Hyperdimensional computing for structured querying on tabular data embeddings

Sebastián Bugedo
UHasselt, DSI
Diepenbeek, Belgium
sebastian.bugedo@uhasselt.be

Stijn Vansummeren
UHasselt, DSI
Diepenbeek, Belgium
stijn.vansummeren@uhasselt.be

ABSTRACT

Tabular data embeddings have become a cornerstone of data profiling and data integration pipelines, enabling tasks such as entity annotation and resolution; schema matching; column type detection; and table search, among others. Existing approaches embed rows, columns, or entire tables into a vector space and rely on nearest-neighbor search to retrieve candidate matches. A fundamental limitation of current embedding methods is the lack of interpretable similarity scores: the concrete similarity value between a query and its nearest neighbour carries no intrinsic meaning, making it impossible to determine whether that neighbour is a true match or simply the least-dissimilar item in a corpus that contains no valid answer. This inability to set principled thresholds for retrieval undermines practical deployment, particularly for zero-match detection.

We investigate the use of HyperDimensional Computing (HDC), specifically the Holographic Reduced Representations (HRR) model, as a framework for tabular row embeddings when the retrieval task corresponds to answering structured select-project queries in vector space. Exploiting the algebraic properties of HDC operations, we derive closed-form expected similarity values for both equality and non-equality retrieval predicates, which converge to interpretable values as dimensionality increases, and use these to identify suitable retrieval thresholds. We evaluate HDC against EmbDI, a graph-based baseline, on two real-world datasets across varying table sizes and predicate lengths. Our results show that HDC matches or outperforms EmbDI for row retrieval across all configurations, handles non-equality predicates more robustly, and achieves perfect attribute projection accuracy at sufficient dimensionality—while uniquely enabling reliable identification of zero-match predicates through its principled thresholds.

VLDB Workshop Reference Format:
Sebastián Bugedo and Stijn Vansummeren. Hyperdimensional computing for structured querying on tabular data embeddings. VLDB 2026 Workshop: Tabular Data Analysis (TaDA).

VLDB Workshop Artifact Availability:
The source code, data, and/or other artifacts have been made available at https://github.com/UHasselt-DSI-Data-Systems-Lab/code-hdc-for-tabular-data.

Proceedings of the VLDB Endowment. ISSN 2150-8097.

1 INTRODUCTION

Tabular data appears ubiquitously in both private enterprise data repositories and on the public Web. In both of these settings, however, tabular data is often found to be very heterogeneous, noisy or incomplete, as well as lacking documentation to guide its correct interpretation. As a consequence, extensive preprocessing that profiles the data, cleans it, or integrates it with other sources has become essential to extract valuable insight from tables "in the wild". Prime examples of such tasks are Schema Mapping [20], the related task of Semantic Type Detection (also called Column Type Annotation) [11], Entity Annotation, Entity Resolution [4], Joinable Column Detection [2], and Table Search [19], among others.

In the past years, the increasing capabilities and popularity of deep learning have motivated the development of machine-learning-based approaches for these tasks [1, 11, 17, 29]. At their core lies the *embedding* of tabular data into vector space, either implicitly or explicitly, with the aim to augment the explicitly stated data with a form of (task-dependent) semantic understanding [1, 7, 14]. In general, a *vector embedding* for a class O of objects is a mapping f from O into some vector space, typically $\mathbb{R}^d$ for some finite dimension d. The idea is to construct a vector embedding in such a way that geometric relationships in the vector space reflect semantic relationships between the objects in O. Importantly, we want similar objects in O to be mapped to vectors that are close to one another with respect to some standard metric on the vector space. This allows to find, given a query object, semantically similar objects in a corpus of known objects using (approximate) nearest neighbor search. In entity annotation, for example, one may hence compute a vector embedding for a single table cell (possibly with information from neighboring cells in the table) and find the nearest neighbors in a known corpus to derive candidate entities to link to, after which these may be passed to other machine learning models or logic-based systems for completing the task in a RAG-like fashion. Similarly, in column type detection, one may embed entire table columns, and in entity resolution one may embed table rows.

Different tabular data embeddings methods hence attempt to capture the information and relationships in tables at different levels of granularity, producing embeddings for cells, rows, columns or sets thereof—possibly entire tables. An important desideratum of tabular embeddings for many of the above-cited tasks is *interpretability of similarity scores*. Indeed, while nearest neighbor search allows to retrieve candidate matches ranked by distance, by itself this does not specify when none of the retrieve candidates are "good enough". For example, a cosine similarity of 0.72 between a query object and its nearest neighbour could mean that the match is excellent (if the embedding space is well-calibrated and 0.72 is high in context), or that the match is poor (if the true answer simply doesn't exist in

the corpus, and 0.72 is the score of the closest wrong answer). The distinction has practical relevance. In the SemTab Entity annotation challenge [24], for example, one needs to correctly identify cells for which no corresponding entity in a knowledge graph is known. Likewise, in column type detection, one needs to recognize when no suitable pre-defined type is known, and in table union search and joinable column detection, one needs to recognize when no suitable table and column matches are known.

For task like these, we are hence in search of *interpretable* embedding methods that specify similarity thresholds below which candidate matches should be ignored. By contrast, most contemporary embedding methods are opaque in this sense. Indeed, broadly speaking we can identify two classes of embedding methods: (i) those that represent cells, rows, and/or columns as nodes in a graph and use graph representation learning to derive the embeddings [1, 3, 5, 25]; and (ii) those that serialize the table as a text string, and feed this into an LLM for attention-based embedding [6, 10, 12, 28]. Neither class provides generally interpretable distance scores.

We postulate that *hyperdimensional computing* (HDC) [13, 26] may offer a general framework for designing interpretable vector embeddings of tabular data. HDC is a symbolic approach to representation learning that encodes structured information in a high-dimensional space through simple vector operations: *binding*, *bundling*, and *unbinding*. It has been successfully applied in various domains, including natural language processing, computer vision, and robotics, and thanks to its efficient computational properties, it offers a compelling alternative to traditional neural approaches [16]. Moreover, it provides a principled way to define similarity thresholds based on the algebraic properties of the operations, as well as an interpretable framework to extract information from encoded entities with probabilistic guarantees [26].

In this paper, we make an initial study of the the suitability of HDC for defining tabular embeddings, focusing in particular on row embeddings. We draw inspiration from the work of Mellouli and Papotti [18], who argue that, for row embeddings, a reasonable notion of "semantic similarity" corresponds to being able to answer simple select-project queries directly on the embedded vector space instead of the original tabular data. To illustrate, consider a table with schema $\{A, B, C\}$. Then finding the row most similar to (the embedding of) the partial row $(A\colon a, B\colon b)$ corresponds to executing the relational algebra selection query $\sigma_{A=a,B=b}(T)$ on the input table T. Moreover, given the embedding of a row in T and an attribute A, finding the embedding of the value in the row at attribute A corresponds to being able to execute projection queries on embeddings. In particular, Mellouli and Papotti showcase that embeddings generated by the EmbDI embedding framework have this property of being able to execute structural queries on them, while it is not clear how to do this directly over LLM-based approaches. EmbDI is a graph-based embedding method designed for general data integration tasks that does not require extensive training [1]. Therefore, we consider it as a suitable baseline to compare HDC to, as it offers a high-level definition for row embeddings, and it is not LLM-based.

Our contributions are threefold:

(1) We discuss how HDC can be used for row-embeddings that allow structured select-project queries.
(2) We theoretically derive thresholds for such queries for HDC, allowing to distinguish matches from non-matches.
(3) We empirically compare the performance of HDC to EmbDI for structured querying. Our results illustrate that HDC matches or outperforms EmbDI for row retrieval across all configurations; handles non-equality predicates more robustly; and improves attribute projection accuracy, especially when the projected column has low cardinality. Furthermore, HDC enables reliable identification of zero-match predicates through its principled thresholds, something that is not available for EmbDI.

Overall, this illustrates the suitability of HDC for row-based embeddings. Its appropriateness for other kinds of embeddings remains to be studied; we discuss necessary future work in this respect in Section 7.

