GeoBloom: Revisiting Lightweight Models for Geographic Information Retrieval

Yi Li
liyi0067@e.ntu.edu.sg
Nanyang Technological University
Singapore

Gao Cong*
gaocong@ntu.edu.sg
Nanyang Technological University
Singapore

ABSTRACT

Geographic Information Retrieval (GIR) systems process text queries with geographic location to identify relevant geographic objects for users. Although recent advancements have leveraged Pre-trained Language Models (PLMs) for their robust semantic comprehension, these models typically depend on extensive *labeled queries* and require considerable computational resources. Deviating from this prevailing trend, we propose GeoBloom, a lightweight framework that surpasses the effectiveness of PLMs with fewer or no labeled queries, with remarkable efficiency in both time and space.

GeoBloom tackles critical challenges such as the lack of labeled queries, low data (labeled) efficiency, and high computational demands. At its core, it employs Bloom filters to encode text at a fine-grained term level and uses intersecting bits to create a robust unsupervised text similarity metric. A specialized Bloom Filter Evaluator is proposed to assess the importance of each intersecting bit, focusing on those associated with ground truth, improving effectiveness with fewer training labels. For enhanced search efficiency, the evaluator exploits the inherent sparsity of Bloom filters, achieving remarkably low time and space complexities. This efficiency is further boosted by a tree-based index that partitions the search space while preserving effectiveness. Extensive experiments show that GeoBloom surpasses state-of-the-art baselines in both unsupervised (up to 15.66% improvement) and supervised settings (up to 10.94% improvement) on real datasets in terms of NDCG@5. Furthermore, GeoBloom operates up to 80x faster and saves up to 74.72% memory and 87.64% disk space over PLM-based alternatives, rendering it highly potent for real-world applications.

PVLDB Reference Format:
Yi Li and Gao Cong. GeoBloom: Revisiting Lightweight Models for Geographic Information Retrieval. PVLDB, 18(5): 1348 - 1361, 2025.
doi:10.14778/3718057.3718064

PVLDB Artifact Availability:
The code and data are available at https://github.com/pkuliyi2015/GeoBloom.

1 INTRODUCTION

Geographic Information Retrieval (GIR) systems are essential in location-based services, powering online maps like Google Maps, geo-tagged webpages, ride-hailing applications such as Uber, and geo-tagged social media platforms like Twitter. These systems provide access to a variety of geographic objects, such as Points of interest (POIs) on the map and geo-tagged texts or reviews. A typical GIR system processes *spatial keyword queries* that combine text-based queries with geographic locations to identify relevant geographic objects. For instance, users searching for a "coffee shop" along with preferred locations, such as their current GPS coordinates, would receive a list of nearby coffee shops.

At the core of GIR systems lies the relevance model, which computes the *relevance score* between user queries and geographic objects. This score integrates text similarity metrics and geographic distances to determine the order in which results are displayed to users. The effectiveness of GIR models can be evaluated and enhanced using *labeled queries*, where each query is labeled with one or more user-selected geographic objects. Tremendous progress has been made on GIR models, including (1) Classical methods [2, 8, 15, 25, 28, 37, 42] that combine unsupervised text similarity metrics (e.g., BM25 [36]) with geographic distances. These methods are based on word matching, ignoring the semantics of user queries (e.g., car and automobile are considered two different words although they have similar semantics). (2) Early learning-based models [44, 47] that learn relevance scores from labeled queries with lightweight neural networks. Some text-based IR methods [17, 18] are also adapted to GIR. Yet, empirical studies [26] show that most of them [17, 18, 47] fail to rival classical methods in effectiveness. (3) Models based on Pre-trained language models (PLMs) [11, 19, 26, 43], which utilize BERT [10] and ERNIE [46] to encode a profound level of semantic knowledge, and are fine-tuned on labeled queries to adapt to GIR tasks. PLM-based models have outperformed previous work in effectiveness by a large margin.

Despite the improvements in effectiveness, existing methods fail to address the longstanding challenges in the GIR domain: (1) *Lack of Labeled Queries*, as labeled queries often include sensitive information such as geographic locations and personal preferences, making them rarely publicly available. Platforms like Google Maps and OpenStreetMap refrain from providing such data due to privacy concerns. Consequently, most learning-based methods rely on limited private data sources (i.e., Baidu Map search logs [19, 44] and Meituan App interactions [17, 18, 26, 43, 47]), with an exception [24] that invites experts to synthesize queries and labels, which are city-specific and difficult to obtain. The reliance on labeled queries often renders these models impractical to deploy when gathering such data is unfeasible or cost-prohibitive. (2) *Low Data (labeled) Efficiency*. Leading machine learning models are notoriously data-hungry [1]. Although GIR systems can accumulate labeled queries over time via user interactions, a small number

*Corresponding author.

Proceedings of the VLDB Endowment, Vol. 18, No. 5 ISSN 2150-8097.
doi:10.14778/3718057.3718064

Table 1: A Qualitative Comparison of Geographic Information Retrieval Methods

Feature	Traditional Methods	Early Learning-based Methods	Pretrained LMs	GeoBloom
Effectiveness in Unsupervised Settings	High	N/A	Medium	High
Overall Effectiveness	Medium	Low	High	High
Data (labeled) Efficiency	N/A	Low	Medium	High
Time & Space Efficiency	High	Low	Low	High

of labeled queries is typically insufficient to improve the model's effectiveness. This insufficiency arises because labeled queries incorporate both general semantics and city-specific terms, such as addresses, landmarks, and trademarks. Classical methods often fail to leverage deep semantics, while learning-based models struggle with city-specific terms rarely included in common pre-training corpora. Consequently, early learning-based models [17, 18, 44, 47] without extensive pre-training fail to rival classical methods, while PLM-based models still require a large number of labeled queries to be as effective as classical methods, as shown in section 5.1.2. (3) *Time and Space Efficiency*. The effectiveness of PLM-based models comes at the cost of substantial increases in both parameter count and computation complexity, leading to higher inference time, GPU costs, and memory and storage needs. In addition, GIR models often rank a large number of geographic objects, presenting a non-trivial challenge to inference scalability at query time.

Our goal is to design a model with robust effectiveness in the absence of labeled queries while being data-efficient, i.e. significantly improving its performance with a few labeled queries. We propose a novel method GeoBloom, which comprises three main components. (1) We first introduce *Bloom filters* [4] as fine-grained and expressive representations for query and geographic object texts. In contrast to dense representations used in previous models (e.g, Word2Vec [29] in [17, 18] and PLM outputs in [11, 19, 26, 43]) that rely on pre-training corpora, Bloom filters directly encode the existence of text terms with binary bits, effectively handling words of low frequency such as addresses, landmarks, and trademarks. Bloom filters are also more space-efficient than classical Bag-of-Words (BOW) representations (as in BM25 [36]) as they convert extensive geographic vocabulary into sparse vectors with fixed length. We further show that the pairwise intersection of the Bloom filters between queries and objects can effectively estimate text similarity without labeled queries. Combined with geographic distances and normalization techniques, it shows better effectiveness than strong unsupervised baselines. (2) The second component is a novel *Bloom Filter Evaluator*, a lightweight neural network that utilizes the query and object Bloom filters to compute relevance scores. Unlike all the recent learning-based models [11, 17–19, 26, 43, 44, 47] that compute scores based on dense representations, our evaluator conducts bit-by-bit evaluations on the intersecting bits of the two input Bloom filters. Our evaluator is able to be responsive to crucial bits associated with ground truth labels, enabling significant effectiveness improvements with fewer labeled queries. We adopt the Efficiently Updatable Neural Network (NNUE) [31] to leverage the sparsity of Bloom filters, ensuring low time and space efficiency. (3) The last component is the *Bloom Filter Tree*, a hierarchical index that aggregates the object Bloom filters by spatial proximity to speed up the search. We adapt our proposed Bloom Filter Evaluator to compute query-node relevance scores, which greatly reduces the comparisons needed while maintaining the same level of effectiveness as a brute-force search. Additionally, it also allows for a beam search algorithm that explicitly utilizes *geographic context* [11] (i.e., objects near the ground truth) to deliver better effectiveness. Table 1 presents a qualitative comparison of various GIR models.

In summary, we make the following contributions:

- The introduction of GeoBloom, a novel method that leverages Bloom filters to represent text terms effectively and uses the intersecting bits for text similarity modeling, which achieves great effectiveness without labeled queries. To the best of our knowledge, this is the first work that successfully adopts Bloom filters to represent queries and geographic objects as the input and the output to a designed relevance evaluator, rather than the bag-of-words representations [36] or dense representations [11, 17–19, 26, 43, 44, 47] used in previous work.
- A new data-efficient Bloom Filter Evaluator to evaluate each intersecting bit within Bloom filters, which prioritizes important bits associated with ground truth labels, enabling significant effectiveness improvements with fewer labeled queries. We also propose optimization techniques tailored for Bloom filters, achieving significantly low time and space complexity.
- A Bloom Filter Tree that further enhances the search efficiency while maintaining the same level of effectiveness as a brute-force search. The tree also allows for an additional feature leveraging geographic context for better effectiveness (see Section 4.4).

Our experiments on real and synthetic datasets demonstrate that GeoBloom outperforms strong unsupervised baselines, achieving improvements of up to 11.92% in Recall@10 and 19.39% in NDCG@1. In supervised settings, GeoBloom's effectiveness improves steadily with more labeled queries, consistently surpassing state-of-the-art PLM-based models across both small (i.e., 2%-10%) and large (i.e., 30%-100%) training queries, with gains of up to 7% in Recall@10 and 11% in NDCG@1. Further experiments show that GeoBloom is up to 80x faster and saves up to 74.72% in runtime memory and 87.64% in disk space, confirming its remarkable time and space efficiency.

2 RELATED WORK

2.1 Geographic Information Retrieval

Geographic Information Retrieval (GIR) systems handle text-based queries with geographic proximity [35] (also known as spatial keyword queries). Early research [6–9, 27] mainly focuses on retrieval efficiency, where the relevance between queries and geographic objects is defined as a linear combination of geographic

distances [38] and unsupervised text similarities measured by TF-IDF [40], BM25 [36], or cosine similarity [37]. Despite their robust effectiveness in unsupervised settings, these methods are constrained by a lack of understanding of semantic similarity.

