# SKYPE: Top-k Spatial-keyword Publish/Subscribe Over Sliding Window

†Xiang Wang, ‡Ying Zhang, †Wenjie Zhang, †Xuemin Lin, †Zengfeng Huang

† *The University of New South Wales* ‡ *The University of Technology, Sydney*

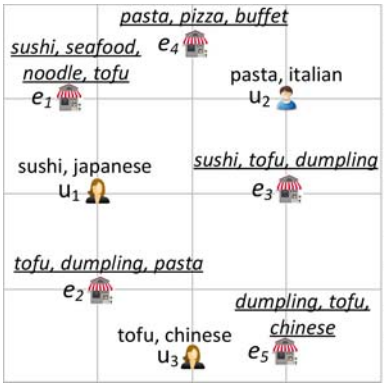
xiangw@cse.unsw.edu.au Ying.Zhang@uts.edu.au {zhangw, lxue}@cse.unsw.edu.au huangzengfeng@gmail.com

## ABSTRACT

As the prevalence of social media and GPS-enabled devices, a massive amount of *geo-textual* data has been generated in a stream fashion, leading to a variety of applications such as location-based recommendation and information dissemination. In this paper, we investigate a novel real-time top-$k$ monitoring problem over sliding window of streaming data; that is, we continuously maintain the *top-k* most relevant *geo-textual messages* (e.g., geo-tagged tweets) for a large number of *spatial-keyword subscriptions* (e.g., registered users interested in *local events*) simultaneously. To provide the most recent information under controllable memory cost, sliding window model is employed on the streaming geo-textual data. To the best of our knowledge, this is the first work to study top-$k$ spatial-keyword publish/subscribe over sliding window. A novel system, called Skype (Top-k **S**patial-**k**e**y**word **P**ublish/Subscrib**e**), is proposed in this paper. In Skype, to continuously maintain top-$k$ results for massive subscriptions, we devise a novel indexing structure upon subscriptions such that each incoming message can be immediately delivered on its arrival. Moreover, to reduce the expensive top-$k$ re-evaluation cost triggered by message expiration, we develop a novel *cost-based k-skyband* technique to reduce the number of re-evaluations in a cost-effective way. Extensive experiments verify the great efficiency and effectiveness of our proposed techniques.

## 1. INTRODUCTION

Recently, with the ubiquity of social media and GPS-enabled mobile devices, large volumes of geo-textual data have been generated in a stream fashion, leading to the popularity of *spatial-keyword publish/subscribe system* (e.g., [20, 8, 29, 19, 9]) in a variety of applications such as location-based recommendation and social network. In such a system, each individual user can register her interest (e.g., favorite food or sports) and location as a *spatial-keyword subscription*. A stream of *geo-textual messages* (e.g., e-coupon promotion and tweets with location information) continuously generated by publishers (e.g., local business) are rapidly fed to the relevant users.




| | $e_1$ | $e_2$ | $e_3$ | $e_4$ |
|---|---|---|---|---|
| $u_1$ | ***0.6*** | 0 | 0.4 | 0 |
| $u_2$ | 0 | 0.3 | 0 | ***0.7*** |
| $u_3$ | 0.3 | ***0.6*** | 0.5 | 0 |

$e_5$ arrives and $e_1$ expires

| | $e_2$ | $e_3$ | $e_4$ | $e_5$ |
|---|---|---|---|---|
| $u_1$ | 0 | ***0.4*** | 0 | 0 |
| $u_2$ | 0.3 | 0 | ***0.7*** | 0 |
| $u_3$ | 0.6 | 0.5 | 0 | ***0.9*** |

**Figure 1: E-coupon recommendation system**

The spatial-keyword publish/subscribe system has been studied in several existing work (e.g., [20, 8, 29]). Most of them are geared towards boolean matching, thus making the size of messages received by users unpredictable. This motivates us to study the problem of top-$k$ spatial-keyword publish/subscribe such that only the top-$k$ most relevant messages are presented to users. Moreover, we adopt the popular sliding window model [2] on geo-textual stream to provide the fresh information under controllable memory usage. In particular, for each subscription, we score a message based on their spatial and textual similarities, and the top-$k$ messages are continuously maintained against the update of the sliding window (i.e., message arrival and expiration). Below is a motivating example.

**Example** 1. *Figure 1 shows an example of location-aware e-coupon recommendation system. Three users interested in nearby restaurants are registered with their locations and favorite food, intending to keep an eye on the most relevant e-coupon issued recently. We assume the system only stores the most recent four e-coupons. An e-coupon e will be delivered to a user u if e has the highest score w.r.t. u according to their spatial and textual similarity (detailed score function will be introduced in Section 3). Initially, we have four e-coupons, and the top-1 answer of each user is shown in bold in the upper-right table, where the relevance score between user and e-coupon is depicted. When a new e-coupon $e_5$ arrives and the old e-coupon $e_1$ expires, the updated results are shown in bottom-right table. Particularly, the top-1 answer of $u_1$ is replaced by $e_3$ since $e_1$ is discarded from the system, while the answer of $u_3$ is replaced by $e_5$, as $e_5$ is the most relevant to $u_3$. The top-1 answer of $u_2$ remains unchanged.*

**Challenges.** Besides the existing challenges in spatial-keyword query processing [36, 14, 12, 26, 11, 34], our problem presents two new challenges.

The first challenge is to devise an efficient indexing structure for a huge number of subscriptions, such that each message from the high-speed stream can be disseminated immediately on its arrival. The only work that supports top-$k$ spatial-keyword publish/subscribe is proposed by Chen *et al.* [9]. In a nutshell, they first deduce a textual bound for each subscription and then employ DAAT (Document-at-a-time [5]) paradigm to traverse the inverted file built in each spatial node. However, we observe that the continuous top-$k$ monitoring problem is essentially a *threshold-based similarity search problem* from the perspective of message; that is, a new message will be delivered to a subscription if and only if its score is not less than the current threshold score (e.g., $k$-th highest score) of the subscription. Consequently, although the DAAT paradigm has been widely used for top-$k$ search (e.g., [13]), it is not suitable to our problem because the advanced threshold-based pruning techniques cannot be naturally integrated under DAAT paradigm.

The second challenge is the top-$k$ re-evaluation problem triggered by frequent message expiration from the sliding window. For example, in Figure 1, the expiration of $e_1$ invalidates the current top-1 answer (i.e., $e_1$) of $u_1$, and thus the system has to re-compute the new result for $u_1$ over the sliding window. It is cost-prohibitive to re-evaluate all the affected subscriptions from scratch when a message expires. Some techniques have been proposed to solve this problem (e.g., [32, 23, 4, 25]). Yi *et al.* [32] introduce a kmax strategy, trying to maintain top-$k'$ results, with $k'$ being a value between $k$ and kmax, rather than buffering the exact top-$k$ results. Later, Mouratidis *et al.* [23] notice that kmax ignores the dominance relationship between messages, and propose a novel idea to convert top-$k$ maintenance into *partial $k$-skyband* maintenance to reduce the number of re-evaluations. Nevertheless, they simply use the $k$-th score of a continuous query (i.e., subscription in our paper) as the threshold of its $k$-skyband without theoretical underpinnings, which may result in poor performance in practice.

In this paper, we propose a novel framework, namely Skype, to efficiently support top-$k$ **S**patial-**k**e**y**word **P**ublish/Subscrib**e** over sliding window. Two key modules, *message dissemination module* and *top-k re-evaluation module*, are designed to address the above two challenges. Specifically, the message dissemination module aims to rapidly deliver each arriving message to its affected subscriptions on its arrival. We devise efficient subscription indexing techniques which carefully integrate both spatial and textual information. Following the TAAT (Term-at-a-time [6]) paradigm, we significantly reduce the number of non-promising subscriptions for the incoming message by utilizing a variety of spatial and textual similarity-based pruning techniques. On the other hand, the top-$k$ re-evaluation module is designed to refill the top-$k$ results of subscriptions when their results expire. To alleviate the frequent re-evaluations, we develop a novel *cost-based $k$-skyband* technique which carefully selects the messages to be buffered based on a threshold value determined by a cost model, considering both top-$k$ re-evaluation cost and $k$-skyband maintenance cost.

**Contributions.** Our principal contributions are summarized as follows:

- We propose a novel framework, called Skype, which continuously maintains top-$k$ geo-textual messages for a large number of subscriptions over sliding window model. To the best of our knowledge, this is the first work to integrate sliding window model into spatial-keyword publish/subscribe system. (Section 4)
- For message dissemination module, we propose both *individual pruning technique* and *group pruning technique* to significantly improve the dissemination efficiency following the TAAT paradigm. (Section 5)
- For top-$k$ re-evaluation module, a novel *cost-based $k$-skyband* method is developed to determine the best threshold value with in-depth theoretical analysis. It is worth mentioning that our technique is a general approach which can be applied to other continuous top-$k$ problems over sliding window. (Section 6)
- We conduct extensive experiments to verify the efficiency and effectiveness of our proposed techniques. It turns out that Skype can achieve up to orders of magnitude improvement compared to the competitors. (Section 7)

# 2. RELATED WORK

## 2.1 Spatial-keyword Search

Spatial-keyword search has been widely studied in literatures. Most of traditional spatial-keyword queries aim to retrieve all the relevant geo-textual objects from a static database based on boolean matching (e.g., [36, 18, 14]) or score function (e.g., [12, 26, 11, 34]). The general idea is to combine both spatial index (e.g., R-Tree, Quadtree) and textual index (e.g., inverted file) to prune unpromising objects. A good summary of spatial-keyword query processing can be found in [10]. Some other extensions based on spatial-keyword processing have also been investigated, such as moving spatial-keyword query [16], collective spatial-keyword query [17] and reverse spatial-keyword query [21]. Note that a spatial-keyword search is an ad-hoc/snapshot query (i.e., user-initiated model) while our problem focuses on continuous query (i.e., server-initiated model).

## 2.2 Publish/Subscribe System

In a publish/subscribe system, users can register their interest as long-running queries at the server, and streaming publications (e.g., news) are delivered to relevant users whose interests are satisfied. Most of the existing work focus on boolean matching [30, 27, 33] or similarity-based ranking [24, 28]. These work are different from ours as they do not consider spatial information. Recently, spatial-keyword publish/subscribe system has been studied in a line of work (e.g., [20, 8, 29, 19, 9]). Among them, [20, 8, 29] study the boolean matching problem while [19] studies the similarity search problem, where each subscription has a *pre-given* threshold. These work are inherently different from ours, and it is non-trivial to extend their techniques to support top-$k$ monitoring.