This paper is further organized as follows. Section 2 discusses related work while Section 3 introduces the problem statement and necessary background. Section 4 introduces HDC; discusses how to use this for structured querying; and derives the theoretical threshold values. Experimental setup is discussed in Section 5 and experimental results in Section 6. We conclude and discuss future work in Section 7. The interested reviewer may find the proofs of formal statements, as well as additional experimental results in the Appendix. The Appendix is not to be considered part of the submission, and can be read at the reviewers' discretion.

2 RELATED WORK

Tabular data embeddings. Methods for embedding tabular data fall for the most part into two families. Graph-based approaches construct a graph from table entities and apply representation learning: EmbDI [1] builds a tripartite graph over rows, columns, and values and trains word embeddings on random walks; HyTrel [3] instead uses a hypergraph to represent the table entities and relationships, to then embed using transformer layers; and Tchuitcheu et al. [25] propose heterogeneous graph embeddings that encode cell positions. LLM-based approaches serialize tables as text and exploit attention mechanisms: TaBERT [28], TaPas [10], TURL [6], and TABBIE [12] all fall in this category. Is it not clear how specific similarity values between vectors generated by these methods can be interpreted, and thus, how to set thresholds for retrieval tasks. Our work applies HDC as a third, symbolic alternative that admits principled threshold derivation.

Universal embeddings. Several works aim at embeddings that generalize across datasets for downstream integration tasks. Joinable table discovery [7] rely on column-level similarity; schema matching [20] operates at the schema level; entity resolution [4] and column type annotation [29] require cell- and column-level embeddings respectively. However, these approaches are task-specific, and similarity values cannot be generalized when used in other settings. EmbDI [1] was designed to span multiple tasks with a single embedding, while the authors in [9] propose a *universal embedding for tabular data*, using a graph auto-encoder approach. HDC is a natural candidate for such settings given its lightweight, compositional framework to generate embeddings for different types of structures. In particular, we focus on row-based embeddings.

Thresholding in HDC. For the specific HDC model that we use and describe in Section 4.1, the HRR model, theoretical analyses

have been conducted on similarity values, although for decoding tasks and vector distributions different from the ones considered in this work. We discuss the differences in detail in Section 4.3.

3 PRELIMINARIES

In this section we introduce the notation and the problem statement, and then we present EmbDI, the embedding method that we use as baseline in our experiments.

Notation. We assume given two disjoint sets $\mathcal{A}$ and $\mathcal{B}$ of attribute names and values, respectively, and range over the elements of $\mathcal{A} \cup \mathcal{B}$ by lowercase letters (e.g. a, b, c). Vectors are denoted by bold lowercase letters (e.g. $\mathbf{a}$, $\mathbf{b}$, $\mathbf{c}$) with dimension d. Indexing is done with square brackets (e.g. $\mathbf{a}[i]$ for $0 \leq i \leq d-1$) and whenever clear, $\mathbf{x}$ denotes the vector representation of x. We treat table rows as partial functions $r : \mathcal{A} \rightharpoonup \mathcal{B}$, and we denote the set of all rows by $\mathcal{R}$. For this paper, once a table T with m columns is specified, all rows come from that table.

3.1 Structured querying in vector spaces

Problem statement. We consider the problem of performing structured queries on a set of rows that have been embedded in a vector space. More concretely, for a table with $m \geq 1$ columns, we focus on two notions of querying:

- *Row retrieval.* We define a *selection predicate* of length n as a partial function $q : \mathcal{A} \rightharpoonup \mathcal{B}$ such that $|\mathrm{dom}(q)| = n$, also writing such predicates as

$$q = (a_1 : b_1, a_2 : b_2, \ldots, a_n : b_n).$$

  Given a predicate, the goal is to retrieve the matching rows. According to the interpretation we give to q, we distinguish between *equality predicates*, where we look for rows r that satisfy $r(a_i) = b_i$ for every $1 \leq i \leq n$, and *non-equality predicates*, where we look for rows that satisfy $r(a_i) \neq b_i$ for every $1 \leq i \leq n$. We denote by $\mathcal{Q}$ the set of all selection predicates.
- *Attribute projection.* Given a row and an attribute name, the goal is to retrieve the value of the attribute in the row, i.e., for r and a, we want to obtain $r(a)$.

Both tasks are specified in terms of non-embedded entities, and our goal is to perform them in vector space, after having performed row embedding and without making use of the original table. For this, we assume that we have an embedding function $\varphi : \mathcal{A} \cup \mathcal{B} \cup \mathcal{R} \cup \mathcal{Q} \to \mathcal{H}$ that maps entities to some vector space $\mathcal{H}$. We further assume that we can measure similarity between vectors in $\mathcal{H}$ using a similarity function $\mathrm{sim} : \mathcal{H} \times \mathcal{H} \to \mathbb{R}$. Both the embedding function and similarity measure are discussed in Sections 3.2 and 4.1.

Performance evaluation. To assess how well can we recover original values from encoded structures, we evaluate the performance of the methods for both tasks using the following standard metrics.

For row retrieval, we measure precision, recall and F1-score as a function of a similarity threshold τ, which is used to determine whether a row is selected or not. Specifically, for a table T and predicate q we contrast the set

$$S(T, q, \tau) = \{r \in T \mid \mathrm{sim}(\varphi(r), \varphi(q)) > \tau\}$$

against the set of rows that satisfy the selection predicate q in the original table. Alternatively, we measure the same metrics for top-k retrieval, where instead of using a threshold, we get the k most similar rows to $\varphi(q)$, for some fixed value k.

For attribute projection, we measure accuracy of exact matches: for a row r and attribute a, we obtain a candidate vector $\hat{\mathbf{v}}$ for the attribute value from $\varphi(r)$ and compare

$$\hat{b} = \arg\max_{b \in \mathcal{B}} \{\mathrm{sim}(\hat{\mathbf{v}}, \varphi(b))\} \tag{1}$$

to the ground truth $r(a)$, having an exact match if $\hat{b} = r(a)$. The exact mechanism to obtain $\hat{\mathbf{v}}$ from $\varphi(r)$ depends on the embedding framework, as described in Sections 3.2 and 4.2.

3.2 EmbDI

EmbDI [1], which is short for "embedding for data integration", is a framework proposed to generate relational, local embeddings for different entities in a tabular dataset: rows, column names and values. It is motivated by the need of a general embedding method that spans across different datasets, as well as adapting to downstream tasks such as schema matching, entity resolution and token matching. In [18], Mellouli and Papotti study selection/projection queries on the tabular embeddings generated by EmbDI, showing that the embeddings can be used to answer such queries.

EmbDI creates tabular embeddings in a three-step process. First, each row is identified with a unique ID. Then, table values and their relations are represented in a tripartite graph where nodes correspond to cell values, column names and row IDs. Cell value b is linked to column name a if it appears in a row r with $r(a) = b$. Similarly, b is linked to row ID j if it appears in row r, and j is the assigned ID for that row. Second, random walks are performed on this graph to generate sequences of nodes, with the goal of capturing the proximity between values. Lastly, these sequences are seen as sentences and are pooled to train a standard word embedding model. The default model, and the one employed in [18] is word2vec with the Skip-Gram learning method. The end result is an embedding function φ that assigns an embedding vector to (1) each column name, (2) each pair $(a : b)$ of column name a and value b, and (3) rows.

Using these base embeddings, the authors of [18] perform row and attribute projection as follows.

- For row retrieval, predicate $q = (a_1 : b_1, \ldots, a_n : b_n)$ is embedded as

$$\varphi(q) = \alpha \cdot \sum_{i=1}^{n} w_i \cdot \varphi(a_i, b_i)$$

  where $\varphi(a_i, b_i)$ is the embedding of the association of value b_i with column a_i, i.e. vector `tt_ai_bi`; w_i is a weight calculated as the inverse frequency of value b_i in column a_i in the input table, scaled to $[0.1, 1.0]$; and α takes value 1 for equality predicates and -1 for non-equality predicates.
- For attribute projection, given an attribute name a and a row embedding $\varphi(r)$, this vector is directly used as the candidate in equation (1), i.e. the projected value is

$$\hat{b} = \arg\max_{b \in \mathcal{B}(a)} \{\mathrm{sim}(\varphi(r), \varphi(a, b))\}$$

  where $\mathcal{B}(a)$ is the set of values that appear in column a.

