issue is that the cost of counting the frequency of the q-grams is high, and thus efficient technique is preferred.

| e-ISSN: 2319-8753, p-ISSN: 2320-6710| www.ijirset.com | Impact Factor: 7.512|

|| Volume 9, Issue 9, September 2020 ||

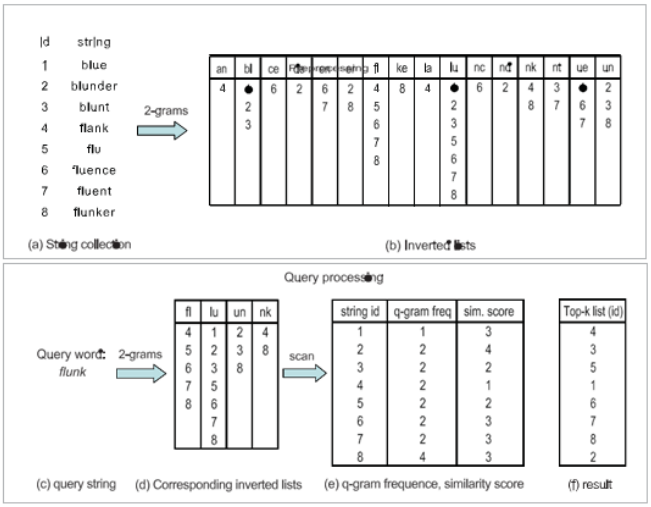

Figure 1: Framework of top k similar string

Example 1. Suppose a user wants to obtain the top-1 similar word to flunk. While querying, the query word flunk is parsed into four 2-grams as f l, lu, un, and nk; their corresponding inverted lists are {4, 5, 6, 7, 8}, {1, 2, 3, 5, 6, 7, 8},{2, 3, 8}, and {4, 8} respectively. The lists are merged to count the frequency of the strings in the collection resulting in 1:1, 2:2, 3:2, 4:2, 5:2, 6:2, 7:2, and 8:4, where each pair of values is of type s id:freq. S id and freq represent the string ID and the frequency value of the n-grams included by the string, respectively. For instance, the string blue (whose ID is 1) includes 1 n-gram of the queried word, giving us 1 : 1. Finally, the edit distances between the candidate strings and the query word are calculated as blue:3, blunder:4, blunt:2, f lank:1, f lu:2, f luence:3, f luent:2, and f lunker:2, where each pair of values is of type string: distance. The top-1 similar string to the query word flunk is thus flank.

III. STRING TRANSFORMATION

There are two processes, learning and generation. In the learning process, rules are first extracted from training string pairs. Then the model of string transformation is constructed using the learning system, consisting of rules and weights. In the generation process, given a new input string, the generation system produces the top k candidates of output string by referring to the model (rules and weights) stored in the rule index shows in figure 2.

The model consists of rules and weights. A rule is formally represented as $\alpha \rightarrow \beta$ which denotes an operation of replacing substring α in the input string with substring β, where $\alpha, \beta \in \{s| s = t, s = \hat{}t, s = t\$, \text{ or } s = \hat{}t\$\}$ and $t \in \Sigma*$ is the set of possible strings over the alphabet, and $\hat{}$ and $ are the start and end symbols respectively. For different applications, we can consider character-level or word-level transformations and thus employ character-level or word-level rules. All the possible rules are derived from the training data based on string alignment. Conditional probability distribution of so and R(si , so) given si and take it as model for string transformation. It is defined as,

$$P\ (so\ , R(si\ , so)|\ si) = \frac{\exp(\Sigma r \epsilon R(si\ , so)\lambda r}{\Sigma(st\ `, R(si\ , st\ `))\epsilon Z(si)\ \exp\ \Sigma o \epsilon R(si\ , st\ `)\lambda o}$$

where , r and o denote rules and λr and λo denote weights

We can consider character-level or word-level transformations and thus employ character-level or word-level rules. All the possible rules are derived from the training data based on string alignment. First, we align the characters in the input string and the output string based on edit-distance, and then derive rules from the alignment. Next, we expand the derived rules with surrounding contexts. Without loss of generality. We only consider expanding 0 − 2 characters from left and right sides as contexts in this paper. Derivation of word-level rules can be performed similarly.

| e-ISSN: 2319-8753, p-ISSN: 2320-6710| www.ijirset.com | Impact Factor: 7.512|

|| Volume 9, Issue 9, September 2020 ||

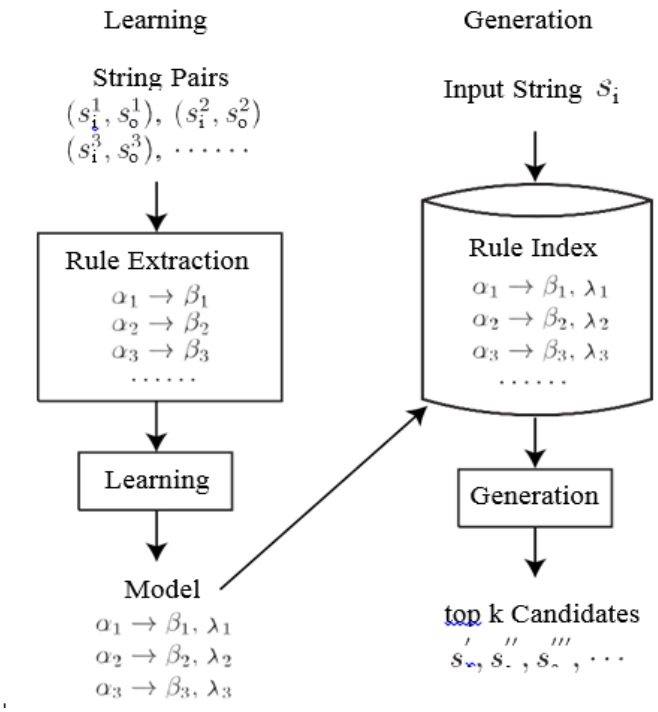

Figure 2: Model of string transformation

We assume that the maximum number of rules applicable to a string pair is predefined. As a result, the number of possible transformations for a string pair is also limited. This is reasonable because the difference between an input string and output string should not be so large. In spelling error correction, for example, the number of possible spelling errors in a word should be rather small.