The CIQ index proposed by Chen *et al.* [9] is the only close work that supports top-$k$ spatial-keyword publish/subscribe (shown in Figure 2). In CIQ, a Quadtree is used to partition the whole space. Each subscription is assigned to a number of covering cells, forming a disjoint partition of the entire space. In Figure 2, we assume all the subscriptions have the same cell covering, i.e., from $c_1$ to $c_7$. A textual bound (e.g., MinT) is precomputed for each subscription w.r.t. each assigned cell, as shown in the tables where the textual bounds

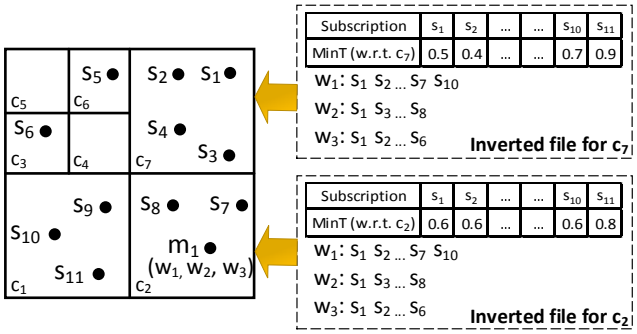



**Figure 2: Example of CIQ index**

w.r.t. $c_2$ and $c_7$ are displayed. An inverted file ordered by subscription id is built to organize the subscriptions assigned to each cell. For a new message (e.g., $m_1$), CIQ traverses all the inverted files with corresponding cells penetrated by message location (e.g., $c_2$) in DAAT paradigm, and finds all the subscriptions with textual similarity higher than the precomputed bound as candidates, which are then verified to get final results. However, we notice that DAAT paradigm employed in CIQ cannot integrate some advanced techniques for threshold-based similarity search, given that the nature of our problem is a threshold-based search problem. Contrary to CIQ, our indexing structure is designed for the TAAT paradigm, combined with advanced techniques for threshold-based pruning, thus enabling us to exclude a significant number of subscriptions. Moreover, CIQ indexes each subscription into multiple cells, taking advantage of precomputed spatial bound. However, the gain is limited since the number of covering cells for each subscription cannot be too large; otherwise, it would lead to extremely high memory cost. Thus, we turn to an *on-the-fly* spatial bound computation strategy, where each subscription is assigned to a single cell with finer spatial granularity. Finally, we remark that CIQ integrates a time decay function rather than a sliding window, which, in the worst case, may overwhelm the limited memory.

## 2.3 Top-k Maintenance Over Sliding Window

One critical problem for top-$k$ maintenance over sliding window is that, when an old element (i.e., message in this paper) expires, we have to recompute the top-$k$ results for the affected continuous queries (i.e., subscriptions in this paper), which is cost-expensive if we simply re-evaluate from scratch. On the flip side, it is also infeasible to buffer all elements and their scores for each individual query to avoid top-$k$ re-evaluation. Several techniques are proposed aiming to identify a trade-off between the number of re-evaluations and the buffer size. In [32], Yi *et al.* introduce a kmax approach. Rather than maintain exact top-$k$ results, they continuously maintain top-$k'$ results where $k'$ is between $k$ and a parameter kmax until the top-$k$ re-evaluation is invoked, i.e., the number of elements in the buffer is less than $k$. However, followed by observation from Mouratidis *et al.* [23], kmax may contain redundant elements due to the overlook of dominance relationship. Thus, Mouratidis *et al.* propose a $k$-skyband based top-$k$ monitoring algorithm to remove redundancy. Since it is very expensive to maintain the *full k-skyband* for each individual query, they only keep elements with scores not lower than the $k$-th highest score determined by the most recent top-$k$ re-evaluation. We observe that this setting is rather ad-hoc and thus may result in unsatisfactory performance in practice. Böhm *et al.* [4] utilize a delay buffer to avoid inserting the newly-arriving objects with low scores into the $k$-skyband. However, since each object has to probe query index twice during its life time, their method is not suitable to our problem given the large number of registered queries (i.e., subscriptions). Pripuzic *et al.* [25] propose a probabilistic $k$-skyband method to drop the data which is unlikely to become top-$k$ results in order to save space and improve efficiency. However, their technique may discard some top-$k$ elements due to its probabilistic nature. In this paper, we propose a novel *cost-based k-skyband* technique to carefully determine the size of $k$-skyband buffer based on a cost model.

# 3. PRELIMINARY

In this section, we formally present some concepts which are used throughout this paper.

Definition 1 (Geo-textual Message). *A geo-textual message is defined as $m = (\psi, \rho, t)$, where $m.\psi$ is a collection of keywords from a vocabulary $\mathcal{V}$, $m.\rho$ is a point location, and $m.t$ is the arrival time.*

Definition 2 (Spatial-keyword Subscription). *A spatial-keyword subscription is denoted as $s = (\psi, \rho, k, \alpha)$, where $s.\psi$ is a set of keywords, $s.\rho$ is a point location, $s.k$ is the number of messages that $s$ is willing to receive and $s.\alpha$ is the preference parameter used in the score function.*

To buffer the most recent data from geo-textual stream, we adopt a count-based sliding window defined as follows.

Definition 3 (Sliding Window). *Given a stream of geo-textual messages arriving in time order, the sliding window $\mathcal{W}$ over the stream with size $|\mathcal{W}|$ consists of most recent $|\mathcal{W}|$ geo-textual messages.*

In the following of the paper, we abbreviate *geo-textual message* and *spatial-keyword subscription* as *message* (denoted as $m$) and *subscription* (denoted as $s$) respectively if there is no ambiguity. We assume that the keywords in vocabulary $\mathcal{V}$, as well as the keywords in subscription and message, are sorted in increasing order of their term frequencies. The $i$-th keyword in $s$ is denoted as $s.\psi[i]$, and we use $s.\psi[i:j]$ to denote a subset of $s.\psi$, i.e., $\cup_{i\le k\le j}\{s.\psi[k]\}$. Particularly, $s.\psi[i:]$ denotes $\cup_{i\le k\le|s.\psi|}\{s.\psi[k]\}$. Message $m$ follows the similar notations.

**Score function.** To measure the *relevance* between a subscription $s$ and a message $m$, we employ a score function defined as follows:

$$\mathsf{Score}(s,m) = s.\alpha\cdot\mathsf{SSim}(s.\rho, m.\rho) + (1-s.\alpha)\cdot\mathsf{TSim}(s.\psi, m.\psi) \quad (1)$$

where $\mathsf{SSim}(s.\rho, m.\rho)$ is the spatial proximity and $\mathsf{TSim}(s.\psi, m.\psi)$ is the textual relevance between $s$ and $m$. Thus, a subscriber can receive messages which are not only close to her location but also fulfil her interest. Meanwhile, the parameter $\alpha$ can be adjusted by subscribers to best satisfy their diverse preferences.

To compute spatial proximity, we utilize Euclidean distance as follows:

$$\mathsf{SSim}(s.\rho, m.\rho) = 1 - \frac{\mathsf{EDist}(s.\rho, m.\rho)}{\mathsf{MaxDist}} \quad (2)$$

where $\mathsf{EDist}(s.\rho, m.\rho)$ is the Euclidean distance between $s$ and $m$, and MaxDist is maximum distance in the space.

For textual similarity, we employ the well-known *cosine similarity* [22], which is computed as:

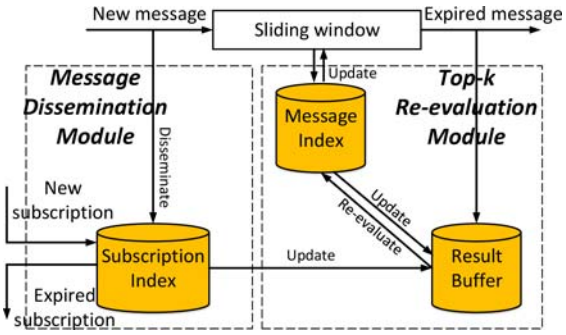



**Figure 3: Framework of Skype**

$$\mathsf{TSim}(s.\psi, m.\psi) = \sum_{w \in s.\psi \cap m.\psi} \mathsf{wt}(s.w) \cdot \mathsf{wt}(m.w) \quad (3)$$

where $\mathsf{wt}(s.w)$ and $\mathsf{wt}(m.w)$ are *tf-idf* weights of keyword $w$ in $s$ and $m$ respectively. Note that the weighting vectors of both $s$ and $m$ are normalized to unit length. Also, same as [9], to guarantee the top-$k$ results are *textual-relevant*, a message must contain at least one common keyword with a subscription to become its top-$k$ results.

**Problem statement.** Given a massive number of spatial-keyword subscriptions and a geo-textual stream, we aim to continuously monitor top-$k$ results for all the subscriptions against the stream over a sliding window $\mathcal{W}$ in real time.

# 4. FRAMEWORK

Figure 3 shows the framework of Skype (Top-$k$ **S**patial-**k**e**y**word **P**ublish/Subscrib**e**). We assume our system already has some registered subscriptions. An arriving message will be processed by **message dissemination module**, where a *subscription index* is built to find all the affected subscriptions and update their top-$k$ results. An expired message will be processed by **top-$k$ re-evaluation module**. Specifically, it will check against a *result buffer*, which maintains the top-$k$ results (possibly including some non-top-$k$ results) of all the subscriptions. For the subscriptions that cannot be refilled through result buffer, their top-$k$ results will be re-evaluated from scratch against a *message index* containing all the messages over the sliding window. Note that the message index can be implemented with any existing spatial-keyword index, such as IR-Tree [12] and S2I [26]. Skype can also support subscription update efficiently. A new subscription will be inserted into subscription index, with its top-$k$ results being initialized against message index, while an unregistered subscription will be deleted from both subscription index and result buffer.

# 5. MESSAGE DISSEMINATION

In this section, we introduce our message dissemination module. We first introduce our data structure in Section 5.1. Individual pruning and group pruning techniques, including early termination strategy, are proposed in Section 5.2 and Section 5.3 respectively. The algorithm details are presented in Section 5.4, followed by index maintenance in Section 5.5.

## 5.1 Subscription Index

Our subscription index is essentially a Quadtree structure integrated with inverted file in each leaf cell, as shown in Figure 4. For each registered subscription, we store its detailed information in a subscription table, and insert it into a leaf cell of Quadtree based on its spatial location. Note that in Quadtree, we only store the subscription id referring to its detailed information in subscription table. Within each leaf cell, an inverted file is built upon all the subscriptions inside the cell. Then each posting list in inverted file is further partitioned into groups based on the subscription preference $\alpha$ to enable group pruning. To facilitate the early termination, the subscriptions within each group are ordered based on their $k$-th highest scores. Moreover, for each subscription and each group, we materialize some statistical information which will be introduced in the following subsections.

## 5.2 Individual Pruning Technique

For each incoming message $m$, the key challenge is to determine all the subscriptions whose top-$k$ results are affected. Specifically, we denote the $k$-th highest score of a subscription $s$ as $\mathsf{kScore}(s)$. Then the top-$k$ results of $s$ need to be updated if $\mathsf{kScore}(s) \leq \mathsf{Score}(s, m)$. In this section, we propose a novel *location-aware prefix filtering* technique to prune an individual subscription efficiently.

### 5.2.1 Location-aware Prefix Filtering

For ease of exposition, we denote a spatial similarity upper bound between a subscription $s$ and a message $m$ as $\mathsf{SSimUB}(s.\rho, m.\rho)$. Based on Equation 1, we can derive a textual similarity threshold for pruning purpose accordingly:

$$\lambda_T(s.\psi, m.\psi) = \frac{\mathsf{kScore}(s)}{1 - s.\alpha} - \frac{s.\alpha}{1 - s.\alpha} \cdot \mathsf{SSimUB}(s.\rho, m.\rho) \quad (4)$$

The following lemma can be immediately derived from Equation 1 and Equation 4.

