Advancing Spatial Keyword Queries: From Filters to Unified Vector Embeddings

Kaan Gocmen, Guanli Liu(✉), and Renata Borovica-Gajic

The University of Melbourne, Parkville VIC 3010, Australia
`kgocmen@student.unimelb.edu.au`
`{guanli.liu1,renata.borovica}@unimelb.edu.au`

Abstract. Effective spatial keyword queries require joint modelling of location constraints and textual semantics. Existing systems typically rely either on spatial indexes with keyword filters or on semantic embeddings, which treat the two dimensions separately and thus lead to rigid spatial boundaries and limited semantic flexibility. To better understand these limitations, we systematically evaluate six representative approaches on large-scale OpenStreetMap (OSM) data from Greater Melbourne. The results show that non-embedding methods deliver exact matches but fail to generalise semantically, while embedding-based methods broaden coverage but capture spatial context poorly. This trade-off is not inevitable. We contribute two fusion strategies, namely concatenation embedding and contrastive fusion, that map spatial coordinates and textual descriptions into a shared vector space, and we integrate them with PostgreSQL, PostGIS and pgvector for evaluation alongside traditional baselines. Experiments demonstrate that fused embeddings improve semantic recall while preserving spatial fidelity, and they remain stable as query intent shifts between location and meaning. These findings indicate that unified spatial-semantic embeddings provide a practical direction for advancing spatial keyword queries.

Keywords: Spatial embedding · Semantic embedding · Hybrid vector search · Contrastive fusion.

1 Introduction

Location-based applications such as Google Maps [12], Foursquare [9], and TripAdvisor [26] rely heavily on geospatial data enriched with semantic annotations such as place descriptions and user-generated tags. To meet user needs effectively, spatial keyword queries must consider both the location and the description of the place, retrieving results that are geographically close and semantically relevant. Traditional approaches handle these components separately, combining spatial indexes with keyword filters. While efficient in simple cases, this separation imposes strict distance boundaries, ignores synonyms and paraphrases, and fails to capture the interaction between spatial and semantic dimensions. As a result, user intent is often only partially reflected in query results.

Recent advances in deep learning and vector databases offer a way forward [20]. Pre-trained language models can generate embeddings that capture semantic similarity, while tools like pgvector [19] enable efficient high-dimensional similarity search within relational databases [23]. This creates the opportunity to embed spatial coordinates and textual attributes together in a unified vector space, supporting retrieval that is both semantically rich and spatially aware.

Embedding-based approaches also broaden the kinds of spatial queries that become feasible. Instead of restricting results to an exact tag such as `cafe`, embeddings can capture related concepts like "coffee shop," "espresso bar," and "quiet place to work," retrieving semantically similar POIs even if their tags differ. Likewise, instead of applying a hard radius filter that excludes slightly more distant but highly relevant places, embeddings model proximity as a continuous similarity, allowing smoother trade-offs between distance and semantics. This enables queries such as *"family-friendly restaurants near the river"*, *"historic buildings around the city centre"*, or *"pharmacies close to hospitals"*, which require nuanced reasoning over both meaning and location.

In this paper, we aim to investigate the following question: *How can spatial and semantic information be embedded into a unified vector space to support efficient and high-quality spatial keyword search?* We address this question by benchmarking fused embedding methods against existing approaches using PostgreSQL, PostGIS, and pgvector on real-world datasets.

This paper makes three main contributions: (i) a systematic comparison of spatial, keyword, and embedding-based methods within PostgreSQL; (ii) the design of fused spatial-semantic embeddings via concatenation and contrastive fusion; (iii) a comprehensive evaluation showing fused methods achieve higher semantic recall without sacrificing efficiency.

The remainder of this paper is organised as follows: Section 2 reviews prior research on spatial keyword indexing and embedding-based retrieval. Section 3 details the indexing and querying strategies considered in this study, spanning both non-embedded and embedded approaches. Section 4 presents the experimental setup and results. Finally, Section 5 summarises our findings and outlines directions for future research.

2 Related Work

This section reviews the related work on spatial keyword indexing and embedding-based retrieval.

