Indoor Top-k Keyword-aware Routing Query

Zijin Feng [1] & Tiantian Liu [2] & Huan Li [2] & Hua Lu [2] & Lidan Shou [3] & Jianliang Xu [1]

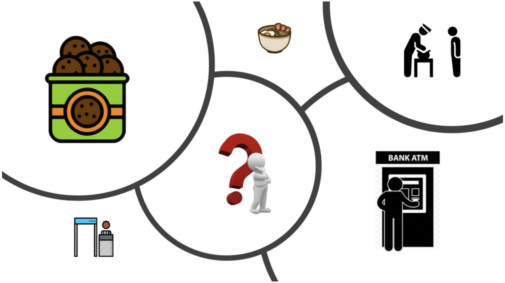
[1]Department of Computer Science, Hong Kong Baptist University

[2]Department of Computer Science, Aalborg University

[3]Department of Computer Science, Zhejiang University

April 2020

Motivation



This paper:

- Formulates indoor top-k keyword-aware routing query(IKRQ)
- Devises mapping structures to organize indoor keywords and compute route keyword relevance
- Derives pruning rules to reduce search space in routing
- Conducts extensive experiments on synthetic and real data sets to evaluate our proposals

Motivation

Indoor Top-k Keyword-aware Routing Query

Given a start point p_s, a terminal point p_t, a distance constraint Δ, and a query keyword list QW, an **indoor top-k keyword-aware routing query** IKRQ(p_s, p_t, Δ, QW, k) returns k regular and prime routes from p_s to p_t in a k-set Θ such that $\forall R \in \Theta$, $\delta(R) \leq \Delta$ and $\Psi(R, \Delta, QW) \geq \Psi(R', \Delta, QW)$ for any route $R' \notin \Theta$ from p_s to p_t with $\delta(R') \leq \Delta$.

High-Level Overview

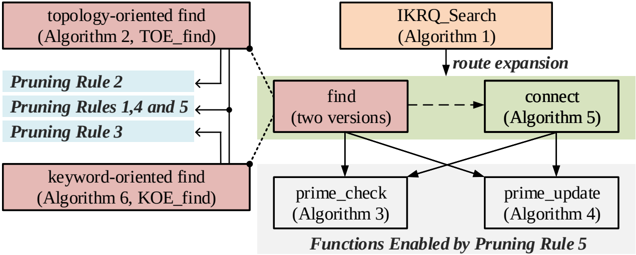


Figure: Architecture of the IKRQ Search Algorithms

Indoor Topology & Route

Homogeneous Routes.Two routes R_i and R_j are **homogeneous routes** if $R_i.head = R_j.head$, $R_i.tail = R_j.tail$, and $KP(R_i) = KP(R_j)$

Prime Route. Suppose HR is a complete set of homogeneous routes for a routing query, we say a route $R_i \in HR$ is **prime** against $R_j \in HR$ if $\delta(R_i) < \delta(R_j)$. R_i is a **prime route** if R_i is prime against all other routes in HR.

Principles of indoor route search

- **Principle of Regularity**. Disqualifies a route that contains one or more doors between two identical doors (e.g. d13-d14-d14-d13)
- **Principle of Diversity**. Avoid homogeneous routes in our indoor routing

TABLE II: Examples of Routes from p_s to p_t

R_1	$(p_s \xrightarrow{v_1} d_2 \xrightarrow{v_2} d_6 \xrightarrow{v_3} d_7 \xrightarrow{v_5} p_t)$
R_2	$(p_s \xrightarrow{v_1} d_2 \xrightarrow{v_2} d_5 \xrightarrow{v_5} d_7 \xrightarrow{v_3} d_7 \xrightarrow{v_5} p_t)$
R_3	$(p_s \xrightarrow{v_1} d_2 \xrightarrow{v_2} d_5 \xrightarrow{v_5} d_9 \xrightarrow{v_6} d_9 \xrightarrow{v_5} d_7 \xrightarrow{v_3} d_7 \xrightarrow{v_5} p_t)$
R_4	$(p_s \xrightarrow{v_1} d_3 \xrightarrow{v_5} d_5 \xrightarrow{v_2} d_5 \xrightarrow{v_5} d_7 \xrightarrow{v_3} d_7 \xrightarrow{v_5} p_t)$

