Fast Error-tolerant Location-aware Query Autocompletion

Jin Wang
University of California, Los Angeles.
jinwang@cs.ucla.edu

Chunbin Lin*
Amazon AWS
lichunbi@amazon.com

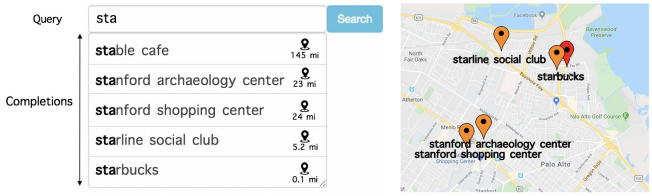
***Abstract*—Location-based services have become ubiquitous in smart life, but typing queries in mobile devices is tedious and error-prone. Therefore, query autocompletion is needed to instantly provide users with query suggestions based on the incomplete user input. A recent trend is to support error-tolerant autocompletion, which could improve the usability by allowing a small number of errors between the query input and prefixes of strings in database. In addition, the query autocompletion should be location-aware for location-based services since it makes more sense to provide query suggestions for nearby objects. Unfortunately, existing query autocompletion algorithms cannot efficiently support both error-tolerate and location-aware features at the same time.**

In this paper, we propose a novel framework AutoEL to support error-tolerant location-aware query autocompletion. The error-tolerate feature is enabled by applying edit distance to evaluate the textual similarity between given query and the underlying data, while the location-aware feature is guaranteed by choosing the k-nearest neighbors. To improve the efficiency, we construct a hybrid data structure to jointly index spatial and textural information. We also propose several optimizations on data partition as well as search algorithm. Extensive experiments on real datasets demonstrate that AutoEL outperforms the baseline methods by up to an order of magnitude.

I. Introduction

With the rapid growth of Location-based Services, it becomes more and more important to support location-aware search engines, which aim at finding point-of-interests (POI) such as hotels, restaurants and gyms near the current location [1]. The input of such search engines is usually text keywords, e.g., the names of hotel and restaurant. However, it is challenging to type complete and correct full names in mobile devices due to the following facts [12]: (i) mobile devices usually have small screens especially for wearable devices, e.g., Apple Watch; (ii) users may type via moving, which makes harder to hit the correct key tokens; and (iii) it is difficult for users to spell the accurate full names as some of those are quite long and hard to spell. Thus, in order to form a correct query, users usually need to repeat the time-consuming process of typing-trying-refining multiple times.

To help users form complete and correct query input, the feature of query autocompletion (QAC) becomes highly necessary. QAC provides a list of suggested queries based on the incomplete query typed by users so far. Then users can just select their intend query to start the search. Location-aware QAC takes the distance between the POI and query place into consideration on the basis of it. Figure 1 shows a scenario of location-aware query autocompletion. When a user types "sta", it shows a list of possible queries, e.g., "stable cafe" and "stanford shopping center" for users to choose. For example, the user's intend query is "starbucks", then the user can directly click the fifth suggested query in the autocompletion to begin a search without typing the complete query. Although QAC indeed improves the user experiences, the existing studies on QAC still have the following limitations.



(a) Example query autocompletion (b) Visualization of suggested completions

Fig. 1. Example of Location-aware Query Autocompletion (QAC). When typing "sta" in the search box, system provides a list of suggested queries (a), and the corresponding objects are shown in (b).

On the one hand, the error tolerate feature is not supported by previous studies about location-aware QAC and instant search [7]. The output of these approaches is a list of queries whose prefix are exactly matched with the user input, which is not user-friendly as it does not allow typos in prefixes. Actually, the real intend queries might not be included in such a list just because of the typos. For example, the nearby "burger king" will not show up in the list when typing "bugger", as they are not exactly matched. As a result, if users do not have enough knowledge of the spelling, it would be difficult for them to get reasonable query results.

On the other hand, although some existing studies support error-tolerate QAC and instant search, they just ignore the location information [2]. It might lead to meaningless search results as the suggested POIs may be thousand miles away from current location. As is observed from Figure 1, the top-1 suggestion is more than 100 miles, which is not quite interesting for users.

One possible solution is to make simple extension on the basis of existing studies. We can first adopt the state-of-

*corresponding author

[1]The current location can be obtained via embed GPS in mobile devices.

