On Efficient Approximate Aggregate Nearest Neighbor Queries over Learned Representations

CARRIE WANG, The University of Hong Kong, Hong Kong
SIHEM AMER-YAHIA, CNRS, Univ. Grenoble Alpes, France
LAKS V. S. LAKSHMANAN, The University of British Columbia, Canada
REYNOLD CHENG, The University of Hong Kong, Hong Kong

We study Aggregation Queries over Nearest Neighbors (AQNN), which compute aggregates over the learned representations of the neighborhood of a designated query object. For example, a medical professional may be interested in *the average heart rate of patients whose representations are similar to that of an insomnia patient.* Answering AQNNs accurately and efficiently is challenging due to the high cost of generating high-quality representations (e.g., via a deep learning model trained on human expert annotations) and the different sensitivities of different aggregation functions to neighbor selection errors. We address these challenges by combining high-quality and low-cost representations to approximate the aggregate. We characterize value- and count-sensitive AQNNs and propose the *Sampler with Precision-Recall in Target* (SPRinT), a query answering framework that works in three steps: (1) sampling, (2) nearest neighbor selection, and (3) aggregation. We further establish theoretical bounds on sample sizes and aggregation errors. Extensive experiments on five datasets from three domains (medical, social media, and e-commerce) demonstrate that SPRinT achieves the lowest aggregation error with minimal computation cost in most cases compared to existing solutions. SPRinT's performance remains stable as dataset size grows, confirming its scalability for large-scale applications requiring both accuracy and efficiency.

CCS Concepts: • **Information systems** → **Query languages**; *Information retrieval.*

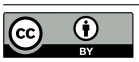
Additional Key Words and Phrases: Aggregation Queries over Nearest Neighbors; Approximate Query Processing; Learned Representations; Oracle and Proxy Models

ACM Reference Format:
Carrie Wang, Sihem Amer-Yahia, Laks V. S. Lakshmanan, and Reynold Cheng. 2026. On Efficient Approximate Aggregate Nearest Neighbor Queries over Learned Representations. *Proc. ACM Manag. Data* 4, 1 (SIGMOD), Article 58 (February 2026), 26 pages. https://doi.org/10.1145/3786672

1 Introduction

Many database applications require computing efficient aggregates of the neighborhood of a designated query object, where the neighborhood is over learned representations. For example, medical professionals may explore *the average heart rate of patients whose representations are similar to that of an insomnia patient* [16]. In online education, instructors may be interested in analyzing performance metrics over groups of students with similar learning trajectories, where similarity is over an embedding space to better capture complex behavioral patterns. Such queries are increasingly common, where the neighborhood is determined, not based on raw data, but on *learned representations*, which are multi-dimensional vectors or embeddings derived from machine learning

Authors' Contact Information: Carrie Wang, The University of Hong Kong, Hong Kong, Hong Kong, carrie07@connect.hku.hk; Sihem Amer-Yahia, CNRS, Univ. Grenoble Alpes, Saint Martin D'Hères, France, sihem.amer-yahia@univ-grenoble-alpes.fr; Laks V. S. Lakshmanan, The University of British Columbia, Vancouver, Canada, laks@cs.ubc.ca; Reynold Cheng, The University of Hong Kong, Hong Kong, Hong Kong, ckcheng@cs.hku.hk.



ACM 2836-6573/2026/2-ART58
https://doi.org/10.1145/3786672

models. We call these queries AQNNs, or *Aggregation Queries over Nearest Neighbors*. AQNNs extend the Fixed-Radius Near Neighbor (FRNN) queries [5] by operating on *learned representations* and applying standard *aggregation functions* (e.g., AVG, SUM, percentage PCT, COUNT, and VAR) over an attribute value of objects in the resulting neighborhood. The evaluation of AQNNs can be costly, as it relies on generating high-quality representations of data objects using an expensive deep learning model, a.k.a. an oracle [8, 18, 20] in order to determine the neighborhood of the input object and compute an aggregate result.

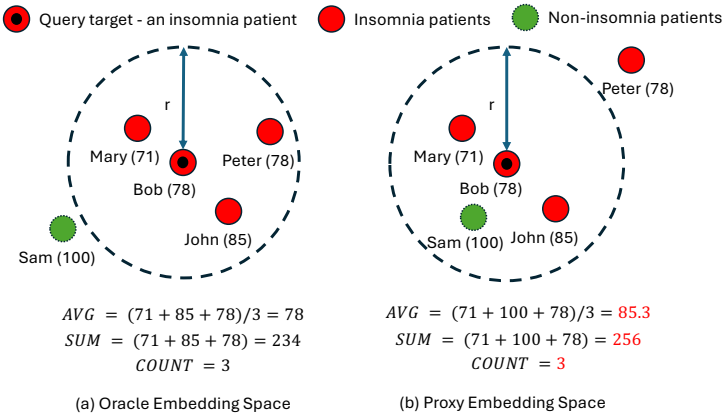


Fig. 1. Nearest neighbors in oracle and proxy embedding spaces and their impact on AQNN results. Numbers in parentheses indicate patients' heart rates (bpm).

In this paper, we study the evaluation of AQNNs by seeking an efficient approximation of the aggregate result. Our approach leverages both an expensive oracle and a cheaper representation learning model, namely a proxy [3, 8, 17, 23], to determine the embeddings and hence the neighborhood for aggregation.

Objectives and Challenges. Effectively answering an AQNN over learned representations requires addressing two key objectives: (**O1**) reducing computational costs, which include *embedding generation* and *query execution costs*, and (**O2**) minimizing the aggregation error between the approximate and true results. A key challenge for **O1** is that learned representations often need to be recomputed at query time, particularly in settings where data evolves rapidly. In healthcare, a patient' physiological state changes frequently, necessitating on-demand embedding updates to ensure accurate neighbor retrieval and attribute aggregation. In online education, student embeddings should be updated with learning progress and interaction history. Using stale embeddings risks inaccurate nearest neighbor (NN) retrieval and unreliable estimation of query answers. Hence, the computational bottleneck lies in generating *oracle embeddings* using high-capacity models such as deep neural networks trained on expert annotations, which can require tens to hundreds of GPU-hours according to throughput benchmarks reported in [27]. Recent work has recognized this issue and proposed the use of *proxy embeddings*, which are less accurate but significantly cheaper to generate for efficient approximate nearest neighbor (ANN) search with probabilistic guarantees [17, 18, 20, 23]. Empirically, generating high-quality oracle embeddings is $2-10\times$ slower than generating proxy embeddings according to existing benchmarks [27, 32]. We will empirically confirm this trend in § 4.2.

However, existing probabilistic ANN algorithms that leverage both proxy and oracle embeddings are typically optimized for either *precision* or *recall*, without carefully considering how neighbor

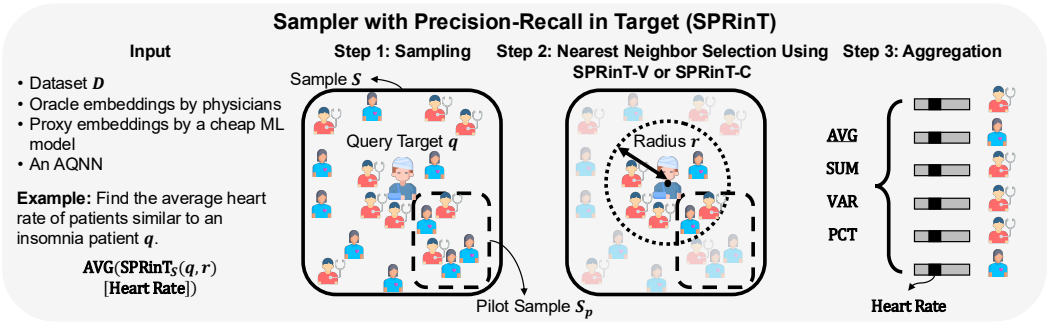


Fig. 2. Overview of the SPRinT framework for answering AQNNs.

selection errors affect the aggregate result. As illustrated in Fig. 1, the neighborhood retrieved in the proxy space may include false positives (e.g., the non-insomnia patient Sam) or omit true positives (e.g., the insomnia patient Peter). These neighbor selection errors propagate to the aggregation step, potentially leading to large aggregation error (**O2**). Moreover, different aggregation functions respond differently to neighbor selection errors. For example, AVG is affected by the attribute values of selected neighbors while COUNT and PCT are more sensitive to the cardinality of the selected neighborhood, regardless of the attribute values. SUM, instead, can be affected by both extreme attribute values and the over- or under-estimation of the neighborhood cardinality. Mitigating such errors can be achieved by jointly controlling the precision and recall of neighbor selection based on the aggregation function's aggregation error sensitivity. Therefore, *our overarching goal is to strike a balance between cost and aggregation error by optimizing precision and recall in a manner that is tailored to different AQNNs.*

Our Contributions. In this work, we address the challenges and make the following contributions:

- We introduce and formalize Aggregation Queries over Nearest Neighbors (AQNNs), a novel query abstraction that unifies learned representation-based NN retrieval with standard SQL aggregation functions (§ 2).
- We propose *Sampler with Precision-Recall in Target* (SPRinT), a framework for answering AQNNs, in three steps (Fig. 2): (1) sampling, (2) NN selection, and (3) aggregation (§ 3.2).
- We analyze aggregation sensitivity and develop two NN selection algorithms: SPRinT-V, which maximizes F1-score for value-sensitive queries (e.g., AVG, VAR), and SPRinT-C, which balances precision and recall for count-sensitive ones (e.g., PCT, COUNT). For SUM, which is both value- and count-sensitive, we heuristically combine SPRinT-V and SPRinT-C (§ 3.2, § 3.4).
- We establish theoretical bounds on the aggregation error for most aggregation functions, deriving minimum sample and pilot sample sizes needed to achieve a target aggregation error with high probability (§ 3.3).
- Extensive experiments on five datasets across three domains (medical, social media, and e-commerce) show that SPRinT achieves significant speedup in terms of embedding generation cost and end-to-end cost, lower aggregation error, and scales efficiently compared to existing methods, corroborating our theoretical analysis. We further showcase how AQNNs enable downstream tasks, particularly one-sample hypothesis testing, which requires neighborhood-level statistics over learned representations (§ 4).

§ 5 discusses related work, while § 6 concludes the paper and outlines directions for future research.

2 Problem Studied

We first discuss Fixed-Radius Near Neighbor (FRNN) queries and then formalize Aggregation Queries over Nearest Neighbors (AQNNs) that build on them. We will also state our problem.

2.1 Nearest Neighbor Queries

Fixed-Radius Near Neighbor (FRNN) queries [5] retrieve all NNs of a query target q within a given radius r under a distance function $dist$. More formally:

$$NN_D(q, r) = \{x \in D \mid dist(x, q) \leq r\},$$

where x is any object in D, and $dist(x, q)$ denotes a standard distance metric such as Euclidean distance. We build on *generalized* FRNN queries by extending them to learned representation spaces, where distances are computed over embeddings produced by machine learning models (e.g., deep neural networks). This enables richer notions of similarity and supports flexible distance metrics, such as cosine similarity. We use $|NN_D(q, r)|$ to denote the neighborhood size of q in D. Fig. 2 shows an example that given the radius r and target patient q, patients in the dotted circle are NNs, and the neighborhood size is 6.

Note that in settings where data evolves rapidly, it is often impractical to assume that all oracle or proxy embeddings are precomputed for all objects in D; instead, they are typically generated on demand during query processing.

2.2 Aggregation Queries over Nearest Neighbors

Given an FRNN query target q in dataset D, a radius r, and an attribute `attr`, an Aggregation Query over Nearest Neighbors (AQNN) is defined as:

$$\text{agg}(NN_D(q, r)[\texttt{attr}])$$

where agg is an aggregation function, such as `AVG`, `SUM`, and `PCT`, and $NN_D(q, r)[\texttt{attr}]$ denotes the bag of values of attribute `attr` of all FRNN results of q within radius r.

