Efficient Location-Aware Web Search

Joel Mackenzie[1] Farhana M. Choudhury[1] J. Shane Culpepper[1]

1. School of Computer Science and Information Technology
RMIT University, Australia

ABSTRACT

Mobile search is quickly becoming the most common mode of search on the internet. This shift is driving changes in user behaviour, and search engine behaviour. Just over half of all search queries from mobile devices have local intent, making location-aware search an increasingly important problem. In this work, we compare the efficiency and effectiveness of two general types of geographical search queries, range queries and k nearest neighbor queries, for common web search tasks. We test state-of-the-art spatial-textual indexing and search algorithms for both query types on two large datasets. Finally, we present a rank-safe dynamic pruning algorithm that is simple to implement and use with current inverted indexing techniques. Our algorithm is more efficient than the tightly coupled best-in-breed hybrid indexing algorithms that are commonly used for top-k spatial textual queries, and more likely to find relevant documents than techniques derived from range queries.

Categories and Subject Descriptors

H.3.1 [**Information Storage and Retrieval**]: Content Analysis and Indexing—*indexing methods*; H.3.2 [**Information Storage and Retrieval**]: Information Storage—*file organization*; H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval—*query formulation, retrieval models, search process*; I.7.3 [**Document and Text Processing**]: Text Processing—*index generation*

General Terms

Spatial textual indexing; location-aware search; experimentation; measurement; performance

1. INTRODUCTION

Location-aware search is an increasingly important problem in Information Retrieval. For mobile devices, more than 50% of all search queries have local intent [1]. Now more than ever, queries can derive a user's location using GPS, and use this information to improve search effectiveness. Location-aware search is not limited to mobile devices. Another recent study has shown that 84% of all computer users have searched with local intent [2]. Location oriented search is popular regardless of the device of choice, which facilitates the need to efficiently and effectively support location-aware queries.

In order to successfully support these search tasks, Information Retrieval (IR) systems cannot depend on just their textual component, as results may be suboptimal, leading to poor user satisfaction. Therefore, spatial relevance should also be considered. One popular approach to constrain search queries spatially is to use a "range constraint". Given a location p and a maximum travel distance, a *range query* returns all items whose location fall within that distance from p. An example of a spatial range query is when a user wants to find all of the petrol stations within 5 km of their current location. Another popular location-aware query is the k *nearest neighbor (knn) query*, where given a specific location p, the query returns the k items closest to p.

Intuitively, both of these spatial query types can be combined with textual relevance for a keyword search. The users' intent can be expressed as a Boolean predicate of keywords to find the useful documents. A Boolean query consists of a set of keywords, where a *conjunctive Boolean query* retrieves the documents that contain all the query keywords, and a *disjunctive Boolean query* retrieves the documents that contain at least one of the query keywords. Based on the combination of these constraints with a spatial query type, hybrid spatial keyword queries can be categorized in four types: (i) Conjunctive Range queries [10]; (ii) Conjunctive knn queries [16]; (iii) Disjunctive Range queries; and (iv) Disjunctive knn queries.

The results of such queries are generally returned as a ranked list of top-k documents. These queries are supported through hybrid systems that support both spatial and textual filtering of the documents to retrieve the top-k items. Different scoring functions can be employed to achieve the final result. One example scoring function is a weighted linear combination of a term-based measure (TF-IDF, Language model, BM25, etc.) and a geographic score – usually based on the Euclidean distance between the query location and the location of the document.

Based on such combined scoring functions, another type of query called a *Top-k Nearest Neighbor* query has been explored in the literature [12, 20, 26, 30]. A top-k knn query returns the k documents with the highest combined spatial and textual relevance score. This is in contrast to previous knn techniques which simply returned the k nearest neighbours containing the query terms. A more IR-centric approach is to rely heavily on the textual scoring component, but use the spatial distance between objects, or use spatially reordered documents to achieve early-termination in the scoring function [9, 10, 32]. This leads us to two research questions.

ADCS '15, December 08 - 09, 2015, Parramatta, NSW, Australia

ACM 978-1-4503-4040-3/15/12 ...$15.00.
DOI: http://dx.doi.org/10.1145/2838931/2838933.

RQ 1:. *Which indexing approaches provide the best efficiency trade-offs for bag-of-words, top-k knn search in large document collections?*

RQ 2:. *What are the efficiency and effectiveness trade-offs for the two most common spatially constrained query types when applied to bag-of-words, location-aware search?*

Contributions. In this work, we consider two alternative approaches to efficiently process bag-of-words queries with local intent. First, we introduce a variation of the *Weak AND* (WAND) [7] dynamic pruning algorithm which incorporates upper bounds estimates on the distance between documents and queries in order to support rank-safe dynamic pruning, and efficiently solve the *Top-k Nearest Neighbor Query* problem. We refer to this algorithm as GEOWAND. Secondly, we re-examine the recent work of Christoforaki et al. [10] which efficiently solves the *Boolean Conjunctive Top-k Range Query* problem. Specifically, we explore the subtle differences between bag-of-words (disjunctive) top-k nearest neighbor queries, and top-k range queries, with respect to both efficiency and effectiveness.

2. BACKGROUND

Related Work. Geographical Information Retrieval (GIR) refers to the problem of efficiently retrieving relevant documents with respect to both textual and spatial relevancy to a query [24]. A significant body of prior work exists in GIR, covering everything from detecting geographical entities in documents and queries using Natural Language Processing, to efficient indexing of documents based on spatial intent. In this work we focus only on spatial-textual keyword indexing and search algorithms. In order to efficiently support hybrid spatial-textual queries, existing work uses both traditional spatial and text indexing techniques. We first give a brief overview of the separate spatial and text indexing methods that are incorporated in several spatial-textual search approaches.

Spatial indexes. The spatial component of spatial-textual objects are indexed using one of these three classes of spatial indexes, namely, space partitioning indexes, R-tree indexes, and space filling curve indexes.