The integration of deep learning models into GIR marked a significant shift, using dense representations to compute text similarities. Early learning-based models [44, 47] encode texts with lightweight neural networks, yet they lack inherent semantic knowledge and rely on extensive labeled queries. Some text-based IR models [17, 18] are adapted to GIR tasks, which use Word2Vec [29] embeddings to estimate semantic relevance. However, empirical studies [26] demonstrate that these approaches [17, 18, 32] fail to compete with classical methods. Moreover, they face scalability issues on large datasets due to their pairwise query-object evaluations. In contrast, our method leverages Bloom filters to represent texts, handling low-frequency terms without training data requirements. It is further boosted by an index based on Bloom filters, significantly reducing the number of necessary evaluations.

The introduction of Pre-trained Language Models (PLMs) like BERT [10] and ERNIE [19] significantly advances GIR models by providing deeper semantic understanding, while with the cost of extensive parameters. Recent studies [26, 43] integrate geographic distances with PLM-based text similarities. DrW [26] leverages BERT word embeddings for text similarities and balances them with geographic distances using an attention mechanism. LIST [43] fine-tunes BERT-based text similarities by aligning them with human geographic preferences. Additionally, some efforts [11, 19] enhance PLMs via pre-training on city-specific data, where MGeo [11] pre-trains BERT-based encoders on OpenStreetMap multi-modality data, and ERNIE-GeoL [19] pre-trains ERNIE on a large-scale heterogeneous graph containing user mobility data and interaction logs. In contrast, our method operates with fewer parameters and sidesteps the need for extensive pre-training, simplifying deployment while maintaining high effectiveness.

2.2 Bloom filters

Bloom filters [4] are space-efficient probabilistic data structures designed for set membership testing, ideal for managing large datasets with small space in large information retrieval systems [16]. Existing work on Bloom filters focuses on sophisticated data structure, including supporting count/delete operations [13], enhancing space efficiency [34], improving spatial privacy [32], and integrating geographic indexing [41]. Recently, Learned Bloom Filter (LBF) [22] emerges to combine vanilla Bloom filters with machine learning models. For example, SandwichedLBF [30] introduced a sandwiched learned bloom filter model by adding a bloom filter before the learned oracle. PA-LBF [45] proposes a prefix-based learned Bloom filter tailored for geographic data. All these advancements aim at reducing the False Negative Rate (FNR), which is orthogonal to this paper. Diverging from previous research, our work takes a novel approach by integrating Bloom filters into relevance modeling, extending their application beyond conventional boundaries.

2.3 Efficiently Updatable Neural Networks

The Efficiently Updatable Neural Networks (NNUE) [31] is a lightweight yet powerful neural network applied in top-tier computer chess engines, which efficiently processes the sparse chessboard representations on the alpha-beta tree to estimate the game-winning probability. This is highly similar to our scenario where the model evaluates query-node relevance scores across the Bloom Filter Tree. However, our model is specially designed to be data-efficient, diverging from a vanilla NNUE that is trained on massive game plays. To the best of our knowledge, this is the first application of the NNUE in the GIR domain.

3 PRELIMINARIES

We introduce the Bloom filter and give the problem statement.

Bloom filter. A Bloom filter is a space-efficient data structure representing a set $A = \{a_1, a_2, ..., a_n\}$ with n elements to support the membership test. Specifically, it allocates a vector B of m bits, all initialized to 0, and chooses k hash functions $H_1, H_2, ..., H_k$, each with range 1,...,m. For each element $a \in A$, the bits at positions $H_1(a), H_2(a), ..., H_k(a)$ in B are set to 1. Given a query q, we check the bits at positions $H_1(q), H_2(q), ..., H_k(q)$. If any of the bits is 0, then q is definitely not in the set A; otherwise, the element is possibly in the set with a false positive rate.

Relevance. Let o be a geo-textual object and q a spatial keyword query, each with a location and a text description. The relevance is defined as $\text{Relevance}(q, o) = f(\text{DistSim}(q, o), \text{TextSim}(q, o))$, where $\text{DistSim}(q, o)$ measures spatial proximity, $\text{TextSim}(q, o)$ quantifies textual relevance, and f combines them into a single score.

Problem Statement. (*Geographic Information Retrieval*) Consider a set of geographic objects S. Given a user's query q consisting of a textual input and a spatial location (e.g., user's current location), the goal of Geographic Information Retrieval is to identify the top-k geographic objects that are most relevant to the user query.

4 METHODOLOGY

We aim to design a model that tackles the key challenges faced by existing GIR models. We first analyze these challenges in detail:

- *Lack of Labeled Queries*: Labeled queries (i.e., queries with user-selected geographic objects) are often unavailable due to privacy or policy constraints. Early learning-based models [17, 18, 44, 47] learn relevance scores directly from labels and cannot function without them. PLM-based methods, while capable of evaluating text similarity, often perform poorly without labeled data, as they emphasize semantics over lexical details [14]. For instance, such methods might view "Unit 1231" and "Unit 1321" as similar, although they refer to different objects.
- *Low Data (labeled) Efficiency*: A substantial portion of user queries and geographic objects contain location-specific text, such as addresses, landmarks, and trademarks. These texts often lack deep semantic meaning and do not cluster semantically in latent spaces. Consequently, learning-based models struggle to generalize from limited training queries to test queries. Our experiments show that early learning-based models fail to compete with classical methods even with sufficient (i.e., >100k) labeled queries, while PLM-based methods also require a significant amount (i.e., >10k) of labeled queries to compete with classical methods.
- *Time and Space Efficiency*: The superior effectiveness of PLM-based methods stems from their hundreds of millions of parameters. However, this increased computational requirement leads to

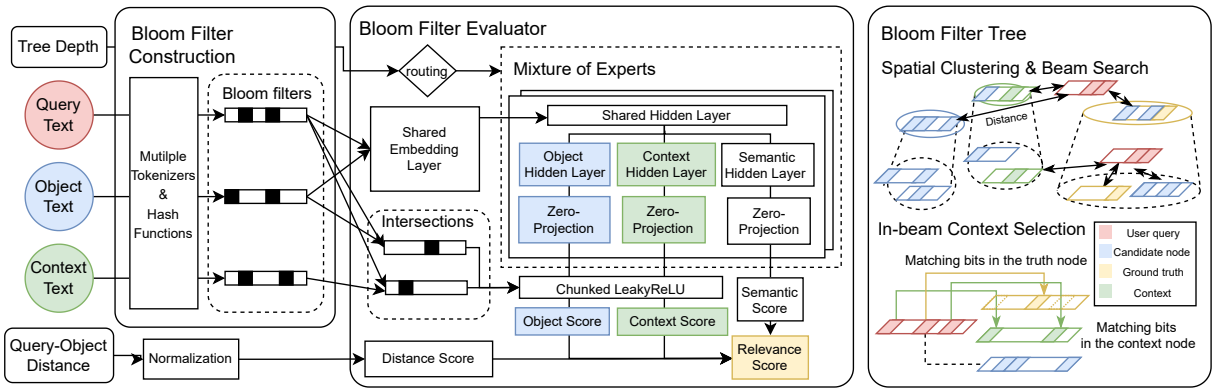


Figure 1: Overview of the GeoBloom Framework with three core components: (1) Bloom filters, constructed by tokenizing and hashing terms, use intersecting bits between filter pairs to form a robust unsupervised text similarity metric; (2) Bloom Filter Evaluator, a lightweight neural network that evaluates the importance of intersecting bits, prioritizing those tied to ground truth in labeled queries; (3) Bloom Filter Tree, a geographic index that partitions the search space and employs beam search to incorporate geographic context.

longer inference time and necessitates expensive GPU hardware, along with substantial memory and storage costs. As the dataset size increases, these challenges are amplified, exacerbating the efficiency issues for answering queries.

We propose GeoBloom to address these challenges. As shown in Figure 1, our method takes the geographic distances, the query text, the candidate object text, and the text from a contextual object (optional) as input. (1) First, we tokenize the texts and encode terms into *Bloom filters*, where each term maps to specific bits in a fixed-length sparse vector. This creates space-efficient, lexically detailed representations. We use intersecting bits in query-object Bloom filter pairs to identify matching terms, providing a robust unsupervised text similarity metric. (2) We develop a *Bloom Filter Evaluator* to assess the importance of intersecting bits. This maintains effectiveness in unsupervised settings while emphasizing key bits associated with ground-truth labels, achieving substantial gains as more labeled queries are available. Inspired by the NNUE [31] chess engine, the evaluator leverages Bloom filter sparsity for low time and space complexity. (3) We further introduce a *Bloom Filter Tree* to reduce comparisons in pairwise Bloom filter evaluations. Constructed like a max-heap, this index preserves evaluator effectiveness comparable to brute-force search. Additionally, it allows for a beam-search algorithm that explicitly utilizes *geographic context* [11] for improved effectiveness in supervised settings.

4.1 Modeling Text Similarity with Bloom filters

We argue that a GIR model needs fine-grained term-matching capabilities to perform well in unsupervised settings, as highlighted by the strong performance of BM25 in our empirical analysis (see Section 5.1.2). While the Bag-of-Words (BOW) representations in BM25 effectively capture key lexical terms (e.g., city-specific keywords) for GIR tasks, integrating BOW into learning-based models is challenging due to the large vocabulary of geographic terms and the resulting computational costs. Methods like Word Hashing [20, 39] can reduce BOW dimensionality and generate dense embeddings for learning-based models, but they still require extensive fine-tuning with labeled queries to achieve effective term-matching.

We introduce Bloom filters [4] to overcome these limitations, offering similar benefits to BOW but with greater space efficiency ideal for learning-based models. Each Bloom filter is constructed by tokenizing the texts within a query (or a geographic object) into short terms using various text tokenizers and encoding these terms into a sparse vector with random hash functions. Unlike existing learning-based GIR models [11, 17–19, 26, 43, 44, 47] that compute relevance scores based on dense representations, we *maintain representations sparse*, directly computing relevance scores with Bloom filter bits. This approach offers several significant advantages:

- *Identifying Matching Terms*: The non-zero bits in each Bloom filter correspond to text terms from the query or geographic objects. *Ideally*, by intersecting Bloom filters from queries and objects, we can uniquely identify and represent matching terms in the underlying texts without the need for training.
- *Integration of Various Tokenizers*: Bloom filters allow for the encoding of different text terms in independent dimensions without interference. This enables the integration of various tokenizer types, such as rule-based (e.g., n-gram letters), dictionary-based (e.g., Jieba[1]), and data-driven (e.g., SentencePiece [23]). Such diversity enhances the effectiveness in unsupervised settings by capturing both common and novel lexical items.