We classify AQNNs by their sensitivity to NN selection errors: (i) *value-sensitive* queries (e.g., `AVG`, `VAR`), which depend critically on the attribute values of neighbors; and (ii) *count-sensitive* queries (e.g., `PCT`, `COUNT`), which are strongly influenced by the cardinality of the selected neighborhood regardless of attribute values. Notably, `SUM`, which is both value- and count-sensitive, exhibits sensitivity to both, as errors in either can distort the aggregate result.

An AQNN expresses aggregation operations to capture key insights about the neighborhood of a query target. For example, `AVG` could reflect the average heart rate of patients in a neighborhood, providing a measure of typical health conditions. `SUM` is useful for assessing cumulative effects, such as the total cost of treatments in the neighborhood, which informs public health policy. `PCT` measures the proportion of patients in the neighborhood relative to the total population in the dataset.

Fig. 2 illustrates an example of an AQNN: "*Find the average heart rate of patients similar to an insomnia patient q*". The aggregation function is `AVG` and the target attribute of interest is heart rate. Exact query evaluation requires computing embeddings for all patients in D using high-quality but expensive models (e.g., deep learning models) or expert annotation [23], followed by identifying q's neighbors within radius r [16]. We refer to such accurate but computationally expensive models as *oracle models* (O) [8, 18, 31] and those at least 2× cheaper as *proxy models* (P) [32]. In the example, if the oracle embedding by a physician costs one unit per patient, the proxy embedding by a lightweight machine learning model would cost at most one-half of that. However, relying solely on proxy embeddings to answer the AQNN may lead to inaccurate neighborhood identification,

Table 1. Table of Notations.

Notation	Description
D	the dataset
O	oracle model
P	proxy model
q	query target
r	query radius
S	random sample of D with size s
ON_D	oracle neighbors in D
ON_S	oracle neighbors in S
SPRINT_S	SPRINT neighbors in S
S_p	pilot sample in S with size s_p
t^*	optimal precision target
R_S, P_S	recall, precision in S
F1_S	F1 score in S

which in turn degrades the aggregate result. Thus, we aim to judiciously generate oracle and proxy embeddings in order to balance computational cost and aggregation error.

Once similar patients are identified, their heart rate values can be averaged and returned as the output. Note that the values of the target attribute `attr` (e.g., heart rate) are not predicted but are instead known quantities. Moreover, they are excluded from the representation learning process to avoid biasing similarity toward the very outcome we wish to analyze. This ensures that AQNN results reflect how similar objects relate w.r.t. `attr`, rather than simply aggregating objects that already share similar values.

2.3 Problem Statement

Given an AQNN, our goal is to return an approximate aggregate result by leveraging both oracle and proxy embeddings while reducing cost and error.

3 Our Solution

In this section, we first discuss our objectives and challenges (§ 3.1), then describe the Sampler with Precision-Recall in Target (SPRINT) framework for answering AQNNs (§ 3.2). For value- and count-sensitive AQNNs, we introduce SPRINT-V (§ 3.2.1) and SPRINT-C (§ 3.2.2), which respectively maximize F1 score and equalize precision and recall with high probability. For `SUM`, we design a heuristic strategy, namely Two-Phase, that combines SPRINT-V and SPRINT-C (§ 3.4). We also theoretically analyze SPRINT-V, SPRINT-C, and Two-Phase by deriving their approximation error bounds and the minimal sample and pilot sample sizes required for `AVG`, `VAR`, `PCT`, `COUNT`, and `SUM` (§ 3.3 and § 3.4).

3.1 Objectives and Challenges

As mentioned in § 1, answering an AQNN involves two key objectives: (**O1**) minimize computational cost and (**O2**) ensure a low approximation error.

Achieving **O1** is challenging because learned representations often need to be computed or refreshed at query time rather than being precomputed or cached. In many applications, data evolves rapidly. In healthcare, a patient' physiological state changes frequently, necessitating on-demand embedding updates to ensure accurate neighbor retrieval and attribute aggregation. In online education, student embeddings should be updated with learning progress and interaction

history. Using stale embeddings risks inaccurate NN retrieval and unreliable estimation of query answers. Therefore, the computational bottleneck lies in generation *oracle embeddings*, which can require tens to hundreds of GPU-hours according to throughput benchmarks reported in [27]. Recent work has recognized this issue and proposed the use of *proxy embeddings*, which are less accurate but significantly cheaper to generate for efficient ANN search with probabilistic guarantees [17, 18, 20, 23]. Empirically, generating high-quality oracle embeddings is $2-10\times$ slower than generating proxy embeddings according to existing benchmarks [27, 32]. We will empirically confirm this trend in § 4.2.

However, existing probabilistic ANN algorithms that leverage both proxy and oracle embeddings are typically optimized for either *precision* or *recall*, without carefully considering how neighbor selection errors affect the aggregate result. As shown in Fig. 1, the neighborhood retrieved in the proxy space may include false positives (e.g., the non-insomnia patient Sam) or omit true positives (e.g., the insomnia patient Peter). Such inaccuracies in neighbor selection directly propagate to the aggregation step, making it challenging to achieve **O2**. In Fig. 1(a), the average heart rate (AVG) of the selected neighbors in the oracle embedding space is 78, which reflects the true health condition of patients similar to the query target. In contrast, in the proxy embedding space (Fig. 1(b)), the inclusion of a non-insomnia patient (Sam, with heart rate 100) inflates the computed average to 85.3, introducing a substantial bias. Similarly, for SUM, the total heart rate increases from 234 to 256 due to this false positive. Interestingly, the COUNT remains unchanged at 3 in both spaces because the number of selected neighbors is the same, even though their composition differs. This illustrates that different aggregation functions respond differently to neighbor selection errors.

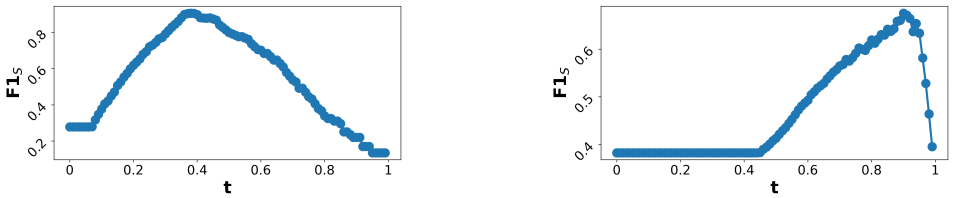
Value-sensitive queries (e.g., AVG, VAR) are prone to distortion from individual attribute values. Hence, both the correctness and completeness of neighbor selection are essential. This requirement can be formally expressed as achieving high precision and recall with high probability. Precision measures the proportion of retrieved neighbors that are correct, while recall measures the proportion of true neighbors that are retrieved, reflecting completeness. Typically, higher precision may exclude true neighbors and hence lower recall, whereas higher recall may include incorrect neighbors and hence reduce precision. Therefore, we aim to maximize both precision and recall as much as possible simultaneously.

Count-sensitive queries (e.g., PCT, COUNT) are less affected by individual false positives or false negatives. Instead, the aggregation error primarily depends on the cardinality of the retrieved neighborhood. This suggests that minimizing error for count-sensitive queries hinges on balancing false positives and false negatives, i.e., equalizing precision and recall. SUM-type queries, in contrast, exhibit dual sensitivity. Errors in both neighbor count and attribute values can contribute to aggregation errors.

To achieve both efficiency and accuracy in answering AQNNs, we propose SPRinT, a three-step framework that combines random sampling with tailored NN selection strategies that optimize precision and recall in accordance with the query's sensitivity to aggregation errors. This design significantly reduces the computational cost of generating proxy and oracle embeddings. We detail the framework in § 3.2.

3.2 The SPRinT Framework

Fig. 2 illustrates SPRinT, our framework for answering AQNNs. It consists of three steps: (1) sampling, (2) NN selection, and (3) aggregation. First, given a dataset D, instead of directly evaluating AQNNs over it, we draw a random sample $S \subset D$ of size s using a random sampler. We then perform neighbor search and aggregation over S. As $s \ll |D|$ and proxy embeddings are computed only for objects in S, this step significantly reduces proxy computation and alleviates the cost burden of **O1**.



(a) eICU (b) Yelp

Fig. 3. F1 score curves w.r.t. the precision target (t).

In the second step, we design NN selection strategies tailored to the aggregation error sensitivity of different AQNNs, addressing **O2**. For value-sensitive queries, we jointly optimize precision and recall by maximizing the F1 score, which harmonizes the two metrics. For count-sensitive queries, we equalize precision and recall to offset over- and under-counting errors. For queries sensitive to both, such as SUM, we combine these strategies heuristically. To enable adaptive NN selection from S, we draw a small pilot sample $S_p \subset S$ of size s_p, on which oracle embeddings are computed. This pilot sample provides lightweight oracle supervision that serves a dual role. First, it enables us to compute precision and recall in S_p, where all objects have both proxy and oracle embeddings. By Hoeffding's inequality [10, 28], the precision and recall in S_p provide reliable estimates of these metrics in S. Second, the true neighbors in S_p can be used to refine NN selection in S according to each AQNNs' error sensitivity. For example, it can further improve the F1 score for value-sensitive queries or balance precision and recall for count-sensitive queries. Overall, the pilot sample incurs minimal oracle embedding generation cost (**O2**) since $s_p < s \ll |D|$, while also playing an important role in controlling the quality of NN selection (**O1**).

In the third step, we apply the desired aggregation function to the target attribute of the selected neighbors in S, and return the aggregate result as an approximation to the AQNN.

3.2.1 SPRINT-V. For value-sensitive AQNNs, we propose SPRINT-V to find neighbors in S. SPRINT-V maximizes the F1 score in S with high probability. However, directly optimizing $\mathrm{F1}_S$ is challenging, as it depends non-linearly on precision and recall, which are often at odds.

To address this, we leverage the insight that a NN search algorithm that *guarantees* one metric, either precision or recall, while *maximizing* the other can effectively navigate this trade-off. We build on two approximate FRNN algorithms: PQE-PT and PQE-RT [8], which, given a query q, a proxy model P, an oracle model O, and a predefined precision (resp. recall) target t, return a neighbor set that satisfies t with high probability while maximizing the complementary[1] metric. Both algorithms achieve statistical guarantees by thresholding proxy distances and validating candidates using oracle embeddings. In our SPRINT framework, the pilot sample S_p, for which oracle embeddings are already available, allows PQE-PT and PQE-RT to run without incurring additional oracle cost. We adopt PQE-PT in SPRINT-V for its implementation stability and consistent empirical performance across datasets. Moreover, we observe empirically that the $\mathrm{F1}_S$ achieved by PQE-PT is unimodal w.r.t. the precision target t (see Fig. 3). This unimodality enables an efficient search for the optimal precision target t^* which maximizes the F1 score via ternary search.

As shown in Algorithm 1, SPRINT-V takes as inputs the sample S, pilot sample S_p, oracle O, proxy P, query target q, query radius r and a tolerance ω_V for terminating the ternary search, and outputs the SPRINT neighbors in S. The algorithm initializes the precision target range with $t_{\min} = 0$ and $t_{\max} = 1$ (Line 3) and iteratively narrows the range until $t_{\max} - t_{\min} \leq \omega_V$ (Line 4). In each iteration,

[1]Precision is complementary w.r.t. recall and vice versa.