The kd-tree [6] is a space partitioning tree, where each node is a k-dimensional point. In each iteration, one of the k-dimensions is chosen as a basis of dividing the rest of the points. For each node, an implicit hyperplane is generated that passes through the point and perpendicular to that dimension's axis. Points to the left of the hyperplane are represented by the left subtree of that node and points to the right are represented by the right subtree. The kd-tree is essentially a type of multidimensional binary search tree. A limitation of the kd-tree is that it only indexes point data, which limits the types of spatial queries that can be supported. If other geometric shapes must be stored, then the R-tree is a more suitable choice.

The R-tree and variants [5, 18, 27] are a commonly used spatial index capable of storing any geometric shape. The idea is to allow spatial pruning using a *Minimum Bounding Rectangle* (MBR). Objects that are spatially close-by are grouped together in a single MBR. The R-tree is a hierarchy of nodes where the entries of a non-leaf node are the identifiers of all child nodes and the corresponding MBRs. The data objects are stored in the leaf nodes. The performance of the index depends on how the MBRs are constructed.

Textual indexes. The inverted index [33] is the most common textual index used in large-scale information retrieval systems. Two alternatives to traverse inverted files have been extensively studied: (i) term-at-a-time (TAAT) processing, where the inverted lists of the query terms are traversed sequentially, and the partial scores of the documents are maintained; and (ii) document-at-a-time (DAAT) processing, where the inverted lists are traversed concurrently, and a document is fully scored before considering the next document.

Broder et al. [7] introduced a dynamic pruning DAAT algorithm called *Weak, or Weighted AND* (WAND) which minimizes the number of fully scored documents in single pass traversal. First, the algorithm identifies a candidate document by maintaining a sorted list of cursors for the next unscored document for each query term. The current smallest document ID is used to estimate the maximum score the query terms can achieve by summing up the maximum scores for each term appearing in the candidate document. If this cumulative upper bound score is greater than the lowest scoring item in a min-heap of the current top-k items, then the document is scored. If one or more query terms are rare in the collection, then the traversal acts like a ranked Boolean disjunctive query. If the query terms are common, then the process acts more like a ranked Boolean conjunctive query. Subsequent work has repeatedly shown that variations of the WAND processing scheme are efficient and effective in practice [14, 15, 23, 28].

Spatial keyword indexes. Existing work in spatial keyword indexes combines a spatial index with a text index to answer the hybrid queries. Several hybrid spatial textual indexes have been studied in the literature [10, 12, 20, 30, 31, 32]. Based on the combination scheme, the hybrid indexes can be categorized into two types, namely, a loose combination and a tight combination.

Loosely combined hybrid indexes maintain separate data structures for the spatial and textual components. In this approach, a query can be answered by either filtering with the spatial index first, then verifying with the textual index, or vice-versa. An example of such an index is to use an R-tree to find the spatially relevant objects, then pass the candidates to an inverted file to score textually relevant documents from this candidate set. This is known as *space-first* processing. Conversely, if the textual index is used first for scoring, it is called *text-first* processing. Zhou et al. [32] describe the loose combination of a spatial index with a textual index in three different ways: (i) separate indexes, (ii) an inverted file on top of an R*-tree for text-first processing, and (iii) an R*-tree on top of an inverted file for space-first processing. According to Zhou et al. [32], the text-first implementation performed slightly better than the space-first.

These indexes can answer all four types of hybrid queries. The main drawback of the approach is that a number of non-relevant objects can be retrieved in the filtering stage. If the spatial index is used first, the retrieved objects are spatially relevant to the query, but may not be textually relevant. The same is true if the text index is used first. However, the advantage of two separate indexes is the ease of maintenance and updates.

Christoforaki et al. [10] propose an alternative hybrid index to answer *conjunctive range queries*, where the resulting documents are ranked using a combined ranking function. They show that a spatial structure is not necessary for text-first processing – A simple array can be used to look up the geo-location of a document, and the document can either be added to the results list, or discarded if the textual component is not relevant. Christoforaki et al. go on to propose two approaches based on space filling curves, SFC-Quad and SFC-SKIP which further improve the runtime performance. The

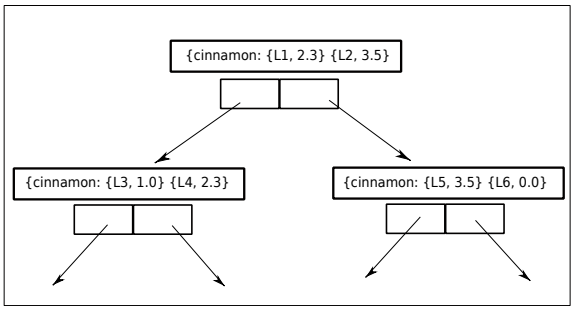

Figure 1: An example IR-Tree and pseudo-nodes for the term '`cinnamon`'. Note that all terms that appear in the tree below the current node will appear in the pseudo-node, a single word is shown here for simplicity. The score 0.0 in the node L6 shows that the term does not appear in any documents in the sub-tree of node L6.

Algorithm 1: Greedy IR-Tree traversal

```
1.1  function GetTopk(q, IR, k)
1.2  Ans ← ∅
1.3  Queue ← INSERT(root(IR), 0)
1.4  while Queue ≠ ∅ do
1.5      Node ← NEXT_BEST_ELEMENT(Queue)
1.6      if Node is an Internal Node then
1.7          for Child ∈ Node do
1.8              Queue ← INSERT(Child, SCORE(Child, q))
1.9      else if Node is a Leaf Node then
1.10         for Object ∈ Node do
1.11             Queue ← INSERT(Object, SCORE(Object, q))
1.12     else
1.13         Ans ← INSERT(Object, SCORE(Object, q))
1.14         if | Ans | = k then
1.15             return Ans
```

SFC-Quad method uses a QuadTree to improve skipping, and the SFC-SKIP method embeds an MBR into block skipping to enable the postings list traversal to skip blocks of documents that do not overlap with the query region.