Keyword Organization & Relevance

Identity word (i-word). Identifies the specific name of a partition (e.g. *Starbucks*)

Thematic word (t-word). Refers to a tag relevant to that partition (e.g. *coffee*, *mocha*)

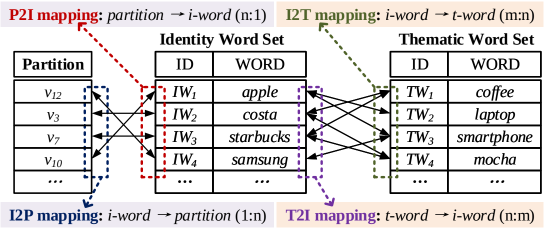


Figure: Indoor Space Keyword Mappings

For **T2I** mapping, we have **direct matching i-words** and **indirect matching i-words**

Keyword Organization & Relevance

Candidate I-word Set $\kappa(w_Q)$. Set of entries each of which is in form of (w_i, s), a pair of a matching i-word w_i and the similarity score s between w_Q and w_i, $s > \tau$.

$\kappa(w_Q)$ has two cases:

- If w_Q is an i-word, $\kappa(w_Q) = \{(w_Q, 1)\}$
- If w_Q is a t-word, $\kappa(w_Q)$ consists of
  - All direct matching i-word, i.e., $(w_i', 1)$, for all $w_i' \in T2I(w_Q)$
  - All indirect matching i-word, i.e., $(w_i'', s(w_i''))$, where

$$s(w_i'') = \frac{|I2T(w_i'') \cap \bigcup_{w_i \in T2I(w_Q)} I2T(w_i)|}{|I2T(w_i'') \cup \bigcup_{w_i \in T2I(w_Q)} I2T(w_i)|} > \tau$$

partition	i-word	t-words
v_3	*costa*	$\{coffee, drinks, macha\}$
v_{10}	*apple*	$\{phone, mac, laptop, watch\}$
v_7	*starbucks*	$\{coffee, macha, latte, drinks\}$
v_{12}	*samsung*	$\{phone, laptop, earphone\}$

Example

$I2T(costa) = \{coffee, drinks, macha\}$ and $\bigcup_{w_i \in T2I(latte)} I2T(w_i) = \{coffee, drinks, macha, latte\}$

Keyword Organization & Relevance

Keyword Relevance.

$$\rho_{QW}(R) = \begin{cases} 0, & \text{if } N_{QW}(R) = 0; \\ N_{QW}(R) + \dfrac{\sum\limits_{w_Q \in QW} \left(\max\limits_{w_i' \in \mathrm{M}(w_Q, R)} s(w_i') \right)}{N_{QW}(R)}, & \text{otherwise.} \end{cases}$$

Ranking Score.

$$\psi(R, \Delta, QW) = \alpha \cdot \frac{\rho(R)}{|QW| + 1} + (1 - \alpha) \cdot (\frac{\Delta - \delta(R)}{\Delta})$$

Search Algorithms for IKRQ

Pruning rules

1. A partial route $R^{\star} = (p_s, d_i, \ldots, d_n)$ in the searching can be pruned if $\delta(R^{\star}) + |d_n, p_t|_L > \Delta$.
2. A door d_n can be pruned out of the search if $|p_s, d_n|_L + |d_n, p_t|_L > \Delta$.
3. An indoor partition v_i can be pruned out of the search if its lower bound distance $\delta(p_s, v_i, p_t) =$

$$\min_{d_i \in P2D_{\sqsupset}(v_i), d_j \in P2D_{\sqsubset}(v_i)} (|p_s, d_i|_L + \delta_{d2d}(d_i, d_j) + |d_j, p_t|_L) > \Delta.$$

4. Given the current k-th highest ranking score ψ_k among the seen complete routes, a partial route $R^{\star} = (p_s, d_i, \ldots, d_n)$ can be pruned if its upper bound ranking score $\psi_U(R^{\star}) = \alpha \cdot 1 + (1-\alpha)(1 - (\delta(R^{\star}) + |d_n, p_t|_L)/\Delta) \leq \psi_k$.
5. A partial route $R^{\star} = (p_s, d_i, \ldots, d_n)$ in the search can be pruned if the search has already obtained a route $R^{\star'}$ from p_s to d_n that is prime against $R^{\star}$.