A primary concern with Bloom filters is *hash collision*, where different text terms might share the same hash value, potentially leading to evaluation inaccuracies with intersecting bits. Previous research [20] avoids this by limiting the vocabulary size (e.g., using tri-gram English letters). However, this approach may break apart crucial keywords like addresses, reducing retrieval accuracy. Fortunately, Bloom filters offer a built-in solution to mitigate hash collisions through the membership test, as explained in Section 3. This allows us to exclude query terms absent from the object's Bloom filter, facilitating more accurate term-matching. The second concern is *the loss of token order*, as Bloom filters do not encode the order of the text tokens. As spatial keyword queries are generally

[1]For more tokenizer details, see https://github.com/fxsjy/jieba

short and concise, we mitigate the problem by n-gram tokenizers with starting and ending marks, as proposed in DSSM [39]. For example, a 3-gram tokenizer with starting and ending marks will turn an address "123456789" into "#123", "456", "789#", thereby encoding the local order of digits to facilitate accurate matching. However, when essential token order information spans longer text sequences (e.g., "shop with coffee and ... but without food"), n-gram tokenizers may struggle to capture such information. To address this, we design an optional component to capture token order. Specifically, before encoding each token into the Bloom filter, we first project them into low-dimensional dense vectors using a linear layer and pass through a stack of 1-D convolution layers, following their order in the input. Since each 1-D convolution kernel assigns different weights to former and latter tokens within its receptive field, the stacked layers can capture orders across longer sequences. The third concern is *term mismatch*, which occurs when users use different words with similar meanings. To address this, we propose learning semantic similarities from labeled queries, extending beyond simple bit intersections. Specifically, we define the text similarity metric as follows:

$$TextSim(q,o) = \sum_{m}^{i} I(B_q, B_o)_i \cdot F_1(B_q, B_o)_i + F_2(B_q, B_o) \quad (1)$$

Here, B_q and B_o denote the Bloom filters with length m for the query and the object text, respectively. $I(B_q, B_o)$ represents the intersection of B_q and B_o, with i denotes the i-th bit. F denotes the neural networks, where F_1 evaluates the importance of the i-th bit in the Bloom filter, and F_2 learns the semantic similarity from labeled queries, both discussed in the following section.

4.2 Bloom Filter Evaluator

4.2.1 Model Architecture. The core of GeoBloom is a lightweight neural network, which is designed to effectively learn the relevance score from a few labeled queries. We use a large, shared embedding layer to learn city-specific information within the Bloom filters and encode them into dense embeddings. This is followed by two non-linear hidden layers that further compress them into low-dimensional space, extracting potential semantic relevance. Finally, the embeddings are expanded back to the dimension of the input Bloom filters, where each dimension is regarded as the importance of intersecting bit at the corresponding position. We propose two key techniques to facilitate the bit importance evaluation:

- *Zero-Projection*: The randomly initialized layer in common learning-based models will arbitrarily distort the importance of the Bloom filter bits, leading to poor effectiveness in unsupervised settings and premature convergence. To address the problem, we designed a *Zero-Projection* layer at the last layer of our model, which outputs a value of 1 when untrained, preserving the term-matching capabilities of Bloom filters in the absence of labeled queries.
- *LeakyReLU Activation*: Our model's relevance score is calculated as the average importance of intersecting bits. During training, the relevance scores for user-selected objects will increase, and others will decrease. However, this approach should be *asymmetric*, as each bit represents a potentially important text term even for non-selected objects. Increasing bit importance for selected objects reveals useful relevance while reducing it for non-selected objects risks missing crucial terms. Therefore, we use LeakyReLU activation, assigning small gradients to less significant bits (importance < 1), thereby emphasizing critical bits.

Formally, the neural network $F_1(B_q, B_o)$ can be defined as:

$$\mathbf{h}_1 = \sigma(\text{CONCAT}(W_1 B_q, W_1 B_o)), \quad W_1 \in \mathbb{R}^{h_1 \times m} \quad (2)$$

$$\mathbf{h}_2 = \sigma(W_2 \mathbf{h}_1 + b2), \quad W_2 \in \mathbb{R}^{h_2 \times 2h_1} \quad (3)$$

$$\mathbf{h}_3 = \sigma(W_3 \mathbf{h}_2 + b3), \quad W_3 \in \mathbb{R}^{h_3 \times h_2} \quad (4)$$

$$F_1(B_q, B_o) = \text{LeakyReLU}(W_4 \mathbf{h}_3) + 1, \quad W_4 \in \mathbb{R}^{m \times h_3} \leftarrow 0 \quad (5)$$

Here, $\mathbf{h}_i$ denotes the output of i-th layer with dimension h_i, weight W_i, and bias b_i. σ denotes the non-linear activation function [2]. Eq 5 specifies the Zero-Projection layer, where $W_4 \in \mathbb{R}^{m \times h_3}$ is set to zero, ensuring each intersecting bit has an equal initial importance of 1. Notably, this doesn't impede the network training, as the gradient of $F_1(B_q, B_o)$ with respect to W_4, given by [3]

$$\frac{\partial F_1}{\partial W_4} = \text{LeakyReLU}'(0) \cdot \mathbf{h}_3 \neq 0, \; if \; \mathbf{h}_3 \neq 0 \quad (6)$$

In the case of *term mismatch*, the intersecting bits between the query's and the user-selected object's Bloom filters can be relatively few, so that F_1 cannot adequately capture the relevance. However, $\mathbf{h}_2$ lies in a dense semantic space, allowing us to roughly estimate the semantic similarity beyond the limitation of intersecting bits. We leverage $\mathbf{h}_2$ to evaluate a *Semantic Score* as follows:

$$F_2(B_q, B_o) = W_4' \sigma(W_3' \mathbf{h}_3), \quad W_3' \in \mathbb{R}^{h_3 \times h_2}, W_4' \in \mathbb{R}^{1 \times h_3} \leftarrow 0 \quad (7)$$

We will further combine the text similarity metric with other metrics (e.g., geographic distances), as detailed in Section 4.3.

4.2.2 Inference Optimizations. We observe that Bloom filters resemble the sparse board representations in computer chess, inspiring us to employ the Efficiently Updatable Neural Network (NNUE) [31]. NNUE is a 4-layer MLP that computes the game-winning probability given a chessboard. When a chess piece moves, it efficiently updates the hidden layers by focusing only on relevant rows in the embedding matrix. In our model, we mimic NNUE's design by gathering and updating only the relevant columns in the proposed Zero-Projection layer. We also follow NNUE's quantization scheme with SIMD (Single Instruction, Multiple Data) for acceleration. We propose key optimizations for inference efficiency:

- *Offline Pre-computing*: As texts of geographic objects seldom update during the query phase, we can offload the embedding process of object Bloom filters to the offline environment. The pre-computed embeddings are saved and reused, eliminating the need for encoding objects during online inference.
- *Sparse Evaluation*: While the Bloom filters of geographic objects can be relatively dense (as they can contain many words), the query Bloom filters are typically sparse as the user input is often confined to a relatively short length. By targeting the non-zero bits in the query Bloom filters and selecting the corresponding columns in the Zero-Projection layer, we can substantially diminish the time complexity of the importance evaluation.

[2] We use $\sigma(x) = \text{ClippedReLU}(x) \in [0, 1]$ to prevent numerical overflow in NNUE.
[3] We can't use standard ReLU or ClippedReLU in this layer as their gradient at x=0 is 0.

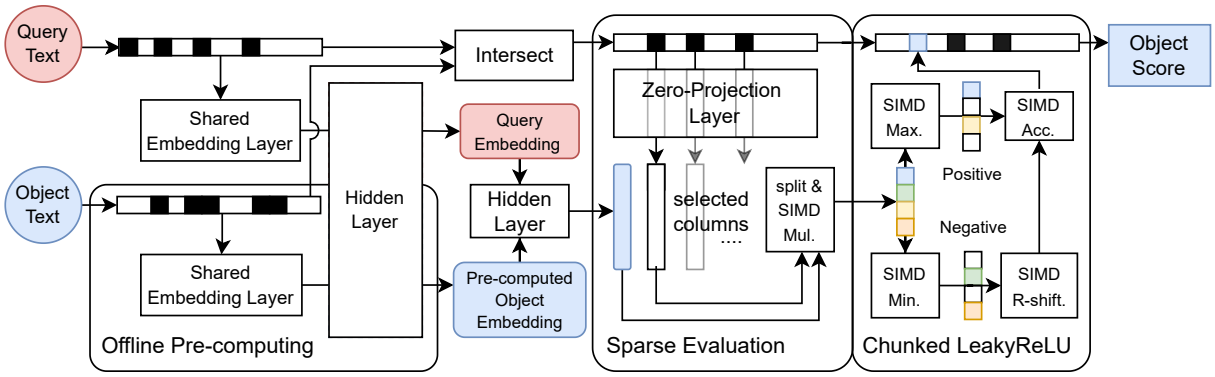


Figure 2: The crucial optimization techniques in Bloom Filter Evaluator: (1) *Offline Pre-computing*, which encodes the Bloom filters of objects into short dense embedding in an offline environment; (2) *Sparse Evaluation*, which focuses on the intersecting bits to eliminate unnecessary computations; and (3) *Chunked LeakyReLU*, which splits the *Zero-Projection* layer output into chunks and computes LeakyReLU with SIMD.

- *Chunked LeakyReLU*: The LeakyReLU function is SIMD-unfriendly due to its special handling of negative values. We propose to split the output vector into multiple chunks and activate separately with bitwise right-shifting. This ensures compatibility with SIMD without compromising the activation function's integrity.

As illustrated in Figure 2, the three optimizations address the major bottlenecks in model inference, enabling low time complexity.