**Lemma** 1. *A message $m$ cannot affect top-$k$ results of a subscription $s$ if $\mathsf{TSim}(s.\psi, m.\psi) < \lambda_T(s.\psi, m.\psi)$.*

Lemma 1 claims that if the textual similarity between $s$ and $m$ is less than $\lambda_T(s.\psi, m.\psi)$, we can safely prune $s$.

To utilize Lemma 1, we employ prefix filtering technique, which is widely adopted in textual similarity join problems (e.g., [7, 3, 31]). The general idea of prefix filtering is to determine the similarity upper bound between two objects simply based on their prefixes. To adopt prefix filtering to our problem, we define a *location-aware prefix* as follows.

Definition 4 (Location-aware Prefix). *Given a subscription $s$, a message $m$ and a textual similarity threshold $\lambda_T(s.\psi, m.\psi)$, we use $\mathsf{pref}(s|m) = s.\psi[1 : p]$ to denote the location-aware prefix of $s$ w.r.t. $m$, where $p = \arg\min_i \sum_{i+1 \leq j \leq |s.\psi|} \mathsf{wt}(s.\psi[j]) < \lambda_T(s.\psi, m.\psi)$.*

The following lemma holds for location-aware prefix.

**Lemma** 2. *Given a subscription $s$ and a message $m$, if $\mathsf{pref}(s|m) \cap m.\psi = \emptyset$, we can safely prune $s$.*

Proof. Since $\mathsf{pref}(s|m) \cap m.\psi = \emptyset$, $\mathsf{TSim}(s.\psi, m.\psi) \leq \sum_{p+1 \leq i \leq |s.\psi|} \mathsf{wt}(s.\psi[i]) \cdot 1.0 < \lambda_T(s.\psi, m.\psi)$, where $p$ is defined in Definition 4 and 1.0 is the maximum weight for keyword in $m$. Then the lemma holds based on Lemma 1. □

**Example** 2. *Figure 5 shows an example of location-aware prefix, with 3 registered subscriptions and 3 incoming messages. The underlined value to the right of each keyword corresponds to its weight, and we do not normalize the keyword weight for simplicity. Assuming $\mathsf{SSimUB}(s_1.\rho, m_1.\rho) = 0.98$, then $\lambda_T(s_1.\psi, m_1.\psi) = \frac{0.7}{1-0.6} - \frac{0.6}{1-0.6} \cdot 0.98 = 0.28$. Thus, $\mathsf{pref}(s_1|m_1) = \{w_1, w_2, w_3\}$. Since $m_1.\psi \cap \mathsf{pref}(s_1|m_1) = \emptyset$, we can prune $s_1$ w.r.t. $m_1$.*

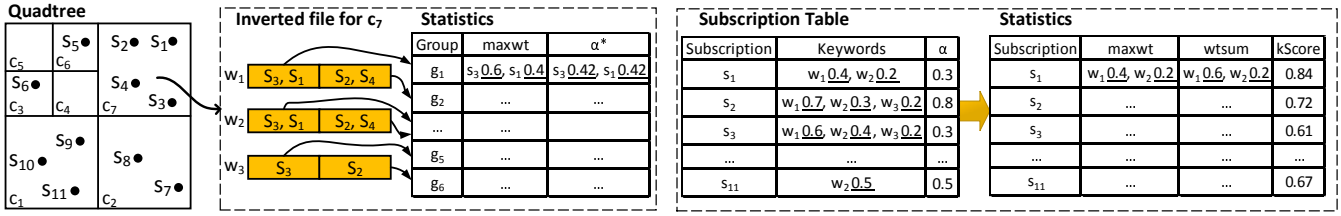



**Figure 4: Subscription index**

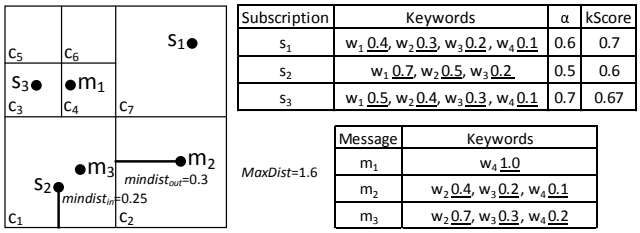



**Figure 5: Example of location-aware prefix**

It is noticed that different from conventional prefix technique (e.g., [3, 31]) where only the prefix of a data entry needs to be indexed, our location-aware prefix is dependent on the spatial location of messages, and different locations may lead to different prefixes. Thus, it is impossible to precompute and index the prefix of subscriptions.

To address this issue, we generate a threshold value for each keyword in $s.\psi$ to indicate whether this keyword should occur in the prefix regarding a message $m$. Specifically, for each keyword $s.\psi[i]$, a threshold value $\mathsf{wtsum}(s.\psi[i])$ is computed as follows:

$$\mathsf{wtsum}(s.\psi[i]) = \sum_{i \le j \le |s.\psi|} \mathsf{wt}(s.\psi[j]) \tag{5}$$

Then, we have the following lemma based on Definition 4.

**Lemma** 3. *Given a subscription $s$, a message $m$ and $\lambda_T(s.\psi, m.\psi)$, if $\mathsf{wtsum}(s.w) < \lambda_T(s.\psi, m.\psi)$, keyword $s.w$ must not be in $\mathsf{pref}(s|m)$.*

In this way, we can dynamically determine the location-aware prefix of a subscription w.r.t. an arriving message. Also, since $\mathsf{wtsum}(s.w)$ is irrelevant to incoming messages, it can be materialized for each subscription.

**Example** 3. *Following the same example in Figure 5, since $\mathsf{wtsum}(s_1.w_4) = 0.1 < \lambda_T(s_1.\psi, m_1.\psi) = 0.28$, $w_4$ is not in $\mathsf{pref}(s_1|m_1)$. Thus, we can prune $s_1$ w.r.t. $m_1$.*

**Max-weight refinement.** We notice that for a specific message $m$, we can compute a better location-aware prefix for $s$ by considering the maximum weight for the keywords in $m$. We first define $\mathsf{maxwt}(m.\psi[i])$ as:

$$\mathsf{maxwt}(m.\psi[i]) = \max_{i \le j \le |m.\psi|} \mathsf{wt}(m.\psi[j]) \tag{6}$$

Then we define a *refined location-aware prefix* as follows.

Definition 5 (Refined Location-aware Prefix). *Given a subscription $s$, a message $m$ and $\lambda_T(s.\psi, m.\psi)$, we use $\mathsf{pref}_+(s|m) = s.\psi[1:p]$ to denote the refined location-aware prefix of $s$ w.r.t. $m$, where $p = \arg\min_i \sum_{w \in s.\psi[i+1:]} \mathsf{maxwt}(m.w) \cdot \mathsf{wt}(s.w) < \lambda_T(s.\psi, m.\psi)$.*

Then, the following theorem holds immediately.

**Theorem** 1. *Given a subscription $s$, a message $m$ and their textual similarity threshold $\lambda_T(s.\psi, m.\psi)$, if $\mathsf{maxwt}(m.w) \cdot \mathsf{wtsum}(s.w) < \lambda_T(s.\psi, m.\psi)$, then keyword $s.w$ must not be in $\mathsf{pref}_+(s|m)$.*

**Example** 4. *Assuming $\mathsf{SSimUB}(s_1.\rho, m_2.\rho) = 0.99$ in Figure 5, then $\lambda_T(s_1.\psi, m_2.\psi) = 0.26$. Based on Lemma 3, $\mathsf{pref}(s_1|m_2) = \{w_1, w_2, w_3\}$, and thus $s_1$ cannot be pruned w.r.t. $m_2$. However, if we consider $\mathsf{maxwt}$, then $\mathsf{pref}_+(s_1|m_2) = \{w_1\}$. Thus, $s_1$ can be pruned w.r.t. $m_2$.*

### 5.2.2 Spatial Bound Estimation

In this section, we discuss the computation of $\mathsf{SSimUB}(s.\rho, m.\rho)$ between a subscription $s$ and a message $m$ in order to get a better threshold $\lambda_T(s.\psi, m.\psi)$ for efficient location-aware prefix filtering. To this end, we employ a spatial index to group subscriptions with similar locations, such that the spatial upper bound for a group of subscriptions can be computed simultaneously. Due to the easy implementation and well-adaptiveness to skewed spatial distributions, we choose Quadtree to index subscriptions. Specifically, each subscription $s$ is assigned into a leaf cell $c$ with range $c.r$ based on its location $s.\rho$. Then the following two types of spatial bounds can be calculated:

**Inner spatial bound $\mathsf{SSimUB}_{in}(s.\rho, c.r)$.** Assuming a subscription $s$ is inside a cell $c$, an inner spatial bound $\mathsf{SSimUB}_{in}(s.\rho, c.r)$ regarding $s$ is computed by considering the *mindist* from $s$ to the nearest boundary of cell range $c.r$. It is obvious that for any message $m$ outside $c$, we have $\mathsf{SSimUB}_{in}(s.\rho, c.r) \ge \mathsf{SSim}(s.\rho, m.\rho)$. An example is shown in Figure 5. Since the *mindist* from $s_2$ to $c_1$ is 0.25, $\mathsf{SSimUB}_{in}(s_2.\rho, c_1.r) = 1 - \frac{0.25}{1.6} = 0.84$ if we assume the MaxDist in the space is 1.6.

**Outer spatial bound $\mathsf{SSimUB}_{out}(m.\rho, c.r)$.** Assuming a message $m$ is outside a cell $c$, an outer spatial bound $\mathsf{SSimUB}_{out}(m.\rho, c.r)$ regarding $m$ is computed by using the *mindist* from $m$ to $c$. It is obvious that for any subscription $s$ inside $c$, we have $\mathsf{SSimUB}_{out}(m.\rho, c.r) \ge \mathsf{SSim}(s.\rho, m.\rho)$. An example is also shown in Figure 5. The *mindist* from $m_2$ to $c_1$ is 0.3, and thus $\mathsf{SSimUB}_{out}(m_2.\rho, c_1.r) = 1 - \frac{0.3}{1.6} = 0.81$. By combining both inner and outer distance, we can get a tighter spatial upper bound between $s_2$ and $m_2$ as $\mathsf{SSimUB}(s_2.\rho, m_2.\rho) = 1 - \frac{0.25+0.3}{1.6} = 0.65$.

Note that the inner spatial bound can be precomputed and materialized, while the outer spatial bound has to be computed on-the-fly as it is relevant to the location of an arriving message. However, the computation cost of $\mathsf{SSimUB}_{out}(m.\rho, c.r)$ is not expensive since we only need to compute this value against each leaf cell. Finally, we remark that when $s$ and $m$ are within the same cell, both $\mathsf{SSimUB}_{in}(s.\rho, c.r)$ and $\mathsf{SSimUB}_{out}(m.\rho, c.r)$ are always 1.0.