Topology-oriented Expansion (ToE)

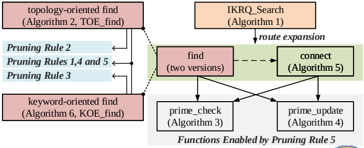
Idea: To reach all accessible doors from the current door based on indoor topology, i.e., always expands from the current door to the next enterable door within one hop

Keyword-oriented Expansion (KoE)

Idea: Focus on the query words that have not been covered by the current stamp, and directly expand to one of the key partitions that can cover some of those uncovered query words

Algorithm 1 IKRQ_Search (p_s, p_t, Δ, QW, k)

initialize priority queue Q
set of all candidate i-words $W_{ci} \leftarrow \bigcup_{w_Q \in QW} \kappa(w_Q).W_i$
$P \leftarrow \left(\bigcup_{w_Q \in QW} I2P(\kappa(w_Q).W_i)\right) \setminus v(p_s) \cup v(p_t)$
door sets $D_n \leftarrow \varnothing$, $D_f \leftarrow \varnothing$
$kbound \leftarrow 0$
initialize hashtable H_{prime}
$R_0 \leftarrow (p_s)$
$S_0 \leftarrow (v(p_s), R_0, 0, \rho(R_0), \psi(R_0))$
$\mathsf{Q}.push(S_0)$
while Q is not empty **do**
  $S_i \leftarrow \mathsf{Q}.pop()$
  $ES \leftarrow$ `find`(S_i) ▷ find the next valid stamps
  for each $S_j \in ES$ **do**
    `connect`(S_j) ▷ connect each valid stamp to terminal
return current top-k results



Experimental Studies

Table: Dataset Information (Indoor Keywords)

	Synthetic Data	Real Data
# of i-word	1120	533
# of t-word	9195	5036

Table: Notations of Comparable Methods

Modification	ToE family	KoE family
–	ToE	KoE
no distance-based Pruning Rules 1 3	ToE\D	KoE\D
no kbound-based Pruning Rule 4	ToE\B	KoE\B
no prime-based Pruning Rule 5	ToE\P	–
with precomputed shortest routes	–	KoE*

Experimental Studies

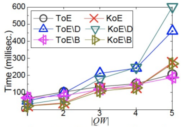


Fig. 6: Time vs. $|QW|$

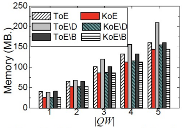


Fig. 7: Memory vs. $|QW|$

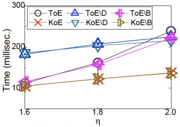


Fig. 8: Time vs. η

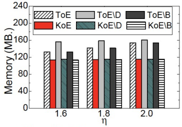


Fig. 9: Memory vs. η

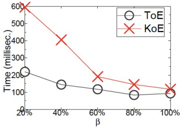


Fig. 10: Time vs. β

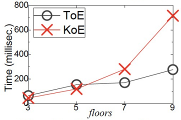


Fig. 11: Time vs. *floor*

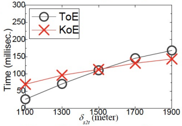


Fig. 12: Time vs. δ_{s2t}

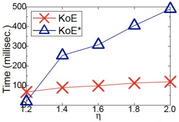


Fig. 13: Time of KoE*

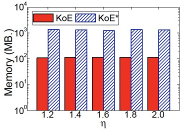


Fig. 14: Memory of KoE*

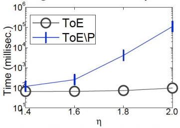


Fig. 15: Time of ToE\P

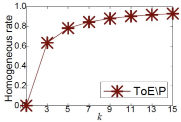


Fig. 16: Homogeneous rate

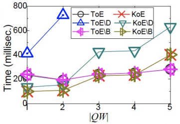


Fig. 17: Time vs. $|QW|$

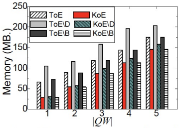


Fig. 18: Memory vs. $|QW|$

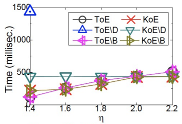


Fig. 19: Time vs. η

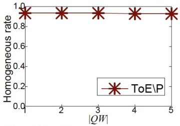


Fig. 20: Homogeneous rate

Thank you!

Thank you!