Accelerating Approximate Analytical Join Queries over Unstructured Data with Statistical Guarantees

YUXUAN ZHU, University of Illinois Urbana-Champaign, USA
TENGJUN JIN, University of Illinois Urbana-Champaign, USA
CHENGHAO MO, University of Illinois Urbana-Champaign, USA
DANIEL KANG, University of Illinois Urbana-Champaign, USA

Analytical join queries over unstructured data are increasingly prevalent in data analytics. Applying machine learning (ML) models to label every pair in the cross product of tables can achieve state-of-the-art accuracy, but the cost of pairwise execution of ML models is prohibitive. Existing algorithms, such as embedding-based blocking and sampling, aim to reduce this cost. However, they either fail to provide statistical guarantees (leading to errors up to 79% higher than expected) or become as inefficient as uniform sampling.

We propose blocking-augmented sampling (BAS), which simultaneously achieves statistical guarantees and high efficiency. BAS optimally orchestrates embedding-based blocking and sampling to mitigate their respective limitations. Specifically, BAS allocates data tuples in the cross product into two regimes based on the failure modes of embeddings. In the regime of false negatives, BAS uses sampling to estimate the result. In the regime of false positives, BAS applies embedding-based blocking to improve efficiency. To minimize the estimation error given a budget for ML executions, we design a novel two-stage algorithm that adaptively allocates the budget between blocking and sampling. Theoretically, we prove that BAS asymptotically outperforms or matches standalone sampling. On real-world datasets across different modalities, we show that BAS provides valid confidence intervals and reduces estimation errors by up to 19× compared to state-of-the-art baselines.

CCS Concepts: • **Information systems → Online analytical processing engines**; **Join algorithms**; *Data analytics*.

Additional Key Words and Phrases: join algorithm, unstructured data, approximate query processing, sampling, machine learning

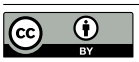
ACM Reference Format:
Yuxuan Zhu, Tengjun Jin, Chenghao Mo, and Daniel Kang. 2026. Accelerating Approximate Analytical Join Queries over Unstructured Data with Statistical Guarantees. *Proc. ACM Manag. Data* 4, 3 (SIGMOD), Article 127 (June 2026), 28 pages. https://doi.org/10.1145/3802004

1 Introduction

The increasing capability of machine learning (ML) and large language models (LLMs) allows data analysts to analyze semantically related objects across multiple unstructured datasets automatically [27, 37, 47, 53, 54, 62, 76, 78, 101]. For example, a business investor can analyze differences in stock prices among related companies [98] by computing the `AVG` over a join of two sets of company records with a join condition based on the company description (Figure 1).

To accurately process queries whose join conditions involve arbitrary semantics, an analyst can manually label or apply an ML model to each pair of records. This pairwise evaluation method

Authors' Contact Information: Yuxuan Zhu, University of Illinois Urbana-Champaign, Urbana, USA, yxx404@illinois.edu; Tengjun Jin, University of Illinois Urbana-Champaign, Urbana, USA, tengjun2@illinois.edu; Chenghao Mo, University of Illinois Urbana-Champaign, Urbana, USA, cmo8@illinois.edu; Daniel Kang, University of Illinois Urbana-Champaign, Urbana, USA, ddkang@illinois.edu.



ACM 2836-6573/2026/6-ART127
https://doi.org/10.1145/3802004

```
SELECT AVG(c1.price - c2.price) FROM companies AS c1
JOIN companies AS c2
ON NL(`Company: {c1.description} and Company: {c2.description} build similar
    products')
```

Fig. 1. An example analytical join query over unstructured data that calculates the differences in stock prices among related companies.

has demonstrated high accuracy across various applications, including small language models for entity matching [49, 65, 83] and text retrieval [37, 62, 78, 103], vision-language models for image-text retrieval [47, 48, 88], and large language models for challenging data wrangling tasks [52, 61, 69, 70, 92]. We refer to this pairwise evaluation method as the *Oracle*. Unfortunately, the Oracle method can be prohibitively expensive since the number of ML model invocations grows at least quadratically with the number of data records and grows exponentially with the number of tables to be joined.[1]

To mitigate these high costs, researchers have proposed two approximation methods that reduce the number of ML model invocations: *embedding-based blocking* and *sampling*. However, when used to process analytical join queries over unstructured data, embedding-based blocking can lead to arbitrary errors, while sampling is inefficient.

Embedding-based Blocking. In efficient entity matching (EM) systems, embedding-based blocking first uses embedding similarities to filter out data pairs that are less likely to match and then applies ML models for pairwise evaluation [49, 50, 59, 85]. In contrast to the Oracle method that applies ML models to each data pair, embedding-based blocking applies embedding models on each data record and only uses the Oracle on a reduced number of data pairs, which effectively reduces the cost of executing expensive ML models.

However, embedding models can be unreliable and are not guaranteed to precisely capture the semantics of interest [30, 45, 60, 84]. Consequently, two data records that satisfy the join condition may have a low embedding similarity (i.e., false negatives), causing embedding-based blocking to filter out such a pair of records. In this case, when used to compute data aggregations, embedding-based blocking skews aggregation results toward data with high embedding similarity. Thus, embedding-based blocking does not provide statistical guarantees. Empirically, we find that blocking can result in errors up to 79% higher than expected even when the threshold of filtering is calibrated on a validation dataset (§7.2).

Sampling. To achieve statistical guarantees, researchers have used sampling techniques to process analytical join queries with approximate semantics [3, 25, 89]. With independently and identically distributed samples, we can calculate confidence intervals (CIs) to provide statistical guarantees. However, due to the sparsity of positive pairs (Figure 7, §7.1), uniform sampling can be inefficient, requiring a large sample size to achieve a small error. In Section 7.3, we show that achieving an average error of 5% for the Company dataset requires uniformly sampling 1.4×10^7 data pairs and processing them with the Oracle (Figure 10a).[2]

We propose *Weighted Wander Join* (WWJ) to improve the efficiency of sampling using embeddings. WWJ extends Wander Join [46], an efficient join algorithm for structured data, with an approximate index computed from embeddings. Intuitively, WWJ follows the idea of importance

[1]For example, running the Oracle on an entity matching dataset, the Company dataset from the Magellan benchmark [40], costs $709K with GPT-4o or $43K with GPT-4o mini. The pairwise evaluation results in 567 million tokens. We estimated the monetary cost according to the state-of-the-art prompt [61] and the pricing from OpenAI [63].
[2]On average, 1.4×10^7 pairs contains 20B tokens. Processing them would cost $26K with GPT-4o or $1.6K with GPT-4o-mini.

sampling to reduce sampling variance by upweighting the probability of sampling data pairs with high embedding similarities [39]. Nevertheless, the effectiveness of WWJ depends heavily on the precision of the embeddings. We empirically show that WWJ becomes as inefficient as uniform sampling, if the embedding similarities contain many false positives (Figure 10, §7.3).

Blocking-augmented Sampling (BAS). To address the limitations of standard statistical methods over imperfect embeddings, we propose *Blocking-augmented Sampling*. Our core novelty lies in the dynamic orchestration of two techniques based on the specific failure modes of embedding models: false negatives (which break blocking) and false positives (which break importance sampling).

BAS addresses this by treating the blocking threshold not as a static filter, but as a dynamic decision boundary that splits the search space into two regimes: (1) a blocking regime (high similarity, high false positive risk), where we deterministically execute the Oracle to eliminate variance caused by frequent non-matches; (2) a sampling regime (low similarity, high false negative risk), where we apply importance sampling to correct the bias introduced by blocking. Unlike standard stratified sampling where strata are fixed, BAS solves a complex optimization problem to determine the optimal boundary between these regimes.

Given a fixed budget for Oracle executions, it is challenging to optimally allocate the data pairs into two regimes and assign the budget between blocking and WWJ. Empirically, the optimal allocation that minimizes the estimation error varies significantly across different datasets. Using the optimal allocation can result in estimation errors that are up to 99% smaller than the worst allocation (Figure 13, §7.5). Theoretically, we show that the optimal budget allocation depends on the sampling variance of WWJ on different parts of the dataset. Unfortunately, the sampling variance is unknown unless the Oracle is executed on the entire dataset.

To automatically determine the optimal allocation, we propose an adaptive allocation algorithm via two-stage stratified sampling. Initially, BAS stratifies data tuples based on the embedding similarity. In the first stage, BAS applies pilot sampling [38] to estimate the sampling variance of each stratum. Based on these estimates, BAS determines the optimal allocation that minimizes the overall estimated sampling variance. In the second stage, BAS executes WWJ and blocking according to the allocated regime and budget. We prove that BAS not only converges to the optimal budget allocation but also outperforms or matches standalone WWJ asymptotically. Finally, to obtain valid CIs as statistical guarantees, BAS adopts bootstrapping that estimates statistical distributions via resampling [26, 73]. We theoretically justify the validity and empirically demonstrate the coverage of the derived CIs.

We implement WWJ and BAS in a prototype system called JoinML to accelerate approximate analytical join queries over unstructured data. We evaluate JoinML using six real-world datasets and two synthetic datasets, including text, image, and bimodal data. Our results reveal that BAS reduces the root mean squared error by up to 19× compared to the uniform sampling, blocking and prior systems for semantic operators. We also show that the CIs provided by BAS achieve the expected coverage. In addition to analytical queries, we show that BAS can improve the precision of selection queries over unstructured data by up to 69%, compared to baselines. In our ablation study, we show that the allocation algorithm closely approaches optimal performance. Our contributions are as follows:

(1) We develop JoinML algorithms, WWJ and BAS, that can accurately and efficiently answer approximate analytical join queries with statistical guarantees.
(2) We prove the convergence rate of BAS to the optimal allocation and that BAS outperforms or matches standalone sampling asymptotically.
(3) We implement and evaluate JoinML in real-world text, images, and bimodal datasets, reducing root mean squared errors by up to 19× compared to state-of-the-art baselines.

```
SELECT {AVG | SUM | COUNT} {field | EXPR(field)}
FROM table_name1 JOIN table_name2 JOIN ...
ON NL(join_condition) AND ...
ORACLE BUDGET b WITH PROBABILITY p
```

Fig. 2. Query syntax of JoinML.

2 Query Syntax and Semantics

JoinML accelerates analytical queries involving joins and similarity-based join conditions. Currently, we support linear aggregates, including `COUNT`, `SUM`, and `AVG`. We illustrate the query syntax in Figure 2. Besides a SQL query, users can specify an execution budget of Oracle and the coverage probability for the CI. To understand the semantics of a JoinML query, we introduce the concepts of Oracle and embedding similarity.

Oracle. We define Oracle as a method that operates on a pair of records to determine if they satisfy the join condition. We assume that the Oracle is the state-of-the-art method that outputs the ground-truth label. For example, the Oracle method can be a foundation model that decides if two sentences contain the same entity [61], a fine-tuned model that accesses whether two images show the same car [53, 54], or a human labeler for particularly challenging tasks [18, 71, 104].

Embedding Similarity. We define the similarity between two records t_1 and t_2 as the cosine similarity of their embedding vectors:

$$\text{sim}\left(E(t_1), E(t_2)\right) = \frac{E(t_1)^\top \cdot E(t_2)}{|E(t_1)| \cdot |E(t_2)|}$$

The embedding vectors $E(\cdot)$ can be obtained by applying a pre-trained embedding model on each data record individually. Therefore, compared to Oracle methods that are executed on each pair of records, using embedding similarity incurs significantly lower computational costs. However, since the embedding model does not consider the joint semantic across data records, the embedding similarity can be less accurate than the Oracle.

Query Semantics. A JoinML query will output an estimate $\hat{\mu}$ of the answer and a CI $[l, u]$. Suppose μ is the result of processing the query exhaustively via the Oracle. With a budget b and a probability p, we provide the following guarantees: (1) The Oracle will not be executed on more than b pairs; (2) $\mathbb{P}\left[l \leq \mu \leq u\right] \geq p$. Our query semantics and guarantees are similar to prior AQP over unstructured data without joins [33, 35, 36].

3 Use Cases

Analytical queries with joins are increasingly prevalent in analytics over unstructured data, such as plagiarism analysis [6, 24, 81], business analysis [43, 98], and traffic condition monitoring [13, 97]. In addition to the business analysis query introduced in Section 1, we demonstrate two more typical use cases. In the example queries, we can use an oracle budget of 1,000,000 and a confidence of 0.95.