4.3 Bloom Filter Tree

Until now, our model has used pairwise query-object evaluations. While each evaluation is computationally cheap, scalability remains challenging for large databases. A common solution is to reduce comparisons using a search index, but designing an efficient index that preserves effectiveness is non-trivial. We propose organizing geographic objects by spatial proximity through hierarchical clustering (e.g., KMeans), forming a tree structure. Each node in the tree is represented by a centroid, a radius r defining its spatial extent, and a Bloom filter aggregated from its child nodes. We then adapt our Bloom Filter Evaluator to efficiently search this tree structure.

- *Text Similarity with Normalization*: The Bloom filters at the higher hierarchy of the tree contain text terms from more geographic objects, leading to a higher hash collision rate and an overestimated similarity that lacks discriminative power between nodes. To counteract the problem, we propose to use beam-search and normalize the text similarities within the search beam:

$$\hat{T}(q,n) = \text{sigmoid}\left(\beta_1 \cdot (TextSim(q,n) - \mu)/\sigma + \beta_2\right) \tag{8}$$

  $\hat{T}(q,n)$ normalizes the original text similarity scores within a search beam based on their mean μ and standard deviation σ, modulated by trainable parameters β_1 and β_2. This process unifies the text similarities across the tree's hierarchies, enabling more consistent comparisons.
- *Geographic Distance with Normalization*: We integrate geographic distances into the relevance score. We observe that individuals are more sensitive to the distances to nearby objects, while this sensitivity diminishes for objects further apart. Hence, we normalize the distances to better align human spatial perception:

$$\hat{D}(q,n) = -\log(1 + \max(0, \text{Dist}(q,n) - r_n)) \tag{9}$$

  Here, $\hat{D}(q,n)$ transforms the distance between the query and a node's boundary using a negative logarithm. This approach models the decreasing sensitivity to increasing distance.
- *Mixture-of-Experts*: The search process from the root to the leaf nodes is a transition from coarse retrieval to precise ranking, where tree nodes at different depths carry diverse information. Hence, we create multiple experts by duplicating the hidden layers and the zero-projection layer, each dedicated to a distinct tree depth, improving the model's ability to discern relevance.

  Finally, we define the relevance score as follows:

$$Relevance(q,n) = \hat{T}(q,n) + \gamma_1 \hat{D}(q,n) + \gamma_2 \hat{T}(q,n)\hat{D}(q,n) \tag{10}$$

The trainable parameters γ_1 and γ_2 balance the influence of two similarities and their first-order interaction, which can exclude those very proximate nodes with little text similarities. Notably, these parameters don't require manual tuning. We initially set $\beta_2 = \gamma_2 = 0$ and $\beta_1 = \gamma_1 = 1$, and train all model parameters together with the neural networks via LambdaRank [5] loss.

It is noteworthy that our beam search on the Bloom Filter Tree has the same level of effectiveness as a brute-force search because it maintains the property akin to a max-heap by satisfying two criteria: For a given query, (1) the number of its intersecting bits with parent nodes is always greater than or equal to that with child nodes, and (2) its distances to parent nodes are always smaller than or equal to that to child nodes. Assuming no normalization or model training, a greedy search on the tree will yield the same results as a brute-force search. While the normalization and the training inevitably break the max-heap property, a beam search with proper beam widths can effectively mitigate inaccuracies.

4.4 Leveraging Geographic Context

We provide an additional feature to leverage *geographic context*, i.e., geographic objects that are spatially close to the user-selected objects. This is particularly useful when users search for objects near

specific addresses or landmarks. For instance, a query "Hotel near New York Central Park" might aim to find "The Manhattan at Times Square Hotel". The "New York Central Park" doesn't share any text terms with the hotel, but it serves as the crucial geographic context to locate the ground truth. Existing PLM-based methods [11, 19] generally pre-train on massive object-context pairs, which introduces substantial costs in data collection and model training.

Our Bloom Filter Tree provides a novel solution to identify useful geographic contexts. For a given query, we not only find the node that best matches the query Bloom filter but also identify a nearby node that best supplements those unmatched bits (i.e., contains useful context terms). We introduce a $P_{context}$ to estimate the probability that a given node contains geographic context:

$$P_{context}(B_q, B_c) = \text{sigmoid}(\sum_{m}^{i} I((B_q \backslash B_n), B_c)_i \cdot F_3(B_q, B_c)_i) \quad (11)$$

Here, B_c represents the Bloom filter of the geographic context. $\backslash$ computes the non-zero bit elements in B_q but not in B_n. The neural network F_3 assesses the importance of each bit by duplicating the last two layers in Bloom Filter Evaluator (i.e. Eq. 4 and 5). Upon selecting an appropriate context using this probability, we further duplicate an F_4 to evaluate the bits within the context object. Our text similarity metric in Eq. 1 is improved as follows:

$$TextSim(q, n) = \underbrace{\sum_{m}^{i} I(B_q, B_n)_i \cdot F_1(B_q, B_n)_i}_{\text{Object Score}} + \underbrace{F_2(B_q, B_n)}_{\text{Semantic Score}} + \underbrace{\sum_{m}^{i} I(B_q \backslash B_n, B_c)_i \cdot F_4(B_q, B_n)_i}_{\text{Context Score}} \quad (12)$$

We construct pseudo labels to optimize $P_{context}$ leveraging labeled queries. For each user-selected object, we find the nearby object (e.g., within 2 km) with the highest $P_{context-untrained} = \sum_{m}^{i} I((B_q \backslash B_n), B_c)_i$ as the ground truth geographic context and optimize $P_{context}$ via BCELoss. As traversing the entire tree for geographic context is computationally intensive, we directly select the object with the largest $P_{context}$ within the search beam. The beam search process in our framework is detailed in Algorithm 1.

4.5 Complexity Analysis

We analyze the time and space complexities of GeoBloom. Let h_1, h_2, h_3 be the output dimension of the i-th model layer, u the count of non-zero bits in the query, b the beam width, and d the tree depth. The overall time complexity is given by $O(uh_1 + h_1h_2 + b^2 \cdot d(h_2h_3 + uh_3))$. $uh_1 + h_1h_2$ accounts for query encoding time, which is negligible as the number of visited nodes $bd \gg 1$. $h_2h_3 + uh_3$ reflects the inference time per node. As h_2 and h_3 are small (e.g., 16 or 32), the inference time is dominated by query length, tree depth, and beam width. The space complexity is affected by the dataset size $|S|$ and the Bloom filter length m. The model's space complexity is $O(mh_1 + d(h_1h_2 + h_2h_3 + mh_3))$, the Bloom Filter Tree's is $O(m|S|)$, and the offline pre-computed embeddings' is $O(h_2|S|)$.

Algorithm 1 Beam Search with In-beam Context Selection

Require: Bloom Filter Tree, query q, spatial distance threshold R
Ensure: Ranked list of nodes based on relevance scores

```
Initialize Beam with root node(s) of the tree
while not at leaf nodes do
    Initialize NewBeam, TextSims, DistSims ← ∅
    for parent in Beam do
        Get child nodes from parent to form CandidateBeam
        for node in CandidateBeam do
            Initialize BestContext ← null
            for context in Beam within R and context ≠ n do
                Compute P_context as Eq. 11 using B_q\B_n
                if P_context > P_context(BestContext) then
                    BestContext ← context
                end if
            end for
            Compute TextSim as Eq. 12 and add to TextSims
            Compute D̂ as Eq. 9 and add to DistSims
        end for
    end for
    Normalize TextSims to get T̂ as Eq. 8
    Combine T̂ with DistSims to get Relevance as Eq. 10
    Rank CandidateBeam by Relevance
    Select top candidates' child nodes to form new Beam
end while
return Beam
```

5 EXPERIMENTS

We conduct experiments to answer the following questions:

- **RQ1**: How effective is the GeoBloom framework in unsupervised and supervised settings with varying sizes of training data, compared to state-of-the-art baselines?
- **RQ2**: How does the inference time and space efficiency of the GeoBloom framework compare to the PLM-based method, and how does it scale with increasing dataset sizes?
- **RQ3**: How do the important parameters, components, and optimization techniques within the GeoBloom framework affect its effectiveness and efficiency?

5.1 Experimental Settings

5.1.1 Datasets. We utilize two real datasets and two open-source synthetic datasets. The real datasets Beijing and Shanghai are provided by Meituan, a leading consumer service platform in China. These datasets are derived from user interactions, where consumers submit keywords along with their current or intended locations to search for products and services, selecting from the geographic objects suggested by the platform. The synthetic datasets are from the GeoGLUE [24], which contains millions of POIs in Hangzhou, China. It invites human experts to synthesize query texts and locations and select ground truth POIs. However, its POI locations are shuffled and over 50% fake POIs are injected, largely diverging from real datasets. To address this, we supplement the GeoGLUE$_{clean}$ dataset, where GeoGLUE queries are aligned with the real Hangzhou POIs collected from the OpenStreetMap [4]. Specifically, GeoGLUE queries

[4] https://www.openstreetmap.org/. Data accessed on Jan 12th, 2024.

are labeled with 17,290 (noisy) POIs. We find the real counterpart for each POI via BM25 and GPT-4-turbo and successfully match 10,278 of them to real Hangzhou POIs, forming GeoGLUE$_{clean}$. The statistics of all datasets are presented in Table. 2.

Table 2: Dataset Statistics

Dataset	#POIs	#Queries	#Train	#Dev	#Test
Beijing	122,420	168,998	136,890	15,209	16,899
Shanghai	116,859	127,183	103,019	11,446	12,718
GeoGLUE	2,849,754	90,000	50,000	20,000	20,000
GeoGLUE$_{clean}$	777,295	53,891	30,048	11,686	12,157

5.1.2 Baselines. We compare with baselines including classical methods, early learning-based methods, and PLM-based methods.

(1) **Classical Methods**.
- BM25 [36]: This classical information retrieval method exclusively ranks geographic objects based on textual relevance.
- SIF [3]: Smooth Inverse Frequency (SIF) represents geo-textual objects through a weighted average of GloVe [33] word embeddings and refine them with PCA or SVD.