4 HYPERDIMENSIONAL COMPUTING

Hyperdimensional computing (HDC), also known as *Vector Symbolic Architectures* (VSA), is a computational framework that represents and processes data using high-dimensional vectors [16]. The core idea is to encode information into *distributed representations*, leveraging the properties of high-dimensional spaces to enable efficient and robust computation. HDC has been applied in various domains, including cognitive modeling, machine learning, and classification, gaining attention as an alternative to the traditional computing paradigm [15]. In addition to its efficient use of resources, low latency and robustness to noise, HDC offers tools to establish theoretical guarantees for the performance of its operations [26].

Specific HDC models differ in how they map data to vectors and in the operations they use to manipulate vectors. A HDC model consist of a mapping $\varphi : A \to \mathcal{H}$ from a set A of atomic symbols to a d-dimensional vector space, producing *atomic vectors* that are stored in a dictionary-like structure called *item memory* or *codebook*.

Additionally, the model defines the following operations:

- *Similarity* $\text{sim}(\mathbf{x}, \mathbf{y})$ measures how close vectors $\mathbf{x}$ and $\mathbf{y}$ are. Based on the possible range $[l, h]$ of similarity values, we say that vectors are *similar* if they have high similarity values, *opposite* for low values, and *pseudo-orthogonal* or *dissimilar* for values around the middle of the range.
- *Bundling* $\oplus : \mathcal{H}^m \to \mathcal{H}$ compiles m vectors into a new vector $\bigoplus_{i=1}^{m} \mathbf{x}_i$ that is similar to every $\mathbf{x}_i$.
- *Binding* $\otimes : \mathcal{H} \times \mathcal{H} \to \mathcal{H}$ generates $\mathbf{x} \otimes \mathbf{y}$ that is pseudo-orthogonal to both $\mathbf{x}$ and $\mathbf{y}$.
- *Unbinding* $\otimes^{-1} : \mathcal{H} \times \mathcal{H} \to \mathcal{H}$ is defined in terms of the binding operator and an inverse element $\mathbf{x}^{-1}$ such that

$$\otimes^{-1}(\mathbf{x} \otimes \mathbf{y},\ \mathbf{x}) = \mathbf{x} \otimes \mathbf{y} \otimes \mathbf{x}^{-1} = \mathbf{y}$$

  Note that $\otimes^{-1}$ is not an inverse in the algebraic sense.

By composing these operations, it is possible to represent data structures such as sets, sequences and trees over atomic symbols in vector space [8, 26]. The resulting *composite vectors* are then stored in an *associative memory*, on top of which similarity search can be performed to retrieve similar vectors, with the objective of identifying similar data structures.

HDC models employ randomization to enable reliable similarity-based retrieval also of composite vectors: when the atomic vectors φ are drawn at random, they are pairwise pseudo-orthogonal with high probability. The HDC operations exploit this property to generate composite vectors that preserve pseudo-orthogonality (resp. similarity, when desired), hence allowing reliable similarity-based retrieval also of composite vectors. The dimensionality of $\mathcal{H}$ is important in this respect, as higher dimensions lead to a reduction in *cross-talk noise* when decoding. We next illustrate the HDC modus operandi on the embedding of tabular rows.

4.1 HRR model

In this paper we adopt the *Holographic Reduced Representations* (HRR) [21] HDC model. Like EmbDI and other common neural (non-HDC) embeddings, but in contrast to other popular HDC models such as Binary Splatter Codes, HRR produce real-valued vectors. Concretely, HRR uses d-dimensional real-valued vectors constructed either by (1) sampling the entries from the normal distribution $\mathbf{x}[k] \sim \mathcal{N}(0, 1/d)$, for $0 \le k \le d-1$, or (2) sampling from the d-dimensional unit sphere $\mathbf{x} \sim \text{Unif}(S^{d-1})$, where $S^{d-1} = \{\mathbf{x} \in \mathbb{R}^d \mid \|\mathbf{x}\| = 1\}$. We use the second approach, as it guarantees that similarity values are always within the range $[-1, 1]$, and vectors are normalized, having unit vector norm. In the first approach this only holds in expectation [22].

- Similarity is the cosine similarity

$$\text{sim}(\mathbf{x}, \mathbf{y}) = \frac{\langle \mathbf{x}, \mathbf{y} \rangle}{\|\mathbf{x}\| \cdot \|\mathbf{y}\|}$$

  where $\langle \cdot, \cdot \rangle$ is the dot product in $\mathbb{R}^d$. Vectors are pseudo-orthogonal if $\text{sim}(\mathbf{x}, \mathbf{y}) \approx 0$.
- Bundling is the normalized element-wise sum, i.e.

$$\left(\bigoplus_{i=1}^{m} \mathbf{x}_i \right)[k] = \sum_{i=1}^{m} \mathbf{x}_i[k], \quad 0 \le k \le d-1$$

- Binding is the circular convolution $\circledast$. For $0 \le k \le d-1$,

$$\begin{aligned}(\mathbf{x} \otimes \mathbf{y})[k] &= (\mathbf{x} \circledast \mathbf{y})[k] \\ &= \sum_{n=0}^{d-1} \mathbf{x}[n]\, \mathbf{y}[(k-n) \bmod d]\end{aligned}$$

- Unbinding is performed by using the pseudo-inverse or *involution* of $\mathbf{x}$, given by $\mathbf{x}^{-1} = (\mathbf{x}[0], \mathbf{x}[d-1], \ldots, \mathbf{x}[1])$. It satisfies $\mathbf{y} \approx (\mathbf{x} \circledast \mathbf{y}) \circledast \mathbf{x}^{-1}$.

Since we work with atomic vectors sampled from the unit sphere, we add a normalization step after bundling and binding to ensure the resulting vectors have unit length. Thus, $\text{sim}(\mathbf{x}, \mathbf{y}) = \langle \mathbf{x}, \mathbf{y} \rangle$ and the similarity values are always in the range $[-1, 1]$. In particular, vectors are similar if $\langle \mathbf{x}, \mathbf{y} \rangle \approx 1$ and opposite if $\langle \mathbf{x}, \mathbf{y} \rangle \approx -1$.

4.2 Encoding and decoding in HRR

Considering the presentation of record encodings proposed by Kanerva [13], we use an equivalent notion for rows in our setting. Given a row $r = (a_1 : b_1, \ldots, a_m : b_m)$, its encoding is given by:

$$\mathbf{r} = \text{norm}\left(\sum_{i=1}^{m} \text{norm}(\mathbf{a}_i \circledast \mathbf{b}_i) \right) \tag{2}$$

where $\text{norm}(\mathbf{x}) = \mathbf{x}/\|\mathbf{x}\|$ and $\mathbf{a}_i, \mathbf{b}_i \sim \text{Unif}(S^{d-1})$ are atomic vectors representing attributes and values. With this,

– Row retrieval is performed by similarity search using the encoding of selection predicates as probes over the associative memory containing the row encodings. For an equality predicate, we encode it using equation (2). Conversely, for a non-equality predicate $q = (a_1 : b_1, \ldots, a_n : b_n)$, we use

$$\mathbf{q} = \text{norm}\left(\sum_{i=1}^{n} \text{norm}(\mathbf{a}_i \circledast -\mathbf{b}_i) \right)$$

  to search for dissimilar values to the ones in the predicate.

– Attribute projection is performed by unbinding the row encoding with the attribute name, i.e. for row r and attribute a, the candidate vector is given by $\text{norm}(\mathbf{r} \circledast \mathbf{a}^{-1})$, thus

$$\hat{b} = \arg\max_{b \in \mathcal{B}} \{\text{sim}(\mathbf{r} \circledast \mathbf{a}^{-1}, \varphi(b))\}$$

  Notice that, in contrast with EmbDI, we search over the full set of values $\mathcal{B}$.