Index-based methods. The first spatial keyword query systems relied heavily on spatial indexing structures such as the R-Tree, which efficiently organises geographic data [10, 13]. Several important variants were subsequently developed, including the R*-Tree, which refines node splitting and re-insertion to reduce overlap [2]. X-Tree introduces supernodes to handle high-dimensional data and combat the curse of dimensionality [3]. To support both spatial and textual constraints, hybrid geo-textual index structures were developed. The IR-Tree combines each R-Tree node with inverted keyword lists to support efficient top-k spatial keyword searches [4,21]. The IR^2-Tree replaces inverted lists with compact signature files, reducing index size and improving query performance [6]. Other variants include the IF-R* tree, which integrates inverted files with R*-Tree structures [29]. Comprehensive surveys of geo-textual indexing provide taxonomies of these approaches and evaluate trade-offs between index size, update cost, and query efficiency [4,10]. In addition to individual index structures, research has also identified common spatial keyword query types, such as Boolean range queries, k-nearest neighbour (kNN) searches, and top-k ranking, and shown how they can be processed using spatial and textual indexes [5]. Göbel et al. [14] proposed an integrated hybrid index that combines R-Trees with inverted files and keyword summary bitmaps, enabling better pruning in both spatial and textual dimensions. In practical systems such as PostgreSQL with PostGIS, these ideas [22] are realised using the Generalized Search Tree (GIST) [15] and Generalized Inverted Index (GIN) [7,18]. GIST provides an extensible framework that supports R-Tree variants for spatial filtering [15], while GIN efficiently indexes composite and multi-valued attributes such as JSONB or tokens [7,18]. Used together, they enable a sequential yet efficient strategy where spatial filtering is applied via GIST and keyword filtering via GIN, forming a strong baseline in DBMS-based spatial keyword search [23,25].

Embedding methods. With growing demand for both spatial and semantic relevance, research has moved from tree-based indexes to hybrid and embedding-based models. Alfarrarjeh et al. combined an R*-Tree with Locality Sensitive Hashing for spatial-visual search [1], while Zhu et al. benchmarked systems such as Milvus, pgvector, ClickHouse, and Elasticsearch on datasets mixing structured attributes with embeddings, highlighting trade-offs in speed and recall [30]. WISK [24] further advances this area by learning workload-aware partitions and building specialised indexes. Embedding models now play a central role in semantic search by representing text and context as vectors stored in similarity-optimised databases. GeoBERT [11] learns geospatial representations from POIs within Geohash grids, while CaLLiPer [27], trained on London, aligns text, location, and POI types via contrastive learning. RegionEncoder [16], built for New York, integrates POIs, mobility, imagery, and spatial graphs into compact embeddings.

Benchmarking. Broader surveys [20] also link text models (Word2Vec, GloVe, FastText) to modern vector databases like Milvus, Weaviate, and pgvector, which rely on Approximate Nearest Neighbour (ANN) indexes such as HNSW and IVF-PQ. Recent work improves HNSW with memory-efficient layouts and dynamic rebalancing, making it suitable for large, frequently updated datasets [8,28]. These developments underpin semantically aware spatial search that extends beyond exact keyword matching, enabling systems to answer vague queries such as "quiet cafe with WiFi." Zhu et al. [30] proposed a comprehensive benchmarking framework, introducing the STFD and MTMD datasets to evaluate hybrid query processing. Their results highlighted trade-offs: Milvus achieved the fastest vector retrieval but sometimes at lower recall; pgvector was slower but integrated tightly with PostgreSQL; ClickHouse delivered high precision at reduced throughput; and Elasticsearch offered a balance between speed and accuracy. Earlier work by Chen et al. [4] benchmarked IR-Tree variants, establishing evaluation metrics and trade-offs for traditional geo-textual indexes. Building on these efforts, our study adopts PostgreSQL with PostGIS and pgvector to benchmark both traditional and embedding-based methods in a unified environment, reflecting the combined spatial and semantic requirements of real-world applications [23, 25].

3 Methodology

Our methodology is built around a PostgreSQL-based framework that integrates spatial, textual, and semantic indexing for spatial keyword search. This framework underpins our fused embedding approaches and allows consistent comparison with established baselines. We first describe the indexing components that support efficient querying, then introduce our unified spatial-semantic embedding strategies.

3.1 Indexing Components

We rely on three primary indexing components that enable efficient query execution across spatial, textual, and semantic dimensions. For spatial operations, we employ a GIST (Generalized Search Tree) index on geometry, which PostGIS implements with an R-Tree variant [15]. This supports predicates such as `ST_DWithin` and `ST_Distance` that prune candidate sets rapidly with near-logarithmic access. For keyword filtering, we build a GIN (Generalized Inverted Index) on the POI descriptions stored in JSONB [7, 18], enabling fast exact-match lookups on frequent attributes (e.g., `amenity`, `highway`). For semantic similarity search, we use vector indexes from the `pgvector` extension [20]. We primarily evaluate (i) HNSW (Hierarchical Navigable Small World), a graph-based index optimised for high recall and low-latency kNN [8, 28]; and (ii) IVF-Flat (Inverted File with Flat Structure), which partitions the embedding space into clusters to reduce memory and build time at some accuracy cost [17]. Each indexing structure is responsible for a distinct aspect of spatial keyword search, spanning spatial locality, textual information, and semantic similarity.

3.2 Fused Embeddings: SE-KE (Spatial & Keyword Embeddings)

To overcome the limitations of existing methods, we propose SE-KE, which jointly encodes spatial coordinates and semantic tags into a unified vector representation. In this study, two SE-KE variants are implemented: *Embedding Concatenation* and *Contrastive Fusion*.