Algorithm 1 SPRINT-V

1: **Input:** S, S_p, O, P, q, r, ω_V
2: **Output:** SPRINT_S
3: $t_{\min} \leftarrow 0$, $t_{\max} \leftarrow 1$
4: **while** $t_{\max} - t_{\min} > \omega_V$ **do**
5: $\quad t_1 \leftarrow t_{\min} + \frac{t_{\max} - t_{\min}}{3}$
6: $\quad t_2 \leftarrow t_{\max} - \frac{t_{\max} - t_{\min}}{3}$
7: $\quad NN(t_1) \leftarrow \text{PQE-PT}(S_p, O, P, q, r, t_1)$
8: $\quad NN(t_2) \leftarrow \text{PQE-PT}(S_p, O, P, q, r, t_2)$
9: $\quad \text{F1}_{S_p}(t_1) \leftarrow \frac{2\text{P}_{S_p}(t_1)\text{R}_{S_p}(t_1)}{\text{P}_{S_p}(t_1)+\text{R}_{S_p}(t_1)}$
10: $\quad \text{F1}_{S_p}(t_2) \leftarrow \frac{2\text{P}_{S_p}(t_2)\text{R}_{S_p}(t_2)}{\text{P}_{S_p}(t_2)+\text{R}_{S_p}(t_2)}$
11: $\quad$ **if** $\text{F1}_{S_p}(t_1) > \text{F1}_{S_p}(t_2)$ **then**
12: $\quad\quad t_{\max} \leftarrow t_2$
13: $\quad$ **else**
14: $\quad\quad t_{\min} \leftarrow t_1$
15: $\quad$ **end if**
16: **end while**
17: $t^* \leftarrow (t_{\min} + t_{\max})/2$
18: **return** $\text{PQE-PT}(S, O, P, q, r, t^*)$

two candidate targets t_1 and t_2 are selected by dividing the range into thirds (Line 5-6). The PQE-PT subroutine is then called with each target to retrieve neighbors in S_p (Lines 7-8). Based on the selected neighbors and the oracle-labeled ground truth in S_p, we compute precision, recall, and F1 scores at both targets in S_p (Lines 9-10). The more promising region of the precision range is retained for the next iteration (Lines 11-15). Once the search converges, the optimal precision target t^* is set as the midpoint of the final interval (Line 17). PQE-PT is then called one last time on S to identify SPRINT_S (Line 18), which will later be used to compute the approximate aggregate.

3.2.2 SPRINT-C. For count-sensitive AQNNs, we propose SPRINT-C to find neighbors in S. Similar to SPRINT-V, SPRINT-C leverages PQE-PT to iteratively find the optimal precision target such that the selected neighbors in S achieve equal precision and recall. The intuition is that balancing the false positives and false negatives in the retrieved neighbor set leads to an unbiased estimate of the true neighbor count. We formalize this insight using PCT as an example, by expressing its approximate estimate as:

$$\widetilde{\text{PCT}_S} = \frac{|\text{SPRINT}_S|}{s} = \frac{|(\text{SPRINT}_S \cap ON_S) \cup (\text{SPRINT}_S \setminus ON_S)|}{s}. \tag{1}$$

where SPRINT_X (resp. ON_X) denotes the NNs found by SPRINT (resp. by the oracle, i.e., the true NNs) in dataset X, $X \in \{S, D\}$. When, in S, the number of false positives ($\text{SPRINT}_S \setminus ON_S$) equals the number of false negatives ($ON_S \setminus \text{SPRINT}_S$), $\widetilde{\text{PCT}_S}$ can accurately reflect PCT_S. This condition holds when precision equals recall, which implies a balance between false positives and false negatives. Formally, the precision and recall of the SPRINT neighbors in S are: $\text{P}_S = \frac{|\text{SPRINT}_S \cap ON_S|}{|\text{SPRINT}_S|}$, and $\text{R}_S = \frac{|\text{SPRINT}_S \cap ON_S|}{|ON_S|}$. By setting $\text{P}_S = \text{R}_S$, we can infer $|\text{SPRINT}_S \setminus ON_S| = |ON_S \setminus \text{SPRINT}_S|$, i.e., the numbers of false positives and false negatives are equal. As a result, when $\text{P}_S = \text{R}_S$, $\widetilde{\text{PCT}_S}$ is an unbiased estimator of PCT_S.

Algorithm 2 SPRINT-C

1: **Input:** S, S_p, O, P, q, r, ω_C
2: **Output:** SPRINT$_S$
3: $t_{\min} \leftarrow 0$, $t_{\max} \leftarrow 1$
4: **while** $|\mathrm{R}_{S_p} - \mathrm{P}_{S_p}| > \omega_C$ **do**
5: $t^* \leftarrow (t_{\min} + t_{\max})/2$
6: $NN \leftarrow$ PQE-PT(S_p, O, P, q, r, t^*)
7: $\mathrm{R}_{S_p} \leftarrow \frac{|NN \cap ON_{S_p}|}{|ON_{S_p}|}$
8: $\mathrm{P}_{S_p} \leftarrow \frac{|NN \cap ON_{S_p}|}{|NN|}$
9: **if** $\mathrm{P}_{S_p} \leq \mathrm{R}_{S_p}$ **then**
10: $t_{\min} \leftarrow t^*$
11: **else**
12: $t_{\max} \leftarrow t^*$
13: **end if**
14: **end while**
15: **return** PQE-PT(S, O, P, q, r, t^*)

As shown in Algorithm 2, SPRINT-C takes similar inputs as SPRINT-V and outputs the SPRINT neighbors in S. Since there is an inverse relationship between precision and recall, where a higher precision leads to a lower recall, a binary search is employed to find the optimal precision target t^* which can equalize precision and recall in S. Specifically, SPRINT-C initializes the search range with $t_{\min} = 0$ and $t_{\max} = 1$ (Line 3) and iteratively narrows this range until the difference between the recall R_{S_p} and precision P_{S_p} in the pilot sample is within a tolerance ω_C (Line 4). In each iteration, a candidate precision target is set to the midpoint of the current range (Line 5). The PQE-PT subroutine is then invoked to retrieve the NNs in S_p (Line 6). The corresponding recall and precision R_{S_p} and P_{S_p} are calculated based on the oracle-determined true NNs and the selected NNs in S_p (Line 7-8), and the search range is updated accordingly (Lines 9–13). Finally, once the optimal t^* is determined, PQE-PT is called on S with t^* to retrieve SPRINT$_S$ (Line 15). The neighbor set will then be used in the aggregation step.

Time Complexity. The time complexity of both algorithms is dominated by the invocation of PQE-PT, which has a complexity of $O(s^2 \log(s))$ [8], $s = |S|$ being the sample size. During the search for t^*, SPRINT-V employs a ternary search, requiring $O(\log(1/\omega_V))$ iterations, while SPRINT-C uses a binary search with $O(\log(1/\omega_C))$ iterations[2]. Assuming $\omega_C = \omega_V = \omega^*$, the overall time complexity is $O(\log(1/\omega^*)s^2 \log(s))$. Evaluating precision and recall over S_p introduces an additional linear term, $O(s_p)$, but since $s_p \ll s$, its impact is negligible in asymptotic analysis. Therefore, SPRINT-V and SPRINT-C scale quadratically with s (up to logarithmic factors), while the number of iterations needed for convergence depends only logarithmically on ω^*.

3.3 Theoretical Analysis

In this section, we derive probabilistic error bounds of SPRINT for value- and count-sensitive queries. Also, we establish the minimum sample and pilot sample sizes required to achieve the error bounds. The detailed proofs are provided in the Appendix.

[2]Ternary search uses $\log_3$, while binary search uses $\log_2$. However, in asymptotic analysis, these differences are absorbed into constant factors.

Key insight. The total approximation error arises from two components: (i) the sampling error ω_S from aggregating over NNs in S instead of the full dataset D, and (ii) the NN selection error ω_{NN} introduced by selecting NNs based on a mixture of oracle and proxy embeddings in S. These two sources of error are bounded separately in our analysis.

To help interpret the theoretical bounds, let us unpack the NN selection error ω_{NN} at a high level. For value-sensitive queries, ω_{NN} depends on how well the F1 score in the sample S is optimized. Using SPRINT-V, we maximize $\mathrm{F1}_{S_p}$ and leverage it as an estimator for $\mathrm{F1}_S$. By ensuring that $\mathrm{F1}_{S_p}$ closely approximates $\mathrm{F1}_S$, i.e. $|\mathrm{F1}_S - \mathrm{F1}_{S_p}| \leq \lambda$, the neighbor selection strategy derived from S_p also performs well on S, keeping ω_{NN} small. For count-sensitive queries, ω_{NN} is determined by the precision-recall gap $|P_S - R_S|$, which reflects the imbalance between false positives and false negatives in the retrieved neighbor set in S. SPRINT-C controls this gap via $|P_{S_p} - R_{S_p}| \leq \omega_C$ in the pilot sample S_p, which ensures with high probability that ω_{NN} is small.

THEOREM 1 (AGGREGATION ERROR BOUND FOR VALUE-SENSITIVE QUERIES). *Let $\widetilde{\mathrm{agg}}_S$ denote the approximate aggregate returned by SPRINT-V for a value-sensitive query (e.g., AVG, VAR). Given user-specified tolerances ω_S and ω_{NN}, and confidence level $1-\alpha$, the approximation error satisfies*

$$\Pr\left[\left|\widetilde{\mathrm{agg}}_S - \mathrm{agg}_D\right| \leq \omega_S + \omega_{NN}\right] \geq 1 - \alpha,$$

provided that the sample size s and pilot sample size s_p satisfy

$$s \geq \kappa_1 \cdot \frac{1}{\omega_S^2}, \quad s_p \geq \kappa_2 \cdot \frac{1}{\lambda^2},$$

where λ is a configurable tolerance on the difference between $\mathrm{F1}_S$ and $\mathrm{F1}_{S_p}$ and $\omega_{NN} = C_1 \cdot \lambda + C_2 \cdot \lambda^2$. Here, κ_1, κ_2, C_1, and C_2 depend on the specific aggregation function as follows:

- *For* AVG,

$$\kappa_1 = \frac{(b-a)^2 \ln(2/\alpha)}{2\rho}, \quad \kappa_2 = 32\ln(2/\alpha),$$
$$C_1 = \frac{2(b-a)}{|ON_S|}, \quad C_2 = 0.$$

- *For* VAR,

$$\kappa_1 = \frac{(b-a)^4 \ln(2/\alpha)}{2\rho}, \quad \kappa_2 = 32\ln(2/\alpha),$$
$$C_1 = \frac{3(b-a)^2}{|ON_S|}, \quad C_2 = \frac{(b-a)^2}{|ON_S|}.$$

where a and b are the lower and upper bounds of the target attribute [3]; $\rho = |ON_S|/|S|$ is the the neighborhood density.

Practical Implications. To illustrate the practical significance of Theorem 1, consider the value-sensitive query in Fig. 2. Suppose we set $\alpha = 0.05$ and choose a radius r such that $\rho = 0.8$. Here, ρ is estimated from the dataset by the proportion of objects in D within radius r. Setting $\omega_S = 5$ bpm and $\omega_{NN} = 0.2$, which are clinically reasonable tolerances, results in a minimum sample size of $s \geq 452$ and pilot sample size of $s_p \geq 111$. These correspond to 10.6% and 2.6% of the population size ($|D| = 4{,}245$), respectively. By applying these bounds, practitioners can configure parameters to balance computational cost and aggregation accuracy in real-world settings.

[3] In clinical settings, adult patients' heart rates are commonly observed in the range of 50-120 bpm, although the typical resting range is 60-100 bpm.