Tight combination. In tightly combined hybrid indexes, both the spatial and textual pruning techniques are applied simultaneously during query processing. Such indexes can answer all four hybrid queries, including top-k knn queries. The most commonly used tightly combined hybrid index is the IR-Tree [12]. The IR-Tree is essentially an R-tree [18], where each node is augmented with an inverted file for the documents stored in the corresponding sub-tree. An inverted list of a term t is a sequence of the form $\langle d, \gamma \rangle$, where d is the document-id, and γ is the weighted score of t in d, computed by a text relevance metric. As the leaf nodes of an R-tree contain the actual data objects, the inverted file of each leaf node indexes the data of the actual document objects. In contrast, each non-leaf node is associated with a pseudo-document, that is, the union of all terms in the documents of its child nodes. Here, the weight of a term t in a pseudo-document is the maximum score for term t in all of the documents contained in the subtree. Figure 1 shows an example inverted list for the term '`cinnamon`' in an IR-Tree.

This data structure reduces the size of the search space quickly by simultaneous pruning both spatially and textually non-relevant branches. The general search algorithm proposed with the IR-Tree and similar structures is the greedy search method, in which the algorithm expands the most promising nodes first. A sketch of the greedy search is shown in Algorithm 1.

Cong et al. also proposed several variants of the IR-Tree, including the DIR-Tree, CIR-Tree, and CDIR-Tree, aimed at optimizing the query performance. During the insertion process, a DIR-tree minimizes the areas of the MBRs, and maximizes the text similarities between the objects of each node. In the other two variants, clustering techniques are used to achieve a better text relevance estimation in the upper level nodes. These techniques result in an improved pruning of the search-space, at the cost of a much longer construction time.

Independently, Li et al. [20] proposed a similar data structure called IR-Tree (referred to as the Li-IR-Tree in this work). The key difference between the Li-IR-Tree and the indexes proposed by Cong et al. [12] is that the posting lists of the non-leaf nodes are stored in a concatenated form in the Li-IR-Tree, whereas they are stored separately in the IR-Tree. It is important to note that all of the IR-Tree variants were initially designed for on-disk use. For the purpose of our experiments, we implement a standard IR-Tree in memory, and store the pseudo-document nodes separately from the postings lists. This approach is derived directly from the hybrid indexing implementations of Chen et al. [8].

3. APPROACH

In this section we describe two different algorithmic approaches to efficiently score location-aware queries. The first set of algorithms are a heuristic two-stage approach where the spatial or textual top-k act as a first-stage filter, and then the initial set of candidates is refined to produce the final ranking. Our second algorithm builds on a safe-to-k, two-component scoring estimation inspired by the two-score language model variants of WAND as originally described by Petri et al. [23].

Textual and Spatial Relevance. For a geographic query q, consisting of a location ℓ_q and keywords $t \in q$, the combined score for candidate document c consisting of a location ℓ_d and a textual document d, is computed using the following ranking function:

$$s(q, c) = \alpha \cdot \beta(\ell_q, \ell_d) + (1 - \alpha) \cdot \sum_{t \in q} \gamma_t$$

where $\beta(\ell_q, \ell_d)$ is the spatial relevance component, γ_t is the textual relevance component for t, and $\alpha \in (0, 1)$ is a parameter that can be used to weight the importance of the spatial and textual components.

This is the most common spatial-textual scoring scheme used in current work [12, 20, 30, 31]. Values of α can influence the efficiency and the effectiveness for a query, as more influence to either component may improve spatial or textual pruning. However, this clearly depends on the type of indexing approach used. Both Li et al. [20] and Zhang et al. [30] found that α did not have any noticeable effects on efficiency, whereas Cong et al. [12] found large differences in run-time with respect to the value of α, and suggest $\alpha = 0.3$ as a reasonable default. Many formulations exist for β and γ, and these are important system design choices. In this work, the spatial metric used is specified as:

$$\beta(\ell_q, \ell_d) = (1 - \frac{\mathit{Euclidean}(\ell_q, \ell_d)}{d_{\max}}) \cdot \sum_{t \in q} U(\gamma_t)$$

where $d_{\max}$ is the maximum distance between any two unique points in the geographical space, and $U(\gamma_t)$ is the textual upper bound score for the query term $t \in q$. The textual relevance metric is computed using BM25 in this work. More information on the exact BM25 formulation used in this work is discussed in Section 4.

Cascaded Filtering. Our baseline approach is a heuristic filter based on knn. Our approach differs from previous loosely coupled scoring methods [9, 10, 32] by guaranteeing that the initial k' documents derived from the spatial index are the k' nearest document neighbors for each query. Previous work used a fixed sized MBR to identify all documents within a pre-defined range, and considered only Boolean conjunctive queries between terms. Our approach is more pragmatic, and aligned with current cascaded web-based ranking approaches which employ a filter to identify a subset of candidate documents that can then be rearranged into a final ordering based on textual relevance using LTR or higher order dependency models [3, 4, 21, 22]. In this approach, a spatial tree is used to find the k' nearest neighbours to the query location ℓ_q, along with the distance of each candidate from the query point, which yields a candidate set C, where $|C| = k'$. Next, the candidate set C is passed to a modified DAAT query processor which runs as follows: For each candidate $c \in C$, the combined spatial and textual score for query q is computed as $s(q, c)$. This process can be seen as a reordering of C. Finally, the top-k items can be simply taken from the reordering and returned. There are two versions of this approach which we refer to as W-R* and W-kd. W-R* uses a bulk loaded R*-tree for knn queries, whereas W-kd uses a kd-tree for the knn filtering.

We also compare and contrast this approach with the more common filter-based approach which uses a range query to define the subset of documents that must be scored in the final text ranking stage. While not explicitly explored in previous work [9, 10, 32], it is relatively easy to support disjunctive, top-k, bag-of-words search queries using MBR constraints. Firstly, an R*-tree is used to issue a range query, in which all documents within the given range are returned as a set of candidates, C. Next, C is passed to a DAAT processor. This processor then evaluates all candidate documents and returns the top-k documents based on the textual score, γ. For example, given an MBR of 10 km covering a query centroid, it is assumed that a user is willing to travel anywhere within this MBR, so the distance to the centroid is not relevant. Three different MBR sizes were tested, namely MBR-1, MBR-10 and MBR-100, which uses MBRs sized at 1 km, 10 km and 100 km respectively, each centered over the query location.