Embedding Concatenation. Let $v \in \mathbb{R}^d$ be the semantic embedding of an item, and let $s \in \mathbb{R}^m$ be a spatial embedding. We fuse them by *repeating* the

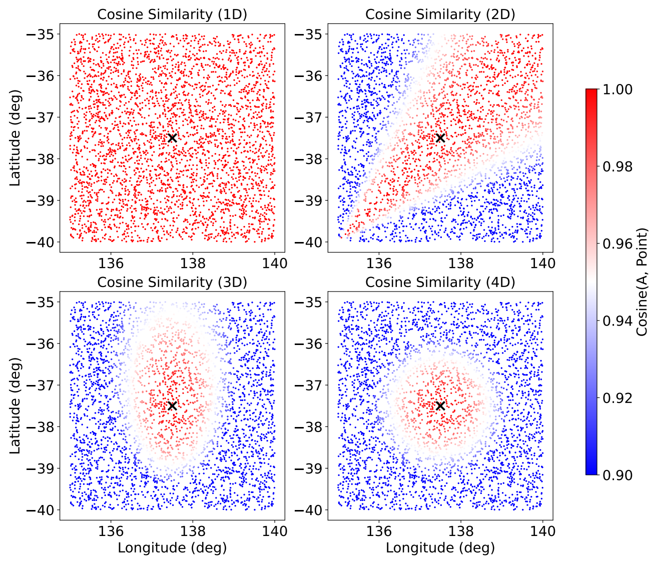

Fig. 1: Cosine similarity decay under increasing spatial encoding dimensions, shown by cosine similarity to $A = (-37.5, 137.5)$ under four spatial encodings.

spatial block λ times and then concatenating with v:

$$u \;=\; \Big[\, \underbrace{s\,;\; s\,;\; \ldots\,;\; s}_{\lambda\ \text{copies}}\,;\; v \,\Big] \;\in\; \mathbb{R}^{\lambda m + d} \tag{1}$$

where $\lambda \in \mathbb{Z}^+$ controls the relative weight of the spatial part under cosine (after normalisation) and, accordingly, changes the dimensionality from $m+d$ to $\lambda m+d$. If we encode spatial position only with signed *2D* latitude/longitude offsets in the range $[-1, 1] \times [-1, 1]$, cosine similarity behaves abnormally: two points that are equally close to the centre (the zero vector) but lie on opposite sides will appear maximally dissimilar. In other words, cosine is dominated by the *direction* of the offset rather than its *magnitude*, so similarity does not decrease smoothly with increasing distance. To address this, each axis is split into nonnegative half–axes: north, south, east, and west. With this representation, all points are mapped to positive vectors of equivalent magnitude, so that moving away from the anchor in any cardinal direction just rotates the vector away from the anchor's code. The resulting *4D* encoding (N, E, S, W) prevents cancellations and produces similarity fields that decay monotonically with distance. Figure 1 shows how the similarity field evolves from *1D* to *4D*: *1D* is uniform, *2D* creates wedge-shaped similarities, *3D* localises better but distorts into an oval shape towards the missing direction. *4D* yields a perfect circle around the point.

We therefore rewrite our definition as:

$$u \;=\; \Big[\, \underbrace{[N, S, E, W]}_{\lambda\ \text{copies}}\,;\; v \,\Big] \;\in\; \mathbb{R}^{4\lambda + d}, \tag{2}$$

with the *4D* spatial code chosen because it is the minimal setting that makes cosine behave like a distance proxy. The idea is to scale latitude-longitude range within the chosen bounding box to $[0, 1]$, and correct the east-west scale by a cosine factor because 1° of longitude shrinks by $\cos(\varphi)$ with latitude. Let the bounding box be $[\varphi_{\min}, \varphi_{\max}] \times [\lambda_{\min}, \lambda_{\max}]$ with latitude $\varphi \in (-90^\circ, 90^\circ)$ and longitude $\lambda \in (-180^\circ, 180^\circ)$

$$\begin{aligned}
\Delta\varphi &= \varphi_{\max} - \varphi_{\min}, \\
\Delta\lambda &= (\lambda_{\max} - \lambda_{\min}) \cos\left(\tfrac{\varphi_{\max}+\varphi_{\min}}{2}\right), \\
c(\varphi) &= \cos\left(\tfrac{\pi\varphi}{180}\right), \\
N(\varphi) &= \tfrac{\varphi_{\max}-\varphi}{\Delta\varphi}, \quad S(\varphi) = \tfrac{\varphi-\varphi_{\min}}{\Delta\varphi}, \\
E(\varphi, \lambda) &= 1 - c(\varphi)\tfrac{\lambda-\lambda_{\min}}{\Delta\lambda}, \quad W(\varphi, \lambda) = 1 - c(\varphi)\tfrac{\lambda_{\max}-\lambda}{\Delta\lambda}.
\end{aligned}$$