4.3 Thresholds for row retrieval

We derive the following description of the expected similarity between selection predicates and rows, in order to correctly discriminate between positive and negative matches. For a row r with m attributes and an equality predicate q with n attributes (all of which are also attributes of r), we say they have an *equality match* if $r(a_i) = b_i$ for every $(a_i : b_i)$ in q. Analogously, for a non-equality predicate q, we say that r and q have a *non-equality match* if $r(a_i) \neq b_i$ for every $(a_i : b_i)$ in q.

Proposition 4.1. *For $1 \leq n \leq m$, let $r = (a_1 : b_1, \ldots, a_m : b_m)$ be a row and $q = (a_{k_1} : v_1, \ldots, a_{k_n} : v_n)$ be an equality predicate such that $\{k_1, \ldots, k_n\} \subseteq \{1, \ldots, m\}$. Let $\mathbf{r}$ and $\mathbf{q}$ be the d-dimensional HRR encodings from Eq. (2) for r and q respectively. Then, as $d \to \infty$,*

$$\mathbb{E}[\langle \mathbf{r}, \mathbf{q} \rangle | \textit{ equality match}] = \sqrt{\frac{n}{m}},$$

$$Var[\langle \mathbf{r}, \mathbf{q} \rangle | \textit{ equality match}] \leq \frac{8dm - 8d + 8m + 4dn + n - 8}{4md^2}.$$

Here, expectation and variance are taken over the random choices of the atomic vectors.

Proposition 4.1 shows that the expected similarity and its variance in the row retrieval task are independent of the contents of r and q, depending only on their sizes and, in the case of the variance, on the dimension d.

In the literature, expectation and variance have been studied for analogous but distinct settings [22]. Specifically, the known analyses (1) adopt the HRR model where vectors have components sampled from $\mathcal{N}(0, 1/d)$, and (2) consider different decoding tasks to ours: similarity between two bound pairs of atomic vectors, or similarity of a vector against bundle, where all vectors involved are either atomic or non-independent linear combinations built from a fixed common vector (presented as a superposition memory with similarity among vectors in [22]). Our contribution lies in the analysis of the normalized unit sphere setting, focusing on decoding of rows, where bundles can be of arbitrary size and are composed of bound pairs of atomic vectors. In this sense, we describe the behavior of a more complex encoding scheme that also benefits from the interpretability of similarity values given by unit sphere normalization.

As a complement to the equality case, an analogous result is obtained for non-equality predicates. The proofs for both results are given in the Appendix.

Proposition 4.2. *For $1 \leq n \leq m$, let r be a row and q be a non-equality predicate that satisfy the hypotheses of Proposition 4.1. Then, as $d \to \infty$, their encodings satisfy*

$$\mathbb{E}[\langle \mathbf{r}, \mathbf{q} \rangle | \textit{ non-equality match}] = 0,$$

$$Var[\langle \mathbf{r}, \mathbf{q} \rangle | \textit{ non-equality match}] \leq \frac{2d + 2}{d^2}.$$

Based on Propositions 4.1 and 4.2, we define the following thresholds for the problem of row retrieval for m-attribute rows with n-attribute predicates.

Definition 4.3. Given $1 \leq n \leq m$, we define the threshold for m-attribute rows and n-attribute equality and non-equality predicates, for d-dimensional HRR encodings, as

$$\tau_{eq}(n, m, d) = \sqrt{\frac{n}{m}} - \sqrt{\frac{8dm - 8d + 8m + 4dn + n - 8}{4md^2}}$$

$$\tau_{neq}(d) = -\sqrt{\frac{2d + 2}{d^2}}.$$

The advised thresholds from Def. 4.3 consider one standard deviation from the expected value. In this way, we expand the range of similarity values that we take as a match (all $\tau \geq \tau_{\text{eq}}(n, m, d)$ or $\tau \geq \tau_{\text{neq}}(d)$, accordingly) in order to address the interference caused by cross-talk noise between bound pairs in bundles. As suggested by the variances, this cross-talk effect is reduced as d increases, so that the proposed thresholds converge to their respective expected values $\sqrt{\frac{n}{m}}$ and 0, as $d \to \infty$.

Because the attribute projection task is performed by searching for the most similar value vector to the unbound candidate $\mathbf{r} \circledast \mathbf{a}^{-1}$, no threshold is involved in the search and an equivalent analysis to the one carried out for row retrieval is not adequate. Instead, the probability of correct decoding for attribute projection could be studied. This is left for future work.

5 EXPERIMENTAL SETTING

Datasets. We generate our tables from the two real-world tabular datasets used in [18] for the evaluation of EmbDI. Both were preprocessed to contain only one table per dataset.

- Movie: $m_{\text{max}} = 15$ columns and 49875 rows
- DBLP: $m_{\text{max}} = 4$ columns and 66876 rows

Then, we generated tables with $1 \leq m \leq m_{\text{max}}$ attributes by considering only the first m columns from the original tables, keeping duplicated rows when they existed. From this, we had a set $\mathcal{T}$ containing 15 tables for the Movie dataset and 4 for DBLP.

Selection predicates. For each table $T \in \mathcal{T}$ of m columns, we generated selection predicates by sampling rows from T and taking $1 \leq n \leq m$ attribute-value pairs from the row to generate a selection predicate that matched those values. Then, we generated 10 equality and 10 non-equality predicates for each value of n and each table T. Additionally, we generated equality predicates with zero matches. For this, we sampled rows, but we modified one attribute to a value present in a different row. For tables with only one column, we sampled values from larger tables. Thus, we had 10 zero-match equality predicates for each n and table T, such that they are as close as possible to an existing row in the table.

Embedding methods. We compare the performance of EmbDI and HDC for different dimensions. For EmbDI, we took dimensions $d \in \{300, 512\}$, sampling 500,000 random walks for the Movie dataset and 1,000,000 for the DBLP dataset. These values were tested so that almost every required entity in the table has an embedding. We used Word2Vec with window size 3, as in [18]. For HDC, we used the HRR model with $d \in \{300, 512, 1024\}$. We executed 3 runs for each dimension on all experiments, to account for the randomness in the generation of the atomic vectors. Results for specific dimensions are always averaged over the 3 runs.

Evaluation. For the row retrieval task, we used two approaches. First, we evaluated the F1-score as a function of k when retrieving the top-k closest rows for $k \in \{1, 2, 5, 10, 20\}$ for each table T and

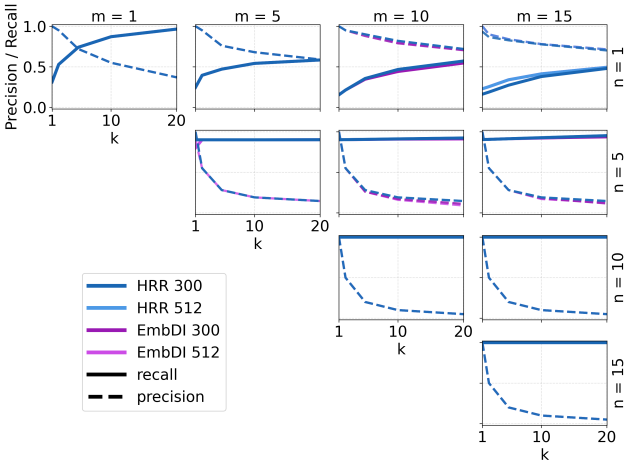


Figure 1: Recall and precision vs top-k for equality predicates on the Movie dataset, for selected values of m, n.

each selection predicate length n. This was done to replicate the evaluation setup from [18]. Second, we obtained the F1-score as a function of the threshold τ when computing the set $S(T, q, \tau)$ for every table T and selection predicate q. For equality predicates, $\tau \in [0.1, 1]$, while $\tau \in [-0.3, 0.2]$ for non-equality predicates, taking values in the specified ranges with a step of $s = 0.05$.

Additionally, we considered the subtask of zero-match predicates. For each HRR dimension d, we measured the number of rows retrieved $|S(T, q, \tau)|$ per predicate q and table T when fixing the threshold value $\tau = \tau_{\text{eq}}(n, \text{cols}(T), d)$, where $\text{cols}(T)$ is the number of columns in table T. Since EmbDI does not have a single threshold description, we only applied this task to HDC.