Plagiarism Analysis. Sentence-level analysis is a common approach in plagiarism detection that involves comparing each sentence of an input article with every sentence in a reference collection [24, 93]. The plagiarism score of the article can be calculated as the percentage of sentences that are paraphrased from the reference collection. To compute the percentage, we need to join the sentences of the article with those in the reference collection and count the number of sentence pairs that are considered paraphrased, as shown in Figure 3.

```
SELECT COUNT(*)
FROM article JOIN db
ON NL(`{article.sentence} is paraphrased from {db.sentence}.')
```

Fig. 3. An example query that estimates the paraphrasing rate of an article.

```
SELECT AVG(video1.ts - video2.ts)
FROM video1 JOIN video2
ON NL(`Frame {video1.frame} and Frame {video2.frame} contains the same car.')
```

Fig. 4. An example query that estimates the travel time of a road segment.

Traffic Analysis. An urban planner may be interested in analyzing the average traffic time across a road segment by calculating the time for the same vehicle to travel from one end to the other [13]. To find the time difference, the urban planner needs to join two videos and identify common vehicles, as shown in Figure 4.

Join Order Optimization. Optimizing multi-way semantic joins requires accurate cardinality estimates to determine efficient execution orders. However, for unstructured predicates requiring ML evaluation, deriving these statistics is prohibitively expensive. BAS resolves this by efficiently estimating COUNT with tight confidence intervals using a minimal budget. These estimates allow the query optimizer to reliably select the optimal join order, preventing performance regressions caused by poor planning without requiring full pairwise evaluation (Section 7.4).

4 Background

Existing algorithms in approximate query processing (AQP) and entity matching (EM) can be applied to accelerate analytical join queries over unstructured data. In this section, we introduce two typical algorithms: sampling and blocking.

4.1 Sampling

Sampling techniques are widely used in AQP to reduce query costs while providing CIs as statistical guarantees. Prior work has developed efficient sampling algorithms for analytical join queries with join keys, such as correlated sampling [89], universe sampling [29], and Wander Join [46]. We can apply these algorithms to accelerate joins over unstructured data.

Limitations. Existing efficient sampling algorithms for joins over structured data require exact indices or hash functions on the join keys. However, there is no join key for semantic join queries over unstructured data. To obtain semantic indices or hash functions, one approach is to apply the Oracle method across all data tuples, which is prohibitively expensive. Without indices or hash functions, we can only use uniform sampling, which is inefficient because positive pairs are often sparse in the cross product of tables. Empirically, given a fixed error budget, uniform sampling can lead to 2.6× higher error than our proposed method (Figure 5).

4.2 Embedding-based Blocking

To reduce the number of pairwise comparisons, traditional blocking algorithms typically use a blocking phase to prune the search space [7, 11, 19, 23, 44, 57, 66–68]. While traditional blocking algorithms cluster records into disjoint buckets (blocks), modern deep learning approaches use

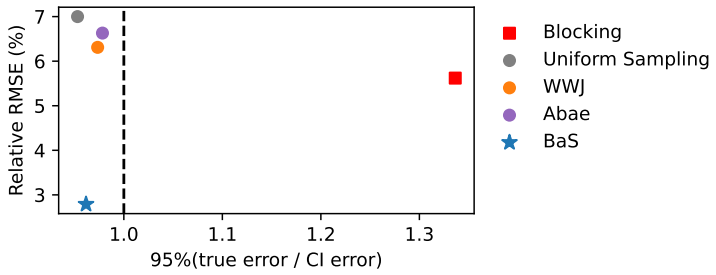


Fig. 5. Given a COUNT query with an Oracle budget of 5,000,000 and a probability of 95%, sampling algorithms leads to high estimation error while blocking fails to achieve statistical guarantees. Achieving statistical guarantees means that the 95th percentile of the true error is less than the error bounded by a 95% CI (to the left of the dashed line).

Algorithm 1: Emedding-based blocking for a SUM aggregation over a join of two tables.

Input : input tables T_1, T_2, Oracle $O(\cdot)$, embedding $E(\cdot)$, attribute $g(\cdot)$, embedding similarity $\text{sim}(\cdot,\cdot)$, similarity score threshold τ

1 $S \leftarrow \emptyset$
2 **for** *each data record* t_1 *in* T_1 **do**
3 **for** *each data record* t_2 *in* T_2 **do**
4 $\tau_i = \text{sim}\left(E(t_1), E(t_2)\right)$
5 **if** $\tau_i \geq \tau$ **then**
6 $S \leftarrow S \cup \{(t_1, t_2)\}$ // Forming data blocks
7 $X \leftarrow \{O(s) \cdot g(s) : s \in S\}$
8 **return** $\sum_{i=1}^{|X|} X_i$

embedding-based blocking [49, 59, 68, 85]. Throughout this paper, blocking refers to this embedding-based filtering with thresholds. It implicitly defines a block of candidate pairs as the set of tuples whose embedding similarities exceed a threshold τ. By filtering out pairs below τ (Line 5 in Algorithm 1 and 2), the algorithm effectively blocks unrelated data from expensive pairwise comparison. We show the procedure for estimating SUM in Algorithm 1, where the Oracle is executed only on the candidate block.

In practice, we often have a budget for Oracle invocations and may need to join more than two tables. To address these, we present a direct extension of the original embedding-based blocking as a baseline in Algorithm 2. As shown, to satisfy the predefined Oracle budget, we perform random sampling over the resulting set of data tuples after filtering and then estimate the target statistic using the sample. We obtain a point estimate and a CI via a standard approach. Furthermore, to support k-table joins ($k \geq 2$) in general, we calculate the similarity of a k-tuple as the joint probability that the k tuples match, assuming embedding similarity approximates the probability that two data records match.

Limitations. Unfortunately, the predefined threshold τ does not guarantee zero false negatives on the evaluation dataset, even when we calibrate it on a validation dataset. Therefore, there is no guarantee on the error of the point estimate or the validity of the CI. Empirically, we show that blocking can result in true errors higher than the errors bounded by the provided CIs (Figure 5).

Algorithm 2: Embedding-based blocking for SUM with a predefined Oracle budget.

Input : input tables $T_1, \ldots, T_k$, Oracle $O(\cdot)$, embedding $E(\cdot)$, attribute $g(\cdot)$, embedding similarity $\text{sim}(\cdot, \cdot)$, similarity score threshold τ, oracle budget b, probability p

$D \leftarrow \text{CrossProduct}(T_1, \ldots, T_k)$
$S \leftarrow \emptyset$
for *each k-tuple* $(t_1, \ldots, t_k)$ *in* D **do**
  $\tau_i \leftarrow \prod_{j=1}^{k-1} \text{sim}\left(E(t_j), E(t_{j+1})\right)$
  if $\tau_i \geq \tau$ **then**
    $S \leftarrow S \cup \{(t_1, \ldots, t_k)\}$ // Forming data blocks
if $|S| \leq b$ **then**
  $X \leftarrow \{O(s) \cdot g(s) : s \in S\}$
  return $\sum_{i=1}^{|X|} X_i$
else
  $\tilde{S} \leftarrow \text{Sample}(S, b)$
  $X \leftarrow \{O(s) \cdot g(s) : s \in \tilde{S}\}$
  return $\text{StandardCI}(X, p, |S|)$
Function $\text{StandardCI}(X, p, N)$**:**
  $\hat{\mu} \leftarrow \frac{1}{|X|} \sum_{i=1}^{|X|} X_i$
  $\hat{\sigma} \leftarrow \frac{1}{|X|-1} \sum_{i=1}^{|X|} (X_i - \hat{\mu})^2$
  $l, u \leftarrow \hat{\mu} \mp z_{(1+p)/2} \cdot \frac{\hat{\sigma}}{\sqrt{|X|}}$
  return $N\hat{\mu}, Nl, Nu$

5 Efficient JoinML Algorithms with Guarantees

In this section, we present our novel JoinML algorithms for analytical join queries over unstructured data which overcomes the limitations of baselines, simultaneously achieving statistical guarantees and high efficiency. Our key insights are:

(1) The sparsity of positive tuples in the cross product of tables makes uniform sampling inefficient. To address it, we design a non-uniform sampling algorithm (i.e., Weighted Wander Join) with the embedding as the approximate index (§5.1).
(2) False positives of the embedding similarity compromise the efficiency of WWJ. Therefore, we augment WWJ with embedding-based blocking (§5.2) and adaptive data allocation (§5.3).

Finally, we extend our algorithms to handle selection queries and TopK heavy hitter queries with recall guarantees (§5.4).

5.1 Weighted Wander Join

As we discussed in Section 4.1, existing sampling algorithms for join queries often fall back to inefficient uniform sampling due to the difficulty and prohibitive costs of obtaining exact semantic indices or hash functions for unstructured data. For example, Wander Join, one of the state-of-the-art join algorithm for structured data, formulates join operations into random walks over the records of different join tables [46]. For each iteration of the Wander Join, we randomly walk to a record that satisfies the join conditions (Figure 6a). Knowing which records can satisfy the join

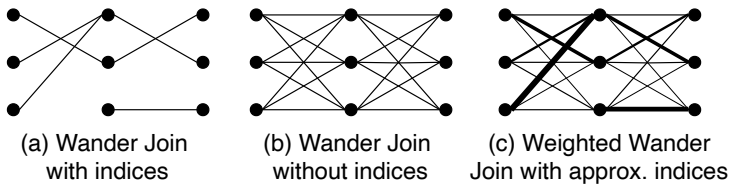


Fig. 6. Illustration of Wander Join with indices, without indices, and with approximate indices. Black dots are data records while solid connections are viable paths during random walks. The widths of edges in (c) show the probability of choosing paths.

Algorithm 3: Weighted Wander Join algorithm (SUM).

Input : input tables $T_1, \ldots, T_k$, embedding $E(\cdot)$, attribute function $g(\cdot)$, embedding similarity $\text{sim}(\cdot,\cdot)$, oracle budget b, probability p

1 $S \leftarrow \emptyset$
2 **for** *i in* $1, \ldots, b$ **do**
3 $\quad t_{i,1} \leftarrow \text{Sample}(T_1, 1)$
4 $\quad$ **for** *j in* $2, \ldots, k$ **do**
5 $\quad\quad W_{i,j} \leftarrow \{\text{sim}\left(E(t_{i,j-1}), E(t)\right) : t \in T_j\}$
6 $\quad\quad t_{i,j} \leftarrow \text{WeightedSample}(T_j, 1, W_{i,j})$
7 $\quad S \leftarrow S \cup \{(t_{i,1}, \ldots, t_{i,k})\}$
8 $\quad W(s) \leftarrow \frac{1}{|T_1|} \prod_{j=2}^{k} \text{sim}\left(E(t_{i,j-1}), E(t_{i,j})\right)$
9 $X \leftarrow \left\{O(s) \cdot g(s) \cdot W(s)^{-1} : s \in S\right\}$
10 **return** $\text{StandardCI}(X, p, \prod_{i=1}^{k} |T_i|)$

condition requires indices over the join columns. Without such indices, Wander Join would simply randomly choose any record in each iteration, which is equivalent to uniform sampling (Figure 6b).

While exact semantic indices are expensive to compute for unstructured data, we can use embeddings to construct approximate indices and extend the Wander Join algorithm accordingly. Specifically, in each iteration, instead of randomly sampling a record satisfying join conditions, we sample a record with a probability proportional to the embedding similarity (Figure 6c). We refer to this algorithm as the *Weighted Wander Join* (WWJ) and illustrate the procedure for estimating a SUM aggregate in Algorithm 3.

Connection to Importance Sampling. We find that WWJ is an instantiation of importance sampling [39]. To understand why, we consider a general case of k join tables: $T_1, \ldots, T_k$. WWJ is equivalent to importance sampling with the sampling space being the set of all tuples in the cross product of $T_1, \ldots, T_k$. Each tuple in the sampling space corresponds to a complete walk from T_1 to T_k. Given a join order, the sampling weight of each tuple is the multiplication of embedding similarity of pairs:

$$W(t_1, \ldots, t_k) = \frac{1}{|T_1|} \cdot \prod_{i=2}^{k} \text{sim}\left(E\left(t_{i-1}\right), E\left(t_i\right)\right)$$

where we normalize the weight by the size of T_1 since T_1 is sampled uniformly. Based on the statistical equivalence between WWJ and importance sampling, we can obtain unbiased estimates and valid CIs in a standard way [39], as shown in Algorithm 3, Lines 9-10.