To combine this with a d–dimensional pre-normalised semantic vector v_x for item x, we build the spatial code $s_x = [N, S, E, W]$, normalise, scale it by λ, and concatenate with v_x. After concatenation, another normalisation is required.

$$u_x = \frac{[\,\lambda\,\hat{s}_x\ ;\ \hat{v}_x\,]}{\|[\,\lambda\,\hat{s}_x\ ;\ \hat{v}_x]\|_2}, \qquad \hat{s}_x = \frac{s_x}{\|s_x\|_2} \tag{3}$$

At query time, build u_q the same way from the query text and anchor location (either taken from the query or inferred by other means), and rank items by the cosine (dot product) $\langle u_q, u_x \rangle$. This yields a single score that includes proximity and meaning components.

Contrastive Fusion. In this approach, contrastive learning is used to align spatial and textual modalities into a common embedding space. For each POI, two complementary views are generated: (i) a semantic embedding $z^{(t)} \in \mathbb{R}^p$ from its tags, and (ii) a spatial embedding $z^{(s)} \in \mathbb{R}^p$ from its coordinates, where p denotes the latent dimension of the shared embedding space. Positive pairs are constructed by matching the text and location of the same POI (same item), while negative pairs are obtained by mismatching different POIs. The model is trained to pull positive pairs closer together and push negative pairs apart. This design is inspired by recent work on multimodal contrastive learning of urban space representations from POI data [27], which demonstrated the effectiveness of aligning geographic and textual modalities through a contrastive objective. The semantic view uses a pre-trained transformer encoder (all-MiniLM-L6-v2), kept frozen to preserve stable semantic representations. The spatial view encodes longitude-latitude coordinates (λ, ϕ) with a sinusoidal scheme similar to transformer positional encodings, which capture spatial patterns at multiple scales:

$$\gamma(\lambda, \phi) = \left[\sin(\lambda \cdot \alpha_k), \cos(\lambda \cdot \alpha_k), \sin(\phi \cdot \alpha_k), \cos(\phi \cdot \alpha_k)\right]_{k=1}^{K},$$

where $\alpha_k = 10000^{-\frac{2k}{p}}$ defines the frequency scales. A feed-forward projection then maps $\gamma(\lambda, \phi)$ into the latent space $z^{(s)}$. Training employs a symmetric InfoNCE loss, applied bidirectionally between $z^{(t)}$ and $z^{(s)}$, using cosine similarity with a learnable temperature τ. This forces text and location views of the same place to lie close in the embedding space, enabling cross-modal retrieval. At inference, the two views are linearly combined to produce a fused representation:

$$u_x = norm\big(w_{\text{text}}\, z_x^{(t)} + w_{\text{spatial}}\, z_x^{(s)}\big),$$

where $w_{\text{text}} \geq 0,\ w_{\text{spatial}} \geq 0$ are modality weights. Since u_x is ℓ_2-normalised, only their ratio $\rho = \frac{w_{\text{text}}}{w_{\text{spatial}}}$ matters: larger ρ emphasises semantic similarity, while smaller ρ emphasises spatial proximity. Queries are encoded in the same way and compared via cosine similarity, allowing flexible trade-offs between meaning and distance.

4 Experiments

4.1 Experimental Setup

Environment. All experiments are executed on a controlled and reproducible computing environment provided by the University of Melbourne's Research Computing Portal (RCP). The virtual machine is hosted on an OpenStack Foundation node equipped with an AMD® EPYC™ 9474F processor (48 cores, 96 threads), 64 GB of RAM, and 200 GB of storage. It runs `Ubuntu 22.04.5 LTS` and uses `PostgreSQL 15`, extended with `PostGIS 3.3` for spatial operations and `pgvector` for vector similarity search. Semantic embeddings are generated using the `sentence-transformers` library (`all-MiniLM-L6-v2`, 384-dimensional) and indexed with either HNSW or IVFFlat. The contrastive fusion model is trained for seven epochs with a learning rate of 1×10^{-4} and weight decay of 0.01, testing projection dimensions $p \in \{128, 256, 384\}$.
Baseline Methods. We benchmark six established approaches that combine spatial filtering, keyword filtering, and semantic embedding in different ways, providing critical baselines for comparison.

1. *SCAN* performs a sequential pass over all n records without index support. It has $O(n)$ complexity but no storage overhead, serving as a worst-case baseline [4].
2. *SF (Spatial Filtering)* uses a GIST index on geometry, which implements an R-Tree [10, 15], reducing candidates with queries such as `ST_DWithin` in $O(\log n + k)$ time, but without semantic awareness.
3. *KF (Keyword Filtering)* builds a GIN index [7,18] over JSONB tags, enabling efficient exact-match lookups (e.g., `amenity`, `highway`) in $O(\log n)$ time, but ignoring spatial context.
4. *SF-KF* combines the two, applying GIST and GIN jointly so results are geographically bounded and keyword-matched, though rigid and semantically limited [5, 14, 23].
5. *KE (Keyword Embedding)* encodes tags into d-dimensional vectors (MiniLM-L6-v2, $d = 384$), stored in pgvector and indexed with HNSW/IVFFlat. Queries are embedded likewise, enabling semantic similarity search but with no spatial notion [8, 11, 16, 20, 27, 28].