Finally, for attribute projection, we sampled 50 rows from each table T and obtain the candidate value for each attribute in every row. We then evaluated performance by measuring the accuracy of exact matches for each table T and attribute name. We executed 3 runs for HRR and averaged the results.

6 RESULTS

6.1 Row retrieval

Equality selection predicates. The overall behavior of both approaches varies with the number of columns m and predicate size n.

Figure 1 shows recall and precision when retrieving the top-k most similar rows on the Movie dataset, for selected values of m, n and for $k \in \{1, 2, 5, 10, 20\}$. Both EmbDI and HDC behave very similarly, with almost identical metrics across all k values, explaining why the curves for EmbDI are often not visible in Figure 1. This behavior is also present in the full grid available in the Appendix in Figure 7. For $k \in \{1, 2\}$ and increasing values of m and n, both approaches have precision and recall approaching 1.0. They are hence comparable in how they correctly select the most similar row(s) for most combinations of m and n. However, as k increases, precision drops rapidly, for all m, n. Note that it is impossible to determine the correct value of k to use, as this depends on the number of answers in the query result. As such, top-k based retrieval is in general inadequate for the general row retrieval problem. This is especially true for predicates with zero matches, where the top-k approach will always incorrectly return $k \geq 1$ rows.

As an alternative to top-k selection, we evaluate the performance of HDC and EmbDI when retrieving rows whose similarity is above a certain threshold τ. Figure 2 shows selected results for equality predicates on the Movie dataset, while Figure 3 corresponds to the full results for the DBLP dataset. Figure 6 in the Appendix shows the full grid for every combination of m and n for the Movie dataset. In these figures, we plot the obtained F1 score as a function of the threshold used, also indicating the expected (Prop. 4.1) and advised threshold (Def. 4.3). Across all scenarios, we confirm that dimension has a positive effect on the performance of HDC, with higher d leading to higher F1-scores across different threshold values, explained by the increased robustness to cross-talk in bundles. The performance of EmbDI is not greatly affected by dimension.

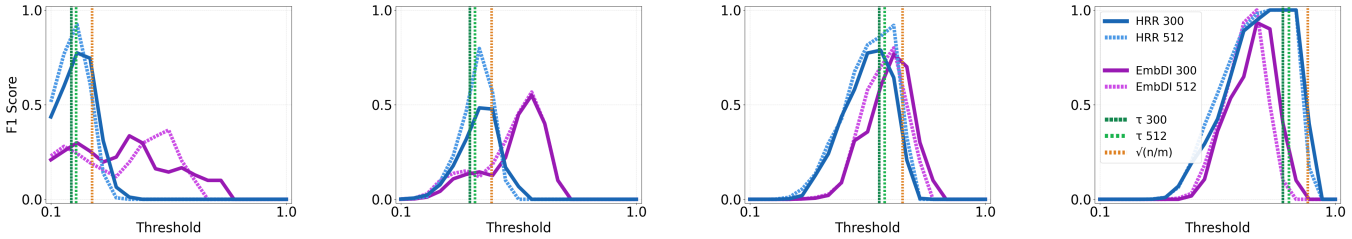
For a fixed table size m, both HDC and EmbDI tend to increase their performance, measured as peak F1-score, as we increase the number of attributes n in the predicate from 1 to m, being more noticeable for larger m values. It is worth noting that for shorter predicates for fixed m, HDC has a slight decrease in performance before reaching perfect retrieval for larger n values. EmbDI has a more consistent increase in performance with predicate length n.

For smaller fixed values of n, the performance of EmbDI degrades as we add more columns to the table. HDC tends to maintain F1-score peaks but the range of threshold values to reach them becomes narrower, particularly for the Movie dataset, which involves longer rows than the DBLP dataset. In particular, $n = 1$ is the predicate size for which HDC performs best when compared with EmbDI. For larger fixed values of n, both methods exhibit a more stable tendency as we increase m and their performance is comparable. This suggests that rows that are similar to a predicate in HDC tend to have comparable similarity scores across different predicates.

This is confirmed by our theoretical thresholds, that are shown in Figures 2 and 3 for dimensions $d \in \{300, 512, 1024\}$, as well as the mean threshold $\tau_{mean} = \sqrt{n/m}$ appearing as the rightmost vertical dashed line. We note that our advised thresholds graphically align with the observed peaks for HDC, showing the displacement of the peaks as we change m and n, and suggesting definition 4.3 is pertinent. Moreover, τ_{mean} indicates threshold values such that F1-score decreases given the partial exclusion of matching rows.

To better illustrate the performance using the proposed thresholds, Table 1 contains the F1-scores for each dimension as a function of m, averaged over values of n for the Movie dataset. For comparison, we include the average over n of the maximum performance for EmbDI on the same scenario given the threshold values $0.2 \leq \tau \leq 1$. We can see that the proposed thresholds for HDC are consistent with the observed peaks, and that they outperform the average best performance of EmbDI in almost every case.

Lastly, to highlight the flexibility of the thresholded approach, we explore the subtask of selecting rows for zero-match predicates. Table 2 shows the average number of retrieved rows for zero-match equality predicates on the Movie dataset, when using the proposed thresholds. We can see that for HDC, the proposed thresholds are effective at retrieving zero rows for most cases, with the expected degradation for larger m caused by the cross-talk of more complex bundles. This goes in line with the increased performance for larger



(a) $n = 1$ **and** $m = 15$ **(b)** $n = 3$ **and** $m = 15$ **(c)** $n = 7$ **and** $m = 15$ **(d)** $n = 12$ **and** $m = 15$

Figure 2: Selected results of F1-score vs similarity threshold for row retrieval over equality predicates on the Movie dataset.

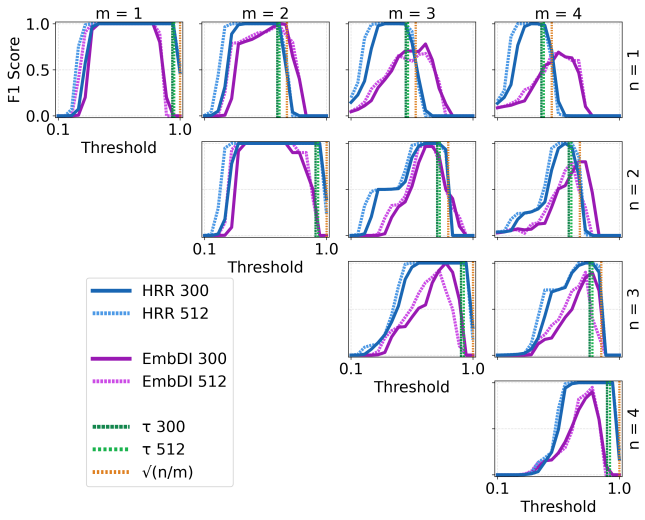


Figure 3: F1-score vs similarity threshold for row retrieval, for the DBLP dataset on equality predicates.

Table 1: F1-scores for row retrieval on the Movie dataset using $\tau_{\mathbf{eq}}(n, m, d)$ for equality predicates, and average best score for EmbDI across all thresholds. Averaged over values of n.

m	HRR 300	HRR 512	HRR 1024	EmbDI 300	EmbDI 512
1	1.00	1.00	1.00	1.00	1.00
2	1.00	1.00	1.00	0.92	0.95
3	1.00	1.00	1.00	0.99	0.98
4	0.99	0.99	1.00	0.89	0.88
5	0.99	0.98	1.00	0.78	0.83
6	0.97	0.99	0.99	0.85	0.83
7	0.94	0.99	1.00	0.79	0.79
8	0.90	0.99	0.99	0.71	0.75
9	0.90	0.97	0.99	0.66	0.65
10	0.85	0.97	0.99	0.71	0.69
11	0.85	0.97	0.99	0.77	0.80
12	0.84	0.93	0.99	0.72	0.74
13	0.85	0.92	0.98	0.79	0.79
14	0.85	0.92	0.98	0.79	0.79
15	0.82	0.92	0.99	0.76	0.82

dimension d. Still, for the different values of d, the number of retrieved rows is very low compared to the 49875 total rows of the table. For EmbDI, it is not clear how to identify zero-match queries.