While WWJ requires computing embedding similarities at each step without pre-built index exists, it is more efficient than importance sampling. Naive sampling requires enumerating the cross-product, resulting in exponential complexity $O(\prod_{i=2}^{k} |T_i|)$. In contrast, WWJ decomposes the problem into sequential steps. In each iteration, we only compute similarities between the current tuple and the N tuples of the next table. Consequently, for a budget b, the complexity scales as $O(b \cdot \sum_{i=1}^{k} |T_i|)$. This cost is linear (or quadratic depending on b) with respect to table sizes, but crucially avoids the exponential explosion of the cross-product.

Limitations. The statistical efficiency of WWJ depends on the precision of the embedding similarity. When the tuples with high similarity scores all satisfy the join conditions, WWJ converges to Wander Join with exact indices, achieving the best efficiency. However, when there are many false positives (i.e., tuples with high similarity scores violate join conditions), WWJ can repetitively sample negative pairs, leading to an efficiency comparable or even worse than uniform sampling. We empirically show such limitation of WWJ in Section 7.

We find that false positives are often unavoidable, particularly when the pre-trained embedding fails to accurately capture the semantics of the join condition. For example, a pre-trained embedding for text data can capture the semantic characteristic. However, two records that appear semantically similar might not satisfy the join condition that requires two companies to build the same product. In Section 7.6, we show that such issues persist even in state-of-the-art embeddings [20, 42, 51, 79, 94].

5.2 Blocking-augmented Sampling

To improve the efficiency of WWJ when there are many false positives, we propose *Blocking-augmented Sampling* (BAS), which augments WWJ with the blocking algorithm. Specifically, we separate the sampling space into two regimes:

(1) The *sampling regime* (D_s), characterized by many false negatives, where we apply WWJ.
(2) The *blocking regime* (D_b), characterized by many false positives, where we execute the Oracle.

Intuition. BAS mitigates the limitation of both WWJ and blocking. First, BAS directly applies the Oracle on the blocking regime to mitigate the high sampling variance due to repetitively sampling false positives. Second, BAS can provide statistical guarantees since WWJ does not ignore false negatives. To execute WWJ only on the sampling regime, we can simply set the sampling probability of tuples in the blocking regime to zero. Finally, we combine the results of both regimes to obtain a final estimate of the aggregate.

Formal Description. We then describe the estimation procedure formally. Suppose we have an Oracle budget b and a separation of the entire sampling space $D = D_s \cup D_b$. Then, we execute WWJ on D_b with a sample size of $b - |D_b|$ and execute the Oracle on D_s. Finally, we can estimate the aggregate over D by combining the estimate of the aggregate over D_s and the true value of the aggregate over D_b:

$$\widehat{\texttt{COUNT}} = \widehat{\texttt{COUNT}}_s + \texttt{COUNT}_b, \quad \widehat{\texttt{SUM}} = \widehat{\texttt{SUM}}_s + \texttt{SUM}_b \tag{1}$$

$$\widehat{\texttt{AVG}} = \left(\widehat{\texttt{SUM}}_s + \texttt{SUM}_b\right) \Big/ \left(\widehat{\texttt{COUNT}}_s + \texttt{COUNT}_b\right) \tag{2}$$

We observe that the combined estimators for COUNT and SUM are unbiased since the estimators over D_s is linear. However, the combined estimator for AVG is a ratio estimator, which is asymptotically unbiased at the order of $O(1/b)$ as proved in the Section 6.8 of [12]. To reduce the bias to the order

of $O(1/b^2)$, we apply the bias correction based on the Taylor expansion [12] to our AVG estimator:

$$\widehat{\mathsf{AVG}}_2 = \widehat{\mathsf{AVG}} \cdot \left(1 - \frac{1}{b - |D_b|} \frac{Var\left[\widehat{\mathsf{COUNT}}\right]}{\widehat{\mathsf{COUNT}}^2}\right) \tag{3}$$

Challenges. To establish the advantages of BAS over WWJ and blocking, we must tackle two challenges. First, we need to find an optimal separation such that the estimation error is minimized given an Oracle budget b. Second, we need to obtain valid CIs for our combined estimators, achieving a specified coverage probability p. To address them, we introduce our adaptive allocation algorithm in the following section.

5.3 BAS with Adaptive Allocation

Given an Oracle budget b, an optimal allocation should minimize the mean squared error (MSE) of the aggregate estimate. In this allocation optimization problem, we need to decide whether a tuple in the sampling space D should be allocated to the sampling regime or the blocking regime. Unfortunately, tuple-level allocation can be prohibitively expensive due to the size of the sampling space. To address it, we formulate the optimization problem into a stratum-level allocation problem, where we first stratify D into a set of strata based on the similarity scores of tuples. In this case, our allocation algorithm determines which strata should be allocated to the blocking regime to minimize the overall MSE in four steps: stratification, adaptive allocation, sampling or blocking execution, and resampling. We explain the entire procedure using a SUM aggregate.

Stratification. Initially, we divide D into a *maximum blocking regime* and a *minimum sampling regime* (D_0). The maximum blocking regime contains tuples with the top $b_2 = \alpha \cdot b$ similarity scores, where α is a parameter between 0 and 1 to control the size of the maximum blocking regime. Based on α, we reserve b_2 for potential blocking and the remaining budget $b_1 = b - b_2$ for adaptive allocation. Next, we stratify the maximum blocking regime into K strata ($D_1, \ldots, D_K$) with equal sizes (Alg. 4, Lines 1-5). Following prior work about stratified sampling for approximate query processing [36], BAS automatically determines the number of strata K to ensure that each stratum has an Oracle budget of at least 1,000.

With the stratification, we formulate the allocation optimization problem. Given the stratified sampling space $D_0, \ldots, D_K$, an aggregate AGG, similarity scores W, and an Oracle budget b, our target is to find an optimal allocation β that minimizes the MSE of the estimated AGG. The allocation β specifies a subset of the maximum blocking regime ($D_1, \ldots, D_K$) that will directly execute the Oracle while the rest of strata (including D_0) will execute WWJ. We formulate this optimization problem as follows:

$$\beta^* = \underset{\beta \subset \{1,\ldots,K\}}{\arg\min}\ MSE_{\mathsf{AGG}}(D, \beta, W, b) \tag{4}$$

Adaptive Allocation via Pilot Sampling. To address the optimization problem, we first calculate the MSE given an allocation β. We observe that our estimator of the SUM aggregate (Equation 1) is unbiased. Therefore, we can calculate overall MSE as the summation of sampling variance of each stratum in the sampling regime:

$$MSE_{\mathsf{SUM}} = \sum_{0 \le i \le k, i \notin \beta} Var\left[\frac{|D_i|}{n_i} \sum_{s \in S_i} \frac{O(s)g(s)}{W(s)}\right] \tag{5}$$

where S_i is the sampled tuple of D_i following the procedure of WWJ and n_i is the (hypothetical) Oracle budget assigned to stratum D_i. For a stratum in the blocking regime, $n_i = |D_i|$. For a stratum

Algorithm 4: BAS with Adaptive Allocation for a SUM Aggregate

Input : input tables $T_1, \ldots, T_k$, similarity score $W(\cdot)$, attribute $g(\cdot)$, number of stratum K, maximum blocking ratio α, oracle budgets b_1 and b_2, probability p

/* Stratification */

1 $D' \leftarrow \text{Top}(\text{CrossProduct}(T_1, \ldots, T_k), W, \alpha(b_1 + b_2))$

2 $D' \leftarrow \text{SortDesc}(D', W)$

3 $D_0 \leftarrow D \backslash D'$

4 **for** i *in* $1, \ldots, K$ **do**

5 $\quad D_i \leftarrow D' \left[\alpha(i-1)(b_1+b_2)/K : \alpha \cdot i(b_1+b_2)/K\right]$

/* Adaptive Allocation via Pilot Sampling */

6 **for** i *in* $0, \ldots, K$ **do**

7 $\quad n_i^{(1)} \leftarrow \frac{\sum_{x \in D_i} W(x)}{\sum_{x \in D} W(x)} b_1$

8 $\quad S_i^{(1)} \leftarrow \text{WeightedSample}(D_i, n_i, W)$

9 $\quad X_i^{(1)} \leftarrow \left\{ g(s) \cdot O(s) \cdot W(s)^{-1} : s \in S_i^{(1)} \right\}$

10 $\quad \hat{\sigma}_i^2 \leftarrow |D_i|^2 \cdot Var\left[X_i^{(1)}\right]$

11 $\beta^* \leftarrow \arg\min_\beta \ \sum_{i \notin \beta} \hat{\sigma}_i^2 \big/ \text{BudgetAssign}(b_2, W, \beta, i, D)$

/* Sampling + Blocking */

12 **for** $i \notin \beta^*$ **do**

13 $\quad n_i^{(2)} \leftarrow \text{BudgetAssign}(b_2, W, \beta^*, i, D)$

14 $\quad S_i^{(2)} \leftarrow \text{WeightedSample}(D_i, n_i, W)$

15 $\quad X_i^{(2)} \leftarrow \left\{ g(s) \cdot O(s) \cdot W(s)^{-1} : s \in S_i^{(2)} \right\}$

16 **for** $i \in \beta^*$ **do**

17 $\quad S_i^{(2)} \leftarrow \text{Sample}(D_i, |D_i|)$

18 $\quad X_i^{(2)} \leftarrow \left\{ g(s) \cdot O(s) : s \in S_i^{(2)} \right\}$

/* CI via Resampling */

19 $X \leftarrow X^{(1)} \cup X^{(2)}$

20 **return** $\text{ResamplingCI}(X, p, D)$

21 **Function** $\text{BudgetAssign}(b, W, \beta, i, D)$**:**

22 $\quad$ **return** $\left(b - \sum_{j \in \beta} |D_j|\right) \frac{\sum_{s \in D_i} W(s)}{\sum_{j \notin \beta} \sum_{s \in D_j} W(s)})$

in the sampling regime, we follow the scheme of importance sampling and assign the budget proportional to the sum of similarity scores:

$$n_i = \frac{(b - \sum_{j \in \beta} |D_i|) \cdot \sum_{s \in D_i} W(s)}{\sum_{1 \le j \le K, j \notin \beta} \sum_{s \in D_j} W(s)}$$

Unfortunately, the sampling variance in Equation (5) is unknown unless we apply the Oracle to the entire sampling regime. To address this, we apply pilot sampling, a sampling procedure that is used to obtain statistics to guide the main sampling later. In our algorithm, we use pilot sampling to estimate the sampling variance with an Oracle budget of b_1 (Algorithm 4, Line 8). Then, we can estimate the optimal allocation using the estimated variance (Algorithm 4, Line 11). In Section 6,

we show that our estimated optimal allocation converges to the optimal allocation at the rate of $O(1/\sqrt{b_1})$. To avoid applying Oracle on the same data tuples twice, we cache the Oracle results obtained in the pilot sampling stage for next stages.

Sampling+Blocking. Given the optimal allocation β^*, we use the remaining Oracle budget b_2 to execute WWJ on strata that are allocated to the sampling regime and execute the Oracle on the strata that are allocated to the blocking regime. We merge the results of using the budget b_1 (for pilot sampling) and the budget b_2 to estimate the aggregate (Alg. 4, Lines 12-18.)

CI via Resampling. As we merge the sample from two dependent stages, the resulting sample is not independently and identically distributed (i.i.d.). This means we cannot calculate CIs using the Central Limit Theorem (CLT) that assumes i.i.d. data. Applying standard CLT on non-i.i.d. data naively can lead to CIs without valid coverage; for example, a 95% CI might cover the true value with a probability lower than 95%. To guarantee valid coverages, we use resampling for CIs.

We apply the bootstrap-t resampling scheme to calculate CIs with coverage guarantees [26]. Given an existing sample X, a confidence p, bootstrap-t resampling works as follows:

(1) Calculate the estimated aggregate $\hat{\mu}$ and its variance $\hat{\sigma}^2$.
(2) Sample $|X|$ data from X with replacement, resulting in a resample R_i
(3) Calculate the estimation $\hat{\mu}_i$ and the variance $\hat{\sigma}_i^2$ using R_i.
(4) Calculate the t-statistic: $t_i = \frac{\hat{\mu}_i - \hat{\mu}}{\hat{\sigma}_i}$
(5) Repeat (2)-(4) for a sufficient number of times. Following prior work [36], we use 1,000 resamples.
(6) Calculate the $\frac{1-p}{2}$ and $1 - \frac{1-p}{2}$ percentile of t-statistics, resulting in t_l and t_r, respectively.
(7) Return the CI as: $[\hat{\mu} - t_l\hat{\sigma}, \hat{\mu} - t_r\hat{\sigma}]$

Intuitively, bootstrap-t resampling estimates the CI by measuring the empirical CI when sampling from the empirical distribution of observed data. It further uses t-statistics to adjust the skewness and thus achieves the coverage guarantee at the order of $O(1/b^2)$ [26].