(2) **Early Learning-based Methods**
- DRMM [17]: This model evaluates text similarities based on pairwise local interactions at the term level. It doesn't account for geographic proximity.
- ARC-I [18]: This model encodes texts in both query and objects and then measures text relevance with multi-layer perceptrons. It doesn't account for geographic proximity.
- PALM [47]: This model leverages both text and geographic proximity to learn representations for queries and objects.

(3) **PLM-based Methods**
- BERT [10]: BERT represents a landmark in PLMs that benefit from pre-training on extensive corpora. Its output embeddings are widely used to assess text similarities.
- OpenAI[5]: OpenAI's text-embedding-3-small is one of its "newest and most performant embedding models". While its technical details remain proprietary, it is well-known for producing high-quality embeddings for retrieval tasks.
- DrW [26]: This method utilizes a frozen BERT encoder to capture term-level text similarities, and then integrates geographic proximity via the attention mechanism.
- MGeo [11]: This method applies multi-task pre-training on a BERT-based encoder and fine-tunes it by user queries. [6]
- DPR [21]: This method fine-tunes BERT through metric learning, which ensures relevant query-object pairs will have smaller distances than the irrelevant ones in the vector space. It does not account for geographic proximity.
- LIST [43]: This method improves DPR by combining it with geographic distances aligned with human preferences. It further proposes a novel learned index to boost efficiency.

[5]https://platform.openai.com/docs/guides/embeddings. Accessed on Mar 3rd, 2024

[6]As we are unable to replicate the results of MGeo with their official code, we reference the evaluation results as presented by the authors.

5.1.3 Evaluation metrics. We employ two widely-used metrics Recall and Normalized Discounted Cumulative Gain (NDCG) to assess the effectiveness. For a given query, Recall@k measures the fraction of user-selected objects retrieved within the top-k results, while NDCG@k additionally accounts for ranking. We use the Recall@20, Recall@10, NDCG@5, and NDCG@1 following [11, 43].

5.1.4 Implementation Details. As the vanilla BM25, SIF, BERT, OpenAI, DRMM, ARC-I, and DPR only consider text similarity, we supplement BM25-D, SIF-D, BERT-D, and OpenAI-D, DRMM-D, ARC-I-D, and DPR-D to incorporate the geographic distance. The relevance score between query q and object o is defined as:

$$Relevance(q, o) = (1-\alpha)(1-D_{norm}(q, o)) + \alpha \cdot TextSim_{norm}(q, o)$$

Here, $Dist_{norm}(q, o)$ denotes the geographic distance between query and object, $TextSim_{norm}$ denotes the text similarity from the vanilla baseline, both are normalized to $[0, 1]$. α is a hyper-parameter balancing the text and the distance similarities. We run a grid search on the dev split to get the optimal α for each baseline. We set $k = 0.3$, $b = 0.1$ for BM25 and BM25-D for better effectiveness on short texts. For other baselines, we adhere to the hyper-parameters in their respective paper. For GeoBloom, we use a combination of 1-gram, 2-gram, and Jieba tokenizers, with $k = 2$ hash functions based on SHA-256, set Bloom filter length to $m = 16384$ and for Beijing, Shanghai, GeoGLUE$_{clean}$, and $m = 32768$ for GeoGLUE. We set the depth of Bloom Filter Tree $d = 4$, the anchor selection threshold $R = 1km$. We perform a grid search on beam widths, setting $b = 400$ for Beijing and Shanghai, $b = 1000$ for GeoGLUE$_{clean}$, and $b = 4000$ for GeoGLUE unless otherwise specified.

5.2 Effectiveness Studies (RQ1)

5.2.1 Effectiveness in Unsupervised Settings. We validate the effectiveness of GeoBloom without labeled queries by comparing it with baselines that support unsupervised settings, excluding those reliant on labeled queries. Notably, we set beam width $b = 6000$ for GeoBloom on GeoGLUE to counteract the datasets' fake POIs. Tables 3 reports the effectiveness of the evaluated methods with the best results in bold and the second-best underlined, which lead us to two key insights: (1) The classical term-matching method BM25-D and our GeoBloom significantly outperforms vector-based methods SIF-D and BERT-D, showcasing a strong effectiveness that rivals the leading commercial product, OpenAI-D. This underscores the importance of term-matching capabilities for GIR models. (2) On two real datasets and GeoGLUE$_{clean}$, GeoBloom significantly surpasses all baselines, notably exceeding BM25-D and OpenAI-D in NDCG@1 by a significant margin. While on GeoGLUE where over 50% of POIs are fake, GeoBloom's advantage narrows, it still performs competitively, showcasing its robustness even when the data quality of geographic objects is compromised.

5.2.2 Effectiveness in Supervised Settings. We compare the effectiveness of supervised GeoBloom against baselines that support supervised learning. For LIST, we use the proposed learned index in their paper, while for other baselines we perform a brute-force search over the full dataset. Notably, DRMM, DRMM-D, ARC-I, PALM, and DrW are not tested on the synthetic dataset due to their scalability issues, which requires more than 24 hours for evaluation. Tables 4 reports the average effectiveness from 10 consecutive

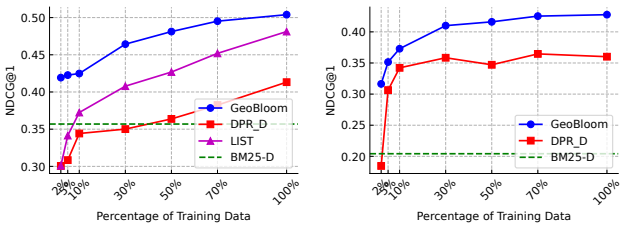
Table 3: Effectiveness in Unsupervised Settings

Method	Beijing				Shanghai			
	Recall@20	Recall@10	NDCG@5	NDCG@1	Recall@20	Recall@10	NDCG@5	NDCG@1
BM25	0.4023	0.3401	0.2199	0.1634	0.4115	0.3274	0.1913	0.1260
BM25-D	0.5904	0.5477	0.4263	0.3569	0.6873	0.6484	0.5215	0.4380
SIF	0.1792	0.1493	0.0944	0.0702	0.1967	0.1587	0.0949	0.0646
SIF-D	0.2148	0.1918	0.1375	0.1108	0.2608	0.2313	0.1626	0.1272
BERT	0.1949	0.1602	0.1169	0.0979	0.1593	0.1277	0.0853	0.0662
BERT-D	0.2801	0.2400	0.1614	0.1298	0.3123	0.2622	0.1687	0.1233
OpenAI	0.3892	0.3265	0.2157	0.1637	0.4067	0.3213	0.1875	0.1258
OpenAI-D	0.5812	0.5206	0.3803	0.3078	0.6755	0.6313	0.4852	0.3864
GeoBloom	**0.6251**	**0.5858**	**0.4730**	**0.4073**	**0.7440**	**0.7160**	**0.6031**	**0.5230**
(Gain)	5.88%	6.96%	10.95%	14.11%	8.26%	10.43%	15.66%	19.39%
	GeoGLUE				**GeoGLUE$_{clean}$**			
BM25	0.5776	0.5430	0.4368	0.3640	0.4651	0.4046	0.2800	0.2086
BM25-D	0.5895	0.5546	0.4442	0.3676	0.4751	0.4083	0.2802	0.2042
SIF	0.1355	0.1215	0.0938	0.0762	0.1047	0.0905	0.0606	0.0427
SIF-D	0.1477	0.1352	0.1021	0.0807	0.1120	0.0954	0.0622	0.0448
BERT	0.2345	0.2121	0.1647	0.1347	0.1093	0.0891	0.0558	0.0380
BERT-D	0.2843	0.2548	0.1872	0.1464	0.1865	0.1524	0.0859	0.0517
OpenAI	0.4739	0.4372	0.3468	0.2878	0.4765	0.4242	0.2922	0.2119
OpenAI-D	0.5118	0.4733	0.3759	0.3103	0.4914	0.4321	0.2934	0.2094
GeoBloom	**0.5925**	**0.5566**	**0.4459**	**0.3704**	**0.5204**	**0.4645**	**0.3254**	**0.2367**
(Gain)	0.51%	0.36%	0.39%	0.78%	5.87%	7.48%	10.94%	11.68%

Table 4: Effectiveness in Supervised Settings

Method	Beijing				Shanghai			
	Recall@20	Recall@10	NDCG@5	NDCG@1	Recall@20	Recall@10	NDCG@5	NDCG@1
DRMM	0.2387	0.1773	0.1105	0.0758	0.2702	0.1921	0.1287	0.0804
DRMM-D	0.4949	0.4357	0.2378	0.1566	0.5107	0.4380	0.2433	0.1595
ARC-I	0.2779	0.2221	0.1317	0.0774	0.3192	0.2551	0.1687	0.0895
ARC-I-D	0.5265	0.4531	0.3685	0.2577	0.5618	0.5079	0.3978	0.2660
PALM	0.3514	0.3098	0.2077	0.1343	0.4617	0.4023	0.2065	0.1223
DrW	0.6968	0.6316	0.4814	0.3791	0.7689	0.7159	0.5394	0.4114
DPR	0.4990	0.4183	0.2775	0.2121	0.4991	0.4087	0.2498	0.1746
DPR-D	0.7382 (± 0.013)	0.6688 (± 0.011)	0.4980 (± 0.005)	0.4132 (± 0.009)	0.7667 (± 0.012)	0.7281 (± 0.010)	0.5641 (± 0.007)	0.4554 (± 0.014)
LIST	0.7711 (± 0.010)	0.7170 (± 0.0009)	0.5668 (± 0.0010)	0.4812 (± 0.010)	0.7721 (± 0.012)	0.7401 (± 0.012)	0.6099 (± 0.010)	0.5139 (± 0.012)
GeoBloom	**0.7719** (±0.002)	**0.7270** (±0.002)	**0.5839** (±0.002)	**0.5039** (±0.003)	**0.8424** (±0.001)	**0.7996** (±0.002)	**0.6629** (±0.002)	**0.5694** (±0.003)
(Gain)	0.10%	1.40%	3.55%	5.91%	8.49%	8.17%	9.39%	12.19%
	GeoGLUE				**GeoGLUE$_{clean}$**			
MGeo	0.7049	N/A	N/A	0.5270	N/A	N/A	N/A	N/A
DPR	0.7864	0.7450	0.6171	0.5199	0.6610	0.6052	0.4585	0.3581
DPR-D	0.7994 (± 0.009)	0.7611 (± 0.008)	0.6318 (± 0.007)	0.5350 (± 0.009)	0.6678 (± 0.009)	0.6131 (± 0.009)	0.4633 (± 0.006)	0.3601 (± 0.007)
LIST	**0.8250** (± 0.009)	**0.7909** (± 0.008)	**0.6747** (± 0.009)	**0.5815** (± 0.010)	0.6922 (± 0.010)	0.6375 (± 0.010)	0.4729 (± 0.008)	0.3593 (± 0.009)
GeoBloom	0.8074 (±0.003)	0.7736 (±0.002)	0.6369 (±0.002)	0.5320 (±0.003)	**0.7792** (±0.004)	**0.7224** (±0.004)	**0.5518** (±0.005)	**0.4276** (±0.007)