2375-026X/20/$31.00 ©2020 IEEE
DOI 10.1109/ICDE48307.2020.00223

IEEE computer society

the-art approaches for error-tolerate QAC to identify textual similar objects, and then find those nearest to current location. However, the search performance could be sub-optimal due to the lack of filtering power. Although there are some pruning techniques in string similarity queries [13], [5], [8], [9], [10] and spatial keyword search [11], [1], they cannot directly be adopted in the QAC problem where the input query needs to be dealt incrementally.

To address above limitations, we propose a novel framework AutoEL to support location-aware and error-tolerated QAC. More precisely, (i) The error-tolerate feature is enabled by applying *edit distance*, which has been proven to be the most widely used metric to capture typographical errors [4]. If the edit distance between the query input and the prefixes of objects in database is within a given threshold, such objects should also be regarded as candidates; and (ii) The location-aware feature is guaranteed by choosing the k-nearest neighbors (KNNs) objects that are candidates from the current location. In real world applications, the suggested queries should be returned within milliseconds even for large amounts of data. Therefore, it is essential to provide an efficient solution. To improve the performance of AutoEL, we construct a hybrid index structure to jointly index the spatial and textual information. Then a top-down search algorithm can be adopted to incrementally find all results. Since the pruning power is closely related to the spatial and textual features, we also study the problem of space segmentation in order to make similar objects closer to each other in the index structure, which will further boost the efficiency. Furthermore, we propose several optimizations to avoid unnecessary accesses to index and fetching redundant results. We perform a comprehensive evaluation on two real-life datasets to demonstrate the superior efficiency of our algorithms. The results show that AutoEL outperforms the baseline methods with $3\times$ to $13\times$ speedup.

II. Framework

A. Problem Formulation

In the ELQA problem, we are dealing with the spatio-textual objects where an object S has two fields, a textual description s and a spatial location s^l, denoted by $S = \langle s, s^l \rangle$. Here s is a string with several characters and s^l is a location descriptor in multi-dimensional space. To enable the error-tolerate feature, we adopt the similarity metric of edit distance. Given a string s, let $s[i]$ denote the i^{th} character of s and $s[i..j]$ denote the substring of s starting from $s[i]$ and ending at $s[j]$. We call the substring $s[1...j]$ $(j \in [1, |s|])$ the prefix of s and denote it as P_s^j. We use the Euclidean distance between two locations to measure the spatial distance. The spatial distance between two spatial textual objects S and Q is denoted as $\text{DIST}(S, Q) = \sqrt{\sum_{i=1}^{n}(S_i^l - Q_i^l)^2}$ where n is the number of dimensions [2]. Next we formally define the problem of Error-tolerate Location-aware Query Autocompletion (ELQA) as Definition 1.

[2]In this paper we focus on the case of n=2

Definition 1 (ELQA): Given a query $Q = (q, q^l)$, a database of spatio-textual objects $\mathcal{S}$, an edit distance threshold τ and a number k, the location-aware query autocompletion aims at finding a set of objects $\mathcal{R}$ s.t. $|\mathcal{R}| = k$, and for any $S \in \mathcal{R}$, $S' \in \mathcal{C} - \mathcal{R}$, we have $\text{DIST}(S, Q) \leq \text{DIST}(S', Q)$, where $\mathcal{C} = \{S | S \in \mathcal{S} \cap \max_{j \in [1,|s|]} \text{ED}(q, P_s^j) \leq \tau\}$.

B. The AutoEL Framework

The cornerstone of this work is the most well-known text-only approach IPCAN [3]. We employ the idea of text-first methodology to construct index by integrating both spatial and textual information and perform search accordingly. To integrate the spatial information into the trie index, we split the whole space into a set of disjoint grids, which are rectangles. Each object in database belongs to one grid. We denote the set of grids as $\mathcal{G}$. Suppose there are M grids in total, the i^{th} grid is denoted as $g_i, i \in [1, M]$ We can attach a bitmap with cardinality of M to each trie node, with the bitmap for a trie node n as G_n. We will also use G_n to denote the set of grids corresponds to node n without ambiguity. If g_i has an object belonging to n, then the value of $G_n[i]$ is set as 1 and otherwise 0.