We justify this validity by leveraging Theorem 23.9 of [87], which states that bootstrap CIs are valid if the statistical functional is *Hadamard differentiable* with respect to the cumulative distribution function. Since our estimators for SUM and COUNT are linear combinations (and AVG is a ratio of differentiable functionals), they satisfy this condition. We defer the full proof to an extended technical report [102].

As shown in Algorithm 4, our resampling step for CI computation operates independent of the adaptive allocation. The allocation objective is to minimize MSE, which inherently optimizes accuracy regardless of whether a CI is requested. Therefore, while skipping CI computation eliminates the resampling overhead, it does not directly improve estimation accuracy. For applications constrained by total latency rather than monetary cost, the computational time saved by omitting resampling could be leveraged to increase the Oracle budget, thereby indirectly improving estimation accuracy.

Handling AVG. We estimate AVG with a ratio estimator (Eq. 2 and 3), which is asymptotically unbiased. Given the independence between SUM and COUNT, we estimate the MSE of AVG as follows:

$$MSE_{\mathsf{AVG}} = \frac{1}{n}\left(\frac{\widehat{\mathsf{SUM}}}{\widehat{\mathsf{COUNT}}}\right)^2\left(\frac{MSE_{\mathsf{COUNT}}}{\widehat{\mathsf{COUNT}}^2} + \frac{MSE_{\mathsf{SUM}}}{\widehat{\mathsf{SUM}}^2}\right)$$

where $n = b - \sum_{i\in\beta}|D_i|$ is the Oracle budget for the sampling regime. This MSE estimator is based on widely used variance estimator for AVG [25], which is accurate to the order of $O(1/b^2)$ [12]. To handle AVG, we just replace estimators in Algorithm 4.

Handling `MIN` and `MAX`. BAS leverages the correlation between embedding similarity and the target attribute. As an example, we estimate `MAX` by $\widehat{\texttt{MAX}} = \max_{t \in D_b \cup S} val(t)$, the maximum observed. Unlike linear aggregates where we minimize global variance, the allocation goal for `MAX` is to maximize the probability that the true maximum is contained within the blocking regime D_b. Utilizing the pilot sample, we model the distribution of values in each stratum using Extreme Value Theory [16] and allocate strata to D_b based on their probability of containing values exceeding the current sample maximum. Since the distribution of the sample maximum is non-normal, bootstrap CIs are invalid. Instead, we construct CIs using the observed maximum as a lower bound and the global maximum of the dataset (ignoring join conditions) as an upper bound.

Handling `MEDIAN`. Estimating the median requires constructing the CDF of the target attribute. BAS estimates the CDF by combining the exact distribution from blocking with the weighted empirical distribution from sampling. Let $F(t)$ be the CDF. We estimate it as:

$$\widehat{F}(t) = \frac{1}{\widehat{N}} \left(\sum_{x \in D_b} \mathbb{I}(x \le t) + \sum_{y \in D_s} \frac{1}{\pi_y} \mathbb{I}(y \le t) \right)$$

where W_y is the sampling probability. The estimated `MEDIAN` is then $\widehat{F}^{-1}(0.5)$. The variance of the median estimator depends on the variance of the CDF estimate at the median value. Therefore, our allocation algorithm focuses on minimizing the variance of the CDF in the strata that overlap with the estimated median range derived from the pilot sample. As the median is a statistical functional that is Hadamard differentiable with respect to the CDF, we apply the same bootstrap-t resampling strategy to derive valid CIs.

Handling `GroupBy`. Group-by queries often suffer from high variance in "tail" groups (e.g., groups with low support) under standard sampling. BAS exploits the semantic clustering of embeddings to target specific groups effectively. We first estimate the aggregate for each group g. Unlike global aggregation where we minimize total variance, our adaptive allocation for group-by queries minimizes the *maximum relative error* across all discovered groups. By using pilot samples to estimate the conditional probability of groups within strata, BAS prioritizes blocking in strata containing high densities of "hard-to-estimate" groups. We use bootstrap-t to provide simultaneous confidence intervals, considering the multiple hypothesis testing in multi-group estimation.

Algorithm Complexity. Consider a chain join of k tables with size $N_1, \ldots, N_k$ and $b \ll \prod_{i=1}^{k} N_i$. BAS has a time complexity of $O\left(\prod_{i=1}^{k} N_i\right)$ and a space complexity of $O\left(\sum_{i=1}^{k-1} N_i N_{n+1} + b \log b\right)$. This complexity is higher than that of uniform sampling, which has both time and space complexities of $O(b)$, because BAS incorporates similarity scores. Additionally, BAS has higher complexity than WWJ, which has time and space complexities $O\left(b \cdot \sum_{i=2}^{k} N_i\right)$, primarily due to the stratification. However, when compared to threshold-based blocking, which shares the same time complexity with BAS and has a space complexity of $O(b)$, BAS only incurs a small overhead to sort the top αb records. Furthermore, BAS is more efficient than ABAE, which involves sorting all similarity scores [36]. Finally, BAS shares the same time complexity with BLAZEIT, which applies control variates over the cross product [33].

Although BAS has a time complexity that increases exponentially with the number of tables, the leading order of the time complexity is attributed to comparing floating-point numbers using CPUs, which is significantly faster than the Oracle. Our latency profiling reveals that BAS's CPU computations take up to 90 seconds on our datasets, costing $0.03 on AWS—making it $10^6\times$ cheaper than running Oracle with GPT-4o mini.

When the table size or the number of tables is so large that the cross product cannot fit into memory, threshold-based blocking in BaS becomes prohibitively expensive. To mitigate this, we apply the nearest neighbor-based blocking [10]. It effectively joins each record of the left table with the top b' records of the right table for each join, where $b' = (\alpha n/N_1)^{1/(k-1)}$. This method has a time complexity of $O\left(b \cdot \sum_{i=2}^{k} N_i\right)$ and a space complexity of $O(b)$, making the complexity of BaS comparable to WWJ and significantly lower than Abae and BlazeIt. We show that BaS with nearest neighbor-based blocking remains more efficient than the baselines (§7.3).

5.4 BaS for Selection Queries

We first review approximate selection queries. Consider a selection query Q that, when executed exactly, outputs a set of records T. If Q is processed approximately, the result is T'. As established in prior work [35], we say that T' achieves the recall target γ and confidence p if $\mathbb{P}[|T \cap T'|/|T| \geq \gamma] \geq p$.

To process such queries, prior work calculates a score for each data and outputs all tuples with scores higher than a threshold τ_s [35]. The score approximately reflects the probability of a tuple satisfying the predicate, while τ_s is estimated using sampling to achieve recall guarantees.

With BaS, we can improve the selection quality by maximize the threshold τ_s. This is achieved by adaptive allocating data to minimize the recall target γ_s for the sampling regime. To formulate the allocation problem, we first translate the overall recall target γ to the recall target of the sampling regime γ_s through the following lemma. We defer the proof to an extended technical report [102].

Lemma 5.1. *With a probability higher than p, we can achieve the overall recall target γ if γ_s satisfies*

$$\gamma_s \geq \gamma - (1-\gamma)\frac{\mathsf{COUNT}_b}{\mathrm{UB}(\mathsf{COUNT}_s, Var[\mathsf{COUNT}_s], b, p)}$$

where

$$\mathrm{UB}(\mu, \sigma^2, b, p) = \mu + \frac{\sigma}{\sqrt{b}}\sqrt{2\log\frac{2}{1-p}}$$

Given a stratification $D_0, \ldots, D_K$ and a budget b_2, we find the optimal allocation β^* to minimize γ_s:

$$\beta^* = \underset{\beta \subset \{1,\ldots,K\}}{\arg\max} \frac{\mathsf{COUNT}_i}{\sum_{i \notin \beta} \mathrm{UB}(\mathsf{COUNT}_i, Var[\mathsf{COUNT}_i], b_i, \frac{p+k-1}{k})}$$

where $\mathsf{COUNT}_i = \sum_{s \in D_i} O(s)$ is the number of matching tuples in stratum i and b_i is the assigned budget for stratum i. Similar to aggregation queries, we approximately solve the allocation problem with estimated COUNT_i and $Var[\mathsf{COUNT}_i]$ using the pilot sampling.

Handling TopK Heavy Hitters Selection. A heavy-hitter TopK query identifies the K entities with the most appearance [15, 91, 96]. This generalizes the selection problem with a dynamic threshold determined by the K-th value. We estimate COUNT for every candidate value using the combined estimator from Equation 1. Let $\widehat{V}_e$ be the estimated value for entity e. We return the set of K entities with the largest $\widehat{V}_e$. To ensure the correctness of the ranking, we minimize the swapping probability between the K-th ranked entity and the $(K+1)$-th ranked entity. We adapt our allocation objective to minimize the estimation variance specifically for candidates whose estimates are close to the threshold τ (the estimated K-th value). We provide guarantees on the recall: with probability p, the returned set contains the true TopK entities. We achieve this by computing simultaneous confidence intervals for the candidate entities using bootstrap-t, ensuring that the lower bound of the K-th item exceeds the upper bound of the $(K+1)$-th item.

5.5 Practical Guidelines

We offer the following deployment guidance. BAS is beneficial when the cost of Oracle significantly outweighs the overhead of embedding similarity calculations. For LLM-based predicates (e.g., GPT-4), Oracle costs are typically orders of magnitude higher than similarity computation, making BAS highly effective. For cold-start scenarios, we recommend allocating 15–30% of the budget to the maximum blocking regime (b_2) to provide a balanced search space for the optimizer without over-constraining the sampling budget. BAS automatically tunes K to ensure each stratum receives > 1000 samples. For small budgets, we enforce a minimum $K = 5$ to ensure sufficient granularity. Finally, BAS is model-agnostic. For semantic joins (e.g., product descriptions), we recommend dense embeddings (e.g., CLIP [76]). For lexical joins, sparse vectors (e.g., TF-IDF) often yield sharper boundaries. To achieve better performance with BAS, we recommend evaluating the false positive and false negative rates of different embeddings on a small data sample.

6 Theoretical Analysis of BAS

In our analysis, we first show MSE of BAS converges to the MSE with optimal allocation asymptotically at the rate $O\left(b_1^{-1/2}\right)$. Furthermore, we compare the MSE of BAS with that of WWJ, showing that BAS outperforms or matches WWJ asymptotically.

6.1 BAS Converges to the Optimal Allocation

BAS applies pilot sampling to approximately solve the minimization problem in Equation 4. Theorem 6.1 shows that the MSE with estimated minimizer $\widehat{\beta}^*$ converges to the MSE with the true minimizer β^* at the rate of $O\left(1/\sqrt{b_1}\right)$.

Theorem 6.1. *The MSE with the estimated minimizer $\hat{\beta}^*$ converges to that with the true minimizer β^* at the rate:*

$$\frac{MSE(D, \widehat{\beta}^*, W, b_2) - MSE(D, \beta^*, W, b_2)}{MSE(D, \beta^*, W, b_2)} = O\left(\frac{1}{\sqrt{b_1}}\right)$$

Intuitively, we observe that statistics for allocation determination, specifically the sample mean and sample variance, converge to their true values at a rate proportional to $1/\sqrt{b_1}$. Theoretically, we demonstrate the MSE of BAS converges to that of the optimal allocation at the same rate. This is because the arithmetic operations involved in calculating the MSE (i.e., summation, multiplication by constants, and division by constants) do not change the convergence rate. We defer a full proof to an extended technical report [102].

6.2 BAS Outperforms or Matches WWJ

Notation. In addition to the notations we introduced in Section 5, We define that $\tilde{D}$ is the set of all positive tuples, the subscript s means the statistic is about the sampling regime of BAS.

We present the comparison of the MSE of BAS and WWJ for the `SUM` aggregate in Theorem 6.2 and defer a full proof to an extended technical report [102]. We can derive the same result for the `COUNT` or `AVG` similarly.