**Example** 5. *An example is shown in Figure 5. If we assume the $\mathsf{SSimUB}(s_2.\rho, m_2.\rho) = 1.0$, we have $\lambda_T(s_2, m_2) = \frac{0.6}{1-0.5} - \frac{0.5}{1-0.5} \cdot 1.0 = 0.20$, and $\mathsf{pref}_+(s_2|m_2) = \{w_1, w_2\}$. Thus, $s_2$ cannot be pruned w.r.t. $m_2$. However, if we utilize the inner spatial bound and outer spatial bound together, we have $\mathsf{SSimUB}(s_2.\rho, m_2.\rho) = 1 - \frac{0.25+0.3}{1.6} = 0.65$, $\lambda_T(s_2, m_2) = 0.55$, and $\mathsf{pref}_+(s_2|m_2) = \{w_1\}$. In this case, we can safely prune $s_2$ w.r.t. $m_2$.*

### 5.2.3 Bound Estimation for Unseen Keywords

Since we employ TAAT paradigm to visit inverted file, we can estimate a textual upper bound for unseen keywords. If this upper bound plus the textual similarity that has already been computed is still less than the required threshold, we can safely prune $s$. The textual upper bound between the unseen keywords of $s$ and $m$ can be computed as follows:

$$\mathsf{TSimUB}(s.\psi[i:], m.\psi[j:]) = min\Big\{\mathsf{wtsum}(s.\psi[i]) \times \mathsf{maxwt}(m.\psi[j]), \mathsf{wtsum}(m.\psi[j]) \times \mathsf{maxwt}(s.\psi[i])\Big\} \quad (7)$$

where $i$ and $j$ are starting positions of unseen keywords. Then we have the following theorem to filter an unpromising subscription based on Lemma 1.

**Theorem** 2. *Given a subscription $s$, a message $m$ and their textual similarity threshold $\lambda_T(s.\psi, m.\psi)$, assuming we have already computed the partial similarity between $s.\psi[1:i]$ and $m.\psi[1:j]$, denoted as $\mathit{TSim}(s.\psi[1:i], m.\psi[1:j])$, if $\mathit{TSim}(s.\psi[1:i], m.\psi[1:j]) + \mathit{TSimUB}(s.\psi[i+1:], m.\psi[j+1:]) < \lambda_T(s.\psi, m.\psi)$, we can safely prune $s$.*

**Example** 6. *In Figure 5, based on location-aware prefix filtering in Theorem 1, $\lambda_T(s_3.\psi, m_3.\psi) = 0.48$, and $\mathit{pref}_+(s_3|m_3) = \{w_1, w_2\}$. Thus $s_3$ cannot be pruned w.r.t. $m_3$, since they have common keyword $w_2$. Assuming current visiting positions of $s_3$ and $m_3$ are 2 and 1 respectively (corresponding to keyword $w_2$), then $\mathit{TSim}(s_3.\psi[1:2], m_3.\psi[1:1]) = 0.4 \cdot 0.7 = 0.28$, and $\mathit{TSimUB}(s_3.\psi[3:], m_3.\psi[2:]) = \min(0.4 \cdot 0.3, 0.5 \cdot 0.3) = 0.12$. As $0.28 + 0.12 < \lambda_T(s_3.\psi, m_3.\psi) = 0.48$, we can immediately prune $s_3$.*

## 5.3 Group Pruning Technique

After applying individual pruning technique, many subscriptions can be pruned without the need to compute their exact similarity w.r.t. a message. To further enhance the performance, we propose a novel *Group Pruning Technique* such that we can skip a group of subscriptions without the need to visit them individually. To begin with, we first define *subscription-dependent prefix* for a message.

Definition 6 (Subscription-dependent Prefix). *Given a message $m$, a subscription $s$ and $\lambda_T(s.\psi, m.\psi)$, we use $\mathit{pref}(m|s) = m.\psi[1:p]$ to denote the subscription-dependent prefix of $m$ w.r.t. $s$, where $p = \arg\min_i \sum_{i+1 \le j \le |m.\psi|} \mathit{wt}(m.\psi[j]) < \lambda_T(s.\psi, m.\psi)$.*

Similar to Definition 5, a *refined subscription-dependent prefix*, denoted as $\mathsf{pref}_+(m|s)$, can be defined by considering the maximum weight of keywords in a subscription. Then the following lemma holds immediately.

**Lemma** 4. *Given a message $m$, a subscription $s$ and $\lambda_T(s.\psi, m.\psi)$, if $\mathit{maxwt}(s.w) \cdot \mathit{wtsum}(m.w) < \lambda_T(s.\psi, m.\psi)$, keyword $m.w$ must not be in $\mathit{pref}_+(m|s)$.*

Let us denote the posting list of keyword $w$ in cell $c$ as $plist(c, w)$. Then based on Lemma 4, for a subscription $s$ in $plist(c, w)$, if $w \notin \mathsf{pref}_+(m|s)$, we can safely skip $s$. Further, if this holds for a group of subscriptions on $plist(c, w)$, we can safely skip the whole group as follows.

**Lemma** 5. *Given a message $m$, a keyword $w \in m.\psi$, a posting list $plist(c, w)$ and a group of subscriptions $\mathcal{G}$ inside $plist(c, w)$, if $\max_{s \in \mathcal{G}} \{\mathit{maxwt}(s.w)\} \cdot \mathit{wtsum}(m.w) < \min_{s \in \mathcal{G}} \{\lambda_T(s.\psi, m.\psi)\}$, the whole group $\mathcal{G}$ can be skipped.*

The left side of the inequality in Lemma 5 can be computed in $O(1)$ time since we can materialize $\max_{s \in \mathcal{G}} \{\mathsf{maxwt}(s.w)\}$ for each group. However, for the right side, it would be quite inefficient if we compute it on the fly for each new message. To avoid this, we propose a lower bound for $\min_{s \in \mathcal{G}} \{\lambda_T(s.\psi, m.\psi)\}$ which can be computed in constant time. In the following, we first present the subscription grouping strategy and then introduce the details of the lower bound deduction.

### 5.3.1 $\alpha$-Partition Scheme

Intuitively, we should group subscriptions with similar $\lambda_T(s.\psi, m.\psi)$ such that we can get a tighter textual threshold for the group. We first let $\mathsf{SSimUB}(s.\rho, m.\rho) = \mathsf{SSimUB}_{out}(m.\rho, c.r)$. It is observed from Equation 4 that, for the computation of $\lambda_T(s.\psi, m.\psi)$, only $\frac{\mathsf{kScore}(s)}{1 - s.\alpha}$ and $\frac{s.\alpha}{1 - s.\alpha}$ are dependent on $s$ while $\mathsf{SSimUB}(s.\rho, m.\rho)$ is irrelevant to $s$. For simplicity, we denote $\frac{\mathsf{kScore}(s)}{1 - s.\alpha}$ as $\mathsf{kScore}^*(s)$ and $\frac{s.\alpha}{1 - s.\alpha}$ as $s.\alpha^*$ respectively. Then, we partition subscriptions into groups based on their $\alpha^*$ values, such that the subscriptions inside a group have similar $\alpha^*$ values. We employ a quantile-based method to partition the domain of $\alpha^*$ to ensure that each group has similar number of subscriptions. Then, for each group $\mathcal{G}$, we derive a lower bound for $\min_{s \in \mathcal{G}} \{\lambda_T(s.\psi, m.\psi)\}$ as stated in the following.

**Theorem** 3. *Given a group $\mathcal{G}$ generated by $\alpha$-partition in a posting list $plist(c, w)$, we denote $\min_{s \in \mathcal{G}} \{\mathit{kScore}^*(s)\}$ as $\mathit{kScore}^*(\mathcal{G})$ and $\max_{s \in \mathcal{G}} \{s.\alpha^*\}$ as $\mathcal{G}.\alpha^*$. If $\max_{s \in \mathcal{G}} \{\mathit{maxwt}(s.w)\} \cdot \mathit{wtsum}(m.w) < \mathit{kScore}^*(\mathcal{G}) - \mathcal{G}.\alpha^* \cdot \mathit{SSimUB}_{out}(m.\rho, c.r)$, the whole group $\mathcal{G}$ can be skipped safely.*

**Time complexity.** The condition checking in Theorem 3 takes $O(1)$ time, since we can precompute the values of $\mathsf{kScore}^*(\mathcal{G})$ and $\mathcal{G}.\alpha^*$.

### 5.3.2 Early Termination Within Group

When a group $\mathcal{G}$ cannot be skipped given a message, we have to check each subscription in it. To avoid this, we propose an *early termination technique* to early stop within a group when the group cannot be skipped totally. To enable early termination, for each group $\mathcal{G}$ in $plist(c, w)$, we sort the subscriptions in $\mathcal{G}$ by their $\mathsf{kScore}^*$ values increasingly. For each subscription $s$ in $\mathcal{G}$, we denote the subscriptions with $\mathsf{kScore}^*$ not less than $\mathsf{kScore}^*(s)$ as $\mathcal{G}[s] = \{s' \in \mathcal{G} | \mathsf{kScore}^*(s') \ge \mathsf{kScore}^*(s)\}$, and maintain two statistics $\mathsf{maxwt}(\mathcal{G}[s])$ and $\mathcal{G}[s].\alpha^*$ w.r.t. keyword $w$ as follows:

$$\mathsf{maxwt}(\mathcal{G}[s]) = \max_{s' \in \mathcal{G}[s]} \{\mathsf{maxwt}(s'.w)\} \quad (8)$$

$$\mathcal{G}[s].\alpha^* = \max_{s' \in \mathcal{G}[s]} \{s'.\alpha^*\} \quad (9)$$

Then we can employ early termination based on the following theorem.

**Theorem** 4. *Given a group $\mathcal{G}$ inside a posting list $plist(c, w)$, and assuming $\hat{s}$ is the subscription with smallest position in $\mathcal{G}$ such that the following inequality holds: $\mathit{maxwt}(\mathcal{G}[\hat{s}]) \cdot \mathit{wtsum}(m.w) < \mathit{kScore}^*(\hat{s}) - \mathcal{G}[\hat{s}].\alpha^* \cdot \mathit{SSimUB}_{out}(m.\rho, c.r)$, then there is no need to check the subscriptions after $\hat{s}$ (including $\hat{s}$ itself).*

Proof. For any subscription $s'$ after $\hat{s}$, the following inequalities hold: $\mathsf{kScore}^*(\hat{s}) \le \mathsf{kScore}^*(s')$, $\mathsf{maxwt}(\mathcal{G}[\hat{s}]) \ge$

$\mathsf{maxwt}(\mathcal{G}[s'])$, $\mathcal{G}[\hat{s}].\alpha^* \geq \mathcal{G}[s'].\alpha^*$. Thus, $\mathsf{maxwt}(s'.w) \cdot \mathsf{wtsum}(m.w) \leq \mathsf{maxwt}(\mathcal{G}[s']) \cdot \mathsf{wtsum}(m.w) \leq \mathsf{maxwt}(\mathcal{G}[\hat{s}]) \cdot \mathsf{wtsum}(m.w) < \mathsf{kScore}^*(\hat{s}) - \mathcal{G}[\hat{s}].\alpha^* \cdot \mathsf{SSimUB}_{out}(m.\rho, c.r) \leq \mathsf{kScore}^*(s') - \mathcal{G}[s'].\alpha^* \cdot \mathsf{SSimUB}_{out}(m.\rho, c.r) \leq \mathsf{kScore}^*(s') - s'.\alpha^* \cdot \mathsf{SSimUB}_{out}(m.\rho, c.r) = \lambda_T(s'.\psi, m.\psi)$. Based on Lemma 4, this theorem holds immediately. □

**Time complexity.** To speed-up the real-time processing, we precompute $\mathsf{maxwt}(\mathcal{G}[s])$ and $\mathcal{G}[s].\alpha^*$ and store them with each subscription in the group $\mathcal{G}$. The condition checking in Theorem 4 can be efficiently computed in $O(log|\mathcal{G}|)$ time with a binary search method.