GEOWAND. A major limitation of the filtering techniques is that the approaches rely on a heuristic method to filter candidate documents before a final ranking is computed. Consequentially, these approaches are not guaranteed to return a rank-safe top-k ranking. However, a carefully modified WAND traversal can guarantee score safety. The GEOWAND algorithm proposed here provides this desirable property.

Algorithm 2 shows the key steps involved in upper bounding the weighted spatial scoring estimate as documents are considered for final scoring, and a second score refinement stage once a candidate document is selected. Given a query q, with terms $t \in q$ and a location ℓ_q, the processing is as follows. In the initial candidate selection phase two contributions must be tracked, the *spatial_limit* and the *textual_limit*. The upper bound for the *spatial_limit* occurs when $\ell_q = \ell_d$. Each term *pivot* in document c_{pivot} has a global maximum textual score added to *textual_limit*. To keep the two scores normalized with respect to α, this upper bound is also added to *spatial_limit* in Line 2.4. In each iteration of the loop, the combined and normalized upper-bound textual and spatial scores are checked against the score of the lowest scoring document in a top-k min-heap, tracked with θ. If the *potential* total score for the current document exceeds θ, then the candidate document must be scored.

Algorithm 2: WAND processing with geospatial weights, replacing steps 10–21 in Algorithm 1 of Petri et al. [23].

```
2.1  text_limit ← 0; spatial_limit ← 0; pivot ← 0
2.2  while pivot < |q| − 1 do
2.3      text_limit ← text_limit + (1 − α) · U[pivot]
2.4      spatial_limit ← spatial_limit + α · U[pivot]
2.5      if text_limit + spatial_limit > θ then
2.6          s ← text_limit + spatial_limit
2.7          break, and continue from Step 2.9.
2.8      pivot ← pivot + 1
2.9  if c_0 = c_pivot then
2.10     t ← 0
2.11     s ← s − spatial_limit + α · β(ℓ_q, ℓ_pivot)
2.12     while t < |q| − 1 and c_0 = c_pivot do
2.13         s ← s − (1 − α) · U[t] + (1 − α) · γ_t
2.14         if s < θ then
2.15             break, and reset all pointers to c_pivot + 1.
2.16         ▷ Update heap and θ . . . remaining computation similar to the
             approach described in Algorithm 1 of of Petri et al. [23].
```

Otherwise, the loop continues until the postings lists are exhausted, or when no more documents can score above θ.

When a candidate document is selected for scoring, the true score of the candidate document can be incrementally refined[1]. First, the true spatial score contribution is updated for c_{pivot}. In the simplest implementation of β, $d_{\max}$ is set as the maximum Euclidean distance between any two documents in the collection (GEOWAND-global). However, early termination in the scoring phase can be improved by finding the distance from the furthermost point in the geospatial dataset to the current query location ℓ_q, and using this distance as $d_{\max}$ (GEOWAND-local). The smaller $d_{\max}$ is, the quicker β will converge to 0.

Next, the textual contribution for each term in c_{pivot} is refined. The upper bound $U[t]$ for each term is iteratively substituted for the true textual score γ_t of the term in Lines $2.12 - 2.15$. If the refined score of the candidate document is less than θ at any stage of this iteration, the document cannot make it into the heap, and execution is returned to the candidate generation loop.

The primary difference between WAND and GEOWAND is that GEOWAND must also compute the spatial relevance dynamically. The scoring operation that occurs for each candidate document is also slightly more complex. WAND must only score the textual relevancy, whereas GEOWAND must compute the normalized sum of both the textual and spatial scores before determining if the candidate document can be added to the top-k heap. As α approaches 0, GEOWAND reduces to WAND. To facilitate spatial scoring, a simple vector of coordinates indexed by document ID can be used to store location data. Then, the coordinates are used to compute the normalized distance score for each candidate document that must be scored.

4. EXPERIMENTS

Experimental Setup. All experiments were executed on a 24-core Intel Xeon E5-2630 with 256 GB of RAM hosting RedHat RHEL-v6.3. Postings lists were generated using Indri, stopping, and Krovetz stemming. Each posting list was then extracted, compressed, and stored in blocks of 128 entries using the FastPFOR library [19] to support skipping. For ranking, a WAND-based variant of BM25 was used with with $k_1 = 0.9$ and $b = 0.4$. [2] All algo-

[1]The observation that text scoring can be incrementally refined was originally made by Gog and Petri [17]. Here we extend this idea to improve pruning for non-textual features.

[2]The values for b and k_1 are different than the defaults reported

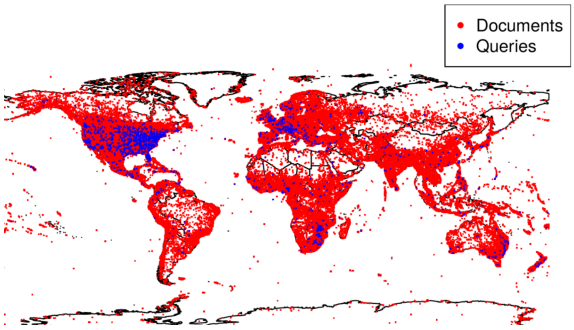

Figure 2: The graphical representation of the document set $\mathcal{D}$ and query set $\mathcal{Q}$ in the CW09BGEO dataset. Note the clustering of both documents and queries around major cities and populous areas.

rithms were implemented in C++ and compiled with `-O3` flags, and all experiments were ran in memory. For all experiments reporting GEOWAND timings, the default algorithm used is GEOWAND-local (G-W).

Search accuracy is measured using Maximized Effectiveness Difference (MED), and scored with RBP and a persistence $p = 0.95$ ($\text{MED}_{\text{RBP}0.95}$) [11, 29]. This approach allows us to quantify the effectiveness differences between runs without requiring relevance judgments.