Table 2: Average number of retrieved rows for zero-match equality predicates on the Movie dataset, using $\tau_{\mathbf{eq}}(n, m, d)$.

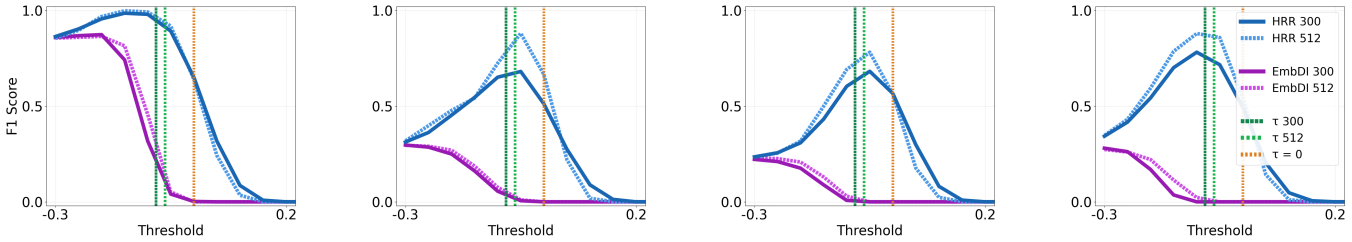
m	HRR 300	HRR 512	HRR 1024
1	0.00	0.00	0.00
2	0.00	0.00	0.00
3	0.00	0.00	0.00
4	0.00	0.00	0.00
5	0.00	0.00	0.00
6	0.01	0.00	0.00
7	0.37	0.00	0.00
8	4.04	0.00	0.00
9	43.04	0.14	0.00
10	20.60	0.66	0.00
11	7.20	0.29	0.00
12	19.20	16.44	0.00
13	30.83	4.43	0.01
14	61.42	2.85	0.06
15	101.99	10.95	0.40

Non-equality selection predicates. In contrast with equality predicates, trends in retrieval performance using non-equality conditions do not vary much when changing n or m. Figure 4 shows selected examples for the Movie dataset, while Figures 8 and 5 contain the full grids for both datasets. Except for short predicates in the Movie dataset, EmbDI has very low performance, while HDC has consistent performance across all m and n values, with results even comparable to the case of equality predicates. Both methods are comparable for $m, n < 4$, being able to retrieve with high F1-score given that most rows are dissimilar to the predicate. Therefore, for a small enough threshold, most retrieved rows are correct. From all, for larger table and predicate sizes, HDC achieves better retrieval for rows being different from specified values, suggesting a better encoding of the notion of being opposite.

As seen in Figure 4, in the same way as in equality predicates, the proposed thresholds for perfect non-equality matching are consistent with the observed peaks for HDC. Furthermore, the mean threshold $\tau_{mean} = 0$ correctly indicates the values of threshold such that recall decreases. Finally, we see a slight increase in performance with dimension for HDC, while EmbDI is again not affected by it.

6.2 Attribute projection

Table 3 shows the results for attribute projection on both datasets. For the Movie dataset, we include a selection of columns exhibiting different attribute cardinalities that have an impact on EmbDI's performance, e.g. *director* has a wide range of values, whereas *rating*



(a) $n = 1$ **and** $m = 15$ **(b)** $n = 3$ **and** $m = 15$ **(c)** $n = 7$ **and** $m = 15$ **(d)** $n = 15$ **and** $m = 15$

Figure 4: Selected results of F1-score vs similarity threshold for row retrieval over non-equality predicates on the Movie dataset. Results are shown for fixed table size $m = 15$ and increasing predicate length n.

Table 3: Average accuracy for attribute projection on full tables

(a) Movie Dataset (selected columns from $m = 15$)

model	overall	director	prod. companies	rating	genres
EmbDI 300	0.78	0.96	0.68	0.30	0.82
EmbDI 512	0.69	0.96	0.62	0.08	0.74
HRR 300	0.50	0.42	0.51	0.57	0.48
HRR 512	0.93	0.93	0.92	0.98	0.97
HRR 1024	1.00	1.00	1.00	1.00	1.00

(b) DBLP dataset ($m = 4$)

model	overall	authors	title	venue	year
EmbDI 300	0.82	1.00	0.98	0.70	0.62
EmbDI 512	0.80	1.00	0.98	0.64	0.56
HRR 300	1.00	1.00	1.00	1.00	1.00
HRR 512	1.00	1.00	1.00	1.00	1.00
HRR 1024	1.00	1.00	1.00	1.00	1.00

has limited numerical values. The table shows the overall accuracy for each model, as well as the accuracy for each column separately.

For both datasets, a higher dimension for EmbDI does not translate into higher accuracy, while for HDC it allows reaching perfect decoding for all columns. The size of the table has an impact on the lower dimensions of HDC, namely $d = 300$ for the larger table of the Movie dataset, which gives HDC lower performance than its neural counterpart with the same dimension. Nonetheless, $d = 512$ is sufficient to overcome this issue in both datasets. Additionally, the performance of HDC is more consistent across columns, while for EmbDI it varies greatly as was already noted in [18]. In particular, EmbDI is noticeably weak at decoding low cardinality columns.

7 CONCLUSIONS AND FUTURE WORK

The experiments presented show that HDC offers a framework for performing simple similarity-based querying on tabular data embeddings, both for row retrieval and attribute projection. Dimension is a key factor in the implementation of HDC, with higher dimensions outperforming EmbDI for both tasks. Even for comparable dimensions, we observe that HDC can generalize over attributes for the projection task, and reach higher performance for row retrieval when using advised thresholds. If we consider even higher dimensions, HDC can achieve perfect decoding for projection, however, vectors that are too large may not be desirable in practice.

The most important strength of HDC in this context is the threshold description for general selection predicates. Retrieving results based on a fixed k for nearest neighbor search is insufficient, as it impacts precision and recall directly whenever the number of results for an arbitrary predicate is different from k. Moreover, it is not even possible to correctly identify predicates with zero results. Our threshold definitions show that it is possible to achieve good performance across different table and predicate lengths, a tool that is not available for EmbDI. In this sense, the principled framework of operations and encoding in HDC is crucial to simulate exact matching in a similarity-based way.

Interestingly, the previous observations apply for the problem of exact matching, both for projection and retrieval, which we argue is not the only environment where HDC shines. The two-task evaluation we present is a method to assess the ability of embeddings to preserve tabular structure, but it does not fully capture other types of querying that can be performed based on similarity. Altogether, our results for exact matching are a strong indicator that HDC preserves and can use similarity in composite entities based on the similarity of their components, which opens the door to (1) queries that are based on semantic relations between values and (2) alternative structure encodings.

With regard to *semantic queries*, the use of random atomic vectors is essential for HDC. In many situations, however, we may want to start from pre-trained vectors for atomic data values (e.g. Word2Vec), such that embedded row vectors become similar even if they contain different, but related, data values. It is an open question to what extent the decoding capabilities of HDC can be coupled with pre-trained atomic embeddings. This motivates the definition of alternative notions of querying that go beyond the exact matching of relational databases, to evaluate embeddings.

We have focused in this paper on row embeddings. The suitability of HDC for other embedding kinds, such as column or table embeddings, is also an open question. In particular, based on the idea of visually traversing tables as an inspiration for a graph representation and embedding in [25], we can think on having a hybrid embedding that navigates in the table structure non-uniformly, e.g. by walking over the grid of the table and encoding positional information about cells. In this context, yet another HDC operation, permutation [8], is relevant to encode positions of tabular entities.

ACKNOWLEDGMENTS

This work was supported by the Flanders AI (FAIR) research program; by the Research Foundation Flanders (FWO) under research project Grant No. G019222N; and by the Bijzonder Onderzoeksfonds (BOF) of Hasselt University Grant No. BOF20ZAP02.