6. *SF-KE* first applies GIST filtering, then ranks survivors by embedding similarity via HNSW/IVFFlat, offering semantic flexibility but still treating spatial and semantic signals separately [1, 24, 30].

These six methods span the main design choices in spatial keyword search, letting us isolate the impact of each indexing or embedding component. Our artifacts are publicly available.[1]

Table 1: Overview of indexing components for baseline methods and proposed SE-KE (see Section 3.2).

Method Name	**Keyword Index** *(GIN)*	**Spatial Index** *(GIST)*	**Keyword Embedding** *(HNSW/IVFFlat)*	**Spatial Embedding** *(HNSW/IVFFlat)*
SCAN				
SF		✓		
KF	✓			
SF-KF	✓	✓		
KE			✓	
SF-KE		✓	✓	
SE-KE			✓	✓

Dataset. The experimental data is derived from OpenStreetMap (OSM). Our main dataset corresponds to Greater Melbourne ($[-38.50, -37.22] \times [144.29, 146.46]$), containing 1.96M records after discarding entries without tags. While the majority of records represent residential addresses, the dataset also includes a substantial number of points of interest (POIs). To evaluate scalability, we additionally use a 100k sample from Melbourne, as well as two further regions: Istanbul, a denser yet geographically more compact city; and Çukurova, a region spanning multiple smaller cities, which is broader but sparser. These datasets provide diverse conditions of density, scale, and semantic richness, enabling us to evaluate robustness under real-world OSM noise and heterogeneity.

Variables. We control a small set of variables to probe accuracy, efficiency, and robustness. Each factor is fixed unless stated, so differences reflect the retrieval method rather than tuning.

1. **Top-k (k).** We report k=10 by default because it stresses precision at the top while remaining comparable across methods. Embedding methods always return exactly k neighbours. Filter-based methods may return fewer if keyword or spatial predicates prune candidates.
2. **Query radius (r, in metres).** Spatial filters use `ST_DWithin` with radius r. Setting r=0 removes distance constraints, so every record is eligible and ranking relies on textual or vector similarity.

[1] Repository: `https://github.com/kgocmen/Advancing-SK-Queries-Benchmark`

3. **Vector index.** We use HNSW (m=16, `ef_construction`=64) and IVF-Flat (`nlist`=300). Keeping these fixed avoids conflating gains from indexing knobs with gains from the embedding strategy.
4. **Query mode.** *Custom* accepts external or free-text queries and infers coordinates when needed. *Dataset* samples coordinates and extracts keywords (e.g., `name`, `addr:street`) to simulate realistic but controlled workloads.
5. **Concatenation weight (λ).** We repeat the spatial vector before concatenation to adjust the balance between proximity and semantics. Larger λ increases spatial influence; smaller values emphasise textual similarity.
6. **Contrastive fusion settings (p, ρ).** We vary embedding dimension $p \in \{128, 256, 384\}$ and fusion ratio $\rho = \frac{w_{\text{text}}}{w_{\text{spatial}}}$ to study how capacity and modality weighting affect retrieval.

Collectively, these choices define the experimental space in which we compare indexing methods and fusion strategies on speed, scalability, and result quality.
Metrics. To measure efficiency, we report *insertion time*, defined as the total duration required to load the dataset into the database. It only accounts for the raw insertion process itself; preprocessing tasks such as parsing the input files or generating semantic embeddings are not included in this measurement. For embedding-based methods, an additional cost arises from generating semantic embeddings for the tags. *Embedding time* measures the total duration required to transform all textual tag data into vector representations of dimension d. *Index creation time* measures the duration required to build the auxiliary data structures that enable efficient querying. *Query execution time* refers to the latency incurred when answering a query using the constructed indexes. We also measure the average execution time across all queries. To assess result quality, we use three metrics. First, *semantic coherence* measures how semantically consistent the top-k results are with one another. High values indicate that results describe similar concepts, while low values indicate semantic diversity or mismatch. Second, *spatial range* evaluates geographic compactness by reporting the distance from the query point to the furthest retrieved item among the top-k. Finally, recall is inspected qualitatively by eye, providing a sanity check on whether relevant items appear in the retrieved lists.

4.2 Experimental Results and Discussion

We first evaluate the non-embedding methods: SCAN, SF (spatial filtering), KF (keyword filtering), and SF-KF (combined). Since these methods cannot process custom semantic queries, the dataset mode is used. As shown in Figure 2, insertion times are similar across all methods, while index creation introduces moderate overheads from GIST (SF) and GIN (KF), which add up in SF-KF. For query execution, SF is efficient at small radii but loses performance at larger ones where most points are included. KF is unaffected by radius, depending mainly on tag selectivity. The combined SF-KF method inherits the behaviour of both filters, performing best when spatial and textual conditions are selective, yet still showing the same efficiency loss as SF at large radii.