### 5.3.3 Cell-based Pruning

Besides the above group pruning technique, we notice that for some cells which are far away from the location of an arriving message, we can safely skip the whole cell. Specifically, for each subscription $s$ within a cell $c$, we can derive a spatial similarity threshold as follows:

$$\lambda_S(s.\rho) = \frac{\mathsf{kScore}(s)}{s.\alpha} - \frac{1 - s.\alpha}{s.\alpha} \tag{10}$$

where we assume the textual similarity achieves the largest value, i.e., 1. Then we can reach the following lemma.

**Lemma** 6. *Given a cell $c$, if $\min_{s \in c} \lambda_S(s.\rho) > \mathsf{SSimUB}_{out}(m.\rho, c.r)$, we can safely prune all the subscriptions in cell $c$.*

## 5.4 Dissemination Algorithm

Algorithm 1 shows our message dissemination algorithm. We follow a *filtering-and-verification* paradigm, where we first generate a set of candidate subscriptions (Lines 1-28), and then compute the exact scores to determine the truly affected ones, with the updated results being disseminated accordingly (Line 29). Specifically, we first initialize an empty map $\mathcal{R}$ to store candidates with their scores (Line 1). Then the maxwt and wtsum values for all the keywords in the arriving message $m$ are computed for later use (Line 3). For each leaf cell $c$ surviving from cell pruning (Line 5), we first compute $\mathsf{SSimUB}_{out}(m.\rho, c.r)$ and then traverse the inverted file in cell $c$ following a TAAT manner. For each group $\mathcal{G}$ encountered in $plist(c, w)$ (Line 10), we prune $\mathcal{G}$ if group pruning can be applied (Line 11); otherwise, we identify $\hat{s}$ for early termination based on Theorem 4 (Line 13 and Line 15). For each surviving subscription $s$, we employ location-aware prefix filtering (Line 20) and bound estimation for unseen keywords (Line 27) to prune it as early as possible. For the surviving subscriptions, we store the accumulated textual similarity so far w.r.t. $m$ in $\mathcal{R}$, while for the pruned subscriptions, we set $\mathcal{R}[s]$ to negative infinity (Line 28). Finally, for each subscription in $\mathcal{R}$ with $\mathcal{R}[s] > 0$, we verify it and update its top-$k$ results if needed (Line 29). Note that when verifying a candidate $s$, we only need to compute the exact spatial similarity to get the final score because the textual similarity, i.e., $\mathcal{R}[s]$, has already been computed. The statistics relevant to pruning techniques are also updated in Line 29.

## 5.5 Index Maintenance

Our indexing structure can also support subscription update efficiently. For a new subscription $s$, we first find the leaf cell containing its location, and then insert it into the inverted file with $O(|s.\psi| \cdot \log |\mathcal{G}|)$ cost. Note that the statistics mentioned above need to be updated accordingly. For an expired subscription, we simply delete it from index and update the statistics if necessary.

**Algorithm 1**: MessageDissemination($m$)

```
   Input : m : a new incoming message
1  R := ∅                                              /* A candidate map */;
2  for 1 ≤ i ≤ |m.ψ| do
3      Compute maxwt(m.ψ[i]) and wtsum(m.ψ[i]);
4  for each leaf cell c in the Quadtree do
5      if c is pruned by Lemma 6 then                  /* Cell pruning */
6          Continue;
7      Compute outer spatial bound SSimUB_out(m.ρ, c.r);
8      for 1 ≤ i ≤ |m.ψ| do
9          w := m.ψ[i] ;
10         for each group G in plist(c, w) do
11             if max_{s∈G} {maxwt(s.w)} · wtsum(m.w) <
               kScore*(G) − G.α* · SSimUB_out(m.ρ, c.r) then
               /* Group pruning based on Theorem 3 */
12                 Continue;
13             Identify ŝ based on Theorem 4 ;
14             for each subscription s in group G do
15                 if s == ŝ then                      /* Early termination
                   based on Theorem 4 */
16                     Break;
17                 if s ∈ R && R[s] == −∞ then
18                     Continue;
19                 Compute λ_T(s.ψ, m.ψ) based on both inner
                   and outer spatial bounds;
20                 if maxwt(m.w) · wtsum(s.w) < λ_T(s.ψ, m.ψ)
                   then                                /* Theorem 1 */
21                     Continue;
22                 if s ∈ R then
23                     R[s]+ := wt(s.w) · wt(m.w);
24                 else
25                     R := R ∪ s; R[s] := wt(s.w) · wt(m.w);
26                 pos := the position of keyword w in s.ψ;
27                 if R[s] + TSimUB(s.ψ[pos + 1 :], m.ψ[i + 1 :
                   ]) < λ_T(s.ψ, m.ψ) then             /* Theorem 2 */
28                     R[s] := −∞;
29 VerifyAndUpdate(R) and disseminate updated results to
   corresponding subscriptions;
```

# 6. TOP-K RE-EVALUATION

In this section, we present the details of top-$k$ re-evaluation module. We first introduce some background knowledge for $k$-skyband in Section 6.1. Then we present our *cost-based skyband* technique in detail in Section 6.2. In the following of this paper, we denote the $k$-skyband buffer (either fully or partially) of a subscription $s$ as $s.\mathcal{A}$ for simplicity, and the exact top-$k$ results are denoted as $s.\mathcal{A}_k$. Meanwhile, $s.k$ is denoted as $k$ if it is clear from context.

## 6.1 K-Skyband

The idea of utilizing $k$-skyband to reduce the number of re-evaluations for top-$k$ queries over a sliding window is first proposed in [23]. In particular, for a given subscription $s$, only the messages in its corresponding $k$-skyband can appear in its top-$k$ results over the sliding window, thus being maintained. Following are formal definitions of *dominance* and *$k$-skyband*.

DEFINITION 7 (DOMINANCE). *A message $m_1$ dominates another message $m_2$ w.r.t. a subscription $s$ if both $\mathsf{Score}(s, m_1) \geq \mathsf{Score}(s, m_2)$ and $m_1.t > m_2.t$ hold.*

DEFINITION 8 ($k$-SKYBAND). *The $k$-skyband of a subscription $s$, denoted as $s.\mathcal{A}$, contains a set of messages which are dominated by less than $k$ other messages.*

**Algorithm 2**: TopkRe-evaluation($m$)

```
Input  : m : an expired message
1 for each subscription s whose k-skyband buffer contains m do
2     Delete m from s.A;
3     if |s.A| < k then
4         Compute the best s.θ based on our cost model;
5         Retrieve B := {m|m ∈ W && Score(s, m) ≥ s.θ};
6         s.A := k-skyband of B;
7     Extract s.A_k from s.A;
```

Instead of keeping $k$-skyband over all the messages in the sliding window, which is cost-prohibitive, Mouratidis *et al.* [23] maintain a *partial $k$-skyband*. Specifically, they only maintain the messages with score not lower than a threshold $s.\theta$, where $s.\theta$ is the $\mathsf{kScore}(s)$ after the most recent top-$k$ re-evaluation for $s$ and remains unchanged until next re-evaluation is triggered. However, as our experiments suggest, the threshold $s.\theta$ in [23] is too high to make $k$-skyband really work.

To alleviate the above problem, we propose a novel *cost-based $k$-skyband* technique, which judiciously selects a best threshold $s.\theta$ for the $k$-skyband maintenance of each subscription. To start with, we present an overview of our top-$k$ re-evaluation algorithm in Algorithm 2. For each subscription $s$ containing the expired message $m$, if the size of $s.\mathcal{A}$ after deleting $m$ is less than $k$, we need to re-evaluate its top-$k$ results from scratch. Specifically, we first compute a proper threshold $s.\theta$ based on our cost model (Line 4), and then re-compute $k$-skyband buffer $s.\mathcal{A}$ based on $B$, which contains all the messages with score at least $s.\theta$ (Line 5 and Line 6) [1]. Note that $B$ can be computed by utilizing message index. Finally, we extract top-$k$ results from $s.\mathcal{A}$ (Line 7). The key challenge here is to estimate a best threshold $s.\theta$, which will be discussed in the following in detail. We remark that we use the term *re-evaluation* to refer in particular to the top-$k$ re-computation against message index.

## 6.2 Cost-based K-Skyband

The general idea of our cost-based $k$-skyband model is to select a best threshold $s.\theta$ for each subscription such that the overall cost defined in the cost model can be minimized. The following theorem guarantees that, as long as we maintain a partial $k$-skyband over all the messages with score not lower than $s.\theta$, we can extract top-$k$ results from partial $k$-skyband safely when some message expires.

**Theorem** 5. *Given a subscription $s$, let $\mathsf{kScore}_{last}(s)$ be the $\mathsf{kScore}(s)$ after the most recent top-$k$ re-evaluation for $s$. We always have $s.\mathcal{A}_k \subseteq s.\mathcal{A}$ if the following conditions hold: (1) $|s.\mathcal{A}| \geq k$; (2) $s.\mathcal{A}$ is a partial $k$-skyband which is built over all the messages with score at least $s.\theta$ in the sliding window, where $0 \leq s.\theta \leq \mathsf{kScore}_{last}(s)$.*

PROOF. By contradiction. Assume there is a message $m \in s.\mathcal{A}_k$ while $m \notin s.\mathcal{A}$, then there are two possible cases: (1) $\mathsf{Score}(s,m) \geq s.\theta$ and $m$ is dominated by more than $k$ messages in $s.\mathcal{A}$, which contradicts with $m \in s.\mathcal{A}_k$; (2) $\mathsf{Score}(s,m) < s.\theta$, which also contradicts with $m \in s.\mathcal{A}_k$ since every message in $s.\mathcal{A}$ has a higher score than $m$. □

Thus, based on Theorem 5, we can safely extract top-$k$ results from $k$-skyband buffer $s.\mathcal{A}$ when $|s.\mathcal{A}| \geq k$; when $|s.\mathcal{A}| < k$, we have to re-evaluate from message index.

[1]The same technique in [23] is used to compute $k$-skyband.