Theorem 2 (Aggregation Error Bound for Count-Sensitive Queries). *Let $\widetilde{\text{agg}}_S$ denote the approximate aggregate returned by SPRinT-C for a count-sensitive query (e.g.,* PCT, COUNT*). Given user-specified tolerances ω_S and ω_{NN}, and confidence level $1-\alpha$, the approximation error satisfies*

$$\Pr\left[\left|\widetilde{\text{agg}}_S - \text{agg}_D\right| \le \omega_S + \omega_{NN}\right] \ge 1-\alpha,$$

provided that the sample size s and pilot sample size s_p satisfy

$$s \ge \kappa_1 \cdot \frac{1}{\omega_S^2}, \quad s_p \ge \kappa_2 \cdot \frac{1}{(\omega_{NN} - \rho \cdot \omega_C)^2},$$

where $\rho = |ON_S|/|S|$ is the the neighborhood density. Here, κ_1 and κ_2 depend on the specific aggregation function as follows:

- *For* PCT,
$$\kappa_1 = \frac{1}{2}\ln\left(\tfrac{2}{\alpha}\right), \quad \kappa_2 = 2\ln\left(\tfrac{2}{\alpha}\right)\cdot\rho^{-2}.$$
- *For* COUNT,
$$\kappa_1 = \frac{|D|^2}{2}\ln\left(\tfrac{2}{\alpha}\right), \quad \kappa_2 = 2\ln\left(\tfrac{2}{\alpha}\right)\cdot\rho^{-2}.$$

Practical Implications. To illustrate the practical significance of Theorem 2, consider a count-sensitive query such as computing the proportion of neighbors in the population of a target insomnia patient. Suppose we set $\omega_S = 0.05$, $\omega_{NN} = 0.1$, and $\omega_C = 0.0001$. These tolerances yield a minimum sample size of $s \ge 738$ and pilot sample size of $s_p \ge 739$, which both correspond to about 17.4% of ($|D| = 4{,}245$). Notably, the lower bound for s_p may exceed that of s because s and s_p serve distinct roles. s controls how well the aggregate computed over S approximates the true aggregate over D. In contrast, s_p ensures the precision and recall estimates obtained from S_p are sufficiently accurate to represent those in S, which is essential for selecting a stable precision target t^*. Hence, to ensure both accuracy and stability, we set $s = \max(s, s_p)$ and the smaller one be the pilot sample.

3.4 Practical Algorithm for SUM Queries

SUM query differs from the value-sensitive or count-sensitive queries because its error depends on both the cardinality of the selected neighborhood and their target attribute values. Hence, we propose a practical strategy, called *Two-Phase*, to balance these two aspects.

Two-Phase Strategy. To handle SUM's sensitivity to both count and value of NNs, we propose Two-Phase: (1) apply SPRinT-C to identify a precision target t that balances precision and recall, aligning the neighbor counts; (2) refine it within a small window (i.e., $t \pm 0.05$) and use SPRinT-V to maximize the F1-score, ensuring that the selected neighbors are value-consistent.

Theorem 3 (Aggregation Error Bound for SUM Queries). *Let $\widetilde{\text{SUM}}_S$ denote the approximate aggregate returned by Two-Phase for the* SUM *query. Given user-specified tolerances ω_S and ω_{NN}, and confidence level $1-\alpha$, the approximation error satisfies*

$$\Pr\left[\left|\widetilde{\text{SUM}}_S - \text{SUM}_D\right| \le \omega_S + \omega_{NN}\right] \ge 1-\alpha,$$

provided that the sample size s and pilot sample size s_p satisfy

$$s \ge \max\left(s^{count},\ s^{avg}\right), \quad s_p \ge \max\left(s_p^{count},\ s_p^{avg}\right).$$

Here:

- $s^{count} = \frac{2|D|^2|\text{AVG}_S|^2\ln(4/\alpha)}{\omega_S^2}$ *bounds the sampling error on the neighbor count.*
- $s^{avg} = \frac{2(b-a)^2|ON_D|^2\ln(4/\alpha)}{\omega_S^2}$ *bounds the sampling error on the attribute mean.*

Table 2. Datasets, oracle, and proxy models.

Datasets	Oracle	Proxy
Mimic-III, eICU	Expert Notations	LIG-Doctor [16]
Jigsaw	User Labels	GLiClass [33]
Yelp, Amazon-E	MiniLM-L12 [32]	MiniLM-L6 [32]

Table 3. Dataset size $|D|$ and default parameters for SPRINT: sample size s and pilot sample size s_p.

	eICU	MIMIC-III	Jigsaw	Yelp	Amazon-E
$\|D\|$	8,234	4,245	16,225	6,990,280	10,000
s	1,000	500	1,000	35,000	2,000
s_p	600	150	600	20,000	1,200

- $s_p^{count} = \kappa_2^{\mathsf{COUNT}} \cdot \frac{1}{\left(\omega_{NN}^{\mathsf{SUM}} - \rho \cdot \omega_C\right)^2}$ *bounds the count selection error, where* $\omega_{NN}^{\mathsf{SUM}} = \frac{\omega_{NN}|S|}{2|D||\overline{\mathsf{AVG}_S}|}$ *and* $\rho = \frac{|ON_S|}{s}$ *denotes the neighborhood density in* S.
- $s_p^{avg} = \kappa_2^{\mathsf{AVG}} \cdot \frac{1}{\lambda^2}$ *bounds the mean selection error, where* $|F1_S - F1_{S_p}| \leq \lambda$.

4 Experiments

The purpose of our experiments is to evaluate our solution for answering AQNNs w.r.t. objectives **O1** and **O2**: cost and error. § 4.1 describes the datasets, baselines, and evaluation measures. § 4.2 evaluates efficiency of SPRINT in terms of embedding generation cost and end-to-end runtime. § 4.3 examines the effectiveness of SPRINT-V and SPRINT-C in lowering aggregation error and query execution time. We also study the impact of dataset size (scalability), sample size, pilot sample size, and NN sparsity (controlled by the query radius r) on the aggregation error. Finally, § 4.4 demonstrates one-sample hypothesis testing as a downstream application of AQNNs.

4.1 Experimental Setup

In our experiments, we use static datasets to simulate settings where embeddings must be recomputed or refreshed on demand at query time. This reflects realistic deployment scenarios, such as in healthcare or education, where data evolves rapidly, making precomputed or cached embeddings stale or infeasible. For settings with static or infrequently updated data, embeddings may be precomputed, cached, and reused for later queries; we analyze the cost of such scenarios in § 4.3.2 by decoupling embedding generation cost and measuring only the query execution cost, assuming embeddings are precomputed.

4.1.1 Datasets. We used five datasets from three different domains, each paired with task-appropriate proxy and oracle models (see Table 2). The dataset sizes are provided in Table 3, top row.
Medical: MIMIC-III [15] and eICU [26] are publicly available clinical datasets containing de-identified ICU records from U.S. hospitals. We adopt LIG-Doctor [16], a lightweight RNN-based model for patient trajectory prediction, to generate proxy embeddings from ICU admission sequences. The oracle embeddings are derived from ground-truth physician diagnoses. Each patient record has demographic and physiological attributes (e.g., heart rate), which enable value-sensitive AQNNs in medical contexts. We remove records with only one ICU stay.
Social media: The Jigsaw toxicity classification dataset [7] contains real-world user comments annotated across five toxicity types (e.g., threat, insult). We employ GLiClass [33], a zero-shot

classifier to generate proxy embeddings without domain-specific training. Oracle embeddings are constructed from the annotated toxicity labels. To support the evaluation of value-sensitive queries, we synthesize numerical engagement metrics (e.g., number of down-votes) using K-means clusters [22, 24].

E-commerce: Yelp [1] and Amazon-Electronics (Amazon-E) [11] are publicly available corpora of user reviews that include text, metadata, and star ratings (1-5). We generate oracle embeddings using MiniLM-L12 [32], a transformer-based encoder known for its strong semantic fidelity, and proxy embeddings using a lighter-weight MiniLM-L6 [32]. Star ratings are used as the target attribute in value-sensitive AQNNs.

4.1.2 Baselines. SPRINT is, to our knowledge, the first framework for cost-efficient and low-error approximation of AQNNs in settings where oracle embeddings are expensive and need to be recomputed on demand. We first conduct a *framework-level comparison* against a Brute-Force baseline, which computes oracle embeddings for all objects and performs exact NN search, w.r.t. embedding generation and the overall end-to-end costs. We then perform an *algorithm-level comparison* to evaluate our proposed NN selection algorithms, SPRINT-V and SPRINT-C, against three probabilistic ANN baselines adapted from prior work. These include:

A. PQE-PT and PQE-RT. These are FRNN approximation algorithms that guarantee user-specified precision targets (PQE-PT) or recall targets (PQE-RT) with high probability [8]. They calibrate the correlation between proxy and oracle distances using a small oracle-labeled sample from the dataset. Once calibrated, proxy distances are used to select neighbors in the full dataset. Although PQE-PT is used as a subroutine in our framework, it is designed to optimize precision only and does not account for aggregation error. We will show its suboptimality for answering AQNNs in § 4.3.1.

B. SUPG-PT and SUPG-RT. SUPG [18] frames FRNN as a binary classification problem: given a query target q and radius r, predict whether each object x lies within the neighborhood of q using proxy-based distances. SUPG-PT and SUPG-RT are trained using oracle-labeled samples to ensure precision or recall guarantees, respectively. After training, the classifier is applied to the entire dataset using only proxy embeddings. Like PQE, SUPG incurs limited oracle cost during calibration but performs inference purely on proxy representations.

C. Top-K. Probabilistic Top-K approximates NNs by returning the top-K closest objects to the query under proxy distances, where K is set to the number of oracle-based neighbors within radius r [20]. It assumes that proxy distances are accurate surrogates of oracle distances. This assumption holds reasonably well in our datasets, as shown in Fig. 4, where $dist^X(x_i)$ represents the distance computed using X embeddings between x_i and q, with $X \in \{O, P\}$.

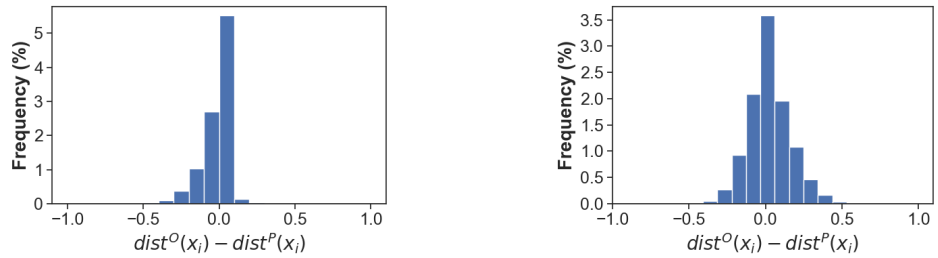
4.1.3 Evaluation Measures. **Relative Error (RE)** is a measure of aggregation error, defined as the proportional difference between the estimated and true aggregates:

$$\text{RE} = \frac{|\widetilde{\text{agg}_S} - \text{agg}_D|}{|\text{agg}_D|} \times 100\% \tag{2}$$

Cost refers to the total computational expense incurred during AQNN evaluation. It consists of two components: (i) *embedding generation cost*, measured by time and also reflected by the number of oracle and proxy model calls required to generate embeddings; and (ii) *query execution cost*, measured by time required to answer AQNNs given computed embeddings.

F1 score, the harmonic mean of precision and recall, measures how accurately the selected NNs match the true NNs. It is defined as:

$$\text{F1} = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \tag{3}$$



(a) eICU (b) MIMIC-III

Fig. 4. Distribution of the difference between proxy and oracle distances.

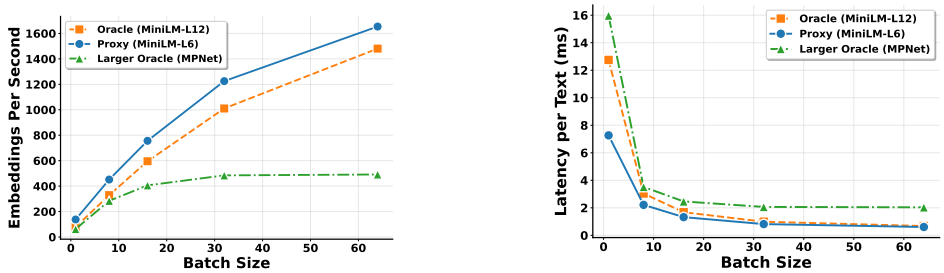


Fig. 5. Oracle vs. Proxy embedding generation cost.