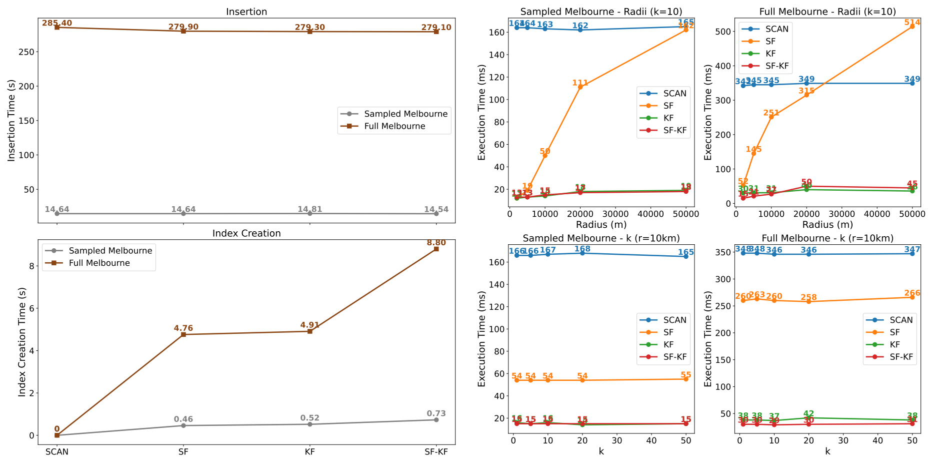

Fig. 2: Timing results for non-embedding methods. Left: insertion and index creation; Right: execution behaviour across radii and k values.

Next, we introduce the semantic embedding methods (KE, SF-KE) and compare them with the non-embedded versions (SCAN, SF-KF) using dataset-derived queries, as shown in Figure 3. The non-embedded methods are faster in insertion, index creation, and query execution, but they are outperformed by the embedding-based approaches in retrieval quality across all ranges, especially at low radii, where they often return no results. Spatial filters remain unreliable in execution time, even when combined with embedding-based methods. This is evident in SF-KE, which showed slightly higher execution times than KE at larger radii due to the additional spatial filtering step. Embedding-based methods also support custom semantic queries, unlike non-embedded ones, which are limited to exact keyword matching and become slower at large radii.

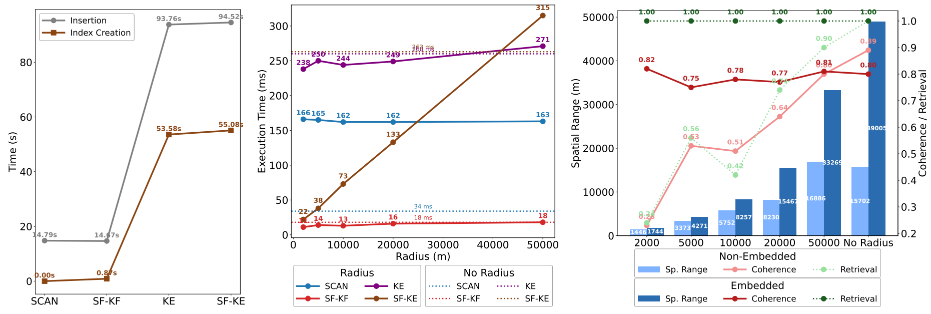

Fig. 3: Comparison of non-embedded (SCAN, SF-KF) and embedding-based (KE, SF-KE) methods using the sampled Melbourne dataset.

To determine SE-KE hyperparameters, we evaluate the trade-off between retrieval quality and efficiency using the full Melbourne dataset. Insertion for the 1.96 million record dataset requires approximately 1800 seconds across all configurations. Index creation is faster with IVFFlat, while query execution times are nearly identical between HNSW and IVFFlat, averaging around one to two seconds per query. Under the fixed settings, both index types deliver comparable retrieval quality, whereas the fused methods consistently achieved higher semantic accuracy than existing embedding approaches. HNSW is ultimately preferred because it allows dynamic insertions and deletions, offering a long-term advantage as it does not require full reindexing when the dataset changes. For hyperparameter selection, both fusion models are tested on the Melbourne, Istanbul, and Çukurova datasets to identify a balanced configuration. Evaluation shows that $\lambda = 2$ for the Concatenation model and $\rho = 5$, $p = 384$ for the Contrastive model achieved the best balance between locality and semantic coherence.