runs, with the standard deviations for GeoBloom and competitive baselines DPR-D and LIST. (1) On real datasets, GeoBloom significantly outperforms early learning-based methods, including DRMM-D, ARC-I-D, and PALM, by a large margin. While all these methods adopt lightweight neural networks, GeoBloom stands out by harnessing the term-matching capability of Bloom filters. (2) GeoBloom can rival or surpass the state-of-the-art PLM-based GIR methods DrW, DPR-D, and LIST. This demonstrates the great effectiveness of our proposed strategy of evaluating Bloom filter bits, which provides strong supervised learning capability without relying on extensive pre-training. (3) On the synthetic datasets GeoGLUE, however, GeoBloom yields similar performance as DPR-D and LIST. This is because GeoBloom is affected by the fake POIs in the GeoGLUE dataset, while DPR-D and LIST can exclude such POIs via negative sampling. Nevertheless, on GeoGLUE$_{clean}$ where fake POIs no longer exist, GeoBloom showcases its significant superiority over DPR-D and LIST.



(a) Beijing **(b) GeoGLUE$_{clean}$**

Figure 3: NDCG@1 vs. Training Data across datasets.

5.2.3 Effectiveness vs. Training Data Size. We assess GeoBloom's data efficiency on Beijing and GeoGLUE$_{clean}$ by starting with 2% of training data and gradually expanding to 100%, simulating the accumulation of labeled queries over time. We compare GeoBloom with DPR-D and LIST using NDCG@1, with BM25-D as a baseline. LIST is not evaluated on GeoGLUE$_{clean}$ due to its high training cost, exceeding 24 hours per data point. (1) On the real dataset Beijing, DPR-D requires about 40% (i.e., 52,000) of training queries to match the effectiveness of BM25-D, whereas LIST uses around 8% (i.e., 10,000). While both methods use BERT for text similarities, LIST additionally learns the user preference of geographic distances, thereby showing better data efficiency. (2) GeoBloom can outperform BM25-D without training data, as it has the same level of term-matching capabilities as BM25-D while better balancing the geographic distances. It outperforms DPR-D and LIST with significantly fewer training queries, using about 2% and 40% queries to surpass DPR-D and LIST with 100%. This highlights its data efficiency in the scarcity of labeled queries. (3) On synthetic dataset GeoGLUE$_{clean}$, we observe a significant leap in effectiveness for both GeoBloom and DPR-D with as little as 2-10% (i.e., < 5,000) of the training data, suggesting that GeoBloom captures essential relevance within expert-synthesized data as effectively as DPR-D. However, DPR-D quickly converges given more training queries, while GeoBloom continues to improve steadily, showing superior data efficiency in handling a growing volume of training data.

5.3 Efficiency Studies (RQ2)

We compare the efficiency of GeoBloom against DPR-D with HNSW, LSH, IVF, and IVFPQ from FAISS [12], and LIST with its proposed learned index. GeoBloom is quantized following Stockfish NNUE scheme [7], while DPR-D and LIST uses ONNX [8] for acceleration.

5.3.1 Time Efficiency. We assess the average time cost for 5,000 random queries under both GPU and CPU-only conditions on an Intel(R) Core(TM) i9-10900K CPU @ 3.70GHz with 128GB RAM and an RTX 3080 with 10GB VRAM. [9] Notably, we test the single-thread performance for CPU evaluation unless otherwise specified. Table 5 illustrates that GeoBloom achieves unparalleled inference efficiency on CPU, comparable to or even 4x faster than the best GPU-accelerated method DPR-IVF. Although GeoBloom requires more time to counteract the fake POIs in GeoGLUE, its runtime in CPU-only scenarios remains superior. Moreover, GeoBloom only uses 1 of the 20 CPU threads, while GPU-based DPR fully utilizes the GPU (via batch encoding). When all 20 CPU threads are used, GeoBloom speeds up linearly and becomes up to 80x faster than the GPU-based DPR-IVF on real datasets. We also evaluate the training efficiency with the time to reach the best effectiveness. As shown in Table 6, while DPR-D benefits from the pre-trained BERT for fast convergence, training GeoBloom is still faster as it has significantly fewer parameters and lower time complexity.

Table 5: Query Runtime (ms) across Datasets.

Method	Beijing	Shanghai	GeoGLUE	GeoGLUE$_{clean}$
Device: GPU				
DPR-D-HNSW	7.2	6.4	14.5	9.3
DPR-D-IVF	2.9	2.9	4.0	3.2
DPR-D-LSH	5.2	5.1	31.1	14.3
DPR-D-IVFPQ	3.6	3.7	5.2	4.5
LIST	3.0	3.0	5.1	6.0
Device: CPU				
DPR-D-HNSW	294.4	293.2	302.4	296.2
DPR-D-IVF	320.0	328.4	340.8	332.8
DPR-D-LSH	303.7	296.3	304.5	300.9
DPR-D-IVFPQ	334.8	323.7	346.2	339.5
GeoBloom **(CPU)**	0.694	0.662	29.532	5.501
+ with 20 threads	0.037	0.036	1.563	0.277

Table 6: Training Time in Supervised Settings across Datasets

Method	Beijing	Shanghai	GeoGLUE	GeoGLUE$_{clean}$
GeoBloom	**1h 59m**	**1h 20m**	**1h 35m**	**26m**
DPR-D	7h 28m	4h 42m	2h 11m	1h 48m

[7] https://github.com/official-stockfish/nnue-pytorch
[8] https://github.com/onnx/onnx
[9] It is not rigorous to directly compare GPU with CPU, but we select the two devices with similar prices. An i9-10900X CPU @ 3.70GHz is priced at around $530 USD (https://www.cpubenchmark.net/) while an RTX 3080 GPU 10GB is priced at $490 USD (https://www.videocardbenchmark.net/gpu_value.html) on June 1st, 2024.

5.3.2 Space Efficiency. We compare the CPU memory and disk usage of GeoBloom with DPR-D and LIST, measured with Linux VmHWM. To efficiently derive relevance scores with text similarities and geospatial distances, DPR-D and LIST load all POI embeddings in memory, following the settings in LIST[43]. As in Table 7, GeoBloom saves up to 74.72% of the runtime memory. Its superior space efficiency stems from the use of Bloom filters. Table 8 presents a comparison of the file sizes. Notably, GeoBloom saves up to 87.63% disk space compared to DPR, making it a practical option for applications where disk space is at a premium.

Table 7: Runtime Memory (MB) across Datasets

Method	Beijing	Shanghai	GeoGLUE	GeoGLUE$_{clean}$
DPR-D-HNSW	556	537	9586	2733
DPR-D-LSH	511	495	8543	2440
DPR-D-IVF	512	496	8518	2437
DPR-D-IVFPQ	516	500	8585	2456
LIST	508	505	8515	2421
GeoBloom	**132**	**125**	**2907**	**692**
(Save)	74.02%	74.72%	65.86%	71.42%
DPR-D Components				
ONNX	149.2	149.2	149.2	149.2
POI Embed.	358.6	342.4	8348.8	2277.2
- Index HNSW	47.7	45.2	1087.5	306.3
- Index LSH	3.1	2.9	44.5	13.2
- Index IVF	4.4	4.3	20.2	10.7
- Index IVFPQ	8.4	8.1	87.1	29.8
GeoBloom Components				
NNUE	21.9	21.9	38.5	21.9
Bloom Filters	24.4	22.9	706.0	149.7
Node Embed.	17.0	16.2	397.5	108.5
Bloom Filter Tree	60.8	58.5	2255.9	399.9

Table 8: Disk Usage (MB) across Datasets

DPR	Beijing	Shanghai	GeoGLUE	GeoGLUE$_{clean}$
Model	193.97	193.97	193.97	193.97
Embeddings	179.33	171.18	4174.44	1138.62
Total	373.29	365.15	4368.41	1332.58
GeoBloom				
Model	20.10	20.10	40.10	20.10
Embeddings	8.90	8.45	198.78	56.12
Bloom Filters	17.98	16.60	319.41	117.74
Total	**46.98**	**45.15**	**558.29**	**193.96**
(Save)	87.41%	87.63%	87.22%	85.45%

5.3.3 Scalability. We explore the scalability of GeoBloom in both the training and inference phases. Figure 4 illustrates that the training time of GeoBloom increases linearly with the volume of training queries. We also examine the effect of dataset size on the inference effectiveness and efficiency of GeoBloom by collecting publicly available POIs in Hangzhou, China, and merging them with the GeoGLUE$_{clean}$ POIs. We observe a sub-linear increase in query time, a benefit derived from our tree-based indexing strategy.