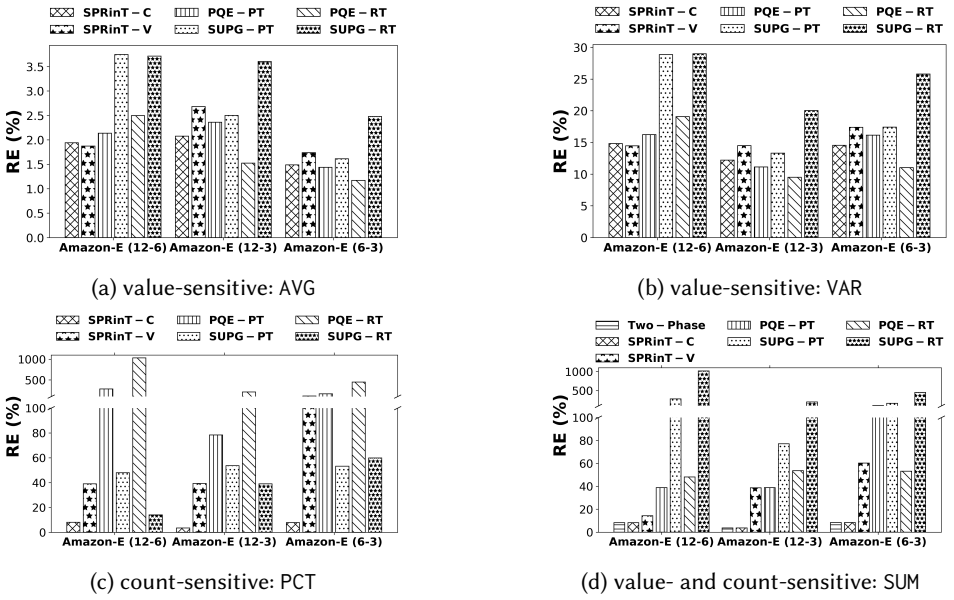
4.1.4 Default Parameters. We evaluate AQNNs using four aggregation functions that span all query types: value-sensitive (AVG, VAR), count-sensitive (PCT), and both value- and count-sensitive (SUM). To use SPRINT, the default values for the sample size s and pilot sample size s_p are shown in Table 3. These values are derived from Theorems 1-3 by setting appropriate values for b, a, ρ, ω_S, and ω_{NN}. Following prior work [8], we set the precision and recall targets to 0.95 for both PQE and SUPG. All experiments are repeated 30 times to ensure statistical reliability under the Central Limit Theorem (CLT) [19]. Reported metrics are averaged over 10 random query targets to account for query-specific variability.

All experiments are conducted on a NVIDIA RTX 2080 GPU with batch size 32, unless otherwise specified.

4.2 Framework-Level Evaluation

We evaluate the embedding generation cost and overall end-to-end runtime of answering AQNNs by comparing SPRINT against Brute-Force. Aggregation error is not compared when evaluating cost, as Brute-Force trivially achieves zero error. Also, we conduct a sensitivity analysis on the choice of oracle-proxy pair to evaluate the robustness of SPRINT to this choice.

Empirical Evidence: Oracle vs Proxy Embedding Generation Cost. To quantify the computational gap between oracle and proxy models, we measure embedding generation latency and throughput across batch sizes for three text-embedding models, MiniLM-L6 (proxy), MiniLM-L12 (oracle), and MPNet [30, 32] using 1,000 Amazon-E reviews. Fig. 5 shows that oracle models are approximately 2× slower in per-text latency than proxy models at small to medium batch sizes,



(a) value-sensitive: AVG

(b) value-sensitive: VAR

(c) count-sensitive: PCT

(d) value- and count-sensitive: SUM

Fig. 6. RE performance under different choices of oracle and proxy models on Amazon-E.

Table 4. Embedding generation cost in terms of oracle and proxy calls. **Bold** indicates the best performance.

Dataset	Framework	Oracle Calls ↓	Proxy Calls ↓	Speedup ↑
eICU	Brute-Force	8,234	**0**	1.0×
	SPRINT	**600**	1,000	**7.5×**
MIMIC-III	Brute-Force	4,245	**0**	1.0×
	SPRINT	**150**	500	**10.6×**
Jigsaw	Brute-Force	16,225	**0**	1.0×
	SPRINT	**600**	1,000	**14.8×**
Yelp	Brute-Force	6,990,280	**0**	1.0×
	SPRINT	**20,000**	35,000	**186.4×**
Amazon-E	Brute-Force	10,000	**0**	1.0×
	SPRINT	**1,200**	2,000	**4.5×**

consistent with the benchmark results reported in [32]. At large batch sizes, per-text latency converges due to GPU amortization, but proxy models still achieve higher throughput and remain significantly more efficient for query-time embedding generation.

4.2.1 Embedding Generation Cost. Table 4 compares the embedding generation costs for all datasets. We assume oracle calls are 2× slower than proxy calls, which is conservative, as prior work reports 2–10× gaps [27, 32]. SPRINT achieves 4.5–186.4× speedup by avoiding the majority of oracle calls. For instance, on Jigsaw, SPRINT uses proxy models for ∼6% of objects to avoid >96% of oracle calls.

4.2.2 End-to-End Runtime. We measure end-to-end latency by running both frameworks on Yelp and Amazon-E, the two datasets for which both oracle and proxy models are executable on GPU. Table 5 shows that SPRINT reduces total query latency by 32.9× on Yelp and 2.8× on Amazon-E. While SPRINT's query execution time is higher due to its additional logic, this overhead is negligible

Table 5. End-to-end cost (s). **Bold** indicates the best performance.

Dataset	Framework	Embedding Generation Time (s) ↓	Query Execution Time (s) ↓	Total Time (s) ↓	Speedup ↑
Amazon-E	Brute-Force	10.88	**0.16**	11.04	1.0×
	SPRINT	**3.01**	0.94	**3.95**	**2.8×**
Yelp	Brute-Force	8996.49	**0.16**	8996.65	1.0×
	SPRINT	**65.54**	207.84	**273.38**	**32.9×**

compared to the savings from reducing the cost of oracle embedding generation, which dominates the total cost when data evolves and the embeddings must be refreshed on query time.

4.2.3 Impact of Oracle and Proxy Choices. To evaluate the robustness of SPRINT under different choices of oracle and proxy models, we conduct a sensitivity analysis on Amazon-E with the following three model pairs:

(1) **MiniLM-L12 (oracle)** and **MiniLM-L6 (proxy)**;
(2) **MiniLM-L6 (oracle)** and **MiniLM-L3 (proxy)**;
(3) **MiniLM-L12 (oracle)** and **MiniLM-L3 (proxy)**.

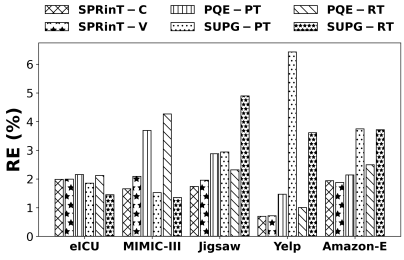
Since all variants use the same default sample and pilot sample sizes under SPRINT, their end-to-end speedup over Brute-Force remains comparable (see Table 5). In terms of aggregation error, Fig. 6 shows that for `PCT` and `SUM` queries, our proposed methods SPRINT-C and Two-Phase consistently outperform baselines across all datasets. Interestingly, we observe that SPRINT-C performs nearly identically to Two-Phase in most cases. This similarity arises because, in these datasets, the attribute values within neighborhoods exhibit low variance, making the benefit over SPRINT-C from the second-phase refinement of the precision target in Two-Phase modest. For value-sensitive queries like `AVG` and `VAR`, different oracle-proxy variants may lead to varying SPRINT-V performance. Particularly, SPRINT-V outperforms all baselines when the oracle is MiniLM-L12 and the proxy is MiniLM-L6. However, precision-targeted methods (e.g., PQE-PT and SUPG-PT) perform better than SPRINT-V for other oracle-proxy pairs. We attribute this to the highly concentrated value distribution of Amazon-E star ratings. When attribute values are similar in the neighborhood, even a small, high-precision set can provide accurate estimates. This highlights an important insight: the optimal neighbor selection strategy for AQNNs may depend not only on the aggregation function but also on the distribution of neighbor attribute values. We defer a deeper analysis of value-sensitive AQNNs and the impact of query targets on aggregation error to § 4.3.1-A.

4.3 Algorithm-Level Evaluation

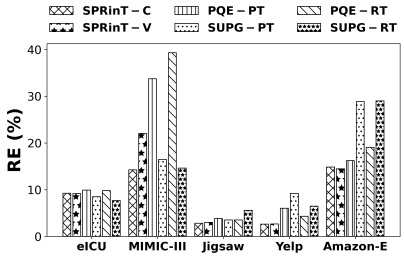
We compare the aggregation error in terms of RE and query execution runtime of our proposed neighbor selection algorithms, including SPRINT-V, SPRINT-C, and Two-Phase, against ANN baselines in § 4.1.2 within the same SPRINT framework. Also, we examine scalability, the impact of sample and pilot sample sizes, and the effect of query radius on RE.

4.3.1 Examining Aggregation Error. Fig. 7 reports the RE of all algorithms except Top-K, which is compared separately in Table 9. This is for a fair comparison as Top-K dynamically incurs oracle calls and does not follow a fixed oracle budget.

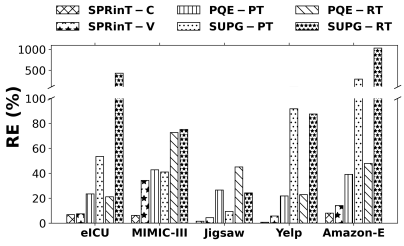
A. Value-sensitive AQNNs. In Fig. 7a and 7b, on `AVG` and `VAR`, SPRINT-V performs best on Amazon-E. On other datasets, recall-targeted methods (e.g., SUPG-RT, PQE-RT) occasionally outperform SPRINT-V. To explain this, we compare the quality of neighbor selection using $F1_S$ scores (Table 6). SUPG-RT and PQE-RT retrieve low-quality neighbors yet achieve lower RE. For instance, on eICU,



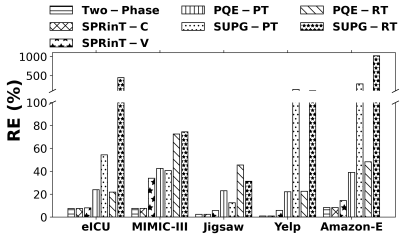
(a) value-sensitive: AVG



(b) value-sensitive: VAR



(c) count-sensitive: PCT

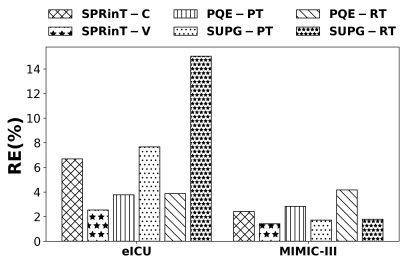


(d) value- and count-sensitive: SUM

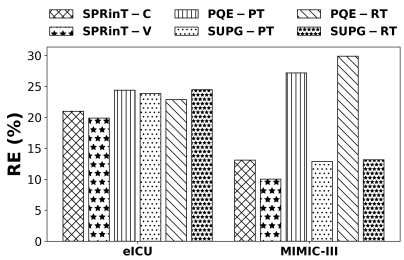
Fig. 7. RE performance for different AQNN queries.

Table 6. $F1_S$ performance (↑) for AVG and VAR. **Bold** indicates the best per dataset.

Dataset	SPRinT-V	SPRinT-C	PQE-PT	PQE-RT	SUPG-PT	SUPG-RT
eICU	**0.95**	**0.95**	0.87	0.84	0.89	0.57
MIMIC-III	**0.82**	0.79	0.67	0.81	0.43	0.75
Jigsaw	**0.96**	**0.96**	0.81	0.95	0.69	0.89
Yelp	**0.93**	**0.93**	0.89	0.83	0.83	0.80
Amazon-E	0.84	**0.85**	0.79	0.47	0.68	0.30



(a) value-sensitive: AVG



(b) value-sensitive: VAR

Fig. 8. RE performance for representative target queries.

SUPG-RT attains the lowest $F1_S$ but surprisingly also the lowest RE. Further investigation reveals that the query target chosen and the target attribute value distribution in its neighborhood can play a decisive role here.