Theorem 6.2. *If there exists an allocation β such that the following two conditions hold*

$$\mathbb{E}_{s\in\tilde{D}_s}\left[\frac{1/|\tilde{D}_s|}{W_s(s)}\right] \le \mathbb{E}_{s\in D}\left[\frac{1/|D|}{W(s)}\right] \tag{6}$$

$$\frac{|\tilde{D}_s|^2}{b_s} \le \frac{|D|^2}{b} \tag{7}$$

BAS outperforms WWJ asymptotically, i.e.,

$$MSE^{(BAS)}_{\mathsf{SUM}} = C \cdot MSE^{(WWJ)}_{\mathsf{SUM}} + O\left(b^{-1}b_1^{-1/2}\right)$$

where W_s is the similarity scores normalized for the sampling regime and C is a coefficient less than 1:

$$C < \frac{|\tilde{D}_s|^2/b_s}{|\tilde{D}|^2/b}\frac{\mathbb{E}_{s\in\tilde{D}_s}\left[\frac{1/|\tilde{D}_s|}{W_s(s)}\right]}{\mathbb{E}_{s\in\tilde{D}}\left[\frac{1/|\tilde{D}|}{W(s)}\right]} \le 1$$

Otherwise, BAS matches WWJ asymptotically, i.e.,

$$MSE^{(BAS)}_{\mathsf{SUM}} \le MSE^{(WWJ)}_{\mathsf{SUM}} + O\left(b^{-1}b_1^{-1/2}\right)$$

Theorem 4 suggests two cases when we compare the MSE of BAS and that of WWJ. We discuss both cases in detail.

Case 1: BAS outperforms WWJ asymptotically. Theorem 6.2 provides sufficient conditions when JOINML achieves asymptotically better MSE than WWJ. We observe that both conditions can match characteristics of the embedding similarity. First, Condition (6) compares the similarity between the proposed sampling distribution W and the ideal distribution (i.e., $1/|\tilde{D}|$) for the matching tuples in the sampling regime or the entire sampling space. When we reduce false positives via blocking, the proposed sampling distribution may become closer to the ideal distribution, satisfying Condition 6. Second, Condition 7 compares the density of matching tuples for tuples in the sampling regime or the entire sampling space. When the similarity scores have a high recall, the blocking regime includes most of the positive tuples, while the positive tuples in the sampling region are sparse, satisfying Condition 7.

Case 2: BAS matches WWJ asymptotically. When sufficient conditions are not fully satisfied, the difference between BAS and WWJ converges to 0 at the rate $O(b^{-1}b_1^{-1/2})$, which is faster than $O(b^{-1})$, the rate at which the MSE converges to 0.

7 Evaluation

In this section, we present evaluation results of our algorithms. We first introduce our experiment setup (§7.1). Then, we demonstrate that BAS achieves statistical guarantees, while BLOCKING fails (§7.2). Moreover, we demonstrate that BAS outperforms baselines in terms of MSE (§7.3). Finally, we perform the ablation study (§7.5) and sensitivity study (§7.6).

7.1 Experiment Settings

Datasets and queries. We curated a comprehensive suite of 16 datasets originating from realistic integration tasks [1, 1, 5, 31, 31, 49, 53, 54, 59, 72, 75, 99], the SemBench benchmark [41], and synthetic stress-tests. To assess robustness across data distributions, our workload spans a wide range of join selectivities (from 5×10^{-10} to 0.5) and scales (up to 1.3 trillion pair cross-products). We summarize the dataset statistics in Figure 7 and defer detailed construction to an extended technical report [102]. For each dataset, we use the embedding model that achieves or matches the

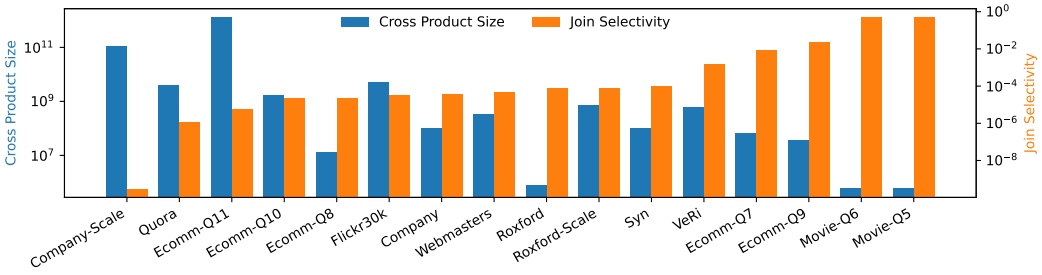


Fig. 7. Cross product size and join selectivity of the 16 evaluated datasets: six real-world datasets, seven SemBench datasets, and three synthetic workloads.

state-of-the-art performance. We evaluated all three aggregates for each dataset and present the results of the most natural aggregate for each dataset. We categorize our queries as follows:

(1) *Real-World Analytical Joins*: Entity resolution, duplicate detection, and video analysis tasks (Company [49], Quora [1], Webmasters [99], Roxford [75], Flickr30K [72], VeRi [53]).
(2) *SemBench*: Queries requiring joint semantic understanding across tables (e.g., matching reviews by sentiment, linking products by images/descriptions). We include all such join queries with a result set of more than 100 records, covering E-commerce and Movie domains.
(3) *Synthetic Stress-Tests*: Company-Scale (6-way join) and Roxford-Scale (10M rows) to test extreme scalability; and $Syn(p_{fp}, p_{fn})$ to systematically vary embedding quality.

Baselines. We compare our algorithms, WWJ and BaS, with the following baselines:

- Uniform: We uniformly sample tuples from the entire cross product of the tables.
- Blocking: A proxy for the state-of-the-art deep EM system, Ditto [49]. It samples tuples above a similarity threshold calibrated on a validation set (10% of positive tuples).
- Abae and BlazeIt: We adapt the state-of-the-art AQP algorithm for aggregation queries with ML predicates by treating the join condition as a selection predicate [33, 36].
- SUPG and LOTUS: We use the state-of-the-art AQP algorithms for selection queries with ML predicates [35, 69] as baselines for the extension of our algorithms for selection queries.

We do not directly compare against full EM pipelines (e.g., Magellan [40]) because they are designed as offline processes that materialize the full result, whereas our algorithms are for an online AQP with a different goal. However, we compare against the core phases of EM (blocking and matching) via our Blocking baseline, which effectively represents the performance of a full EM pipeline on AQP tasks. In addition, we do not consider non-ML text-based EM systems or additional data augmentation used in Ditto [49] because these techniques are specific to the text modality and do not generalize to the multimodal datasets we consider.

7.2 Statistical Guarantees

We analyze whether the baseline method (Blocking) and our proposed method (BaS) can achieve statistical guarantees. Specifically, we focus on whether these methods can provide valid CIs, a standard approach in statistically guaranteed AQP [25, 33, 35, 36, 46].

Metric. We use the ratio between the true error and the CI bounds as our evaluation metric. Given a query with a groundtruth result μ, an estimated result $\hat{\mu}$, and a confidence interval $[l, u]$, the error

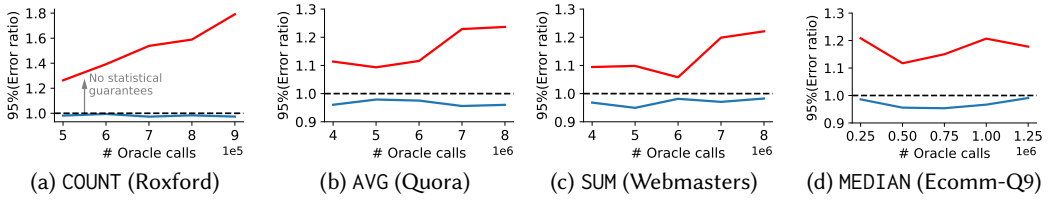


Fig. 8. BAS achieves valid statistical guarantees (Error Ratio ≤ 1) across all evaluated aggregators, whereas standard BLOCKING consistently produces invalid CI.

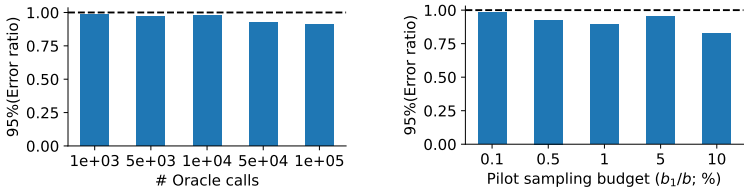


Fig. 9. BAS maintains valid CIs even under extreme conditions, including low Oracle budgets (e.g., 1,000) and small pilot sample sizes (e.g., 0.1%).

ratio is calculated as:

$$\text{Error ratio} := \frac{|\hat{\mu} - \mu|}{|u - l|/2}$$

According to the definition of a CI, a valid 95% CI must include the true value with a probability of 95%. Consequently, the probability that the true error falls within the CI bounds should be at least 95%. Thus, for a method to provide valid CIs, the 95th percentile of the ratio between the true error and the CI bounds must not exceed 1.

BAS Achieves Statistical Guarantees. We present the results for BLOCKING and BAS across various Oracle budgets in Figure 8. For each Oracle budget, we repeated the experiments 500 times. As shown, BAS consistently achieves 95th percentile error ratios of less than 1, indicating the validity of provided CIs. In contrast, BLOCKING fails to achieve valid CIs for every query, leading to true errors that are up to 79% higher than the CI errors.

We observe that the error ratio of BLOCKING increases as we increase the Oracle budget. This trend is expected because of the inherent bias of BLOCKING: it determines the data to execute Oracle through a similarity score threshold. Even when the similarity threshold is calibrated on a validation dataset, BLOCKING can still ignore positive pairs in the evaluation dataset, resulting in inherently biased estimation results. As the Oracle budget increases, the true error of BLOCKING decreases and converges to an unknown inherent bias, while the width of the CI approaches to zero. Consequently, when the Oracle budget is sufficiently large, true errors can be smaller than CI errors, causing invalid CIs. BAS resolves this problem by incorporating sampling algorithms to provide unbiased or asymptotically unbiased estimations, leading to valid CIs.

BAS Achieves Statistical Guarantees Under Limited Budgets. We show that our asymptotic statistical guarantees hold under limited budgets. In the left part of Figure 9, we report the maximum error ratio across all datasets. We show that BAS maintains valid CIs even when the oracle budget is as low as 1,000. These results empirically validate the robustness of the bootstrap-t method used in BAS, ensuring reliability even in small-budget scenarios. In right part of Figure 9, we show

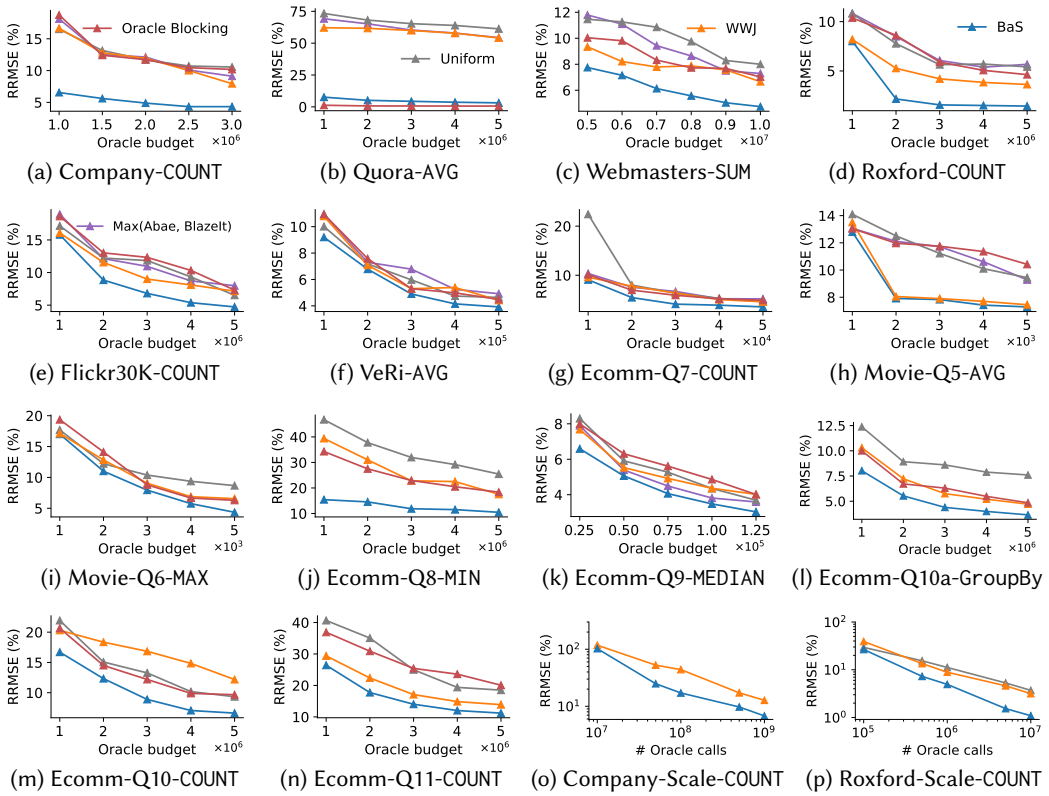


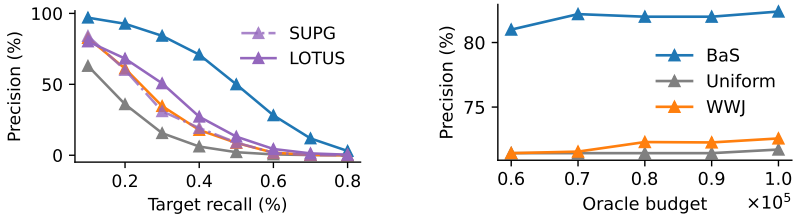
Fig. 10. BaS consistently reduces RMSE across a diverse workload. It outperforms baselines in linear aggregators (10a–10h), extreme aggregators (10i and 10j), MEDIAN (10k), GroupBy (10l), multi-table join (10m–10o), and large-scale join queries (10o and 10p). BaS remains robust in both rare-event (Low Selectivity) and dense (High Selectivity) scenarios.

this validity holds regardless of the pilot sample size (varying from 0.1% to 10% of the budget). Intuitively, this is because the size of the pilot sample does not affect the derivation of confidence intervals, thus not affecting our statistical guarantees.