Dataset. Experiments are conducted using a subset of the TREC[3] ClueWebB collection, known as CW09BGEO. Since this collection is not geotagged, the geotagging process is outlined here. Firstly, a Redis[4] database was created which stored tuples of the form `(location_name, latitude_longitude)`. These tuples were taken from GeoNames[5], and ordered such that the most populous locations would take preference over other locations with the same name. Next, location names in each document were extracted from the Freebase annotations of the ClueWeb Corpora[6], and were used to retrieve geographic coordinates that are associated with the location name from the Redis DB. Finally, the coordinates were associated with the document ID. The query set used is the 2009 Million Query Track, and was geotagged in the same way as the document collection. A single location is used for each document. In queries that have multiple locations specified, the query has been replicated as a one-location query for each specified location. For example, consider the query "`beaumont port arthur airport`". This query would be broken into two separate queries, with the location of `beaumont` supplied for one, and the location of `port arthur` supplied for the other. This resulted in 24 million unique, geotagged documents, and 5,315 unique geo-queries (Figure 2).

An issue with the natural language processing method of geotagging documents and queries is the granularity of location data. Many of the documents have the same location (documents containing `New York` for instance). This is clearly not representative of the granularity we would expect for stores or locations in a large city, and can unfairly skew the results for queries matching a large cluster of documents with identical locations. To make the geotags more realistic, a random ϵ-value is added to the latitude or longitude of any duplicate geotag, with $\epsilon \leq 0.0000000001$. Before "scrambling", there were 150,000 unique locations across the document set, which was expanded to 23,413,629 unique locations. Queries were left unscrambled, which resulted in 2,000 unique query locations across the 5,315 queries.

A second dataset was created for use in preliminary testing. This smaller dataset is a subset of the CW09BGEO dataset, and is simply the first 1 million documents from the CW09BGEO dataset ordered by spam-score [13]. This dataset is referred to as WIKIGEO since a large number of geo-tagged Wikipedia documents are in the collection, as these documents are often the highest ranked documents by spam score. A summary of the datasets can be found in Table 1.

Dataset	Docs	Unique terms	Unique locations
CW09BGEO	23,781,142	53,523,166	23,413,629
WIKIGEO	1,000,000	2,530,440	999,079

Table 1: Statistical summary of the two collections used in the experiments.

System	Dataset	Fanout
kd-Tree	WIKIGEO	100
R*-Tree	WIKIGEO	15
IR-Tree	WIKIGEO	10
kd-Tree	CW09BGEO	150
R*-Tree	CW09BGEO	5

Table 2: Summary of the best tree fanout for each spatial index – collection combination.

Parameters. Before the final experiments were ran, a parameter sweep was done for each spatial index to determine the best fanout for the datasets. The chosen values are summarized in Table 2.

Time and Space Trade-offs Revisited. Our first set of experiments are designed to address RQ1. Specifically, we are interested in the scalability of the IR-Tree as previous research has shown the approach to have a significant space overhead [8], but to be very efficient for smaller k.

Clearly, one large disadvantage of the IR-Tree (denoted IR in the following graphs) with respect to in-memory processing is the number of cache misses that occur. Since the tree structure is stored separately from the pseudo-nodes, which are in turn also separate from the postings lists, a single iteration of the greedy search may access all three data structures, which may be located far apart in RAM. Another issue with the IR-Tree is the necessity of precomputing the textual scores for each term in each document. This leads to a large memory footprint, and compression of the pseudo-nodes is not possible without quantization. Note that postings lists were stored in a similar format to the pseudo-nodes, which explains the increased size of the postings lists as compared to the compressed WAND postings. One redeeming quality of the IR-Tree is that it is safe-to-k. Therefore, every query is guaranteed to have the top-k documents returned with respect to the ranking function. Therefore, the effectiveness of the IR-Tree is optimal with respect to the ranking function used, and better than any of the filtering methods which are not rank-safe, at the cost of both time and space efficiency.

Figure 3 shows the relative costs for each indexing approach for the WIKIGEO collection. We show the different space costs of the IR-Tree as a function of the tuning parameter `fanout`. Note that

by Robertson et al. [25]. These parameter choices were reported for Atire and Lucene in the 2015 IR-Reproducibility Challenge, see `github.com/lintool/IR-Reproducibility` for further details.

[3]https://www.trec.nist.gov

[4]https://redis.io

[5]http://download.geonames.org/export/dump/

[6]https://lemurproject.org/clueweb09/FACC1

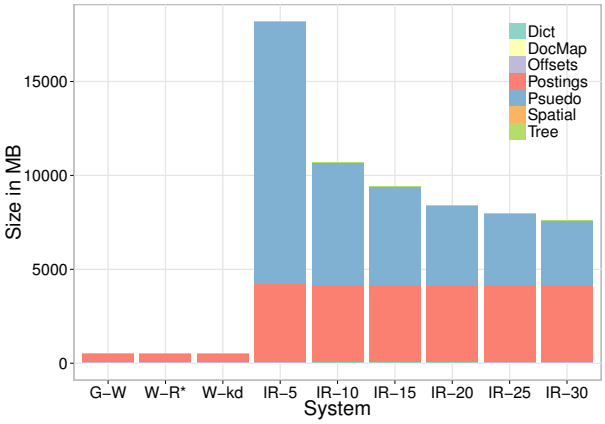

Figure 3: A comparison of the space usage for a GEOWAND inverted index, a loosely coupled WAND inverted index combined with an R-Tree and k-d-Tree, and IR-Tree of varying fanouts. The space difference between the different WAND and GEOWAND indexes is negligible.