Impact of query targets. Value-sensitive queries are indeed a bit nuanced. When a query target's NNs have a target attribute distribution that differs from the global distribution, SPRinT-V consistently

Table 7. $|P_S - R_S|$ performance (↓) for PCT. **Bold** indicates the best per dataset.

Dataset	SPRINT-C	SPRINT-V	PQE-PT	PQE-RT	SUPG-PT	SUPG-RT
eICU	**0.02**	0.03	0.21	0.22	0.18	0.51
MIMIC-III	**0.04**	0.22	0.39	0.25	0.72	0.38
Jigsaw	**0.01**	0.04	0.26	0.08	0.45	0.15
Yelp	**0.00**	0.05	0.16	0.24	0.23	0.25
Amazon-E	**0.05**	0.09	0.23	0.68	0.48	0.81

Table 8. $F1_S$ and $|P_S - R_S|$ performance for SUM. **Bold** indicates the best per dataset.

Dataset	Two-Phase	SPRINT-C	SPRINT-V	PQA-PT	PQA-RT	SUPG-PT	SUPG-RT
$F1_S$ (↑)							
eICU	**0.95**	**0.95**	**0.95**	0.87	0.84	0.89	0.59
MIMIC-III	0.79	0.79	**0.82**	0.67	0.81	0.43	0.75
Jigsaw	**0.96**	**0.96**	**0.96**	0.81	0.95	0.69	0.89
Yelp	**0.93**	**0.93**	**0.93**	0.89	0.83	0.83	0.80
Amazon-E	**0.85**	**0.85**	0.84	0.79	0.47	0.68	0.30
$\|P_S - R_S\|$ (↓)							
eICU	**0.82**	**0.82**	0.03	0.21	0.22	0.18	0.51
MIMIC-III	**0.04**	**0.04**	0.22	0.39	0.25	0.72	0.38
Jigsaw	**0.01**	**0.01**	0.04	0.26	0.08	0.45	0.15
Yelp	**0.00**	**0.00**	0.05	0.16	0.24	0.23	0.25
Amazon-E	**0.05**	**0.05**	0.09	0.23	0.68	0.48	0.81

outperforms baselines (Fig. 8 shows representative queries from eICU and MIMIC-III). The rationale is that algorithms that achieve poorer F1 score will end up picking up "neighbors" which are less representative of the true neighborhood and as a result perform the subsequent aggregation on an attribute value distribution dissimilar to the neighborhood's distribution. In contrast, if the neighborhood and the global distributions are similar, the impact of a poor F1 score diminishes: this is demonstrated in the various cases in Fig. 7 and 6. Indeed, in such cases, NN search may be unnecessary in the first place: directly sampling from the global distribution would already give accurate approximations of aggregates, while SPRINT-V may incur unnecessary overhead. A practitioner needs to know under what conditions computing AQNNs using a sophisticated approach like SPRINT-V is warranted. To help with this decision, we recommend first using pure proxy embeddings to compute global and neighborhood aggregates. If the difference exceeds a meaningful threshold, i.e., 10%, selective oracle refinement using SPRINT framework can significantly improve RE. Otherwise, proxy-based aggregates suffice.

B. Count-sensitive AQNNs. Next we turn to count-sensitive queries. We report PCT results only, as COUNT exhibits similar performance trends due to their proportional relationship. As shown in Fig. 7b, SPRINT-C consistently achieves the lowest RE across all datasets, empirically corroborating Theorem 2. SPRINT-V consistently ranks second, which implies that maximizing the F1 score can be beneficial for count-sensitive queries, though it is less effective than directly controlling the neighbor count. Other baselines perform significantly worse. Particularly, the recall-targeted methods (SUPG-RT and PQE-RT) attain the highest RE because they tend to retrieve overly many false neighbors. These findings are consistent with the precision-recall gap observed in Table 7.

C. Both value- and count-sensitive AQNNs. Fig. 7d reports the RE for SUM, which is sensitive to both neighbor counts and attribute values. Two-Phase strategy consistently achieves the lowest RE across all datasets, demonstrating its ability to handle dual sensitivities effectively. A similar trend is observed here: when attribute variance within neighborhoods is low, SPRINT-C performs

Table 9. RE and cost comparison with Top-K. The algorithm "ours" refers to SPRINT-V for AVG and VAR, SPRINT-C for PCT, and Two-Phase for SUM. **Bold** indicates the best performance.

Dataset	Algorithm	Oracle Calls ↓	Query Execution Cost (s) ↓	AVG RE ↓	VAR RE ↓	PCT RE ↓	SUM RE ↓
eICU	Top-K	991	47.71	2.01	**8.77**	**6.15**	**6.82**
	ours	**600**	**0.24**	**1.99**	9.15	6.80	7.37
MIMIC-III	Top-K	500	1.77	1.60	14.80	**2.87**	**4.98**
	ours	**150**	**0.09**	**1.44**	**12.92**	6.21	7.55
Jigsaw	Top-K	998	2.18	**1.39**	**1.77**	**1.13**	**1.87**
	ours	**600**	**0.27**	1.96	3.01	1.53	2.36
Yelp	Top-K	34,997	947.23	**0.37**	**0.51**	**0.65**	1.14
	ours	**20,000**	**125.34**	0.71	2.67	0.77	**0.96**
Amazon-E	Top-K	1,980	30.05	**1.70**	**13.14**	**5.28**	**5.47**
	ours	**1,200**	**0.94**	1.87	14.47	8.02	8.30

Table 10. Query execution cost (s). **Bold** indicates the lowest cost per dataset.

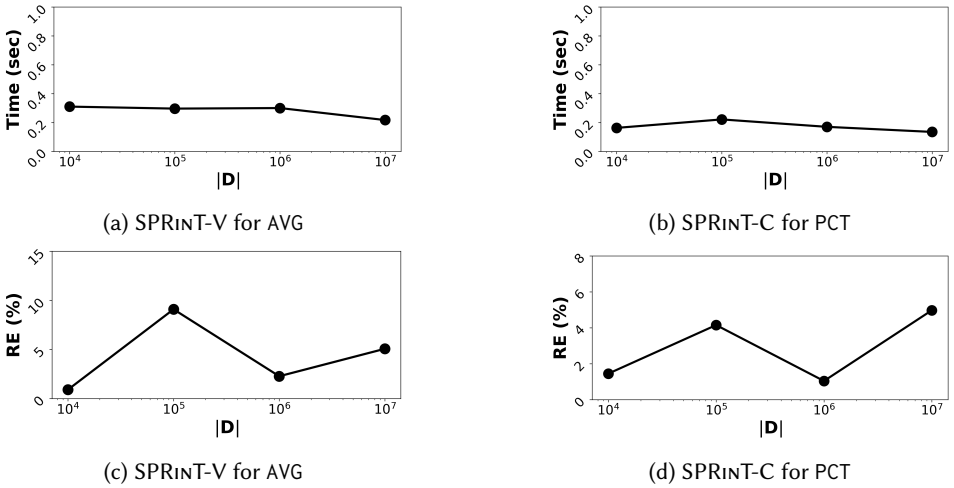
Dataset	SPRINT-V	SPRINT-C	PQE-PT	PQE-RT	SUPG-PT	SUPG-RT
eICU	0.24	0.09	**0.01**	0.09	0.02	0.02
MIMIC-III	0.09	0.03	**0.00**	0.02	0.01	0.01
Jigsaw	0.27	0.14	0.06	0.11	**0.02**	**0.02**
Yelp	207.84	119.12	9.07	120.42	0.45	**0.44**
Amazon-E	0.94	0.54	0.12	0.42	**0.02**	**0.02**

comparably to Two-Phase, consistent with our earlier observation in § 4.2.3. Again, we dig deeper for a better understanding by measuring the F1 score and precision-recall gaps. As shown in Table 8, Two-Phase and SPRINT-C consistently attain the lowest $|P_S - R_S|$ with high $F1_S$ in most datasets. While SPRINT-C seems to suffice in these datasets, Two-Phase becomes critical in settings where attribute distributions are highly skewed or contain extreme outliers, as it provides an additional refinement step to control value errors.

D. Comparison with Top-K. Tables 9 compares the performance of our proposed algorithms against Top-K. On all datasets, our proposed algorithms achieve comparable RE to Top-K, i.e., less than 5% difference, on all aggregation functions, and even slightly outperform Top-K on MIMIC-III for AVG and VAR.

4.3.2 Examining Cost. We first analyze Top-K. In Table 9, Top-K incurs more oracle calls, leading to higher embedding generation cost, and takes significantly higher query execution runtime than SPRINT. While it sometimes yields lower RE, this comes at the expense of considerably more computational overhead. For other baselines executed within the SPRINT framework, Table 10 shows the query execution cost across datasets. By comparing query execution cost only, we simulate settings with static or infrequently updated data, where oracle and proxy embeddings can be precomputed, cached, and reused for later queries. This is the *least favorable setting* for our framework, yet our algorithms remain competitive with baselines in most cases, except on Yelp, whose large sample size (see Table 3) leads to higher runtimes.

Overall, though our proposed algorithms may sometimes spend more time on NN selection, SPRINT, analyzed in § 4.2, saves a significant amount of embedding generation cost and thus reduces end-to-end cost, demonstrating the efficiency of our solution.



(a) SPRINT-V for AVG (b) SPRINT-C for PCT

(c) SPRINT-V for AVG (d) SPRINT-C for PCT

Fig. 9. Impact of dataset size $|D|$ on the execution time and RE using SPRINT-V for AVG (a, c) and SPRINT-C for PCT (b, d) on the Yelp dataset.

4.3.3 Examine Scalability. We evaluate scalability using a semi-synthetic version of the Yelp dataset. Specifically, we start with a sampled subset of $|D| = 10^4$ objects and scale $|D|$ up to 10^7 by duplicating and slightly perturbing records to preserve the original attribute value distribution. The sample size $s = 1000$ and pilot sample size $s_p = 600$ remain fixed as $|D|$ increases. Fig. 9a and 9b report runtime. Fig. 9c and 9d show RE performance.

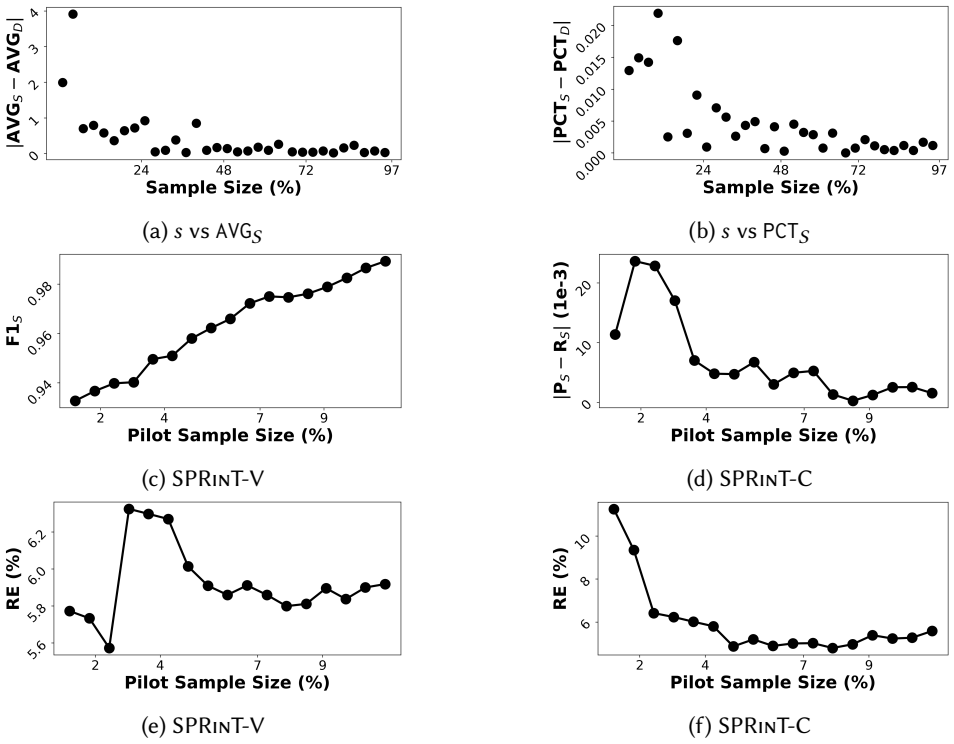
Results confirm that both SPRINT-V and SPRINT-C scale very well w.r.t. $|D|$, validating that our sampling-based framework's cost is independent of the dataset size. The RE also remains fairly stable. The slight fluctuations are due to changes in neighborhood density. This aligns with our theoretical analysis: once s and s_p are sufficiently large, the approximation error and computational cost remain relatively unaffected by $|D|$.