7.3 End-to-end Performance

We analyze end-to-end performance on a diverse workload including real-world analytical joins, SemBench semantic queries, and multi-way joins. We report the average relative RMSE (RRMSE) over 100 repetitions with a maximum blocking ratio $\alpha = 20\%$.

BaS Improves Overall Estimation Errors. As shown in Figure 10, BaS reduces estimation errors by 1.04–19.5× compared to the best performing baseline among Uniform, Abae, and BlazeIt. Compared to WWJ, BaS achieves similar or lower estimation errors by up to 17.2×. Except for Quora, BaS reduces errors by up to 2.2× compared to Oracle Blocking. This improvement is due to the relatively low Oracle threshold to achieve valid confidence intervals. While Oracle Blocking can be more efficient if positive and negative tuples are well distinguished by similarity scores (as in the case of Quora), determining the Oracle threshold requires exhaustive execution of the Oracle, making it prohibitively expensive.



(a) BaS achieves up to 43% higher precision than LOTUS while maintaining the same recalls guarantees.

(b) BaS achieves up to 10% higher precision than baselines, while LOTUS does not support TopK with guarantees.

Fig. 11. BaS extends to improve selection queries.

BaS Reduces Estimation Errors Across Selectivities. We categorize results by join selectivity to demonstrate BaS's adaptability:

(1) *Low-Selectivity* (< 0.001): On datasets with rare matches (Fig. 10a–10e, 10j, and 10l–10p), BaS significantly outperforms all baselines by up to 19.5×. For the 6-way Company-Scale join (selectivity 5×10^{-10}), Uniform sampling failed to retrieve sufficient positive samples to form an estimate, while BaS maintained low error.
(2) *High-Selectivity* ($0.001 - 0.5$): In high-density scenarios (Fig. 10f–10i. and 10k) where blocking offers little theoretical advantage, BaS demonstrates robustness. It adaptively allocates budget towards sampling, preventing the performance regression observed in Blocking. BaS achieves errors 1.05–2.5× lower than baselines and matches WWJ.

BaS Improves Real-World Multi-Way Joins. We evaluated scalability using real-world 3-way (Ecomm-Q10), 4-way (Ecomm-Q11), and synthetic 6-way (Company-Scale) joins. As shown in Figure 10m–10o, BaS reduces error by 1.4–2.1× compared to Uniform and WWJ, while Abae and BlazeIt do not natively support multi-way joins. This performance gap highlights the critical role of blocking in large search spaces. As the number of joined tables increases, the density of valid join paths decreases, causing inefficient sampling.

BaS Improves `MAX`, `MIN`, `MEDIAN`, and `GroupBy` Queries. To demonstrate general applicability of BaS, we evaluated non-linear aggregators and `GroupBy` queries. We show that BaS reduces estimation error by up to 1.4× for `MEDIAN` (Fig. 10k), 2.0× for `MAX` (Fig. 10i), and 3.0× for `MIN` (Fig. 10j). For `GroupBy` queries (Fig. 10l), BaS reduces the mean estimation error across groups by up to 2.1× compared to baselines, demonstrating its utility for OLAP-style analytics.

BaS Improves Selection Queries. We compared BaS against LOTUS [69] on selection queries (Fig. 11a). On a selection query based on the Company dataset, while both methods satisfy the recall target with 95% confidence, BaS achieves significantly higher precision, improving by 2.6–43.8% over LOTUS and 3.0–68.6% over Uniform. In addition, we show that WWJ achieves performance similar to SUPG (dotted purple line), empirically validating that WWJ shares the same statistical characteristics as SUPG for selection queries. Furthermore, for a `TopK` heavy-hitter query based on Ecomm-Q11 with recall guarantees (95% confidence), where LOTUS lacks native support, BaS improves precision by 9.6–10.7% over baselines. This indicates that BaS's variance-optimal allocation is effective not only for aggregation, but for high-precision retrieval.

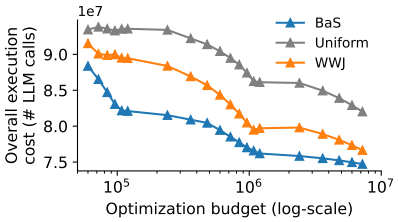


Fig. 12. By providing accurate cardinality estimates, BaS enables the query optimizer to select better join orders, reducing the execution cost of Ecomm-Q11 by 12.7%.

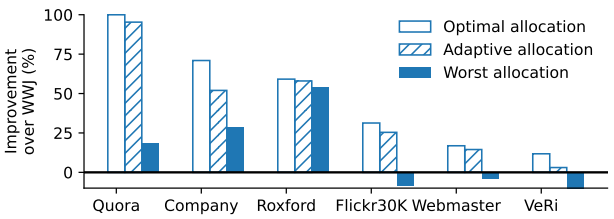


Fig. 13. BaS determines near-optimal allocations: BaS with adaptive allocations achieves improvements that are 1.61–18.9% lower compared to the optimal allocation and 18.6–606.4% higher compared to the worst allocation.

7.4 Application of Approximate `COUNT`: Join Order Optimization

We applied BaS to the problem of join order optimization for exactly executing a multi-way semantic join query (Ecomm-Q11 of SemBench), where cardinality estimation errors can lead to execution costs 80× higher than optimal. By integrating BaS with the DPccp enumeration algorithm [58], we generate more accurate cardinality estimates for sub-joins. As shown in Figure 12, the improved `COUNT` estimation from BaS allows the optimizer to identify efficient execution plans, reducing total execution cost by up to 12.7% compared to plans generated using Uniform or WWJ estimates.

7.5 Ablation Study

We empirically validate the theoretical result in Section 6 that BaS converges to the optimal allocation. To achieve this, we compare the adaptive allocation with the optimal allocation and the worst allocation. To approximate the optimal and worst allocations, we executed BaS with fixed blocking ratios ranging from 10% to 50%. For example, a fixed blocking ratio of 10% means we directly execute Oracle on pairs with top 10% similarity scores while applying WWJ on the rest. For each ratio, we repeated experiments 100 times.

In Figure 13, we show the improvements of BaS using optimal, adaptive, and worst allocations over WWJ. We calculate the improvement as the relative reduction of RMSE. As shown, BaS with adaptive allocations achieves improvements that are only 1.61-18.9% (7.04% on average) less compared to BaS with optimal allocations. In contrast, BaS with the worst allocations underperforms (i.e., negative improvements) by up to 98.4% compared to WWJ and up to 99.9% compared to BaS with optimal allocations.[4]

7.6 Sensitivity Analysis

We analyze the sensitivity of BaS in terms of the quality of embedding models, strategies to use the similarity scores as sampling weights, and the maximum blocking ratio α.

Impact of False Negatives and False Positives. We examined the impact of embedding quality using synthetic datasets with controlled False Negative Rates (FNR) and False Positive Rates (FPR). As shown in Figure 15, when FNR and FPR reach 50%, BaS outperforms baselines by more than 20%. Specifically, BaS dominates Blocking when FNR is high (by correcting for missed matches via sampling) and dominates WWJ when FPR is high (by avoiding the overweighting of false positives).

Impact of the Embedding Quality. We executed algorithms on the Company dataset with 9 more embeddings, including traditional textual similarity [14, 77], encoder-only embedding models [78],

[4]We remove the majority of the negative improvements of the worst allocations in the figure for simplicity.

Fig. 14. BaS consistently outperforms Uniform and WWJ across various types of embeddings.

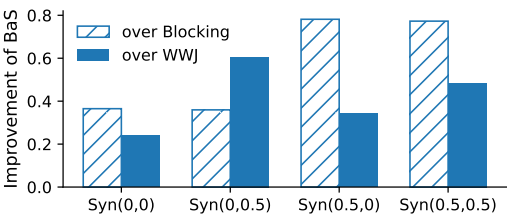
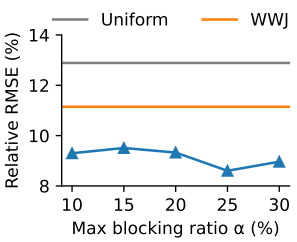
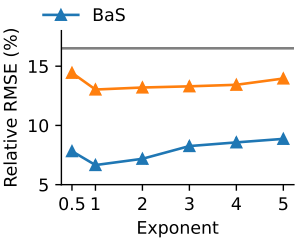


Fig. 15. BaS consistently outperforms baselines by more than 20% as embedding quality degrades.



(a) α vs relative RMSE for the Webmasters dataset.



(b) Weight exponent vs relative RMSE for the Company dataset.

Fig. 16. Sensitivity of BaS in terms of α and the weight exponents. BaS consistently outperforms baselines.

and large-scale embedding models [20, 42, 51, 64, 79, 94]. For each embedding method, we executed WWJ and BaS 100 times with an Oracle budget of 1,000,000. In Figure 14, we demonstrate the performance of BaS, WWJ, and Uniform with various embeddings on the Company dataset with an Oracle budget of 1,000,000 and a maximum blocking ratio of 20%. As shown, BaS consistently outperforms Uniform and WWJ across all embeddings. We find that BaS benefits from improved performance with better embeddings.

Furthermore, we find that traditional TF-IDF sometimes outperforms modern dense embeddings (e.g., OpenAI). This counter-intuitive result is caused by the specific join semantics of the Company dataset, which requires high *lexical precision* (e.g., distinguishing a radio company from a television company), a task where traditional methods excel by strictly penalizing token mismatches. In contrast, dense embeddings prioritize *semantic relatedness*, often clustering non-matching companies from the same industry, creating extra false positives (e.g., 99% FPR for OpenAI vs. 27% for TF-IDF).[1] Therefore, traditional methods or embedding models with instructional prompts (e.g., `gte-Qwen2` [51] and `stella_en` [20]) work better on this dataset. Nevertheless, BaS provides statistical guarantees regardless of the embedding's quality.

Impact of the Maximum Blocking Ratio (α). We executed BaS with maximum blocking ratios (α) ranging from 10% to 30% on the Flickr30K datasets. For each value of α, we repeated experiments 100 times and measured the relative RMSE. As shown in Figure 16a, the RMSE of BaS fluctuates by 9.5-14.2% compared with the optimal one. Despite the fluctuation, BaS consistently outperforms baselines, demonstrating that BaS is insensitive to α.

[1]We measured false positive rates at a fixed recall of 50%.

Impact of the Weight Exponents. We executed WWJ and BaS on the Company dataset with different weight exponents. For example, the exponent=0.5 means that the sampling weight and budget allocation is proportional to the square root of the similarity score. For each exponent, we repeated experiments 100 times and measure the relative RMSE. As shown in Figure 16b, BaS achieves relatively better performance with exponent equal to 1 and consistently outperforms baselines by 36.5–59.7%.

8 Related Work

We draw inspiration from classic join optimization algorithms and leverages recent advancements in EM and AQP. We summarize the relevant literature on these areas.

Optimization and acceleration of approximate join queries. Joins are fundamental but notoriously expensive operations in relational database management systems [2, 56]. To optimize and accelerate join queries with structured data, various algorithms and systems are developed in both offline and online scenarios.