We use a max heap $\mathcal{R}$ to keep the current top-k results ranked by the spatial distance. The spatial distance between the object on top of $\mathcal{R}$ and q^l is denoted as $\text{UB}_{\mathcal{R}}$. Since each trie node n corresponds to a set of objects that share the common prefix, it also has a corresponding spatial area that encloses all the objects. Then we can use the lower bound of the spatial distance between q^l and the area corresponding to n, denoted as $\text{MINDIST}(q^l, n)$, to perform pruning. If $\text{MINDIST}(q^l, n) > \text{UB}_{\mathcal{R}}$, we can prune the whole sub-trie of n even n is still an active node. The reason is that it is guaranteed that no object in the sub-trie can have smaller spatial distance than the current k objects in $\mathcal{R}$. Given a location q^l and a grid g, the minimum distance between q^l and any object in g can be calculated with Equation 1 as illustrated in [6].

$$\text{MINDIST}(q^l, g) = \sqrt{|q_x^l - g_{r,x}|^2 + |q_y^l - g_{r,y}|^2} \quad (1)$$

where

$$g_{r,x}(g_{r,y}) = \begin{cases} g.l_x(g.l_y) & q_x^l(q_y^l) < g.l_x(g.l_y) \\ g.u_x(g.u_y) & q_x^l(q_y^l) > g.u_x(g.u_y) \\ q_x^l(q_y^l) & otherwise \end{cases} \quad (2)$$

In order to integrate above computation into IPCAN, we need to compute the value of $\text{MINDIST}(q^l, n)$ once we reach a new active node n. If the given node n satisfies $\text{MINDIST}(q^l, n) \geq \text{UB}_{\mathcal{R}}$, we can safely exclude n from the active node set. Note that the value of $\text{UB}_{\mathcal{R}}$ can be updated by checking the bitmap of a trie node without fetching the results. To reach this goal, we maintain the cardinality of each grid $g \in \mathcal{G}$ as $|g|$. We can compute the upper bound of distance $\text{MAXDIST}(q^l, g)$ in a similar manner with Equation 1 according to [6]. By computing this upper bound, we can know that there are $|g|$ objects with $\text{MAXDIST}(q^l, g)$ distance. Consequently, we can update the value of $\text{UB}_{\mathcal{R}}$ from node n

by estimating the upper bound of distance for the k^{th} smallest object from all grids in G_n.

Algorithm 1: Location-aware QAC($\mathcal{S}$, Q, τ, k)

Input: $\mathcal{S}$: The collection of strings; $Q = \langle q, q^l \rangle$: The query; τ: The edit distance threshold; k: The number of results
Output: $\mathcal{R}$: The top-k results

```
1  begin
2      Initialize A with ⟨root, 0, ∅, 0⟩;
3      Initialize UB_R as ∞;
4      for i = 1 to |q| do
5          Initialize A' = ∅;
6          foreach A ∈ A do
7              if A.d_n + 1 ≤ τ then
8                  Add ⟨n, d_n + 1, j, d_{n,j} + 1⟩ into A';
9              Traverse the subtrie rooted by n within height
               τ − A.d_{n,j} + 1, check each node n';
10             if MinDist(q^l, n') ≥ UB_R then
11                 continue;
12             Update the value of UB_R with
               MaxDist(q^l, n');
13             d' = A.d_{n,i} + max(|n'| − |n| − 1, i − j)
14             if d' ≤ τ then
15                 Add ⟨n', d', i + 1, d'⟩ into A';
16
17         Remove non-pivotal active nodes and duplicates
           from A';
18         A = A';
19     Traverse leaf nodes reachable from all active nodes
       in A;
20     Compute the spatial distance and obtain the top-k
       results in R;
21     return R;
22 end
```

Algorithm 1 shows the process of ELQA. It first initializes the active node set with the dummy trie root (line 2). And the value of $\mathsf{UB}_\mathcal{R}$ can be initialized as the infinity distance (line 3). For each input character, it incrementally computes the active nodes with the IPCAN [3] method (line 6-18). To utilize the textual information, it identifies new active nodes by considering insertion (line 9) and deletion (line 8) where the last character is matched. In this process, when accessing a new trie node n, if $\text{MinDist}(q^l, n') \geq \mathsf{UB}_\mathcal{R}$ the sub-trie rooted by n will be pruned (line 11). Otherwise, the value of $\mathsf{UB}_\mathcal{R}$ is updated and the process of collecting active node continues (line 12).

Next it fetches the results by visiting all the reachable leaf nodes from the set of active node (line 19). Finally, it performs verification using the true spatial information of each candidate and return the top-k results (line 20). Actually, the steps of fetching and verification do not necessarily happen after seeing the whole query. Since the algorithm is incremental, it can happen in the process of typing any character $q[i]$ upon practical requirement.