4.3.4 Impact of Sample and Pilot Sample Sizes. In § 3, we present the theoretical minimum s and s_p required to guarantee a certain aggregation error at $1 - \alpha$ confidence level. Here, we empirically validate how varying s and s_p affects RE using the eICU dataset for AVG and PCT in Fig. 10.

A. Effect of Sample Size. Fig. 10a and 10b plot the absolute difference $|\mathsf{agg}_S - \mathsf{agg}_D|$ against the sample size as a percentage of the dataset. As s increases, both the absolute difference and its variance decrease, as the aggregate over S better approximates D. Notably, using around 25% of the data already provides a close approximation to agg_D. Beyond a certain threshold (e.g., $s > 25\%$), the improvement plateaus, indicating diminishing returns. This suggests that low RE in AQNN evaluation can be achieved with a moderate sample size s, which also means fewer proxy calls and less overhead.

B. Effect of Pilot Sample Size. Fig. 10c and 10e show that for SPRINT-V, increasing s_p improves $\mathrm{F1}_S$ and consequently lowers RE. For SPRINT-C, Fig. 10d and 10f indicate that increasing s_p narrows the precision-recall gap in S, which in turn reduces RE. Early instabilities are observed across all plots when s_p is very small, i.e., lacking true neighbors to reliably approximate aggregates.

Overall, sufficiently large s and s_p ensure stable and low-error aggregation. But interestingly, using as little as 5% of the data as the pilot sample already achieves stable RE in most cases. The key message is not simply that larger samples are better, but that even small samples, once above a

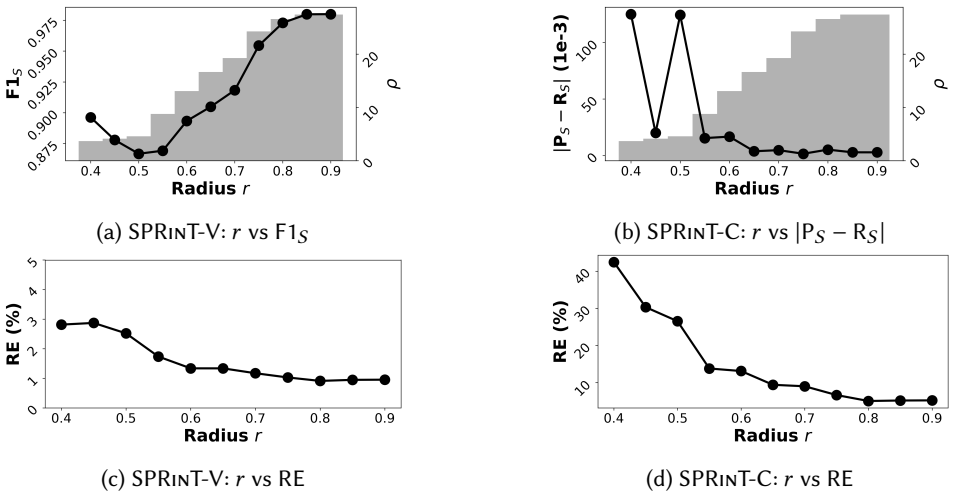


(a) s vs AVG$_S$ (b) s vs PCT$_S$

(c) SPRINT-V (d) SPRINT-C

(e) SPRINT-V (f) SPRINT-C

Fig. 10. Impact of s_p and s on RE using SPRINT-V for AVG (a, c, e) and SPRINT-C for PCT (b, d, f) on the eICU dataset.



(a) SPRINT-V: r vs F1$_S$ (b) SPRINT-C: r vs $|P_S - R_S|$

(c) SPRINT-V: r vs RE (d) SPRINT-C: r vs RE

Fig. 11. Impact of radius on RE using SPRINT-V for AVG (a, c) and SPRINT-C for PCT (b, d) on the eICU dataset.

certain threshold, suffice to yield accurate estimates. This phenomenon is not directly predicted by the theory and it reflects the efficiency of our framework.

4.3.5 Impact of Radius. The radius controls the sparsity of the set of NNs for a given query target. A smaller radius makes it challenging to compute AQNN with a low RE. We conduct an experiment to assess this impact on AVG and PCT. In Fig. 11, we report RE, $F1_S$, and precision-recall gap across different radii $r \in [0.4, 0.9]$. We denote ρ as the effective neighborhood size ratio, i.e., the proportion of true neighbors in S, which increases with the radius. It is visualized as bars in Fig. 11a and 11b.

For both AVG and PCT, RE is high at small radii. This can be explained by the fluctuating $F1_S$ and precision-recall gap. As r increases, the neighborhood size grows and these metrics stabilize, i.e., lower RE, higher $F1_S$, and smaller precision-recall gaps. These observations demonstrate that aggregation error is sensitive to the effective neighborhood size. Too few neighbors lead to unreliable estimates, whereas larger neighborhoods yield more stable and accurate aggregates.

4.4 Application: Hypothesis Testing

AQNNs have a natural downstream application in hypothesis testing because they provide aggregate statistics over the local neighborhood of a query target, which are key inputs for statistical inference. Hypothesis testing is a statistical method used to evaluate whether there is sufficient evidence in a sample to support or reject a specific claim about a population. When the population of interest is the neighborhood of a query target, this nearest neighbor hypothesis (NNH) is formulated directly from an AQNN. Hence, accurate AQNN results are essential for accurate hypothesis testing. Testing NNHs enables practitioners to uncover meaningful patterns, validate assumptions, and make reliable and trustworthy decisions. For example, it allows clinicians to assess whether the average heart rate of patients similar to a given individual significantly deviates from normal, thereby guiding personalized treatment. Similarly, it enables instructors to evaluate whether the average grade of students with similar learning trajectories is significantly lower than the class average, prompting timely interventions.

To evaluate the effectiveness of SPRINT-V and SPRINT-C in this application, we perform one-sample t-tests and proportion z-tests, respectively, with hypotheses of the form:

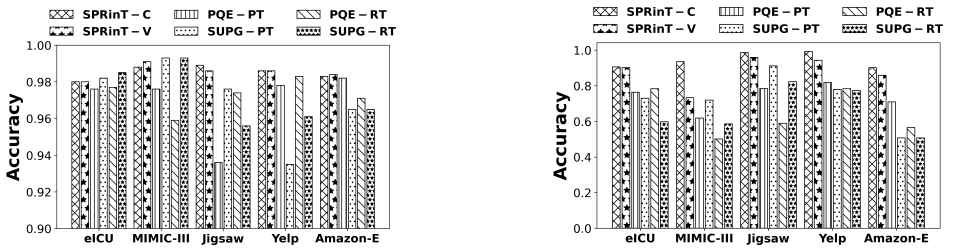
$$\text{agg}(\text{NN}_D(q, r)[\texttt{attr}]) \text{ op } c$$

where agg is AVG or PCT, op $\in \{\geq, \leq, \neq\}$, and $c \in \mathbb{R}$ is a constant. Each hypothesis comes as a pair of a null hypothesis (H_0) and an alternative one (H_a). For instance, H_0 might state that "*the average heart rate of patients similar to q is 100 bpm*", while H_a posits that "*it is strictly less than 100 bpm*". We measure performance using **accuracy**, defined as the proportion of test decisions (accept/reject) based on approximate aggregates that match those based on true aggregates:

$$\text{Accuracy} = \frac{1}{k} \sum_{i=1}^{k} \mathbb{1}_{H(\text{agg}_D)==H(\widetilde{\text{agg}_S})}$$

where $k = 30$ is the number of samples and $H(\cdot)$ is the accept/reject test decision. The constant c is set as a multiple of the ground-truth aggregate, i.e., $c = \textit{factor} \times$ ground truth, with the *factor* ranging from 0.5 to 1.5 in steps of 0.05. Accuracy is averaged over 10 random queries under two operators ($\geq$ and $\leq$).

Fig. 12 presents hypothesis testing accuracy across datasets and algorithms. As shown in Fig. 12a and 12b, testing accuracy closely tracks RE performance in § 4.3.1. For PCT, SPRINT-C achieves the lowest RE and highest testing accuracy across all datasets. Similarly, for AVG, SPRINT-V outperforms baselines on Amazon-E in both RE and testing accuracy. Methods with low RE (e.g., SUPG-RT, PQE-RT) tend to yield higher testing accuracy. These results validate that low-error AQNN estimation directly enables accurate downstream hypothesis testing. Combined with SPRINT's low computational cost (§ 4.2.1 and § 4.2.2), these support our framework's practicality and scalability for real-world NNH applications.



(a) t-test (AVG) (b) Proportion z-test (PCT)

Fig. 12. Hypothesis testing performance.

4.5 Deployment Considerations

Selecting appropriate values for the sample size s and pilot sample size s_p involves balancing cost and error. The theoretical bounds in § 3.3 and § 3.4 provide guidelines for minimal values of s and s_p to meet target error guarantees in practice.

For deployment, SPRinT can be integrated into a query engine where proxy and oracle embeddings need to be recomputed or refreshed at query time. To further optimize latency in time-sensitive applications, the search tolerance parameters (ω_V and ω_C) can be tuned to accelerate convergence while maintaining acceptable approximation quality.

Finally, although an AQNN computes aggregates with respect to a single designated query target, it can be extended to support SQL-style group-by semantics. Here, each group key in a group-by query may be treated as a query target, which enables an AQNN to compute aggregates over the neighborhood of each target in the embedding space. This extension is particularly useful for computing AQNNs over multiple query targets in batch.

5 Related Work

Approximate Query Processing (AQP). There is a rich body of prior work in the DB community on AQP methods that aim to compute approximate answers to data intensive Online Analytical Processing (OLAP) queries with high accuracy and low computational overhead. We note that AQNNs differ from OLAP queries fundamentally in both scope and operation.

AQP techniques can generally be categorized into two types: online and offline. Online methods dynamically sample data at query time to compute approximate answers [2]. Offline methods, on the other hand, rely on precomputed synopses such as wavelets to answer queries efficiently [6]. In contrast, AQNNs operate in embedding spaces where aggregation accuracy depends on both sampling and neighbor selection quality. Existing AQP methods do not address these challenges.

Nearest Neighbor Search methods can be categorized into exact and approximate approaches. Common methods, such as brute force, k-d trees, and cell techniques, are conceptually straightforward [5, 9] but impractical due to their high computational cost. Exact NN search typically relies on accurate embeddings generated by expensive oracle models and builds indexes over the entire database, leading to significant computational overhead [21].

As databases and data dimensionality grow, approximation algorithms have been used to balance accuracy and efficiency. These approaches include hashing-based [12], tree-based [29], and index-based methods [4]. Additionally, [13] proposes a locality-sensitive hashing based approach and delivers sublinear query times and polynomial time processing costs, while [14] proposes space-efficient data structures guaranteeing compactness for Euclidean distances under specific accuracy

constraints. FLANN [25] is a library leveraging randomized kd-trees and priority search k-means trees for scalable approximate NN finding over large datasets.