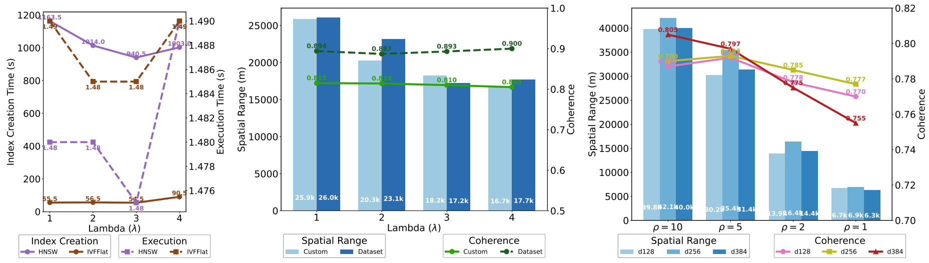

Fig. 4: Evaluation of SE-KE configurations. Left: index creation and execution time of concatenation embedding; Middle: retrieval quality of concatenation embedding; Right: retrieval quality of contrastive learning.

Finally, we compare the best-performing settings: $r = 20000$ for SF-KE, $\lambda = 2$ for Concatenation Embedding, and $p = 384$, $\rho = 5$ for Contrastive Fusion. In addition to the sampled Melbourne dataset, we used two other datasets: Istanbul, a geographically smaller urban area containing only points of interest (no residential address records), and Çukurova, which spans multiple cities separated by wide rural zones. The results in Figure 5 reveal complementary strengths among the methods. SE-KE (Concat) consistently achieves the highest coherence values, confirming that combining spatial and semantic components within a unified vector space yields the most accurate retrieval behaviour. Its performance remains stable across different spatial densities, showing that $\lambda = 2$ offers a well-balanced trade-off between locality and meaning. SE-KE (Contrast) produces slightly lower peak scores but generalised more effectively across datasets of varying scales and densities. By learning a shared representation of text and coordinates, it adapts better to unseen regions such as Çukurova, where spatial cues are less consistent. SF-KE remains competitive in dense urban datasets like full Melbourne, where spatial filtering efficiently prunes the candidate space, but its performance degraded in large, sparse regions due to reduced selectivity and inconsistent latency. Overall, the fused embedding approaches overcome these limitations by integrating spatial and semantic similarity within a single retrieval process, achieving higher recall without compromising efficiency.

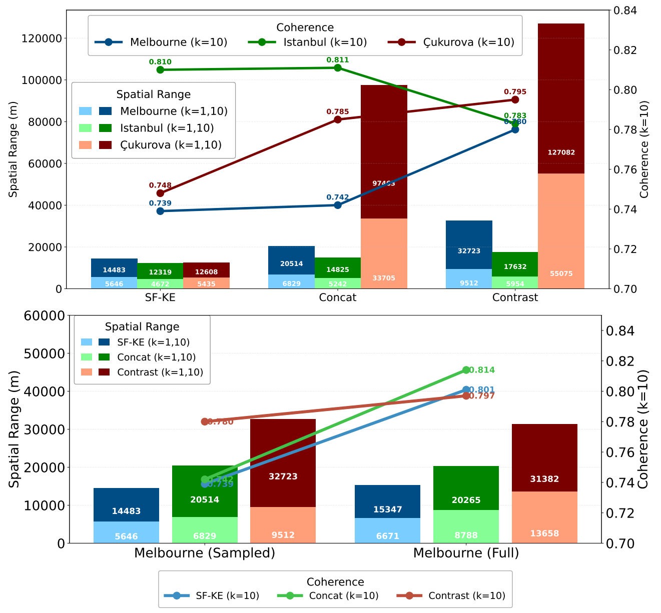

Fig. 5: Spatial and semantic performance of fused embedding methods across three regions (Melbourne, Istanbul, Çukurova) and two dataset scales (sampled and full Melbourne).

5 Conclusions and Future Work

We study unifying spatial and semantic signals in one vector space for spatial keyword queries. On Greater Melbourne, separate spatial filters with keyword indexes show rigid boundaries and exact-match limitations. Fused embeddings improve retrieval by aligning proximity with meaning, with most overhead confined to offline embedding generation and ingestion. Among fusion strategies, simple concatenation is highly accurate and often exceeds hybrid baselines such as SF-KE, but it requires tuning the λ weight between spatial and semantic components. Contrastive fusion is initially less accurate but more stable across datasets, indicating room for gains with stronger training.

In the future, we will pursue adaptive retrieval that balances spatial and semantic signals based on query intent and geographic scale. We will also develop geospatial foundation models to capture place semantics beyond general-purpose text embeddings. Finally, we will focus on large-scale system integration, including new index structures, cost models, and robust pipelines that maintain performance over time, aiming to make spatial semantic vector querying a core capability of database systems.

6 Acknowledgments

We gratefully acknowledge support from the Australian Research Council Discovery Early Career Researcher Award DE230100366.