Prior work developed offline sampling algorithms to address the challenge that joining uniform samples does not constitute an independent sample of the join [9, 29, 100]. Specifically, researchers proposed the Stratified-Universe-Bernoulli sampling that achieves the theoretical optimal sampling performance within a constant factor [29]. For joins with specific types, prior work proposed precomputed synopses for foreign key joins [3] and the histogram method for temporal joins [82]. To accelerate join queries in the online scenario, prior work proposed ripple join [25]. Wander Join improved the sampling efficiency using indices and random walks [46].

Recently, for unstructured data, researchers have focused on efficient implementations of the semantic join operator [69, 86]. Notably, Trummer [86] proposes a batch-oriented block nested loop algorithm that optimizes prompt construction to evaluate multiple row pairs in a single LLM invocation. This line of work addresses the *physical execution efficiency* of the join operator. In contrast, BaS addresses the logical approximate execution of such queries via reducing variance. These approaches are highly complementary. BaS optimizes which pairs to check, while [86] reduces the cost of checking those pairs, particularly in the dense blocking regime.

Entity Matching. EM seeks to identify common objects across datasets [22, 68]. Earlier approaches have modeled EM problems as similarity joins [8, 90]. However, EM is prohibitively expensive due to pair-wise comparisons [80]. In response, the blocking method is developed to reduce the number of pair-wise comparisons. Traditional blocking methods relied on heuristics and expert knowledge to develop rule-based block constructions, such as suffix arrays [17, 66], sorted neighborhood [74, 95], and sorted block [19].

Recent advances in ML models, especially LLMs, have facilitated learning-based methods in EM systems [49, 50, 59, 85]. Existing work proposed to use language models for textual data embedding, classification [49, 50, 59], and blocking [85]. Additionally, recent work improved the robustness of learning-based methods using data augmentation [4].

AQP for unstructured data. Recent work addressed AQP for unstructured data using machine learning models. To reduce the cost of executing expensive machine learning models, prior work leveraged approximate indexes [28], binary classifiers [55], and specialized models [34] for AQP without statistical guarantees. Furthermore, advancements including query-driven object tracking algorithms and segmentation-based tracking queries, further reduced the query cost. To provide guarantees on accuracy, existing work used proxy models with control variate to process aggregation and limit queries on video frames [33], importance sampling to process selection queries [35], and stratified sampling to process aggregation queries [36].

Join operations have been investigated for unstructured data in prior AQP systems [32, 52, 69]. However, prior work either focused on join with structured join keys [32, 52], which is orthogonal to our work, or applies calibration techniques without statistical guarantees [69].

9 Conclusion

In this work, we propose Weighted Wander Join and Blocking-augmented Sampling to approximately process analytical join queries over unstructured data with statistical guarantees. We address the inaccuracy of embedding models by integrating sampling and blocking, and propose an adaptive allocation algorithm to minimize the errors. Theoretically, we prove the optimality and superiority of BAS. Our evaluation results demonstrate that BAS outperforms baselines by up to 21×, saving costs by up to 188× on real-world datasets. These results highlight the potential of embeddings and cost-effective algorithms in analyzing unstructured data with joins.

10 Acknowledgements

We are grateful to the CloudLab for providing computing resources for experiments [21]. We thank Kaimeng Zhu and Siheng Pan for their help during the initial implementation and experimentation of this work. This research was supported in part by Google.

References

[1] [n. d.]. First Quora Dataset Release: Question Pairs. https://quoradata.quora.com/First-Quora-Dataset-Release-Question-Pairs. Accessed: 2024-01-08.
[2] Serge Abiteboul, Richard Hull, and Victor Vianu. 1995. *Foundations of databases*. Addison-Wesley Reading.
[3] Swarup Acharya, Phillip B Gibbons, Viswanath Poosala, and Sridhar Ramaswamy. 1999. Join synopses for approximate query answering. In *SIGMOD*.
[4] Mehdi Akbarian Rastaghi, Ehsan Kamalloo, and Davood Rafiei. 2022. Probing the Robustness of Pre-trained Language Models for Entity Matching. In *CIKM*.
[5] Alberto Bacchelli. 2013. Mining Challenge 2013: Stack Overflow. In *MSR*.
[6] Alberto Barrón-Cedeño, Marta Vila, M Antònia Martí, and Paolo Rosso. 2013. Plagiarism meets paraphrasing: Insights for the next generation in automatic plagiarism detection. *Computational Linguistics* (2013).
[7] Mikhail Bilenko, Beena Kamath, and Raymond J Mooney. 2006. Adaptive blocking: Learning to scale up record linkage. In *Sixth International Conference on Data Mining (ICDM'06)*. IEEE, 87–96.
[8] Surajit Chaudhuri, Venkatesh Ganti, and Raghav Kaushik. 2006. A primitive operator for similarity joins in data cleaning. In *ICDE*.
[9] Surajit Chaudhuri, Rajeev Motwani, and Vivek Narasayya. 1999. On random sampling over joins. *SIGMOD* (1999).
[10] Peter Christen. 2011. A survey of indexing techniques for scalable record linkage and deduplication. *IEEE transactions on knowledge and data engineering* 24, 9 (2011), 1537–1555.
[11] Vassilis Christophides, Vasilis Efthymiou, and Kostas Stefanidis. 2015. *Entity resolution in the web of data*. Vol. 5. Springer.
[12] William Gemmell Cochran. 1977. *Sampling techniques*. john wiley & sons.
[13] Benjamin Coifman. 1998. Vehicle re-identification and travel time measurement in real-time on freeways using existing loop detector infrastructure. *Transportation Research Record* (1998).
[14] Courtney D Corley and Rada Mihalcea. 2005. Measuring the semantic similarity of texts. In *Proceedings of the ACL workshop on empirical modeling of semantic equivalence and entailment*. 13–18.
[15] Graham Cormode, Flip Korn, Shanmugavelayutham Muthukrishnan, and Divesh Srivastava. 2003. Finding hierarchical heavy hitters in data streams. In *Proceedings 2003 VLDB Conference*. Elsevier, 464–475.
[16] Laurens De Haan and Ana Ferreira. 2006. *Extreme value theory: an introduction*. Springer.
[17] Timothy De Vries, Hui Ke, Sanjay Chawla, and Peter Christen. 2009. Robust record linkage blocking using suffix arrays. In *CIKM*.
[18] Dorottya Demszky, Devyani Sharma, Jonathan H Clark, Vinodkumar Prabhakaran, and Jacob Eisenstein. 2020. Learning to recognize dialect features. *arXiv preprint arXiv:2010.12707* (2020).
[19] Uwe Draisbach and Felix Naumann. 2011. A generalization of blocking and windowing algorithms for duplicate detection. In *International Conference on Data and Knowledge Engineering*.

[20] DunZhang. 2024. *dunzhang/stella_en_1.5B_v5*. https://huggingface.co/dunzhang/stella_en_1.5B_v5 Accessed: 2024-11-20.
[21] Dmitry Duplyakin, Robert Ricci, Aleksander Maricq, Gary Wong, Jonathon Duerig, Eric Eide, Leigh Stoller, Mike Hibler, David Johnson, Kirk Webb, Aditya Akella, Kuangching Wang, Glenn Ricart, Larry Landweber, Chip Elliott, Michael Zink, Emmanuel Cecchet, Snigdhaswin Kar, and Prabodh Mishra. 2019. The Design and Operation of CloudLab. In *Proceedings of the USENIX Annual Technical Conference (ATC)*. 1–14. https://www.flux.utah.edu/paper/duplyakin-atc19
[22] Ahmed K Elmagarmid, Panagiotis G Ipeirotis, and Vassilios S Verykios. 2006. Duplicate record detection: A survey. *IEEE TKDE* (2006).
[23] Ivan P Fellegi and Alan B Sunter. 1969. A theory for record linkage. *J. Amer. Statist. Assoc.* 64, 328 (1969), 1183–1210.
[24] Tomáš Foltýnek, Norman Meuschke, and Bela Gipp. 2019. Academic plagiarism detection: a systematic literature review. *Comput. Surveys* (2019).
[25] Peter J Haas and Joseph M Hellerstein. 1999. Ripple joins for online aggregation. *ACM SIGMOD Record* (1999).
[26] Peter Hall. 1988. Theoretical comparison of bootstrap confidence intervals. *The Annals of Statistics* (1988).
[27] Shuting He, Hao Luo, Pichao Wang, Fan Wang, Hao Li, and Wei Jiang. 2021. Transreid: Transformer-based object re-identification. In *CVPR*.
[28] Kevin Hsieh, Ganesh Ananthanarayanan, Peter Bodik, Shivaram Venkataraman, Paramvir Bahl, Matthai Philipose, Phillip B Gibbons, and Onur Mutlu. 2018. Focus: Querying large video datasets with low latency and low cost. In *OSDI*.
[29] Dawei Huang, Dong Young Yoon, Seth Pettie, and Barzan Mozafari. 2019. Joins on samples: a theoretical guide for practitioners. *PVLDB* (2019).
[30] Lei Huang, Weijiang Yu, Weitao Ma, Weihong Zhong, Zhangyin Feng, Haotian Wang, Qianglong Chen, Weihua Peng, Xiaocheng Feng, Bing Qin, et al. 2023. A survey on hallucination in large language models: Principles, taxonomy, challenges, and open questions. *ACM Transactions on Information Systems* (2023).
[31] Haoming Jiang, Pengcheng He, Weizhu Chen, Xiaodong Liu, Jianfeng Gao, and Tuo Zhao. 2020. Smart: Robust and efficient fine-tuning for pre-trained natural language models through principled regularized optimization. In *ACL*.
[32] Saehan Jo and Immanuel Trummer. 2024. ThalamusDB: Approximate Query Processing on Multi-Modal Data. *Proceedings of the ACM on Management of Data* 2, 3 (2024), 1–26.
[33] Daniel Kang, Peter Bailis, and Matei Zaharia. 2019. BlazeIt: optimizing declarative aggregation and limit queries for neural network-based video analytics. *PVLDB* (2019).
[34] Daniel Kang, John Emmons, Firas Abuzaid, Peter Bailis, and Matei Zaharia. 2017. NoScope: Optimizing Neural Network Queries over Video at Scale. *PVLDB* (2017).
[35] Daniel Kang, Edward Gan, Peter Bailis, Tatsunori Hashimoto, and Matei Zaharia. 2020. Approximate selection with guarantees using proxies. *PVLDB* (2020).
[36] Daniel Kang, John Guibas, Peter Bailis, Tatsunori Hashimoto, Yi Sun, and Matei Zaharia. 2021. Accelerating approximate aggregation queries with expensive predicates. *PVLDB* (2021).
[37] Omar Khattab and Matei Zaharia. 2020. Colbert: Efficient and effective passage search via contextualized late interaction over bert. In *SIGIR*.
[38] Leslie Kish. 2011. Survey sampling.
[39] Teun Kloek and Herman K Van Dijk. 1978. Bayesian estimates of equation system parameters: an application of integration by Monte Carlo. *Econometrica: Journal of the Econometric Society* (1978), 1–19.
[40] Pradap Konda, Sanjib Das, AnHai Doan, Adel Ardalan, Jeffrey R Ballard, Han Li, Fatemah Panahi, Haojun Zhang, Jeff Naughton, Shishir Prasad, et al. 2016. Magellan: toward building entity matching management systems over data science stacks. *PVLDB* (2016).
[41] Jiale Lao, Andreas Zimmerer, Olga Ovcharenko, Tianji Cong, Matthew Russo, Gerardo Vitagliano, Michael Cochez, Fatma Özcan, Gautam Gupta, Thibaud Hottelier, et al. 2025. SemBench: A Benchmark for Semantic Query Processing Engines. *arXiv preprint arXiv:2511.01716* (2025).
[42] Chankyu Lee, Rajarshi Roy, Mengyao Xu, Jonathan Raiman, Mohammad Shoeybi, Bryan Catanzaro, and Wei Ping. 2024. NV-Embed: Improved Techniques for Training LLMs as Generalist Embedding Models. *arXiv preprint arXiv:2405.17428* (2024).
[43] Zengyu Lei, Caiming Zhang, Yunyang Xu, and Xuemei Li. 2024. DR-GAT: Dynamic routing graph attention network for stock recommendation. *Information Sciences* (2024).
[44] Bo-Han Li, Yi Liu, An-Man Zhang, Wen-Huan Wang, and Shuo Wan. 2020. A survey on blocking technology of entity resolution. *Journal of Computer Science and Technology* 35 (2020), 769–793.
[45] Chenglin Li, Yuanzhen Xie, Chenyun Yu, Bo Hu, Zang Li, Guoqiang Shu, Xiaohu Qie, and Di Niu. 2023. One for all, all for one: Learning and transferring user embeddings for cross-domain recommendation. In *Proceedings of the Sixteenth ACM International Conference on Web Search and Data Mining*. 366–374.
[46] Feifei Li, Bin Wu, Ke Yi, and Zhuoyue Zhao. 2016. Wander join: Online aggregation for joins. In *SIGMOD*.