Unlike our work, these approaches assume that the ground-truth representations (aka oracle embeddings) of data objects are precomputed and can be efficiently accessed. In many real-world scenarios, however, embeddings need to be recomputed at query time because data evolves rapidly, which poses a unique challenge. To reduce the embedding generation cost, some recent works leverage cheap proxy models to approximate ground-truth oracle labels and find accurate approximations of NNs [17, 18, 20, 23]. For instance, Lai et al. propose probabilistic Top-K to train proxy models to generate oracle label distribution and output approximate Top-K solutions for video analytics [20]. In [18], the authors introduce statistical accuracy guarantees, such as meeting a minimum precision or recall target with high probability, for approximate selection queries. Recently, Ding et al. introduce a framework for approximate FRNN queries that leverages proxy models to minimize the reliance on expensive oracle models [8]. They propose four algorithms to find high-quality NNs for precision-target and recall-target queries.

Notice that while SPRINT-V and SPRINT-C build on PQE-PT [8] for NN selection, our goal is fundamentally different: PQE-PT focuses on retrieving a high-quality neighbor set, whereas we aim to minimize the aggregation error of the *aggregate value* over neighbors. Building on this, our methods introduce two key innovations. First, unlike PQE-PT, which assumes a *given* precision target and identifies neighbors that satisfy it, our algorithms *dynamically optimize both precision and recall targets*. Second, whereas PQE-PT focuses on achieving a given precision target while maximizing recall, our methods explore the precision-recall space to directly optimize higher-level objectives for accurate approximation of aggregate results: SPRINT-V maximizes the F1 score, and SPRINT-C equalizes precision and recall. None of the prior algorithms, including PQE-PT, can support these objectives.

6 Conclusion

We presented SPRINT, a scalable framework for answering AQNNs, a new class of queries that compute aggregates over neighborhoods defined by learned representations. Exact evaluation of AQNNs using high-quality oracle embeddings is often costly, especially when data evolves rapidly. Hence, SPRINT combines sampling with a judicious mix of high-quality oracle and efficient proxy embeddings to approximate aggregates with low error and computational cost. We categorized AQNNs into value- and count-sensitive queries and designed tailored strategies, SPRINT-V and SPRINT-C, to optimize F1 score or precision-recall balance. Extensive experiments as well as our theoretical analysis demonstrate that SPRINT enables accurate and efficient aggregation, which also translates to high accuracy in downstream applications such as hypothesis testing.

SPRINT supports aggregation functions stable under sampling, functions like `MIN`, `MAX` and `MEDIAN` pose unique challenges due to outlier sensitivity and rank instability. Extending SPRINT to handle such queries with theoretical guarantees is an important future direction. We also plan to explore sparsity-aware sampling for diverse neighborhood structures and attribute-value-aware heuristics that directly minimize aggregation error.

Acknowledgments

This work was partially supported by DataGEMS, funded by the European Union's Horizon Europe Research and Innovation programme, under grant agreement No 101188416. Carrie Wang and Reynold Cheng were supported by the Research Grant Council of Hong Kong (RGC Project HKU 17202325), the University of Hong Kong (Project 2409100399), and the HKU Faculty Exchange Award 2024 (Faculty of Engineering). Lakshmanan's research was supported in part by a grant from the Natural Sciences and Engineering Research Council of Canada (Grant Number RGPIN-2020-05408).

References

[1] 2015. Yelp Dataset. https://www.yelp.com/dataset.

[2] Sameer Agarwal, Barzan Mozafari, Aurojit Panda, Henry Milner, Samuel Madden, and Ion Stoica. 2013. BlinkDB: queries with bounded errors and bounded response times on very large data. In *Eighth Eurosys Conference 2013, EuroSys '13, Prague, Czech Republic, April 14-17, 2013*. ACM, 29–42. doi:10.1145/2465351.2465355

[3] Michael R. Anderson, Michael J. Cafarella, Germán Ros, and Thomas F. Wenisch. 2019. Physical Representation-Based Predicate Optimization for a Visual Analytics Database. In *35th IEEE International Conference on Data Engineering, ICDE 2019, Macao, China, April 8-11, 2019*. IEEE, 1466–1477. doi:10.1109/ICDE.2019.00132

[4] Akhil Arora, Sakshi Sinha, Piyush Kumar, and Arnab Bhattacharya. 2018. Hd-index: Pushing the scalability-accuracy boundary for approximate knn search in high-dimensional spaces. *arXiv preprint arXiv:1804.06829* (2018).

[5] Jon L Bentley. 1975. *A survey of techniques for fixed radius near neighbor searching*. Technical Report. Stanford University, Stanford, CA, USA.

[6] Kaushik Chakrabarti, Minos N. Garofalakis, Rajeev Rastogi, and Kyuseok Shim. 2000. Approximate Query Processing Using Wavelets. In *VLDB 2000, Proceedings of 26th International Conference on Very Large Data Bases, September 10-14, 2000, Cairo, Egypt*. Morgan Kaufmann, 111–122.

[7] cjadams, Jeffrey Sorensen, Julia Elliott, Lucas Dixon, Mark McDonald, nithum, and Will Cukierski. 2017. Toxic Comment Classification Challenge. https://kaggle.com/competitions/jigsaw-toxic-comment-classification-challenge.

[8] Dujian Ding, Sihem Amer-Yahia, and Laks V. S. Lakshmanan. 2022. On Efficient Approximate Queries over Machine Learning Models. *Proc. VLDB Endow.* 16, 4 (2022), 918–931. doi:10.14778/3574245.3574273

[9] Jerome H. Friedman, Jon Louis Bentley, and Raphael A. Finkel. 1977. An Algorithm for Finding Best Matches in Logarithmic Expected Time. *ACM Trans. Math. Softw.* 3, 3 (1977), 209–226. doi:10.1145/355744.355745

[10] Wassily Hoeffding. 1994. Probability inequalities for sums of bounded random variables. *The collected works of Wassily Hoeffding* (1994), 409–426.

[11] Yupeng Hou, Jiacheng Li, Zhankui He, An Yan, Xiusi Chen, and Julian McAuley. 2024. Bridging Language and Items for Retrieval and Recommendation. *arXiv preprint arXiv:2403.03952* (2024).

[12] Qiang Huang, Jianlin Feng, Yikai Zhang, Qiong Fang, and Wilfred Ng. 2015. Query-Aware Locality-Sensitive Hashing for Approximate Nearest Neighbor Search. *Proc. VLDB Endow.* 9, 1 (2015), 1–12. doi:10.14778/2850469.2850470

[13] Piotr Indyk and Rajeev Motwani. 1998. Approximate nearest neighbors: towards removing the curse of dimensionality. In *Proceedings of the thirtieth annual ACM symposium on Theory of computing*. 604–613.

[14] Piotr Indyk and Tal Wagner. 2018. Approximate nearest neighbors in limited space. In *Conference On Learning Theory*. PMLR, 2012–2036.

[15] Alistair EW Johnson, Tom J Pollard, Lu Shen, Li-wei H Lehman, Mengling Feng, Mohammad Ghassemi, Benjamin Moody, Peter Szolovits, Leo Anthony Celi, and Roger G Mark. 2016. MIMIC-III, a freely accessible critical care database. *Scientific data* 3, 1 (2016), 1–9.

[16] José F. Rodrigues Jr., Marco A. Gutierrez, Gabriel Spadon, Bruno Brandoli, and Sihem Amer-Yahia. 2021. LIG-Doctor: Efficient patient trajectory prediction using bidirectional minimal gated-recurrent networks. *Inf. Sci.* 545 (2021), 813–827. doi:10.1016/J.INS.2020.09.024

[17] Daniel Kang, John Emmons, Firas Abuzaid, Peter Bailis, and Matei Zaharia. 2017. NoScope: Optimizing Deep CNN-Based Queries over Video Streams at Scale. *Proc. VLDB Endow.* 10, 11 (2017), 1586–1597. doi:10.14778/3137628.3137664

[18] Daniel Kang, Edward Gan, Peter Bailis, Tatsunori Hashimoto, and Matei Zaharia. 2020. Approximate Selection with Guarantees using Proxies. *Proc. VLDB Endow.* 13, 11 (2020), 1990–2003.

[19] Sang Gyu Kwak and Jong Hae Kim. 2017. Central limit theorem: the cornerstone of modern statistics. *Korean journal of anesthesiology* 70, 2 (2017), 144–156.

[20] Ziliang Lai, Chenxia Han, Chris Liu, Pengfei Zhang, Eric Lo, and Ben Kao. 2021. Top-K Deep Video Analytics: A Probabilistic Approach. In *SIGMOD '21: International Conference on Management of Data, Virtual Event, China, June 20-25, 2021*, Guoliang Li, Zhanhuai Li, Stratos Idreos, and Divesh Srivastava (Eds.). ACM, 1037–1050. doi:10.1145/3448016.3452786

[21] Conglong Li, Minjia Zhang, David G. Andersen, and Yuxiong He. 2020. Improving Approximate Nearest Neighbor Search through Learned Adaptive Early Termination. In *Proceedings of the 2020 International Conference on Management of Data, SIGMOD Conference 2020, online conference [Portland, OR, USA], June 14-19, 2020*. ACM, 2539–2554. doi:10.1145/3318464.3380600

[22] Stuart Lloyd. 1982. Least squares quantization in PCM. *IEEE transactions on information theory* 28, 2 (1982), 129–137.

[23] Yao Lu, Aakanksha Chowdhery, Srikanth Kandula, and Surajit Chaudhuri. 2018. Accelerating Machine Learning Inference with Probabilistic Predicates. In *Proceedings of the 2018 International Conference on Management of Data, SIGMOD Conference 2018, Houston, TX, USA, June 10-15, 2018*. ACM, 1493–1508. doi:10.1145/3183713.3183751

[24] James B McQueen. 1967. Some methods of classification and analysis of multivariate observations. In *Proc. of 5th Berkeley Symposium on Math. Stat. and Prob.* 281–297.

[25] Marius Muja and David G Lowe. 2014. Scalable nearest neighbor algorithms for high dimensional data. *IEEE transactions on pattern analysis and machine intelligence* 36, 11 (2014), 2227–2240.

[26] Tom J Pollard, Alistair EW Johnson, Jesse D Raffa, Leo A Celi, Roger G Mark, and Omar Badawi. 2018. The eICU Collaborative Research Database, a freely available multi-center database for critical care research. *Scientific data* 5, 1 (2018), 1–13.

[27] Vijay Janapa Reddi, Christine Cheng, David Kanter, Peter Mattson, Guenther Schmuelling, Carole-Jean Wu, Brian Anderson, Maximilien Breughe, Mark Charlebois, William Chou, et al. 2020. Mlperf inference benchmark. In *2020 ACM/IEEE 47th Annual International Symposium on Computer Architecture (ISCA)*. IEEE, 446–459.

[28] Robert J Serfling. 1974. Probability inequalities for the sum in sampling without replacement. *The Annals of Statistics* (1974), 39–48.

[29] Chanop Silpa-Anan and Richard I. Hartley. 2008. Optimised KD-trees for fast image descriptor matching. In *2008 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2008), 24-26 June 2008, Anchorage, Alaska, USA*. IEEE Computer Society. doi:10.1109/CVPR.2008.4587638

[30] Kaitao Song, Xu Tan, Tao Qin, Jianfeng Lu, and Tie-Yan Liu. 2020. Mpnet: Masked and permuted pre-training for language understanding. *Advances in neural information processing systems* 33 (2020), 16857–16867.

[31] Vivienne Sze, Yu-Hsin Chen, Tien-Ju Yang, and Joel S Emer. 2017. Efficient processing of deep neural networks: A tutorial and survey. *Proc. IEEE* 105, 12 (2017), 2295–2329.

[32] Wenhui Wang, Furu Wei, Li Dong, Hangbo Bao, Nan Yang, and Ming Zhou. 2020. MiniLM: Deep Self-Attention Distillation for Task-Agnostic Compression of Pre-Trained Transformers. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

[33] Urchade Zaratiana, Nadi Tomeh, Pierre Holat, and Thierry Charnois. 2023. Gliner: Generalist model for named entity recognition using bidirectional transformer. *arXiv preprint arXiv:2311.08526* (2023).

Received July 2025; revised October 2025; accepted November 2025