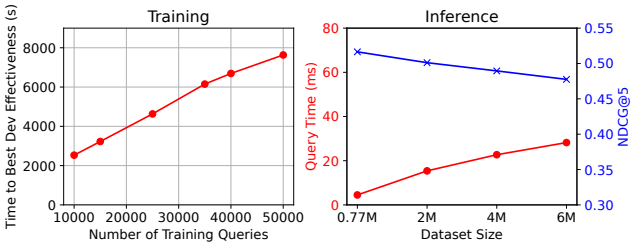


Figure 4: Scalability Analysis on Hangzhou Data

5.4 Ablation Studies (RQ3)

5.4.1 Effect of Model Components. We remove various components from the Bloom Filter Evaluator to analyze their effectiveness and efficiency impact, measured by NDCG@5 and Query Per Second (QPS). As shown in Table 9, in unsupervised settings, removing *Zero Projection* breaks the model as the initial relevance scores are randomized. The *Text Norm.* and *Dist Norm.* significantly affect the effectiveness as they ensure a fair balance between text similarities and geographic distances. In supervised settings, however, such utility diminishes where the model inherently learns this balance from data. Replacing LeakyReLU with Sigmoid (where $\sigma'(x) = \sigma(-x)$) or removing it reduces effectiveness, as LeakyReLU introduces non-linearity and mitigates negative impacts from unselected objects with its asymmetric gradient about x=0. Removing *Semantic* and *Context Score* both markedly impact the effectiveness on GeoGLUE$_{clean}$ but not on Beijing, as GeoGLUE queries are rich in geographic context. Table 10 shows that *Sparse Evaluation* and *Offline Pre-computing* are crucial for high efficiency, and *Chunked LeakyReLU* brings up to 19% acceleration. Removing the *Context Score* enhances speed but compromises effectiveness. In summary, *Zero-Projection*, *Offline Pre-computing*, *Sparse Evaluation* are crucial for GeoBloom to work efficiently.

Table 9: Impact on Effectiveness by Model Components

Component	**Beijing**		**GeoGLUE$_{clean}$**	
	NDCG@5	Impact	NDCG@5	Impact
GeoBloom-unsupervised	0.4730		0.3254	
w/o Zero Projection	0.0211	-95.54%	0.0254	-92.19%
w/o Text. Norm.	0.4591	-2.94%	0.2709	-16.75%
w/o Dist. Norm.	0.4231	-10.55%	0.1618	-50.28%
GeoBloom-supervised	0.5873		0.5518	
w/o Zero Projection	0.5815	-0.98%	0.4920	-10.83%
w/o Text. Norm.	0.5872	-0.02%	0.5139	-6.86%
w/o Dist. Norm.	0.5874	0.02%	0.5144	-6.77%
w/o LeakyReLU	0.5357	-8.79%	0.4641	-15.88%
- replace with Sigmoid	0.5668	-3.49%	0.5049	-8.49%
w/o Semantic Score	0.5821	-0.88%	0.5034	-8.76%
w/o Context Score	0.5772	-1.73%	0.5133	-6.98%

5.4.2 Effect of the Token Order Component. We further evaluate the effectiveness of the optional token order component in Section 4.1. We perform a grid search over the number of stacked 1-D convolution layers and kernel sizes, setting them to 3 and 5, respectively. Since long texts with critical token orders rarely appear in our evaluation datasets, we use GPT-4o to synthesize 1,000 long queries,

Table 10: Impact on Efficiency by Model Components

Component	Beijing		GeoGLUE$_{clean}$	
	QPS	Impact	QPS	Impact
GeoBloom	1418.75	0.00%	191.46	0.00%
w/o Offline Pre-computing	43.64	-96.92%	4.42	-97.69%
w/o Sparse Evaluation	8.24	-99.42%	1.14	-99.41%
w/o Chunked LeakyReLU	1145.42	-19.27%	177.33	-7.38%
w/o Context Score	1538.43	8.44%	225.53	17.79%

each paired with 1 matching and 4 non-matching POIs, using the following prompt: *"Generate examples for testing the model's ability to capture token orders. Each example should include: (1) Query: A description of an entity with a specific attribute order (e.g., 'Cafe with food but without Wi-Fi'). (2) 1 Match: A description of the same entity with attributes in the same order but rephrased (e.g., 'Cafe without Wi-Fi and ... but with food'). (3) 4 Non-Matches: Descriptions of the same entity where attributes appear in reversed or incorrect orders (e.g., 'Cafe with Wi-Fi ... but without food')."* The synthetic data is split into a 700/100/200 train/dev/test set. For a fair comparison, we also randomly select 1,000 labeled queries from the Beijing and GeoGLUE$_{clean}$ datasets (each has already labeled with 1 matching POI) and use BM25-D to select 4 non-matching POIs. Intuitively, GeoBloom with the proposed component should capture the token order in long text sequences (which BM25-D cannot capture), and perform better on all datasets. However, as shown in Table 11, while the proposed component effectively captures useful token order information in synthetic data, it fails to improve the effectiveness on Beijing and GeoGLUE$_{clean}$ queries, suggesting that the token order in long sequences is not important for these two datasets.

Table 11: NDCG@5 with the Token Order Component

Dataset	Synthetic	Beijing queries	GeoGLUE$_{clean}$ queries
GeoBloom	0.5353	0.4624	0.4350
+ Token Order	**0.9408**	0.4533	0.4332

Table 12: Effect of the Bloom Filter Tree

Search strategy	Beijing		GeoGLUE$_{clean}$	
	NDCG@5	QPS	NDCG@5	QPS
Brute-force	0.4671	16.53	0.3010	1.98
Bloom Filter Tree	0.4730	1418.75	0.3254	191.46

5.4.3 Effect of the Bloom Filter Tree. We investigate the effect of Bloom Filter Tree effectiveness and efficiency in unsupervised settings. Table 12 shows that the Bloom Filter Tree is up to 97 times faster than a brute-force search with similar or even better effectiveness, as the in-beam normalization more accurately identifies relevant POI clusters in remote areas and excludes nearby areas without relevant texts, leading to marginal gains in effectiveness.

5.4.4 Effect of Tokenizers. We analyze how the choice of tokenizers affects the effectiveness of GeoBloom in unsupervised settings. Table 9 shows that the combination of 1-gram, 2-gram, and dictionary-based tokenizers achieves the best Recall@20 and NDCG@5, enhancing the term-matching capability of Bloom filters.

Table 13: Effect of Tokenizers

Tokenizer	Beijing		GeoGLUE$_{clean}$	
	Recall@20	NDCG@5	Recall@20	NDCG@5
1-gram	0.5835	0.4404	0.3831	0.2114
2-gram	0.6180	0.4675	0.4898	0.3046
3-gram	0.4542	0.3689	0.4116	0.2540
Dict. (Jieba)	0.6164	0.4646	0.4463	0.2627
1,2,3-gram	0.6064	0.4594	0.4601	0.2880
1,2-gram+Dict.	**0.6251**	**0.4730**	**0.5204**	**0.3254**

5.4.5 Effect of Beam-widths. We examine how varying the beam width impacts the balance between effectiveness and efficiency. Figure 5 demonstrates that upon reaching a specific threshold (200 for Beijing and 400 for GeoGLUE$_{clean}$), any further increase in beam width only marginally improves effectiveness. Consequently, we can choose a faster speed with a slight decrease in effectiveness.

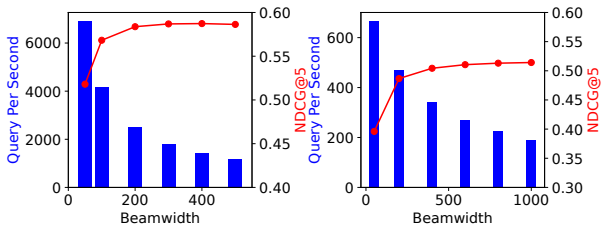


Figure 5: Effect of Beam-widths on Beijing and GeoGLUE$_{clean}$

6 CONCLUSIONS

In conclusion, this paper introduces GeoBloom, a novel framework tailored for the challenges of Geographic Information Retrieval, which equips the unsupervised strengths and the space efficiency of Bloom filters with the power of deep learning. Extensive experiments demonstrate that GeoBloom not only achieves superior effectiveness in both supervised and unsupervised settings but also showcases remarkable time and space efficiency. Future work directions include the adoption of learnable hash functions, advanced Bloom filter variants, sophisticated index structures, and multi-modality data to facilitate all-in-one retrieval platforms.

ACKNOWLEDGMENTS

This research/project is supported by the National Research Foundation, Singapore under its AI Singapore Programme (AISG Award No: AISG2-PhD-2021-08-020[T]) and Singapore MOE AcRF Tier-2 grant MOE-T2EP20223-0004. We would like to thank Shang Liu for providing processed data, and Ziqi Yin for providing the experimental results of baselines DrW, PALM, and LIST [43].

REFERENCES

[1] Amina Adadi. 2021. A survey on data-efficient algorithms in big data era. *J. Big Data* 8, 1 (2021), 1–54.

[2] Leonardo Andrade and Mário J Silva. 2006. Relevance ranking for geographic IR.. In *In Workshop on Geographic Information Retrieval, SIGIR '06*.

[3] Sanjeev Arora, Yingyu Liang, and Tengyu Ma. 2017. A Simple but Tough-to-Beat Baseline for Sentence Embeddings. (2017).

[4] Burton H. Bloom. 1970. Space/time trade-offs in hash coding with allowable errors. *Commun. ACM* 13, 7 (jul 1970), 422–426.

[5] Christopher Burges, Robert Ragno, and Quoc Le. 2006. Learning to Rank with Nonsmooth Cost Functions. In *Advances in Neural Information Processing Systems*, Vol. 19. MIT Press.

[6] Yen-Yu Chen, Torsten Suel, and Alexander Markowetz. 2006. Efficient query processing in geographic web search engines. In *Proceedings of the 2006 ACM SIGMOD international conference on Management of data*. 277–288.

[7] Maria Christoforaki, Jinru He, Constantinos Dimopoulos, Alexander Markowetz, and Torsten Suel. 2011. Text vs. space: efficient geo-search query processing. In *Proceedings of the 20th ACM International Conference on Information and Knowledge Management* (Glasgow, Scotland, UK) *(CIKM '11)*. Association for Computing Machinery, New York, NY, USA, 423–432.

[8] Gao Cong, Christian S. Jensen, and Dingming Wu. 2009. Efficient retrieval of the top-k most relevant spatial web objects. *Proc. VLDB Endow.* 2, 1 (aug 2009), 337–348.

[9] Ian De Felipe, Vagelis Hristidis, and Naphtali Rishe. 2008. Keyword search on spatial databases. In *2008 IEEE 24th International conference on data engineering*. IEEE, 656–665.

[10] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. Association for Computational Linguistics, Minneapolis, Minnesota, 4171–4186.

[11] Ruixue Ding, Boli Chen, Pengjun Xie, Fei Huang, Xin Li, Qiang Zhang, and Yao Xu. 2023. MGeo: A Multi-Modal Geographic Pre-Training Method. *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval* (2023).