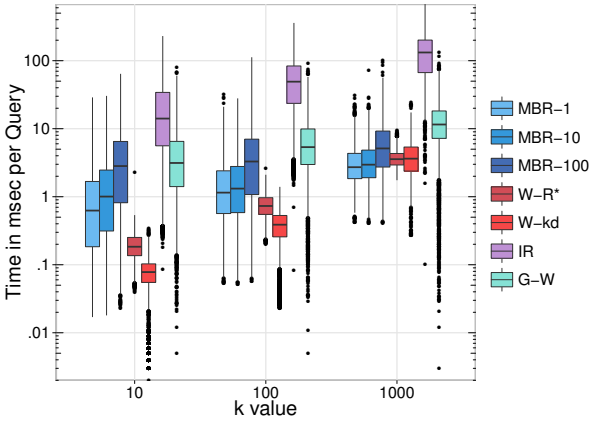

Figure 4: The mean running time per query as a function of k for all query type and index combinations on the WIKIGEO dataset, with $\alpha = 0.5$.

the total size of the original uncompressed document collection is around 9 GB. So the space overhead for the IR-Tree is significant, approaching 2x the size of the original collection. In fact, the space overhead for the IR-Tree is so substantial, we were unable to construct the index for the full CW09BGEO collection. It is clear that more focus should be placed on using well known methods for text indexing and compression in future hybrid indexing experimental comparisons, as the current methods are neither scalable nor friendly to in-memory traversal.

Figure 4 shows the average processing time per query for $k = 10$, 100, and 1,000. Note that the MBR and knn filtering methods MBR-1, MBR-10, MBR-100, W-R*, and W-kd are not rank-safe. So, despite the clear efficiency advantages, some loss of effectiveness does occur. Effectiveness will be explored further in the next section. The IR-Tree (IR) and GEOWAND (G-W) methods are both safe-to-k. We can see that the efficiency gap between GEOWAND and IR-Tree grows as k increases. At $k = 1{,}000$, there is an order of magnitude difference between the running time of the two algorithms.

As we saw in Figure 4, the hybrid IR-Tree index is less efficient than all of other approaches we tested, and the performance gap

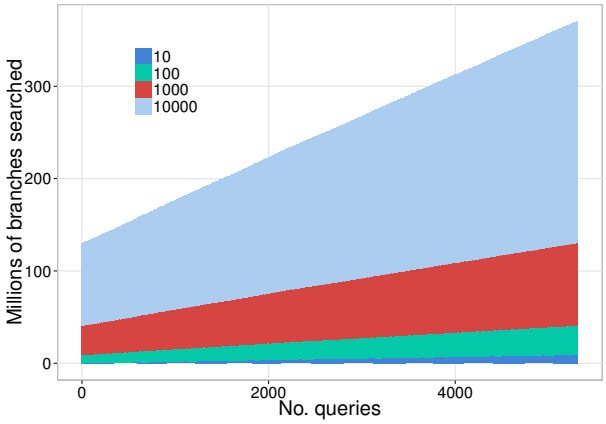

Figure 5: The cumulative number of branches evaluated for the IR-Tree traversal for $\alpha = 0.5$ on the WIKIGEO collection. The greedy search must search many more nodes as k increases, which results in a significant increase in the number of cache misses incurred when walking the tree.

grows with respect to k. To better understand why the algorithm performed so poorly, we computed the number of branches in the IR-Tree that must be evaluated to determine the final top-k list. Figure 5 shows the number of branches as a function of k and queries processed. We can see that the number of branches traversed is significant for values of k greater than 1,000, which are commonly used in multi-stage retrieval scenarios.

So, despite several previous papers reporting that IR-Tree is an attractive time and space trade-off for top-k knn queries, our experiments do not reach the same conclusion. The IR-Tree does not appear to be a competitive search algorithm for in-memory location-aware search in large text databases. It should be re-iterated that prior experimental studies of the IR-Tree reported times that were disk-based and not memory resident, and the textual objects were considerably smaller than the documents in our collection. So, the poor performance we have observed could be an artifact of the problem we are trying to solve, which may not match the original intent. Since we are unable to construct the IR-Tree index on larger datasets, and the performance is not competitive, we do not consider it further in the following sections.

Filter Effectiveness. We now turn our attention to RQ2, and the effectiveness of the filter-based queries. Recall that two major types of queries dominate the literature for location-aware search – range and knn spatial-textual queries. Both of these query types can be processed conjunctively or disjunctively. Here we focus on only bag-of-words disjunctive top-k querying.

As shown in Figure 4, filtered range queries are efficient, but there is inevitably some loss in effectiveness. This is due to the diversity in queries that are observed in large query streams. Some queries cover very densely populated areas, while other query locations may originate in sparsely populated areas. However, many potential documents close to the query origin do not guarantee that these documents also match the keywords in the query. The highest scoring documents for the combined textual-spatial score could be just outside of the MBR range which is fixed at query time. This makes automatically bounding the size of the MBR difficult in practice.

Table 3 shows the Maximum Expected Difference (MED) for two common utility-based effectiveness metrics, ERR and RBP. For both metrics, we see that the effectiveness loss for all of the filtering methods is substantial when compared to a rank-safe knn

α	MBRF 1 km	MBRF 10 km	MBRF 100 km	KNNF $k = 10$	KNNF $k = 100$	KNNF $k = 1000$
			$\text{MED}_{\text{RBP0.95}}$			
0.2	0.7164	0.6978	0.6601	0.9947	0.9936	0.9703
0.5	0.5991	0.5746	0.5246	0.9921	0.9902	0.9551
0.8	0.4412	0.4085	0.3432	0.9880	0.9849	0.9319
			$\text{MED}_{\text{ERR@20}}$			
0.2	0.6844	0.6646	0.6264	0.9655	0.8892	0.8746
0.5	0.5589	0.5332	0.4825	0.9473	0.8354	0.8149
0.8	0.4021	0.3682	0.3042	0.9220	0.7625	0.7406

Table 3: The Maximum Expected Difference (MED) as measured with $\text{MED}_{\text{RBP0.95}}$ and $\text{MED}_{\text{ERR@20}}$. A MED value less than 0.2 is considered negligible. The higher the value, the more likely there is a noticeable effectiveness difference.

approach. When comparing the MBR Range Query methods, we see two important trends. First, the overall effectiveness is never less than 0.2, even when the queries are heavily biased towards the spatial score. This is largely an artifact of query diversity. Some queries have many thousands of potential documents that fall within the query range, while other queries have only a few. Even when there are many potential documents, there is no guarantee that the documents contain the keywords with the highest impact. Even fewer would be conjunctive Boolean matches.