REFERENCES

[1] Riccardo Cappuzzo, Paolo Papotti, and Saravanan Thirumuruganathan. 2020. Creating Embeddings of Heterogeneous Relational Datasets for Data Integration Tasks. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data* (Portland, OR, USA) *(SIGMOD '20)*. Association for Computing Machinery, New York, NY, USA, 1335–1349. https://doi.org/10.1145/3318464.3389742

[2] Emanuele Cavalleri, Matteo Castagna, and Marco Mesiti. 2026. A Semantic Schema-Based Catalog for Identifying Joinable Columns via LLMs. In *Flexible Query Answering Systems*, Guy De Tré, Sotir Sotirov, Janusz Kacprzyk, Giuseppe Psaila, Grégory Smits, Troels Andreasen, Gloria Bordogna, and Henrik Legind Larsen (Eds.). Springer Nature Switzerland, Cham, 206–218.

[3] Pei Chen, Soumajyoti Sarkar, Leonard Lausen, Balasubramaniam Srinivasan, Sheng Zha, Ruihong Huang, and George Karypis. 2023. HyTrel: Hypergraph-enhanced Tabular Data Representation Learning. In *Advances in Neural Information Processing Systems*, A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (Eds.), Vol. 36. Curran Associates, Inc., 32173–32193. https://proceedings.neurips.cc/paper_files/paper/2023/file/66178beae8f12fcd48699de95acc1152-Paper-Conference.pdf

[4] Vassilis Christophides, Vasilis Efthymiou, Themis Palpanas, George Papadakis, and Kostas Stefanidis. 2020. An Overview of End-to-End Entity Resolution for Big Data. *ACM Comput. Surv.* 53, 6, Article 127 (Dec. 2020), 42 pages. https://doi.org/10.1145/3418896

[5] Tamara Cucumides and Floris Geerts. 2025. From Features to Structure: Task-Aware Graph Construction for Relational and Tabular Learning with GNNs. In *VLDB 2025 Workshop: Tabular Data Analysis (TaDA)*. https://www.vldb.org/2025/Workshops/VLDB-Workshops-2025/TaDA/TaDA25_5.pdf

[6] Xiang Deng, Huan Sun, Alyssa Lees, You Wu, and Cong Yu. 2022. TURL: Table Understanding through Representation Learning. *SIGMOD Rec.* 51, 1 (June 2022), 33–40. https://doi.org/10.1145/3542700.3542709

[7] Yuyang Dong, Chuan Xiao, Takuma Nozawa, Masafumi Enomoto, and Masafumi Oyamada. 2023. DeepJoin: Joinable Table Discovery with Pre-Trained Language Models. *Proc. VLDB Endow.* 16, 10 (June 2023), 2458–2470. https://doi.org/10.14778/3603581.3603587

[8] E. Paxon Frady, Spencer J. Kent, Bruno A. Olshausen, and Friedrich T. Sommer. 2020. Resonator Networks, 1: An Efficient Solution for Factoring High-Dimensional, Distributed Representations of Data Structures. *Neural Comput.* 32, 12 (Dec. 2020), 2311–2331. https://doi.org/10.1162/neco_a_01331

[9] Astrid Franz, Frederik Hoppe, Marianne Michaelis, and Udo Göbel. 2025. Universal Embeddings of Tabular Data. In *VLDB 2025 Workshop: Tabular Data Analysis (TaDA)*. https://www.vldb.org/2025/Workshops/VLDB-Workshops-2025/TaDA/TaDA25_7.pdf

[10] Jonathan Herzig, Pawel Krzysztof Nowak, Thomas Müller, Francesco Piccinno, and Julian Eisenschlos. 2020. TaPas: Weakly Supervised Table Parsing via Pre-training. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, Dan Jurafsky, Joyce Chai, Natalie Schluter, and Joel Tetreault (Eds.). Association for Computational Linguistics, Online, 4320–4333. https://doi.org/10.18653/v1/2020.acl-main.398

[11] Madelon Hulsebos, Kevin Hu, Michiel Bakker, Emanuel Zgraggen, Arvind Satyanarayan, Tim Kraska, Çagatay Demiralp, and César Hidalgo. 2019. Sherlock: A Deep Learning Approach to Semantic Data Type Detection. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining* (Anchorage, AK, USA) *(KDD '19)*. Association for Computing Machinery, New York, NY, USA, 1500–1508. https://doi.org/10.1145/3292500.3330993

[12] Hiroshi Iida, Dung Thai, Varun Manjunatha, and Mohit Iyyer. 2021. TABBIE: Pretrained Representations of Tabular Data. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, Kristina Toutanova, Anna Rumshisky, Luke Zettlemoyer, Dilek Hakkani-Tur, Iz Beltagy, Steven Bethard, Ryan Cotterell, Tanmoy Chakraborty, and Yichao Zhou (Eds.). Association for Computational Linguistics, Online, 3446–3456. https://doi.org/10.18653/v1/2021.naacl-main.270

[13] Pentti Kanerva. 2009. Hyperdimensional Computing: An Introduction to Computing in Distributed Representation with High-Dimensional Random Vectors. *Cognitive Computation* 1, 2 (June 2009), 139–159. https://doi.org/10.1007/s12559-009-9009-8

[14] Aamod Khatiwada, Grace Fan, Roee Shraga, Zixuan Chen, Wolfgang Gatterbauer, Renée J. Miller, and Mirek Riedewald. 2023. SANTOS: Relationship-based Semantic Table Union Search. *Proc. ACM Manag. Data* 1, 1, Article 9 (May 2023), 25 pages. https://doi.org/10.1145/3588689

[15] Denis Kleyko, Dmitri Rachkovskij, Evgeny Osipov, and Abbas Rahimi. 2023. A Survey on Hyperdimensional Computing aka Vector Symbolic Architectures, Part II: Applications, Cognitive Models, and Challenges. *ACM Comput. Surv.* 55, 9, Article 175 (Jan. 2023), 52 pages. https://doi.org/10.1145/3558000

[16] Denis Kleyko, Dmitri A. Rachkovskij, Evgeny Osipov, and Abbas Rahimi. 2022. A Survey on Hyperdimensional Computing Aka Vector Symbolic Architectures, Part I: Models and Data Transformations. *ACM Comput. Surv.* 55, 6 (Dec. 2022), 130:1–130:40. https://doi.org/10.1145/3538531

[17] Yuliang Li, Jinfeng Li, Yoshihiko Suhara, AnHai Doan, and Wang-Chiew Tan. 2020. Deep entity matching with pre-trained language models. *Proc. VLDB Endow.* 14, 1 (Sept. 2020), 50–60. https://doi.org/10.14778/3421424.3421431

[18] Mariam Mellouli and Paolo Papotti. 2025. Evaluating SQL Selection/Projection over Table Embeddings. In *VLDB 2025 Workshop: Tabular Data Analysis (TaDA)*. https://www.vldb.org/2025/Workshops/VLDB-Workshops-2025/TaDA/TaDA25_4.pdf

[19] Fatemeh Nargesian, Erkang Zhu, Ken Q. Pu, and Renée J. Miller. 2018. Table union search on open data. *Proc. VLDB Endow.* 11, 7 (March 2018), 813–825. https://doi.org/10.14778/3192965.3192973

[20] Marcel Parciak, Brecht Vandevoort, Frank Neven, Liesbet M. Peeters, and Stijn Vansummeren. 2025. LLM-Matcher: A Name-Based Schema Matching Tool using Large Language Models. In *Companion of the 2025 International Conference on Management of Data* (Berlin, Germany) *(SIGMOD/PODS '25)*. Association for Computing Machinery, New York, NY, USA, 203–206. https://doi.org/10.1145/3722212.3725112

[21] Tony A. Plate. 1995. Holographic reduced representations. *IEEE Transactions on Neural Networks* 6, 3 (1995), 623–641. https://doi.org/10.1109/72.377968

[22] Tony A Plate. 2003. *Holographic Reduced Representation: Distributed Representation for Cognitive Structures.* Center for the Study of Language and Information (CSLI), Stanford, CA.

[23] John A Rice. 2007. *Mathematical statistics and data analysis* (3rd ed.). Brooks/Cole, Belmont, CA.

[24] SemTab Challenge. 2025. SemTab 2025: Semantic Web Challenge on Tabular Data to Knowledge Graph Matching — LLMs & Tabular Data Matching. https://sem-tab-challenge.github.io/2025/. Co-located with the 24th International Semantic Web Conference (ISWC 2025). Accessed: May 2026.