C. Optimization

To further improve the performance, we also made several optimizations. The high level ideas of such techniques are summarized as following.

1) Space Segmentation: To improve the pruning power of Algorithm 1, a key step is to produce a high quality segmentation of the space to generate grids. A straightforward solution is to use some rule based methods, e.g. put all objects with the same city into one grid. However, it might result in data skewness problem which will reduce the filter power. To address this issue, we propose a weight based approach to perform space segmentation so as to generate the grids. Specifically, we propose a weighting mechanism to describe the data distribution of each grid by considering both spatial and textual similarity. Based on it, we then devise a greedy algorithm to decide the grids accordingly.

2) Avoid Unnecessary Bitmap Access: Recall that in Algorithm 1, we need to check the bitmap of each trie node in order to update the value of $\mathsf{UB}_\mathcal{R}$, which might involve heavy overhead considering the large number of trie nodes. To improve the performance, we propose three bitmap-based improvements: Firstly, we remove the duplicate bitmaps if a parent has only one child node; Secondly, we will make a bitwise XOR operation between the bitmap of a parent node and its child where it only needs to check the bits with 1 for the child. Lastly, once we find a value larger than $\mathsf{UB}_\mathcal{R}$, we can stop earlier as this trie node cannot further lower the spatial bound.

3) Improve Results Fetches: To avoid redundant node traversal, we propose another optimization to eliminate descendant active nodes before fetching results. The basic idea is to identifying the ancestor-descendant relationship in the process of finding active nodes. If an ancestor node and its descendant co-exist in the set of active nodes, we remove all descendant active nodes and only maintain the ancestor. To keep record of this relationship, we maintain a forest data structure where we can only use the root of each tree in the forest for result fetching. This optimization strategy can be integrated into Algorithm 1 without much space overhead by modifying the places where results fetching is required (line 7-8, 14-15 and 18).

III. Evaluation

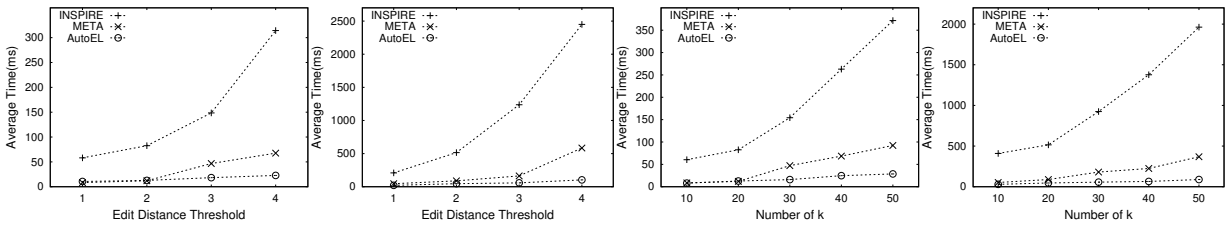
A. Experiment Setup

TABLE I
Datasets Statistics

Dataset	Cardinality	Avg String Length	Size (MB)
OSM	2 million	17	183
SGP	12.9 million	19	866

We use two public available POI datasets to evaluate our proposed techniques, which have been widely applied in previous studies. OpenStreetMap [3] (OSM) is from the

[3]http://www.openstreetmap.org/



(a) varying τ, OSM (b) varying τ, SGP (c) varying k, OSM (d) varying k, SGP

Fig. 2. Comparison with State-of-the-art Methods

project of OpenStreet open-source spatial database. It consists of POIs for multiple kinds of places in the United States, such as parks, schools and restaurants. SimpleGeo Place [4] (SGP) is a database of business listings and POIs. The detailed information is shown in Table I.

As there is no previous work for the ELQA problem, we extend two previous studies as the baseline methods: INSPIRE [14] and META [2]. The evaluation metric is the average processing time per query. We implement the two baselines by ourselves since there is no public available code. All experiments are conducted on a server with an Intel i7-4770 CPU processor, 32 GB RAM, running Ubuntu 14.04. All the algorithms are implemented in C++ and compiled with GCC 4.8.4.