The second important trend is that the Range queries tend to perform better than the loosely coupled knn query approach, even with a conservative k' of $2.5 \times k$. This is because it is even less likely that the documents very close to a query are textually relevant in a dense document region.

These two observations further strengthen our belief that rank-safe scoring methods such as GEOWAND are in fact the simplest and most intuitive approach to ranking documents with more than one type of weighting constraint. The importance of location can be easily determined independent of locational document clustering.

The virtues of GEOWAND. So, GEOWAND has a few clear advantages over any of the filtered range query approaches evaluated. Firstly, GEOWAND allows a user to submit a location-aware query without the need to input any distance parameters. Clearly, this is more intuitive than using the knn or MBR filters, in which a k' or MBR size must be selected by the user or decided a priori by the retrieval system. The obvious caveat here is that the α normalization must be carefully selected in order to provide the correct spatial and textual relevancy levels. Secondly, GEOWAND can be applied to an existing index without much effort. Reindexing is not necessary, and the only addition that must be made to the system is an efficient representation of the document locations. Thirdly, a GEOWAND index can be used to service standard textual queries without a loss of runtime efficiency or effectiveness compared to a plain WAND index. Finally, GEOWAND scales similarly to WAND, so it can be applied to much larger indexes efficiently. Figure 6 shows the relative efficiency of filter-based and GEOWAND approaches on a much larger collection. We can also see the additional cost of GEOWAND over WAND, and the scalability of the various approaches.

Figure 7 shows the number of postings examined for both GEOWAND methods. Clearly GEOWAND benefits from the dynamic refinement of the spatial bound when $d_{\max}$ is query specific. This enhancement results in fewer candidate documents being fully scored since β decreases faster as the distance between the document and the query increases.

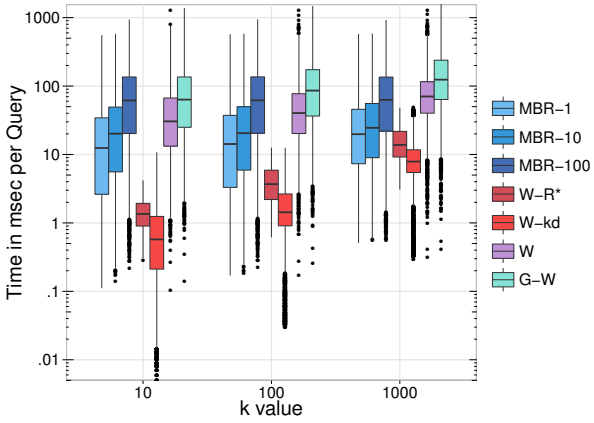

Figure 6: A comparison of the time with respect to k for filtering and GEOWAND systems on the CW09BGEO dataset. Note that $\alpha = 0.5$ in this experiment, and that W represents a standard textual WAND timing run across the textual query only.

Figure 8 shows the effectiveness and efficiency trade-offs for all of the algorithms examined with the CW09BGEO dataset. Clearly, there is a tension between efficiency and effectiveness for the methods tested. The knn filters provide fast but less relevant results. The MBR filter is less efficient, but finds more relevant results. The larger the MBR, the more documents on average must be scored, but with improved overall effectiveness. With the knn filter, as k' increases, the results would become closer to those of GEOWAND, but efficiency would continue to tail off. The IR-Tree would deliver the same results as GEOWAND but less efficiently.

5. CONCLUSION

In this paper, we have explored the efficiency and effectiveness trade-offs for knn and range-based location-aware search queries (RQ2). Our experimental study has shown that while range-based queries are fast, the effectiveness of the queries are dependent on the spatial properties of the queries and the collection. On the other hand, safe-to-k methods derived from top-k knn spatial-textual queries are capable of finding relevant documents regardless of the collection properties.

We have also presented a variation of the WAND processing algorithm called GEOWAND which is easy to implement, efficient, and scalable. This algorithm provides an attractive trade-off for processing location-aware queries, and provides a suitable answer to RQ1. GEOWAND is an order of magnitude faster than a tightly coupled IR-Tree index, and up to 15 to 20 times smaller. Note that one important optimization was not explored in this work – document reordering based on location. Using space filling curves or similar approaches has been shown to further improve efficiency in location-aware search. We believe these efficiency gains are orthogonal to our experiments as all of the approaches tested would benefit in a similar manner from this enhancement, including GEOWAND. We intend to explore these and other enhancements in future work.

Acknowledgment. This work was supported by the ARC *Discovery Projects* Scheme (DP140101587). Shane Culpepper is the recipient of an ARC DECRA Research Fellowship (DE140100275).

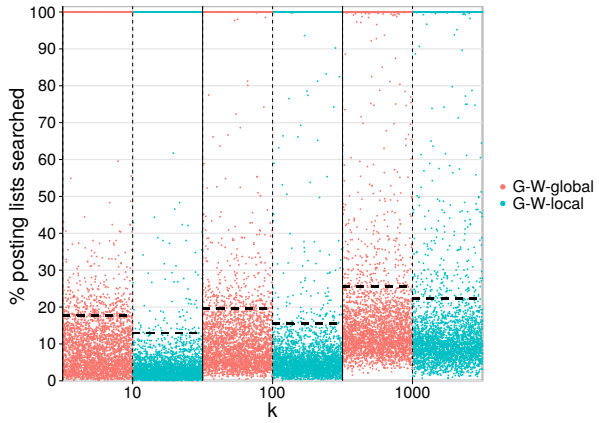

Figure 7: This graph shows the percentage of postings lists that GEOWAND evaluated with respect to an exhaustive run over all queries for $k = 10, 100, 1000$ on the CW09BGEO dataset with $\alpha = 0.5$. G-W-global refers to the version of GEOWAND that normalizes the spatial score using the maximum distance between any two documents in the collection. G-W-local is the improved version, which normalizes the spatial score relative to the query location. The dotted line shows the average percentage of postings lists evaluated for each k.