Our cost-based $k$-skyband model, based on Theorem 5, aims to find the best $s.\theta$ such that the overall cost can be minimized for each subscription. We mainly consider two costs. The first one is *$k$-skyband maintenance cost*, denoted as $\mathcal{C}_{sm}(s)$, which is triggered upon message arrival and expiration. The second one is *top-$k$ re-evaluation cost*, denoted as $\mathcal{C}_{re}(s)$, which is triggered when some message expires and the top-$k$ results can no longer be retrieved from $k$-skyband buffer. We aim to estimate the expected overall cost w.r.t. each message update, i.e., message arrival and message expiration, each of which we assume occurs with probability $\frac{1}{2}$ as the window slides. To simplify the presentation, we denote as $\mathsf{prob}(s.\theta)$ the probability that the score between a random message and a subscription $s$ is at least $s.\theta$. We may immediately derive $\mathsf{prob}(s.\theta)$ for a given $s.\theta$ from historical data, assuming the score follows previous distribution. The details of these two costs are presented in the following respectively.

### 6.2.1 K-Skyband Maintenance Cost

The maintenance of $k$-skyband is triggered when the following two types of updates happen, both with probability $\frac{1}{2} \cdot \mathsf{prob}(s.\theta)$.

The first type of update is triggered when a message $m$ with score at least $s.\theta$ arrives. Apart from the insertion of $m$ into $s.\mathcal{A}$, the dominance counters of all the messages in $s.\mathcal{A}$ with score not higher than $\mathsf{Score}(s,m)$ will increase by 1, and the messages with dominance counter equal to $k$ will be evicted. Since we implement our $k$-skyband buffer with a linked list sorted by $\mathsf{Score}(s,m)$. The above operations can be processed in $O(|s.\mathcal{A}|)$ time with a linear scan. The next challenge is to estimate $|s.\mathcal{A}|$. Based on the independence assumption between score dimension and time dimension, the expected number, i.e., $|s.\mathcal{A}|$, of messages in the partial $k$-skyband is $k \cdot \ln(\frac{|\mathcal{W}| \cdot \mathsf{prob}(s.\theta)}{k})$ [35], where $|\mathcal{W}|$ is the size of sliding window.

The second type of update occurs when an old message $m$ among the $k$-skyband buffer of $s$ expires. In this case, we only need to delete $m$ from $s.\mathcal{A}$ in $O(|s.\mathcal{A}|)$ time. Note that $m$ does not dominate any remaining messages and therefore the dominance counters of the remaining messages are not affected. Finally, we get the total cost of $k$-skyband maintenance as follows:

$$\begin{aligned} \mathcal{C}_{sm}(s) &= \frac{1}{2} \cdot \mathsf{prob}(s.\theta) \cdot |s.\mathcal{A}| + \frac{1}{2} \cdot \mathsf{prob}(s.\theta) \cdot |s.\mathcal{A}| \\ &= \mathsf{prob}(s.\theta) \cdot k \cdot \ln(\frac{|\mathcal{W}| \cdot \mathsf{prob}(s.\theta)}{k}) \end{aligned} \quad (11)$$

### 6.2.2 Top-k Re-evaluation Cost

The top-$k$ re-evaluation cost can be formalized as:

$$\mathcal{C}_{re}(s) = \mathcal{C}_{topk}(s) \cdot \frac{1}{\mathbb{Z}(s)} \quad (12)$$

where $\mathcal{C}_{topk}(s)$ is the average top-$k$ computation cost over message index for subscription $s$, and $\mathbb{Z}(s)$ is the expected number of message updates that is required to trigger top-$k$ re-evaluation, i.e., leading to $|s.\mathcal{A}| < k$. The value of $\mathcal{C}_{topk}(s)$ can be estimated by the average of previous top-$k$ computation cost against message index. The remaining issue is how to estimate $\mathbb{Z}(s)$, which is non-trivial.

To solve this problem, we model the streaming updating process as a simple *random walk*. A random walk is a stochastic sequence $RW_n$, with $RW_0$ being the starting position, defined by

$$RW_n = \sum_{i=1}^{n} X_i \quad (13)$$

where $X_i$ are independent and identically distributed random variables (i.e., i.i.d.). The random walk is *simple* if $\mathsf{prob}(X_i = 1) = p$, $\mathsf{prob}(X_i = -1) = q$ and $\mathsf{prob}(X_i = 0) = r$, where $p + q + r = 1$. We map the estimation of $\mathbb{Z}(s)$ into a simple random walk as follows. We model the change of $k$-skyband buffer $s.\mathcal{A}$ w.r.t. each message update as an i.i.d. variable $X_i$. $X_i$ is set to 1 when the size of $s.\mathcal{A}$ is increased by 1 at $i$-th step, while $X_i$ is set to $-1$ when the size of $s.\mathcal{A}$ is decreased by 1. When the size of $s.\mathcal{A}$ does not change, $X_i$ is set to 0. Unfortunately, it is difficult to estimate the probability of $\mathsf{prob}(X_i = 1)$ and $\mathsf{prob}(X_i = -1)$ for each message update, due to the eviction of messages by dominance relationship. For example, for a new message, the size of $s.\mathcal{A}$ may decrease rather than increase due to the eviction of messages with dominance counter reaching $k$. To address this problem, rather than estimating $\mathbb{Z}(s)$ for $s.\mathcal{A}$ maintenance, we estimate $\mathbb{Z}'(s)$ for $s.\mathcal{A}'$, which contains all the messages with score not lower than $s.\theta$. Specifically, when we maintain $s.\mathcal{A}'$, we do not consider the dominance relationship between messages for each message update, and thus the messages dominated by $k$ (or more) messages are not evicted. Clearly, $s.\mathcal{A}'$ is a superset of $s.\mathcal{A}$, i.e., $s.\mathcal{A} \subseteq s.\mathcal{A}'$. The following theorem guarantees that $\mathbb{Z}(s)$ is equal to $\mathbb{Z}'(s)$.

**Theorem** 6. *The expected number of message updates that is required to trigger top-k re-evaluation for $s.\mathcal{A}$ maintenance is the same as that for $s.\mathcal{A}'$ maintenance, i.e., $\mathbb{Z}(s) = \mathbb{Z}'(s)$.*

Proof. To show $\mathbb{Z}(s)$ is equal to $\mathbb{Z}'(s)$, it is sufficient to prove that $|s.\mathcal{A}| < k$ *if and only if* $|s.\mathcal{A}'| < k$ at any point. Initially, we have $s.\mathcal{A} \cup s.\mathcal{A}_{evict} = s.\mathcal{A}'$ and $s.\mathcal{A} \cap s.\mathcal{A}_{evict} = \emptyset$, where $s.\mathcal{A}_{evict}$ is a set of messages evicted by messages in $s.\mathcal{A}$. We prove that after each message update, $s.\mathcal{A} \cup s.\mathcal{A}_{evict} = s.\mathcal{A}'$ always holds. As to the arrival of a new message $m$, if $m$ is inserted into $s.\mathcal{A}'$, it will also be inserted into $s.\mathcal{A}$ since $m$ will not be dominated by any existing message due to its freshness; and vice versa. Note that some messages may be evicted from $s.\mathcal{A}$ to $s.\mathcal{A}_{evict}$ due to dominance relationship. Thus, $s.\mathcal{A} \cup s.\mathcal{A}_{evict} = s.\mathcal{A}'$ holds. As to the expiration of an old message $m$, (1) If $m \in s.\mathcal{A}$, $m$ will also expire from $s.\mathcal{A}'$, while $s.\mathcal{A}_{evict}$ does not change. (2) If $m \notin s.\mathcal{A}$, $m$ will expire from both $s.\mathcal{A}_{evict}$ and $s.\mathcal{A}'$. Thus, $s.\mathcal{A} \cup s.\mathcal{A}_{evict} = s.\mathcal{A}'$ still holds. Therefore, when $|s.\mathcal{A}| < k$ occurs, $s.\mathcal{A}_{evict}$ must be empty because there is no $k$ messages in $s.\mathcal{A}$ that can dominate any message in $s.\mathcal{A}_{evict}$. Thus, $|s.\mathcal{A}'| = |s.\mathcal{A}| < k$ holds. Contrarily, when $|s.\mathcal{A}'| < k$ occurs, it is immediate that $|s.\mathcal{A}| < k$ because $s.\mathcal{A}$ is always a subset of $s.\mathcal{A}'$. Therefore, the theorem holds. □

Based on the above theorem, we turn to estimate $\mathbb{Z}'(s)$, which is much easier. Now the probability distribution of $X_i$ can be estimated as:

$$\mathsf{prob}(X_i) = \begin{cases} \frac{1}{2} \cdot \mathsf{prob}(s.\theta) & \text{if } X_i = 1, \\ \frac{1}{2} \cdot \mathsf{prob}(s.\theta) & \text{if } X_i = -1, \\ 1 - \mathsf{prob}(s.\theta) & \text{if } X_i = 0. \end{cases} \quad (14)$$

We denote the initial size of $s.\mathcal{A}'$ as $|s.\mathcal{A}'_{init}|$. Now, the estimation of $\mathbb{Z}'(s)$ is equivalent to a well-known random walk problem, namely *Monkey at the cliff with reflecting barriers* [15]. Specifically, we set the starting position $RW_0$ as $|s.\mathcal{A}'_{init}|$ and the destination position as $k - 1$; the i.i.d. variable $X_i$ is defined as Equation 14; and the reflecting barrier is set as $2 \cdot RW_0$. By applying some mathematical reduction based on the property of random walk [15], we get the following result.

$$\begin{aligned} \mathbb{Z}'(s) &= \frac{2 \cdot (|s.\mathcal{A}'_{init}| - k + 1) \cdot |s.\mathcal{A}'_{init}|}{\mathsf{prob}(s.\theta)} \\ &+ \frac{(|s.\mathcal{A}'_{init}| - k + 1) \cdot (|s.\mathcal{A}'_{init}| - k + 2)}{\mathsf{prob}(s.\theta)} \end{aligned} \quad (15)$$

where $|s.\mathcal{A}'_{init}|$ can be estimated as $\mathsf{prob}(s.\theta) \cdot |\mathcal{W}|$. Thus, the top-$k$ re-evaluation cost in Equation 12 can be estimated by replacing $\mathbb{Z}(s)$ with $\mathbb{Z}'(s)$.

Based on Equation 11 and Equation 12, we get our final cost model:

$$\mathcal{C}(s) = \mathcal{C}_{sm}(s) + \mathcal{C}_{re}(s) \quad (16)$$

To minimize Equation 16 where the only variable is $s.\theta$, we employ an incremental estimation algorithm similar to gradient descent [1] to compute the best value of $s.\theta$.

**Remark.** To accommodate our cost-based skyband model with the message dissemination algorithm, we need to replace $\mathsf{kScore}(s)$ in Section 5 with $s.\theta$ such that any message with score not lower than $s.\theta$ will be considered to possibly affect the top-$k$ results of $s$. Moreover, since our dominance definition simply depends on the 2-dimensional score-time space while is irrelevant to the exact score function, our technique can be easily applied to other top-$k$ monitoring problems with different score functions.