IV. ALGORITHM

We propose two efficient algorithms for top- k approximate string search. We first introduce two filtering strategies (i.e., count filtering and length filtering) for the first algorithm and then we present the adaptive q selection strategy for the second algorithm. we introduce how to efficiently generate the top k output strings. We employ top k pruning, which can guarantee to find the optimal k output strings. We also exploit two special data structures to facilitate efficient generation.

a) Count Filtering

The intuition of count filtering is that strings within a small edit distance of each other share a large number of q-grams in common. The q-gram count filtering for the top-k similar string search is derived from that of the traditional threshold- based framework.

Algorithm 1: String Search by Q-gram Matching

Input: String collections SC , query string s, top-k, q Output: Top-k similar strings

1. construct q-gram dictionary as trie t, based on SC;
2. parse the query string s into a candidate q-gram list C L;
3. set q-gram distance t=1; //to include all cand. strs;
4. for each x ∈ C L do
5. find the string list SL of x in the triet;
6. for each y ∈ SL do
7. frequency[y]++;
8. if frequency[y] ≥ t then
9. put y into a candidate result list C RL;
10. for each z ∈ C RL do
11. compute the similarity ed(z, s);

| e-ISSN: 2319-8753, p-ISSN: 2320-6710| www.ijirset.com | Impact Factor: 7.512|

|| Volume 9, Issue 9, September 2020 ||

12. Output the top-k strings with smallest similarities

b) Length Filtering

The intuition of length filtering is that if there are two strings within a small edit distance τ of each other, then the difference of their lengths will not exceed τ.

When scanning the inverted lists, we choose to test the candidate strings in ascending order of the difference between their lengths and that of the query string.

Algorithm 2: Branch and bound algorithm (BB)
Input: String collections S, a query string P , k
Output: Top-k similar strings

1. build index I DX to record strings based on their lengths from S;
2. construct q-gram dictionary T ;
3. parse P into a set of q-grams, t′ =1, ld=0;
4. while true do
5. Inv = the inverted lists in T with any string S has ||S| − |P || = ld;
6. apply skip-merge-skip strategy to get the candidate strings from Inv;
7. rank the candidate strings, Q=the top-k string;
8. τ ′ p−k =ed(P, Q);
9. if ed(P, Q) ≤ (ld + 1) then
10. Output the top-k strings; break;
11. End

The query processing is executed as follows:
a) Initialization: Set the frequency threshold t=1, and the length difference ld=0. P is parsed into a set of q-grams. The following steps are executed iteratively until k similar strings are output.
b) Branch: Extract the corresponding inverted lists in which any string S has ||S| − |P || = ld.1 Scan these inverted lists and discover the candidate similar strings whose frequencies are no less than t=2.
c) Bound: Rank the candidate strings and obtain the temporal edit distance threshold t. Terminate the process if the top-k string Q in the candidate list so far has ed(P, Q) ≤ (ld + 1).

V. CONCLUSION

In this paper we have studied the efficient top-k approximate string problem. Several efficient strategies are proposed, i.e., length filtering and adaptive q-gram selection. We present two algorithms in a general framework by combining these strategies. It shows that our approaches can efficiently the top-k string queries.

ACKNOWLEDGEMENT

I want to genuinely thank my guide Prof. Shyam Gupta for his inspiration and proposals which really helped me in enhancing the quality and viability of this paper. I take this chance to express my thanks my instructor, family and companions for their consolation and backing.

REFERENCES

[1] Ziqi Wang, Gu Xu, Hang Li, and Ming Zhang,A probabilistic approach to string transformation ,vol pp no 99 year 2013.
[2] Z. Yang, J. Yu, and M. Kitsuregawa, “Fast algorithms for top-k approximate string matching,” in Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence, ser. AAAI ’10, 2010, pp.1467–1473.
[3] R. Vernica and C. Li, “Efficient top-k algorithms for fuzzy search in string collections,” in Proceedings of the First International Work- shop on Keyword Search on Structured Data, ser. KEYS ’09. New York, NY, USA: ACM, 2009, pp. 9–14.
[4] H. Duan and B.-J. P. Hsu, “Online spelling correction for query completion,” in Proceedings of the 20th international conference on World wide web, ser. WWW ’11. New York, NY, USA: ACM,2011, pp. 117–126.

INNO SPACE
SJIF Scientific Journal Impact Factor

Impact Factor:
7.512

ISSN INTERNATIONAL STANDARD SERIAL NUMBER INDIA

निस्केयर
NISCAIR

INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH

IN SCIENCE | ENGINEERING | TECHNOLOGY

9940 572 462 6381 907 438 ijirset@gmail.com

www.ijirset.com

Scan to save the contact details