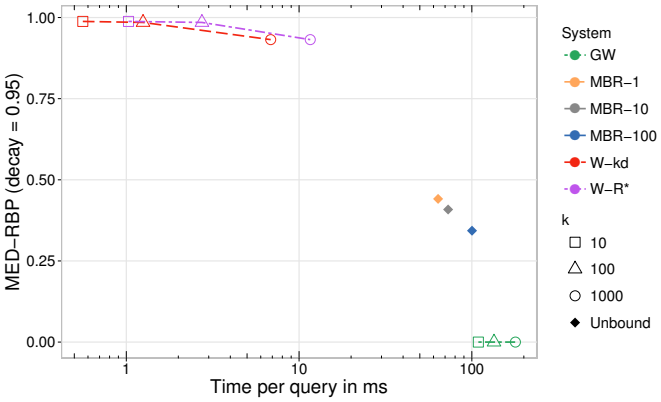

Figure 8: Effectiveness versus Efficiency for the CW09BGEO collection for $\alpha = 0.8$.

References

[1] Microsoft: 53 percent of mobile searches have local intent. `http://searchengineland.com/microsoft-53-percent-of-mobile-searches-have-local-intent-55556`. Accessed: 2015-09-03.

[2] Understanding consumers' local search behavior. `https://www.thinkwithgoogle.com/research-studies/how-advertisers-can-extend-their-relevance-with-search.html`. Accessed: 2015-09-03.

[3] N. Asadi and J. Lin. Document vector representations for feature extraction in multi-stage document ranking. *Information Retrieval*, 16 (6):747–768, 2013.

[4] N. Asadi and J. Lin. Effectiveness/efficiency tradeoffs for candidate generation in multi-stage retrieval architectures. In *Proc. SIGIR*, pages 997–1000, 2013.

[5] N. Beckmann, H.-P. Kriegel, R. Schneider, and B. Seeger. The R*-tree: an efficient and robust access method for points and rectangles. In *Proc. SIGMOD*, pages 322–331, 1990.

[6] J. L. Bentley. Multidimensional binary search trees used for associative searching. *Comm. of the ACM*, 18(9):509–517, 1975.

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

[8] L. Chen, G. Cong, C. S. Jensen, and D. Wu. Spatial keyword query processing: an experimental evaluation. In *Proc. VLDB*, pages 217–228, 2013.

[9] Y.-Y. Chen, T. Suel, and A. Markowetz. Efficient query processing in geographic web search engines. In *Proc. SIGMOD*, pages 277–288, 2006.

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

[11] C. L. A. Clarke, J. S. Culpepper, and A. Moffat. Assessing efficiency-effectiveness tradeoffs in multi-stage retrieval systems without using relevance judgements. `http://arxiv.org/abs/1506.00717`, 2015.

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

[13] G. V. Cormack, M. D. Smucker, and C. L. A. Clarke. Efficient and effective spam filtering and re-ranking for large web datasets. *Information Retrieval*, 14(5):441–465, October 2011.

[14] C. Dimopoulos, S. Nepomnyachiy, and T. Suel. Optimizing top-k document retrieval strategies for block-max indexes. In *Proc. WSDM*, pages 113–122, 2013.

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

[16] Ian De Felipe, Vagelis Hristidis, and Naphtali Rishe. Keyword search on spatial databases. In *Proc. ICDE*, pages 656–665, 2008.

[17] S. Gog and M. Petri. Compact indexes for flexible top-k retrieval. In *Proc. CPM*, pages 207–218, 2015.

[18] A. Guttman. R-trees: a dynamic index structure for spatial searching. In *Proc. SIGMOD*, pages 47–57, 1984.

[19] D. Lemire and L. Boytsov. Decoding billions of integers per second through vectorization. *Soft. Prac. & Exp.*, 41(1):1–29, 2015.

[20] Z. Li, K. C. K. Lee, B. Zheng, W.-C. Lee, D. Lee, and X. Wang. IR-tree: An efficient index for geographic document search. *TKDE*, 23 (4):585–599, April 2011.

[21] C. Macdonald, R. L. T. Santos, and I. Ounis. The whens and hows of learning to rank for web search. *Information Retrieval*, 16(5):584–628, 2013.

[22] C. Macdonald, R. L. T. Santos, I. Ounis, and B. He. About learning models with multiple query-dependent features. *ACM Trans. Information Systems*, 31(3):1–39, 2013.

[23] M. Petri, J. S. Culpepper, and A. Moffat. Exploring the magic of WAND. In *Proc. ADCS*, pages 58–65, 2013.

[24] R. Purves and C. B. Jones. Geographic information retrieval. *SIGSPATIAL Special*, 3(2):2–4, 2011.

[25] S. E. Robertson, S. Walker, S. Jones, M. Hancock-Beaulieu, and M. Gatford. Okapi at TREC-3. In *Proc. TREC-3*, 1994.

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

[27] T. Sellis, N. Roussopoulos, and C. Faloutsos. The R+-tree: A dynamic index for multi-dimensional objects. In *Proc. VLDB*, pages 507–518, 1987.

[28] D. Shan, S. Ding, J. He, H. Yan, and X. Li. Optimized top-k processing with global page scores on block-max indexes. In *Proc. WSDM*, pages 423–432, 2012.

[29] L. Tan and C. L. A. Clarke. A family of rank similarity measures based on maximized effectiveness difference. `http://arxiv.org/abs/1408.3587`, 2014.

[30] D. Zhang, K.-L. Tan, and A. K. H. Tung. Scalable top-k spatial keyword search. In *Proc. EDBT*, pages 359–370, 2013.

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

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

[33] J. Zobel and A. Moffat. Inverted files for text search engines. *ACM Computing Surveys*, 38(2):6–1 – 6–56, 2006.