# 7. EXPERIMENTS

In this section, we conduct extensive experiments to verify the efficiency and effectiveness of our proposed techniques. All experiments are implemented in C++, and conducted on a PC with 3.4GHz Intel Xeon 2 cores CPU and 32GB memory running Red Hat Linux. Following the typical setting of publish/subscribe systems (e.g., [19, 9]), we assume indexes are fit in main memory to support real-time response.

## 7.1 Experimental Setup

As this is the first work to study top-$k$ spatial-keyword publish/subscribe over sliding window, we extend previous work [9] to sliding window setting. We implement and compare the following algorithms. For message dissemination module:

- **CIQ.** The subscription index proposed in [9] [2].
- **IGPT.** The subscription index proposed in our paper, which combines both **I**ndividual and **G**roup **P**runing **T**echniques.

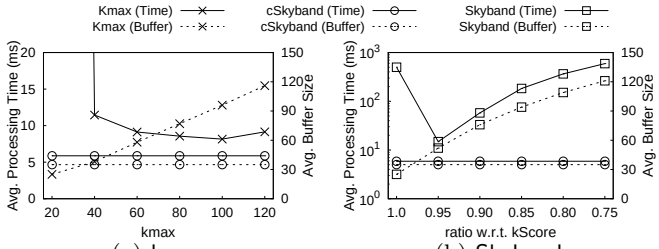
For top-$k$ re-evaluation module:

- **Skyband.** The $k$-skyband algorithm proposed in [23].
- **kmax.** The kmax algorithm proposed in [32].
- **cSkyband.** The cost-based $k$-skyband algorithm proposed in our paper.

Thus, we may have 6 different algorithms by combining different message dissemination algorithm and top-$k$ re-evaluation algorithm. Note that our Skype algorithm is the combination of IGPT and cSkyband. We apply techniques in IR-Tree [12] to index messages.

**Datasets.** Three datasets are deployed for experimental evaluations. *TWEETS* is a real-life dataset collected from

[2]The time decay function and related index in CIQ are removed to adapt to our problem.




(a) kmax (b) Skyband

**Figure 6: Tuning baseline algorithms**

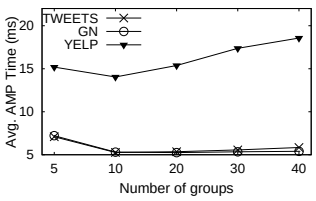



**Figure 7: Vary # groups**

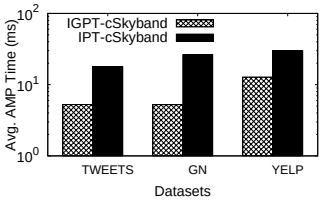
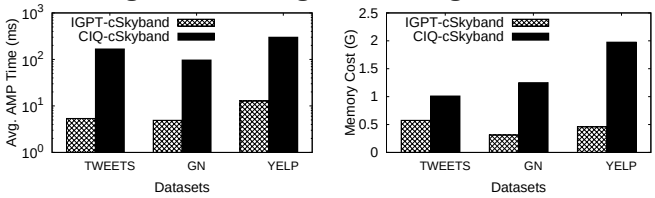



**Figure 8: Compare pruning techniques**




(a) Arriving msg processing (b) Memory cost

**Figure 9: Compare dissemination algorithms**

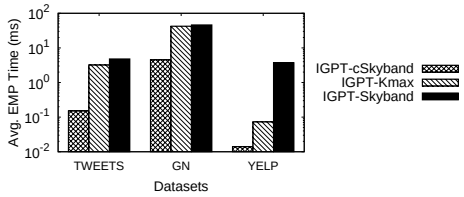



**Figure 10: Compare re-evaluation algorithms**

**Table 1: Datasets Statistics**

| Datasets | TWEETS | GN | YELP |
|---|---|---|---|
| # of msg | 12.7M | 2.2M | 1.6M |
| Vocabulary size | 1.7M | 208K | 85K |
| Avg. # of msg keywords | 9 | 7 | 37 |

Twitter [20], containing millions of tweets with geo-locations from May 2012 to August 2012. *GN* is obtained from the US Board on Geographic Names[3] in which each message is associated with a geo-location and a short text description. *YELP* is obtained from Yelp[4], which contains user reviews and check-ins for thousands of businesses. The statistics of three datasets are summarized in Table 1.

**Subscription workload.** We generate top-$k$ subscriptions based on the above datasets. For each dataset, $1M$ geo-textual messages are randomly selected. For each selected message, we randomly pick $j$ keywords as subscription keywords with $1 \leq j \leq 5$. The weight of each keyword is computed according to *tf-idf* weighting scheme[5]. The subscription location is the same as message location. For each subscription, the preference parameter $\alpha$ is randomly selected between 0 and 1, while the default value of $k$, i.e., the number of top-$k$ results, is set to 20.

**Message workload.** Our simulation starts when the sliding window with default size of $1M$ is full and continuously runs for $100,000$ arriving messages over the sliding window.

We report the average processing time, including average arriving message processing time (i.e., ***AMP***) and average expired message processing time (i.e., ***EMP***), as well as the index size. By default, the number of $\alpha$-partition groups is set to 10. The maximum number of subscriptions that can be stored in each cell is set to 1000.

## 7.2 Experimental Tuning

**Tuning kmax and Skyband.** In the first set of experiments, we tune the performance of both kmax and Skyband techniques in Figure 6 on *TWEETS* dataset, where IGPT algorithm is employed for message dissemination. For better understanding, we evaluate average processing time (denoted as solid line) and average buffer size (denoted as dotted line) in the same figure. We also show the results of our IGPT-cSkyband algorithm under default settings which remains unchanged. For kmax algorithm (Figure 6(a)), we vary kmax from 20 to 120. It is noticed that a small kmax leads to high top-$k$ re-evaluation cost while a large kmax results in high message dissemination cost and buffer maintenance cost. We set 60 as the default kmax value since it strikes a good trade-off between performance and buffer size (i.e., memory cost). For Skyband algorithm, we vary the threshold score $s.\theta$ from $1.0\times$kScore to $0.75\times$kScore where the smaller value leads to larger buffer size. It is noticed that when the ratio is 1.0 which is the original setting in [23], the performance of Skyband is poor due to the frequent re-evaluations. For comparison fairness, $s.\theta$ is set to its sweet point $0.95\times$kScore for Skyband in the following experiments. It is worth mentioning that our cSkyband always outperforms Skyband under all settings because cSkyband can tune a best threshold for each individual subscription based on the cost model while there is no sensible way to tune Skyband for millions of subscriptions. The similar trends are also observed on other datasets.

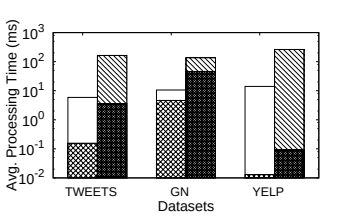
**Vary the number of groups in $\alpha$-partition.** Figure 7 reports the $AMP$ time of Skype algorithm against three datasets where the number of groups varying from 5 to 40. It is shown that we can achieve a good trade-off between the group filtering effectiveness and group checking costs when the number of groups is set to 10, which is used as default value in the following experiments.

**Effect of pruning techniques.** In this experiment, we compare the $AMP$ time of different pruning techniques employed in our message dissemination module. Specifically, we compare IGPT with IPT, which only employs individual pruning technique in Figure 8. We observe that IGPT algorithm can achieve at least three times improvement compared with IPT over all the datasets, which verifies the efficiency of our group pruning techniques. This is mainly because the group pruning technique can skip the whole group without the need to check individual subscription, and can terminate early within a group. In the following experiments, we always use IGPT as our dissemination algorithm.

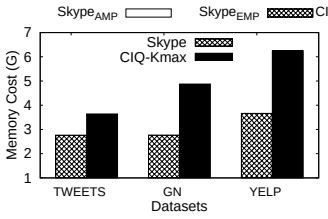
## 7.3 Performance Evaluation

**Compare message dissemination algorithms.** In this experiment, we compare the performance of different dissemination algorithms. Specifically, we compare CIQ and

[3]http://geonames.usgs.gov
[4]http://www.yelp.com/
[5]https://en.wikipedia.org/wiki/Tfidf

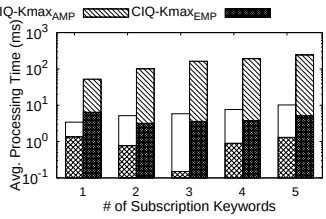



(a) Avg. processing time

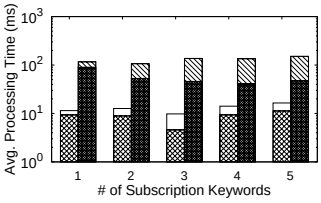



(b) Total Memory cost

**Figure 11: Evaluation over all datasets**

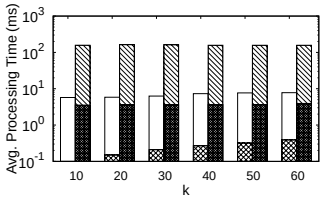



(a) *TWEETS*

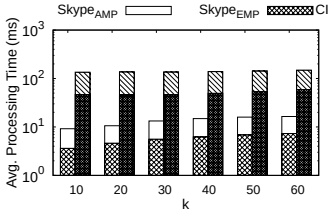



(b) *GN*

**Figure 12: Effect of number of subscription keywords**

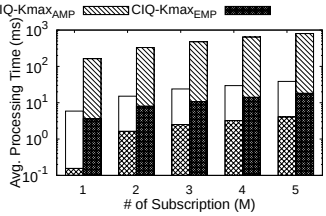



(a) *TWEETS*

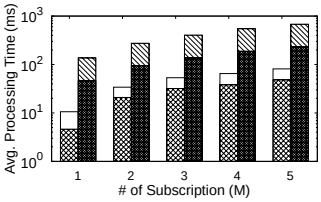



(b) *GN*

**Figure 13: Effect of number of top-$k$ results**




(a) *TWEETS*

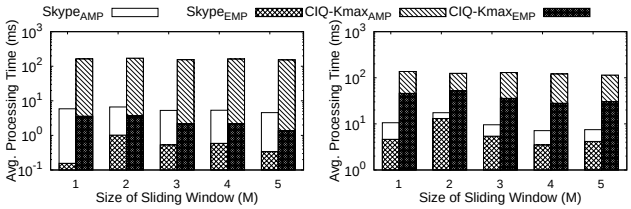



(b) *GN*

**Figure 14: Effect of number of subscriptions**

**Table 2: Buffer size of diff. re-evaluation algorithms**

| Algorithm | *TWEETS* | *GN* | *YELP* |
|---|---|---|---|
| IGPT-cSkyband | **35** | **33** | **28** |
| IGPT-Kmax | 58 | 58 | 59 |
| IGPT-Skyband | 52 | 58 | 56 |




(a) *TWEETS*




(b) *GN*

**Figure 15: Effect of sliding window size**

IGPT with cSkyband being top-$k$ re-evaluation algorithm. As shown in Figure 9(a), our algorithm can achieve about 10 times faster than CIQ algorithm, due to the benefit of individual pruning technique and group pruning technique. On the other hand, as shown in Figure 9(b), even if we need to maintain some additional statistics, the memory cost of our subscription index is much smaller than that of CIQ, since our algorithm only indexes each subscription into single cell, rather than multiple cells.