B. Results and Analysis

We compare our framework with the extension of two state-of-the-art methods. From the results shown in Figure 2, we observe that our AutoEL achieves the best performance. For example, as shown in Figure 2(c) when $\tau = 2$ and $k = 50$ on OSM, the average time per query for INSPIRE and META is 371.54 ms and 92.32 ms, respectively, while AutoEL takes only 28.64 ms. The reason is that compared with previous approaches, AutoEL can perform pruning with spatial and textual information simultaneously. Then we can avoid visit redundant trie nodes with the help of spatial information. In almost all the experiment settings, the performance of META ranks second and obviously outperforms INSPIRE. The reason could be that in the ELQA problem, finding similar strings is more expensive than identifying spatial distance. As META is specifically optimized for text-only task, the performance would benefit greatly from its simplified trie index. Meanwhile, INSPIRE enables pruning on the textual dimension with inverted index, which is not as good at supporting incremental search as trie-based indexes.

However, META cannot make use of the spatial information to perform pruning. Therefore, it has much larger search space and thus worse overall performance than AutoEL. We can see that simply extending text-only approach is not efficient enough Therefore, it also illustrates the necessity of devising effective techniques to perform pruning with both spatial and textual information.

[4]https://archive.org/details/2011-08-SimpleGeo-CC0-Public-Spaces

IV. Conclusion

In this paper, we study the problem of Error-tolerate Location-aware Query Autocompletion and propose the AutoEL framework. We jointly index the spatial and textual features in a trie-based index structure to perform pruning with both information simultaneously. We devise a weight-based strategy to split the space before index construction so as to improve pruning power. We further develop effective search strategies to accelerate the query processing. Experimental results on real world datasets demonstrate the superiority of our proposed framework over alternative solutions.

References

[1] L. Chen, S. Shang, C. Yang, and J. Li. Spatial keyword search: a survey. *GeoInformatica*, 24(1):85–106, 2020.

[2] D. Deng, G. Li, H. Wen, H. V. Jagadish, and J. Feng. META: an efficient matching-based method for error-tolerant autocompletion. *PVLDB*, 9(10):828–839, 2016.

[3] G. Li, S. Ji, C. Li, and J. Feng. Efficient fuzzy full-text type-ahead search. *VLDB J.*, 20(4):617–640, 2011.

[4] G. Navarro. A guided tour to approximate string matching. *ACM Comput. Surv.*, 33(1):31–88, 2001.

[5] C. Rong, C. Lin, Y. N. Silva, J. Wang, W. Lu, and X. Du. Fast and scalable distributed set similarity joins for big data analytics. In *ICDE*, pages 1059–1070, 2017.

[6] N. Roussopoulos, S. Kelley, and F. Vincent. Nearest neighbor queries. In *SIGMOD*, pages 71–79, 1995.

[7] S. B. Roy and K. Chakrabarti. Location-aware type ahead search on spatial databases: semantics and efficiency. In *SIGMOD*, pages 361–372, 2011.

[8] J. Wang, C. Lin, M. Li, and C. Zaniolo. An efficient sliding window approach for approximate entity extraction with synonyms. In *EDBT*, pages 109–120, 2019.

[9] J. Wang, C. Lin, and C. Zaniolo. Mf-join: Efficient fuzzy string similarity join with multi-level filtering. In *ICDE*, pages 386–397, 2019.

[10] J. Wu, Y. Zhang, J. Wang, C. Lin, Y. Fu, and C. Xing. Scalable metric similarity join using mapreduce. In *ICDE*, pages 1662–1665, 2019.

[11] J. Yang, Y. Zhang, X. Zhou, J. Wang, H. Hu, and C. Xing. A hierarchical framework for top-k location-aware error-tolerant keyword search. In *ICDE*, pages 986–997, 2019.

[12] A. Zhang, A. Goyal, R. A. Baeza-Yates, Y. Chang, J. Han, C. A. Gunter, and H. Deng. Towards mobile query auto-completion: An efficient mobile application-aware approach. In *WWW*, pages 579–590, 2016.

[13] Y. Zhang, X. Li, J. Wang, Y. Zhang, C. Xing, and X. Yuan. An efficient framework for exact set similarity search using tree structure indexes. In *ICDE*, pages 759–770, 2017.

[14] Y. Zheng, Z. Bao, L. Shou, and A. K. H. Tung. INSPIRE: A framework for incremental spatial prefix query relaxation. *IEEE Trans. Knowl. Data Eng.*, 27(7):1949–1963, 2015.