References

1. Alfarrarjeh, A., Shahabi, C., Kim, S.H.: Hybrid Indexes for Spatial-Visual Search. In: Proceedings of the on Thematic Workshops of ACM Multimedia. p. 75–83 (2017)
2. Beckmann, N., Kriegel, H.P., Schneider, R., Seeger, B.: The R*-tree: an efficient and robust access method for points and rectangles. SIGMOD Rec. **19**(2), 322–331 (1990)
3. Berchtold, S., Keim, D.A., Kriegel, H.P.: The X-tree: an index structure for high-dimensional data, p. 451–462 (2001)
4. Chen, L., Cong, G., Jensen, C., Wu, D.: Spatial keyword query processing: An experimental evaluation. vol. 6, pp. 217–228 (2013)
5. Chen, Z., Chen, L., Cong, G., Jensen, C.S.: Location- and keyword-based querying of geo-textual data: a survey. VLDBJ **30**(4), 603–640 (2021)
6. De Felipe, I., Hristidis, V., Rishe, N.: Keyword Search on Spatial Databases. In: ICDE. pp. 656–665 (2008)
7. Fittl, L.: Understanding Postgres GIN Indexes: The Good and the Bad (2021), `https://pganalyze.com/blog/gin-index`
8. Foster, C., Kimia, B.: Computational Enhancements of HNSW Targeted to Very Large Datasets, pp. 291–299 (10 2023)
9. Foursquare Labs Inc.: Foursquare. `https://foursquare.com`
10. Gaede, V., Günther, O.: Multidimensional Access Methods (1999)
11. Gao, Y., Xiong, Y., Wang, S., Wang, H.: GeoBERT: Pre-Training Geospatial Representation Learning on Point-of-Interest. Applied Sciences **12**, 12942 (2022)
12. Google LLC: Google maps. `https://maps.google.com`
13. Guttman, A.: R-trees: a dynamic index structure for spatial searching. In: SIGMOD. p. 47–57 (1984)
14. Göbel, R., Henrich, A., Niemann, R., Blank, D.: A Hybrid Index Structure for Geo-Textual Searches. In: CIKM. p. 1625–1628 (2009)
15. Hellerstein, J.M., Naughton, J.F., Pfeffer, A.: Generalized search trees for database systems (1995)
16. Jenkins, P., Farag, A., Wang, S., Li, Z.: Unsupervised Representation Learning of Spatial Data via Multimodal Embedding. In: CIKM. pp. 1993–2002 (2019)
17. Johnson, J., Douze, M., Jégou, H.: Billion-scale similarity search with gpus. In: Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). pp. 2617–2621 (2017)
18. Kanchan, D., Paikrao, R.: Survey paper on Generalized Inverted Index for Keyword Search. International Journal of Engineering Research and Development **10**(4), 69–73 (2014)
19. Kane, A.: pgvector: Open-source vector similarity search for postgresql (2024)
20. Kukreja, S., Kumar, T., Bharate, V., Purohit, A., Dasgupta, A., Guha, D.: Vector Databases and Vector Embeddings-Review. pp. 231–236 (12 2023)
21. Li, Z., Lee, K.C., Zheng, B., Lee, W.C., Lee, D., Wang, X.: IR-Tree: An Efficient Index for Geographic Document Search. TKDE **23**(4), 585–599 (2011)
22. PostGIS Project: Introduction to PostGIS Indexing (2025), `https://postgis.net/workshops/postgis-intro/indexing.html`
23. Sakr, M., Vaisman, A., Zimanyi, E.: Query Processing and Indexing, pp. 187–246 (2025)
24. Sheng, Y., Cao, X., Fang, Y., Zhao, K., Qi, J., Cong, G., Zhang, W.: WISK: A Workload-aware Learned Index for Spatial Keyword Queries. PACMMOD **1**(2) (2023)

25. Tampakis, P., Spyrellis, D., Doulkeridis, C., Pelekis, N., Kalyvas, C., Vlachou, A.: A Novel Indexing Method for Spatial-Keyword Range Queries. In: Proceedings of the 17th International Symposium on Spatial and Temporal Databases. p. 54–63 (2021)
26. Tripadvisor LLC: Tripadvisor. `https://www.tripadvisor.com`
27. Wang, X., Cheng, T., Law, S., Zeng, Z., Yin, L., Liu, J.: Multimodal Contrastive Learning of Urban Space Representations from POI Data (2024)
28. Xiao, W., Zhan, Y., Xi, R., Hou, M., Liao, J.: Enhancing HNSW Index for Real-Time Updates: Addressing Unreachable Points and Performance Degradation (07 2024)
29. Zhou, Y., Xie, X., Wang, C., Gong, Y., Ma, W.Y.: Hybrid index structures for location-based Web search. pp. 155–162 (2005)
30. Zhu, Z., Wang, Y., Liu, D.: Hybrid Query of Boolean Filter and Vector Similarity Search: Benchmark, Comparison and Direction. In: Asia Conference on Information Engineering. pp. 117–124 (2025)