**Compare top-$k$ re-evaluation algorithms.** In this experiment, we compare the performance of different top-$k$ re-evaluation strategies combined with our IGPT algorithm. Specifically, we compare kmax algorithm [32], $k$-skyband algorithm [23] and our cost-based $k$-skyband algorithm, which are denoted as IGPT-Kmax, IGPT-Skyband, IGPT-cSkyband respectively. The average *EMP* time is reported in Figure 10. We observe that our cSkyband algorithm can achieve about 4-20 times improvement compared to the second best algorithm. This is mainly due to the adaptiveness of our cost model which can tune a best threshold for each subscription. Table 2 demonstrates the average buffer size of each algorithm. Our algorithm maintains much fewer number of messages than other competitors due to the advantage of our cost model.

In the following experiments, we only compare our Skype algorithm (i.e., IGPT-cSkyband) with CIQ-Kmax, which performs best among all the baselines.

**Evaluate over various datasets.** In this experiments, we evaluate the average processing time and memory cost against all datasets. To facilitate detailed comparisons, we decompose average processing time into average *AMP* time and average *EMP* time respectively. It is noticed from Figure 11(a) that for different dataset, the relative proportions of *AMP* and *EMP* time are quite different. However, our algorithm can always achieve about an order of magnitude improvement than the baseline algorithm over all the datasets. The overall memory consumptions (including subscription index, result buffer and message index) are recorded in Figure 11(b). Our algorithm consumes lower memory compared to CIQ-Kmax, since we only index each subscription into one cell, and the buffer size of our cost-based $k$-skyband is smaller than kmax. In the following, we conduct detailed experiments under diverse parameter settings against only *TWEETS* and *GN* datasets due to space constraint.

**Effect of number of subscription keywords.** We assess the effect of number of subscription keywords in Figure 12. We notice that the *AMP* time increases as we vary the number of keywords from 1 to 5. This is obvious since more candidates will be encountered during traversing posting lists when the number of keywords is large. As to the *EMP* time, we observe that the selectivity is low and fewer messages are *relevant* at initial, thus leading to high cost. With increasing number of keywords, the selectivity increases, thus reducing the number of re-evaluations accordingly. Finally, when the number of keyword reaches 4 or 5, a message is less likely to have a high score w.r.t. a subscription due to the smaller weight assigned to each subscription keyword on average, resulting in the increase of *EMP* time. The overall processing time increases decently for a large number of keywords.

**Effect of number of top-$k$ results.** In this set of experiments, we analyse the effect of number of top-$k$ results, i.e., $k$, in Figure 13. For *AMP* time, as we increment $k$ from 10 to 60, the average kScore of subscriptions decreases; therefore, an arriving message is more likely to influence more subscriptions, leading to high *AMP* time in our algorithm. Meanwhile, a large $k$ usually results in high *EMP* time, because the subscriptions with low selectivity are more likely to expire and incur top-$k$ re-evaluations. Besides, the $k$-skyband maintenance cost also increases for large $k$. Overall, the average processing time increases slowly w.r.t. $k$.

**Effect of number of subscriptions.** We evaluate the scalability of our system in Figure 14, where we vary the number of subscriptions from $1M$ to $5M$. As shown in the figure, our algorithm scales very well with increasing number of subscriptions, thus making it practical to support real-life applications with fast response.

**Effect of sliding window size.** We turn to evaluate the effect of sliding window size $|\mathcal{W}|$ in this set of experiments. The results are demonstrated in Figure 15, where we vary $|\mathcal{W}|$ from $1M$ to $5M$. It is observed that, when we increase $|\mathcal{W}|$, the *AMP* time decreases, which is due to the fact that a large sliding window usually leads to better top-$k$ results with higher kScore. Thus, a new message will affect less subscriptions, resulting in lower *AMP* cost. Regarding the *EMP* time, it fluctuates around a value due to the competitive results of fewer number of re-evaluations and high query cost against the message index as we increase $|\mathcal{W}|$.

# 8. CONCLUSION

The popularity of streaming geo-textual data offers great opportunity for applications such as information dissemination and location-based campaigns. In this paper, we study a novel problem of continuous top-$k$ spatial-keyword publish/subscribe over sliding window. To maintain top-$k$ results for a large number of subscriptions over a fast stream simultaneously and continuously, we propose a novel indexing structure, which employs both individual pruning technique and group pruning technique, to process a new message instantly on its arrival. Moreover, to handle the re-evaluations incurred by expired messages from the sliding window, we develop a novel cost-based $k$-skyband model with theoretical analysis to judiciously maintain a partial $k$-skyband for each subscription. The experiments demonstrate that our techniques can achieve a high throughput performance over geo-textual stream.

# 9. REFERENCES


[1] M. Avriel. *Nonlinear programming: analysis and methods*. Courier Corporation, 2003.

[2] B. Babcock, S. Babu, M. Datar, R. Motwani, and J. Widom. Models and issues in data stream systems. In *ACM PODS*, pages 1–16, 2002.

[3] R. J. Bayardo, Y. Ma, and R. Srikant. Scaling up all pairs similarity search. In *WWW*, pages 131–140, 2007.

[4] C. Böhm, B. C. Ooi, C. Plant, and Y. Yan. Efficiently processing continuous k-nn queries on data streams. In *ICDE*, pages 156–165, 2007.

[5] A. Z. Broder, D. Carmel, M. Herscovici, A. Soffer, and J. Y. Zien. Efficient query evaluation using a two-level retrieval process. In *CIKM*, pages 426–434, 2003.

[6] C. Buckley and A. F. Lewit. Optimization of inverted vector searches. In *SIGIR*, pages 97–110, 1985.

[7] S. Chaudhuri, V. Ganti, and R. Kaushik. A primitive operator for similarity joins in data cleaning. In *ICDE*, page 5, 2006.

[8] L. Chen, G. Cong, and X. Cao. An efficient query indexing mechanism for filtering geo-textual data. In *SIGMOD*, pages 749–760, 2013.

[9] L. Chen, G. Cong, X. Cao, and K. Tan. Temporal spatial-keyword top-k publish/subscribe. In *ICDE*, pages 255–266, 2015.

[10] L. Chen, G. Cong, C. S. Jensen, and D. Wu. Spatial keyword query processing: An experimental evaluation. *PVLDB*, 6(3):217–228, 2013.

[11] M. Christoforaki, J. He, C. Dimopoulos, A. Markowetz, and T. Suel. Text vs. space: efficient geo-search query processing. In *CIKM*, pages 423–432, 2011.

[12] G. Cong, C. S. Jensen, and D. Wu. Efficient retrieval of the top-k most relevant spatial web objects. *PVLDB*, 2(1):337–348, 2009.

[13] S. Ding and T. Suel. Faster top-k document retrieval using block-max indexes. In *SIGIR*, pages 993–1002, 2011.

[14] I. D. Felipe, V. Hristidis, and N. Rishe. Keyword search on spatial databases. In *ICDE*, pages 656–665, 2008.

[15] W. Feller. *An introduction to probability theory and its applications*, volume 2. John Wiley & Sons, 2008.

[16] L. Guo, D. Zhang, G. Li, K. Tan, and Z. Bao. Location-aware pub/sub system: When continuous moving queries meet dynamic event streams. In *SIGMOD*, pages 843–857, 2015.

[17] T. Guo, X. Cao, and G. Cong. Efficient algorithms for answering the m-closest keywords query. In *SIGMOD*, pages 405–418, 2015.

[18] R. Hariharan, B. Hore, C. Li, and S. Mehrotra. Processing spatial-keyword (SK) queries in geographic information retrieval (GIR) systems. In *SSDBM*, page 16, 2007.

[19] H. Hu, Y. Liu, G. Li, J. Feng, and K. Tan. A location-aware publish/subscribe framework for parameterized spatio-textual subscriptions. In *ICDE*, pages 711–722, 2015.

[20] G. Li, Y. Wang, T. Wang, and J. Feng. Location-aware publish/subscribe. In *SIGKDD*, pages 802–810, 2013.

[21] J. Lu, Y. Lu, and G. Cong. Reverse spatial and textual k nearest neighbor search. In *SIGMOD*, pages 349–360, 2011.

[22] C. D. Manning, P. Raghavan, H. Schütze, et al. *Introduction to information retrieval*, volume 1. Cambridge university press Cambridge, 2008.

[23] K. Mouratidis, S. Bakiras, and D. Papadias. Continuous monitoring of top-k queries over sliding windows. In *SIGMOD*, pages 635–646, 2006.

[24] K. Mouratidis and H. Pang. Efficient evaluation of continuous text search queries. *TKDE*, pages 1469–1482, 2011.

[25] K. Pripuzic, I. P. Zarko, and K. Aberer. Time- and space-efficient sliding window top-k query processing. *ACM Trans. Database Syst.*, 40(1):1, 2015.

[26] J. B. Rocha-Junior, O. Gkorgkas, S. Jonassen, and K. Nørvåg. Efficient processing of top-k spatial keyword queries. In *SSTD*, pages 205–222, 2011.

[27] M. Sadoghi and H. Jacobsen. Be-tree: an index structure to efficiently match boolean expressions over high-dimensional discrete space. In *SIGMOD*, pages 637–648, 2011.

[28] A. Shraer, M. Gurevich, M. Fontoura, and V. Josifovski. Top-k publish-subscribe for social annotation of news. *PVLDB*, 6(6):385–396, 2013.

[29] X. Wang, Y. Zhang, W. Zhang, X. Lin, and W. Wang. Ap-tree: Efficiently support continuous spatial-keyword queries over stream. In *ICDE*, pages 1107–1118, 2015.

[30] S. Whang, C. Brower, J. Shanmugasundaram, S. Vassilvitskii, E. Vee, R. Yerneni, and H. Garcia-Molina. Indexing boolean expressions. *PVLDB*, 2(1):37–48, 2009.

[31] C. Xiao, W. Wang, X. Lin, J. X. Yu, and G. Wang. Efficient similarity joins for near-duplicate detection. *ACM Trans. Database Syst.*, 36(3):15, 2011.

[32] K. Yi, H. Yu, J. Yang, G. Xia, and Y. Chen. Efficient maintenance of materialized top-k views. In *ICDE*, pages 189–200, 2003.

[33] D. Zhang, C. Chan, and K. Tan. An efficient publish/subscribe index for ecommerce databases. *PVLDB*, 7(8):613–624, 2014.

[34] D. Zhang, C. Chan, and K. Tan. Processing spatial keyword query as a top-k aggregation query. In *SIGIR*, pages 355–364, 2014.

[35] Y. Zhang, X. Lin, Y. Yuan, M. Kitsuregawa, X. Zhou, and J. X. Yu. Duplicate-insensitive order statistics computation over data streams. *TKDE*, 22(4):493–507, 2010.

[36] Y. Zhou, X. Xie, C. Wang, Y. Gong, and W. Ma. Hybrid index structures for location-based web search. In *CIKM*, pages 155–162, 2005.