[47] Junnan Li, Dongxu Li, Caiming Xiong, and Steven Hoi. 2022. Blip: Bootstrapping language-image pre-training for unified vision-language understanding and generation. In *ICML*.

[48] Junnan Li, Ramprasaath Selvaraju, Akhilesh Gotmare, Shafiq Joty, Caiming Xiong, and Steven Chu Hong Hoi. 2021. Align before fuse: Vision and language representation learning with momentum distillation. *Advances in neural information processing systems* 34 (2021), 9694–9705.

[49] Yuliang Li, Jinfeng Li, Yoshihiko Suhara, AnHai Doan, and Wang-Chiew Tan. 2020. Deep entity matching with pre-trained language models. *PVLDB* (2020).

[50] Yuliang Li, Jinfeng Li, Yoshihiko Suhara, Jin Wang, Wataru Hirota, and Wang-Chiew Tan. 2021. Deep entity matching: Challenges and opportunities. *Journal of Data and Information Quality* (2021).

[51] Zehan Li, Xin Zhang, Yanzhao Zhang, Dingkun Long, Pengjun Xie, and Meishan Zhang. 2023. Towards general text embeddings with multi-stage contrastive learning. *arXiv preprint arXiv:2308.03281* (2023).

[52] Shu Liu, Asim Biswal, Audrey Cheng, Xiangxi Mo, Shiyi Cao, Joseph E Gonzalez, Ion Stoica, and Matei Zaharia. 2024. Optimizing llm queries in relational workloads. *arXiv preprint arXiv:2403.05821* (2024).

[53] Xinchen Liu, Wu Liu, Tao Mei, and Huadong Ma. 2016. A deep learning-based approach to progressive vehicle re-identification for urban surveillance. In *ECCV*.

[54] Xinchen Liu, Wu Liu, Tao Mei, and Huadong Ma. 2017. Provid: Progressive and multimodal vehicle reidentification for large-scale urban surveillance. *IEEE Transactions on Multimedia* (2017).

[55] Yao Lu, Aakanksha Chowdhery, Srikanth Kandula, and Surajit Chaudhuri. 2018. Accelerating machine learning inference with probabilistic predicates. In *SIGMOD*.

[56] David Maier. 1983. *The theory of relational databases*. Computer science press Rockville.

[57] Matthew Michelson and Craig A Knoblock. 2006. Learning blocking schemes for record linkage. In *AAAI*, Vol. 6. 440–445.

[58] Guido Moerkotte and Thomas Neumann. 2006. Analysis of two existing and one new dynamic programming algorithm for the generation of optimal bushy join trees without cross products. In *Proceedings of the 32nd international conference on Very large data bases*. 930–941.

[59] Sidharth Mudgal, Han Li, Theodoros Rekatsinas, AnHai Doan, Youngchoon Park, Ganesh Krishnan, Rohit Deep, Esteban Arcaute, and Vijay Raghavendra. 2018. Deep learning for entity matching: A design space exploration. In *SIGMOD*.

[60] Niklas Muennighoff, Nouamane Tazi, Loic Magne, and Nils Reimers. 2023. MTEB: Massive Text Embedding Benchmark. In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*. 2014–2037.

[61] Avanika Narayan, Ines Chami, Laurel Orr, and Christopher Ré. 2022. Can Foundation Models Wrangle Your Data? *PVLDB* (2022).

[62] Rodrigo Nogueira and Kyunghyun Cho. 2019. Passage Re-ranking with BERT. *arXiv preprint arXiv:1901.04085* (2019).

[63] OpenAI. [n. d.]. *Pricing*. https://openai.com/api/pricing/ Accessed: 2024-10-22.

[64] OpenAI. 2024. New embedding models and API updates. https://openai.com/index/new-embedding-models-and-api-updates/

[65] Matteo Paganelli, Francesco Del Buono, Andrea Baraldi, Francesco Guerra, et al. 2022. Analyzing how BERT performs entity matching. *Proceedings of the VLDB Endowment* 15, 8 (2022), 1726–1738.

[66] George Papadakis, George Alexiou, George Papastefanatos, and Georgia Koutrika. 2015. Schema-agnostic vs schema-based configurations for blocking methods on homogeneous data. *PVLDB* (2015).

[67] George Papadakis, Ekaterini Ioannou, Themis Palpanas, Claudia Niederée, and Wolfgang Nejdl. 2012. A blocking framework for entity resolution in highly heterogeneous information spaces. *IEEE Transactions on Knowledge and Data Engineering* 25, 12 (2012), 2665–2682.

[68] George Papadakis, Dimitrios Skoutas, Emmanouil Thanos, and Themis Palpanas. 2020. Blocking and filtering techniques for entity resolution: A survey. *Comput. Surveys* (2020).

[69] Liana Patel, Siddharth Jha, Carlos Guestrin, and Matei Zaharia. 2025. Semantic Operators: A Declarative Model for Rich, AI-based Data Processing. *arXiv preprint arXiv:2407.11418* (2025).

[70] Ralph Peeters, Aaron Steiner, and Christian Bizer. 2025. Entity matching using large language models. In *28th International Conference on Extending Database Technology*.

[71] Mohammad Taher Pilehvar and Jose Camacho-Collados. 2018. WiC: the word-in-context dataset for evaluating context-sensitive meaning representations. *arXiv preprint arXiv:1808.09121* (2018).

[72] Bryan A. Plummer, Liwei Wang, Christopher M. Cervantes, Juan C. Caicedo, Julia Hockenmaier, and Svetlana Lazebnik. 2017. Flickr30K Entities: Collecting Region-to-Phrase Correspondences for Richer Image-to-Sentence Models. *ICCV* (2017).

[73] Abhijit Pol and Christopher Jermaine. 2005. Relational confidence bounds are easy with the bootstrap. In *Proceedings of the 2005 ACM SIGMOD international conference on Management of data*. 587–598.

[74] Sven Puhlmann, Melanie Weis, and Felix Naumann. 2006. XML duplicate detection using sorted neighborhoods. In *International Conference on Extending Database Technology*.
[75] Filip Radenović, Ahmet Iscen, Giorgos Tolias, Yannis Avrithis, and Ondřej Chum. 2018. Revisiting oxford and paris: Large-scale image retrieval benchmarking. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 5706–5715.
[76] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. 2021. Learning transferable visual models from natural language supervision. In *ICML*.
[77] Juan Ramos et al. 2003. Using tf-idf to determine word relevance in document queries. In *ICML*.
[78] Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. In *EMNLP*.
[79] Meng Rui*, Ye* Liu, Shafiq Rayhan Joty, Caiming Xiong, Yingbo Zhou, and Semih Yavuz. 2024. SFR-Embedding-2: Advanced Text Embedding with Multi-stage Training. https://huggingface.co/Salesforce/SFR-Embedding-2_R
[80] Wei Shen, Jianyong Wang, and Jiawei Han. 2014. Entity linking with a knowledge base: Issues, techniques, and solutions. *IEEE TKDE* (2014).
[81] Moohebat Shoyukhi, Paul Hubert Vossen, Abdol Hossein Ahmadi, Reza Kafipour, and Kyle Albert Beattie. 2023. Developing a comprehensive plagiarism assessment rubric. *Education and Information Technologies* (2023).
[82] Inga Sitzmann and Peter J Stuckey. 2000. Improving Temporal Joins Using Histograms. In *International Conference on Database and Expert Systems Applications*.
[83] Kai-Sheng Teong, Lay-Ki Soon, and Tin Tin Su. 2020. Schema-agnostic entity matching using pre-trained language models. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*. 2241–2244.
[84] Nandan Thakur, Nils Reimers, Andreas Rücklé, Abhishek Srivastava, and Iryna Gurevych. [n. d.]. BEIR: A Heterogeneous Benchmark for Zero-shot Evaluation of Information Retrieval Models. ([n. d.]).
[85] Saravanan Thirumuruganathan, Han Li, Nan Tang, Mourad Ouzzani, Yash Govind, Derek Paulsen, Glenn Fung, and AnHai Doan. 2021. Deep learning for blocking in entity matching: a design space exploration. *PVLDB* (2021).
[86] Immanuel Trummer. 2025. Implementing Semantic Join Operators Efficiently. *arXiv preprint arXiv:2510.08489* (2025).
[87] Aad W Van der Vaart. 2000. *Asymptotic statistics*. Vol. 3. Cambridge university press.
[88] Edward Vendrow, Omiros Pantazis, Alexander Shepard, Gabriel Brostow, Kate Jones, Oisin Mac Aodha, Sara Beery, and Grant Van Horn. 2025. INQUIRE: A Natural World Text-to-Image Retrieval Benchmark. *Advances in Neural Information Processing Systems* 37 (2025), 126500–126514.
[89] David Vengerov, Andre Cavalheiro Menck, Mohamed Zait, and Sunil P Chakkappen. 2015. Join size estimation subject to filter conditions. *PVLDB* (2015).
[90] Jiannan Wang, Guoliang Li, Jeffrey Xu Yu, and Jianhua Feng. 2011. Entity matching: How similar is similar. *PVLDB* (2011).
[91] Sibo Wang and Yufei Tao. 2018. Efficient algorithms for finding approximate heavy hitters in personalized pageranks. In *Proceedings of the 2018 International Conference on Management of Data*. 1113–1127.
[92] Tianshu Wang, Xiaoyang Chen, Hongyu Lin, Xuanang Chen, Xianpei Han, Hao Wang, Zhenyu Zeng, and Le Sun. 2025. Match, Compare, or Select? An Investigation of Large Language Models for Entity Matching. In *International Conference on Computational Linguistics*.
[93] Daniel R White and Mike S Joy. 2004. Sentence-based natural language plagiarism detection. *Journal on Educational Resources in Computing (JERIC)* 4, 4 (2004), 2–es.
[94] Shitao Xiao, Zheng Liu, Peitian Zhang, and Niklas Muennighoff. 2023. C-Pack: Packaged Resources To Advance General Chinese Embedding. arXiv:2309.07597 [cs.CL]
[95] Su Yan, Dongwon Lee, Min-Yen Kan, and Lee C Giles. 2007. Adaptive sorted neighborhood methods for efficient record linkage. In *ACM/IEEE-CS joint conference on Digital libraries*.
[96] Ke Yi and Qin Zhang. 2009. Optimal tracking of distributed heavy hitters and quantiles. In *Proceedings of the twenty-eighth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*. 167–174.
[97] Dominik Zapletal and Adam Herout. 2016. Vehicle re-identification for automatic video traffic surveillance. In *CVPR workshops*.
[98] Boyao Zhang, Zhongrui Li, Chao Yang, Zongguo Wang, Yonghua Zhao, Jingqi Sun, and Lihua Wang. 2021. Similarity Analysis of Knowledge Graph-based Company Embedding for Stocks Portfolio. In *IEEE International Conference on Smart Cloud*.
[99] Yun Zhang, David Lo, Xin Xia, and Jian-Ling Sun. 2015. Multi-factor duplicate question detection in stack overflow. *Journal of Computer Science and Technology* (2015).
[100] Zhuoyue Zhao, Robert Christensen, Feifei Li, Xiao Hu, and Ke Yi. 2018. Random sampling over joins revisited. In *SIGMOD*.

[101] Kaiyang Zhou, Yongxin Yang, Andrea Cavallaro, and Tao Xiang. 2019. Omni-scale feature learning for person re-identification. In *ICCV*.
[102] Yuxuan Zhu, Tengjun Jin, Chenghao Mo, and Daniel Kang. 2026. Accelerating Approximate Analytical Join Queries over Unstructured Data with Error Guarantees. *arXiv preprint* (2026).
[103] Honglei Zhuang, Zhen Qin, Rolf Jagerman, Kai Hui, Ji Ma, Jing Lu, Jianmo Ni, Xuanhui Wang, and Michael Bendersky. 2023. Rankt5: Fine-tuning t5 for text ranking with ranking losses. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 2308–2313.
[104] Caleb Ziems, William Held, Omar Shaikh, Jiaao Chen, Zhehao Zhang, and Diyi Yang. 2024. Can large language models transform computational social science? *Computational Linguistics* 50, 1 (2024), 237–291.

Received October 2025; revised January 2026; accepted February 2026