[12] Matthijs Douze, Alexandr Guzhva, Chengqi Deng, Jeff Johnson, Gergely Szilvasy, Pierre-Emmanuel Mazaré, Maria Lomeli, Lucas Hosseini, and Hervé Jégou. 2024. The Faiss library. (2024). arXiv:2401.08281 [cs.LG]

[13] Li Fan, Pei Cao, Jussara Almeida, and Andrei Z. Broder. 1998. Summary cache: a scalable wide-area Web cache sharing protocol. *SIGCOMM Comput. Commun. Rev.* 28, 4 (oct 1998), 254–265.

[14] Yao Fu and Mirella Lapata. 2022. Latent Topology Induction for Understanding Contextualized Representations. arXiv:2206.01512 [cs.CL]

[15] Yunpeng Gao, Yao Wang, and Shengwei Yi. 2016. Preference-Aware Top-k Spatio-Textual Queries. In *Web-Age Information Management*. Springer International Publishing, Cham, 186–197.

[16] Bob Goodwin, Michael Hopcroft, Dan Luu, Alex Clemmer, Mihaela Curmei, Sameh Elnikety, and Yuxiong He. 2017. BitFunnel: Revisiting Signatures for Search. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval* (Shinjuku, Tokyo, Japan) *(SIGIR '17)*. Association for Computing Machinery, New York, NY, USA, 605–614.

[17] Jiafeng Guo, Yixing Fan, Qingyao Ai, and W. Bruce Croft. 2016. A Deep Relevance Matching Model for Ad-hoc Retrieval. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management* (Indianapolis, Indiana, USA) *(CIKM '16)*. Association for Computing Machinery, New York, NY, USA, 55–64.

[18] Baotian Hu, Zhengdong Lu, Hang Li, and Qingcai Chen. 2014. Convolutional neural network architectures for matching natural language sentences. In *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2* (Montreal, Canada) *(NIPS'14)*. MIT Press, Cambridge, MA, USA, 2042–2050.

[19] Jizhou Huang, Haifeng Wang, Yibo Sun, Yunsheng Shi, Zhengjie Huang, An Zhuo, and Shikun Feng. 2022. ERNIE-GeoL: A Geography-and-Language Pre-trained Model and its Applications in Baidu Maps. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining* (Washington DC, USA) *(KDD '22)*. Association for Computing Machinery, New York, NY, USA, 3029–3039.

[20] Po-Sen Huang, Xiaodong He, Jianfeng Gao, Li Deng, Alex Acero, and Larry Heck. 2013. Learning deep structured semantic models for web search using clickthrough data. In *Proceedings of the 22nd ACM International Conference on Information & Knowledge Management* (San Francisco, California, USA) *(CIKM '13)*. Association for Computing Machinery, New York, NY, USA, 2333–2338.

[21] Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense Passage Retrieval for Open-Domain Question Answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, Online, 6769–6781.

[22] Tim Kraska, Alex Beutel, Ed H. Chi, Jeffrey Dean, and Neoklis Polyzotis. 2018. The Case for Learned Index Structures. In *Proceedings of the 2018 International Conference on Management of Data* (Houston, TX, USA) *(SIGMOD '18)*. Association for Computing Machinery, New York, NY, USA, 489–504.

[23] Taku Kudo and John Richardson. 2018. SentencePiece: A simple and language independent subword tokenizer and detokenizer for Neural Text Processing. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, EMNLP 2018: System Demonstrations, Brussels, Belgium, October 31 - November 4, 2018*. 66–71.

[24] Dongyang Li, Ruixue Ding, Qiang Zhang, Zheng Li, Boli Chen, Pengjun Xie, Yao Xu, Xin Li, Ning Guo, Fei Huang, and Xiaofeng He. 2023. GeoGLUE: A GeoGraphic Language Understanding Evaluation Benchmark. *CoRR* abs/2305.06545 (2023). arXiv:2305.06545

[25] Zhisheng Li, Ken C. K. Lee, Baihua Zheng, Wang-Chien Lee, Dik Lee, and Xufa Wang. 2011. IR-Tree: An Efficient Index for Geographic Document Search. *IEEE Trans. on Knowl. and Data Eng.* 23, 4 (apr 2011), 585–599.

[26] Shang Liu, Gao Cong, Kaiyu Feng, Wanli Gu, and Fuzheng Zhang. 2023. Effectiveness Perspectives and a Deep Relevance Model for Spatial Keyword Queries. *Proceedings of the ACM on Management of Data* 1, 1 (2023), 1–25.

[27] Thomas Mandl, Paula Carvalho, Giorgio Maria Di Nunzio, Fredric Gey, Ray R Larson, Diana Santos, and Christa Womser-Hacker. 2009. GeoCLEF 2008: The CLEF 2008 cross-language geographic information retrieval track overview. In *Evaluating Systems for Multilingual and Multimodal Information Access: 9th Workshop of the Cross-Language Evaluation Forum, CLEF 2008, Aarhus, Denmark, September 17-19, 2008, Revised Selected Papers 9*. Springer, 808–821.

[28] Bruno Martins, Mário J. Silva, and Leonardo Andrade. 2005. Indexing and ranking in Geo-IR systems. In *Proceedings of the 2005 Workshop on Geographic Information Retrieval* (Bremen, Germany) *(GIR '05)*. Association for Computing Machinery, New York, NY, USA, 31–34.

[29] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed Representations of Words and Phrases and their Compositionality. In *Advances in Neural Information Processing Systems*, Vol. 26. Curran Associates, Inc.

[30] Michael Mitzenmacher. 2018. A Model for Learned Bloom Filters and Optimizing by Sandwiching. In *Neural Information Processing Systems*.

[31] Yu Nasu. 2018. Efficiently updatable neural-network-based evaluation functions for computer shogi. *The 28th World Computer Shogi Championship Appeal Document* 185 (2018).

[32] Paolo Palmieri, Luca Calderoni, and Dario Maio. 2015. Spatial Bloom Filters: Enabling Privacy in Location-Aware Applications. In *Information Security and Cryptology*. Springer International Publishing, Cham, 16–36.

[33] Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. GloVe: Global Vectors for Word Representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, Doha, Qatar, 1532–1543.

[34] Ely Porat. 2009. An Optimal Bloom Filter Replacement Based on Matrix Solving. In *Computer Science - Theory and Applications*. Springer Berlin Heidelberg, Berlin, Heidelberg, 263–273.

[35] Ross S Purves, Paul Clough, Christopher B Jones, Mark H Hall, Vanessa Murdock, et al. 2018. Geographic information retrieval: Progress and challenges in spatial search of text. *Foundations and Trends® in Information Retrieval* 12, 2-3 (2018), 164–318.

[36] S. E. Robertson and S. Walker. 1994. Some Simple Effective Approximations to the 2-Poisson Model for Probabilistic Weighted Retrieval. In *SIGIR '94*. Springer London, London, 232–241.

[37] João B. Rocha-Junior, Orestis Gkorgkas, Simon Jonassen, and Kjetil Nørvåg. 2011. Efficient Processing of Top-k Spatial Keyword Queries. In *Advances in Spatial and Temporal Databases*. Springer Berlin Heidelberg, Berlin, Heidelberg, 205–222.

[38] Angela Schwering. 2008. Approaches to semantic similarity measurement for geo-spatial data: a survey. *Transactions in GIS* 12, 1 (2008), 5–29.

[39] Yelong Shen, Xiaodong He, Jianfeng Gao, Li Deng, and Grégoire Mesnil. 2014. A Latent Semantic Model with Convolutional-Pooling Structure for Information Retrieval. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management* (Shanghai, China) *(CIKM '14)*. Association for Computing Machinery, New York, NY, USA, 101–110.

[40] Karen Sparck Jones. 1988. *A statistical interpretation of term specificity and its application in retrieval*. Taylor Graham Publishing, GBR, 132–142.

[41] Zengjie Wang, Wen Luo, Linwang Yuan, Hong Gao, Fan Wu, Xu Hu, and Zhaoyuan Yu. 2021. Query the trajectory based on the precise track: a Bloom filter-based approach. *GeoInformatica* 25, 2 (01 Apr 2021), 397–416.

[42] Mingyang Yang, Long Zheng, Yanchao Lu, Minyi Guo, and Jie Li. 2015. Cloud-assisted spatio-textual k nearest neighbor joins in sensor networks. In *2015 1st International Conference on Industrial Networks and Intelligent Systems (INISCom)*. IEEE, 12–17.

[43] Ziqi Yin, Shanshan Feng, Shang Liu, Gao Cong, Yew Soon Ong, and Bin Cui. 2024. LIST: Learning to Index Spatio-Textual Data for Embedding based Spatial Keyword Queries. arXiv:2403.07331 [cs.IR]

[44] Zixuan Yuan, Hao Liu, Yanchi Liu, Denghui Zhang, Fei Yi, Nengjun Zhu, and Hui Xiong. 2020. Spatio-Temporal Dual Graph Attention Network for Query-POI Matching. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval* (Virtual Event, China) *(SIGIR '20)*. Association for Computing Machinery, New York, NY, USA, 629–638.

[45] Meng Zeng, Beiji Zou, Xiaoyan Kui, Chengzhang Zhu, Ling Xiao, Zhi Chen, and Jingyu Du. 2023. PA-LBF: Prefix-Based and Adaptive Learned Bloom Filter for Spatial Data. *International Journal of Intelligent Systems* 2023 (29 Mar 2023), 4970776.

[46] Zhengyan Zhang, Xu Han, Zhiyuan Liu, Xin Jiang, Maosong Sun, and Qun Liu. 2019. ERNIE: Enhanced Language Representation with Informative Entities. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, Florence, Italy, 1441–1451.

[47] Ji Zhao, Dan Peng, Chuhan Wu, Huan Chen, Meiyu Yu, Wanji Zheng, Li Ma, Hua Chai, Jieping Ye, and Xiaohu Qie. 2019. Incorporating Semantic Similarity with Geographic Correlation for Query-POI Relevance Learning. *Proceedings of the AAAI Conference on Artificial Intelligence* 33, 01 (Jul. 2019), 1270–1277.