[25] Willy Carlos Tchuitcheu, Tan Lu, and Ann Dooms. 2024. Table representation learning using heterogeneous graph embedding. *Pattern Recognition* 156 (2024), 110734. https://doi.org/10.1016/j.patcog.2024.110734

[26] Anthony Thomas, Sanjoy Dasgupta, and Tajana Rosing. 2022. A Theoretical Perspective on Hyperdimensional Computing. *J. Artif. Int. Res.* 72 (Jan. 2022), 215–249. https://doi.org/10.1613/jair.1.12664

[27] Roman Vershynin. 2018. *High-Dimensional Probability: An Introduction with Applications in Data Science.* Cambridge University Press, Cambridge. https://doi.org/10.1017/9781108231596

[28] Pengcheng Yin, Graham Neubig, Wen-tau Yih, and Sebastian Riedel. 2020. TaBERT: Pretraining for Joint Understanding of Textual and Tabular Data. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, Dan Jurafsky, Joyce Chai, Natalie Schluter, and Joel Tetreault (Eds.). Association for Computational Linguistics, Online, 8413–8426. https://doi.org/10.18653/v1/2020.acl-main.745

[29] Dan Zhang, Madelon Hulsebos, Yoshihiko Suhara, Çağatay Demiralp, Jinfeng Li, and Wang-Chiew Tan. 2020. Sato: contextual semantic type detection in tables. *Proc. VLDB Endow.* 13, 12 (July 2020), 1835–1848. https://doi.org/10.14778/3407790.3407793

Appendix A PROOF OF SIMILARITY RESULTS

On nomenclature. In the following results, we refer to the circular convolution $\mathbf{c} = \mathbf{a} \circledast \mathbf{b}$ as a *binding* or *bound pair*. We refer to the sum of normalized bound pairs $\mathbf{r} = \sum_{i=1}^{m} \text{norm}(\mathbf{a}_i \circledast \mathbf{b}_i)$ as a *bundle of normalized bound pairs*. The similarity between two unit vectors $\mathbf{x}, \mathbf{y}$ is given by their inner product $\langle \mathbf{x}, \mathbf{y} \rangle$.

Lemma A.1. *If* $\mathbf{a} = (\mathbf{a}[0], \ldots, \mathbf{a}[d-1]) \sim \mathit{Unif}(S^{d-1})$, *then the individual components* $\mathbf{a}[i] \sim \mathcal{N}(0, 1/d)$ *for every* $0 \leq i \leq d-1$ *as* $d \to \infty$.

Proof. It follows from Theorem 3.3.9 in [27]. □

Lemma A.2. *For vectors* $\mathbf{a}, \mathbf{b}, \mathbf{c}, \mathbf{d} \sim \mathit{Unif}(S^{d-1})$, *the dot product* $X = \langle \mathbf{a} \circledast \mathbf{b}, \mathbf{c} \circledast \mathbf{d} \rangle$ *satisfies the following properties as* $d \to \infty$*:*

- *If* $\mathbf{a} = \mathbf{c}, \mathbf{b} = \mathbf{d}$, *and* $\mathbf{a}, \mathbf{b}$ *are independent, then* $X \sim \mathcal{N}\left(1, \frac{6d+4}{d^2}\right)$.

- *If* $\mathbf{a} = \mathbf{c}$ *and* $\mathbf{a}, \mathbf{b}, \mathbf{d}$ *are independent, then* $X \sim \mathcal{N}\left(0, \frac{2d+2}{d^2}\right)$.
- *If* $\mathbf{a}, \mathbf{b}, \mathbf{c}, \mathbf{d}$ *are independent, then* $X \sim \mathcal{N}\left(0, \frac{1}{d}\right)$.

Proof. Follows from Lemma A.1 and rows (6), (8) and (10) of Table 3 in [22]. □

Lemma A.3. *For* $\mathbf{a}, \mathbf{b} \sim Unif(S^{d-1})$ *independent, the coordinates of their circular convolution* $\mathbf{c} = \mathbf{a} \circledast \mathbf{b}$ *are independent and distributed as* $\mathcal{N}(0, 1/d)$ *for* $d \to \infty$.

Proof. An arbitrary coordinate of the binding $\mathbf{c} = \mathbf{a} \circledast \mathbf{b}$ are given by

$$\mathbf{c}[i] = \sum_{k=0}^{d-1} \mathbf{a}[k]\ \mathbf{b}[(i-k) \bmod d],$$

for $0 \le i \le d-1$. Let $X_k = \mathbf{a}[k]$ and $Y_k = \mathbf{b}[(i-k) \bmod d]$ so that

$$\mathbf{c}[i] = \sum_{k=0}^{d-1} X_k Y_k.$$

Since $\mathbf{a}$ and $\mathbf{b}$ are independent, the pairs (X_k, Y_k) are independent for $0 \le k \le d-1$. By Lemma A.1 we know each $X_k, Y_k \sim \mathcal{N}(0, 1/d)$ as $d \to \infty$. Noticing each product can be written as

$$X_k Y_k = \frac{1}{4}\left((X_k + Y_k)^2 - (X_k - Y_k)^2\right),$$

let $U_k = X_k + Y_k$ and $V_k = X_k - Y_k$. Then $U_k, V_k \sim \mathcal{N}(0, 2/d)$ and $U_k^2, V_k^2 \sim \frac{2}{d}\chi^2(1)$. With this, the sum of d independent products can be written as

$$\mathbf{c}[i] = \sum_{k=0}^{d-1} X_k Y_k = \left(\sum_{k=0}^{d-1} \frac{1}{4} U_k^2\right) - \left(\sum_{k=0}^{d-1} \frac{1}{4} V_k^2\right) = U - V,$$

where U and V are sums of d independent $\frac{1}{2d}\chi^2(1)$ variables, thus $U, V \sim \frac{1}{2d}\chi^2(d)$. As $d \to \infty$, both variables approach $\mathcal{N}(\frac{1}{2}, \frac{1}{2d})$ and $\mathbf{c}[i] \sim \mathcal{N}(0, \frac{1}{2d} + \frac{1}{2d}) = \mathcal{N}(0, \frac{1}{d})$, thus the coordinates of $\mathbf{c}$ are independent and distributed as $\mathcal{N}(0, 1/d)$. □

Lemma A.4. *For* $\mathbf{a}, \mathbf{b} \sim Unif(S^{d-1})$ *independent, the norm of their circular convolution* $\mathbf{c} = \mathbf{a} \circledast \mathbf{b}$ *distributes as*

$$\|\mathbf{c}\| \sim \mathcal{N}\left(1, \frac{1}{2d}\right)$$

as $d \to \infty$.

Proof. By Lemma A.3, as $d \to \infty$ the components of of $\mathbf{c}$ are independent and distributed as $\mathcal{N}(0, 1/d)$. To approximate the norm for large d, consider the standardized variables

$$Z_i = \frac{\mathbf{c}[i] - \mathbb{E}[\mathbf{c}[i]]}{\sqrt{\mathrm{Var}[\mathbf{c}[i]]}} = \frac{\mathbf{c}[i]}{\sqrt{1/d}} \sim \mathcal{N}(0, 1)$$

Then,

$$\|\mathbf{c}\| = \sqrt{\sum_{i=0}^{d-1} \frac{1}{d} Z_i^2} = \sqrt{\frac{1}{d}}\, Y$$

where $Y = \sqrt{\sum_{i=0}^{d-1} Z_i^2} \sim \chi(d)$. By the expectation and variance of the Chi distribution and Stirling's approximation of the Gamma function, we have that

$$\mathbb{E}[Y] = \sqrt{2}\frac{\Gamma(\frac{d+1}{2})}{\Gamma(\frac{d}{2})} \xrightarrow{d\to\infty} \sqrt{d - \frac{1}{2}} \approx \sqrt{d}$$

$$\mathrm{Var}[Y] = d - \mathbb{E}[Y]^2 \xrightarrow{d\to\infty} d - (d - \frac{1}{2}) = \frac{1}{2}$$

Thus,

$$\mathbb{E}[\|\mathbf{c}\|] = \sqrt{\frac{1}{d}}\mathbb{E}[Y] = \sqrt{\frac{1}{d}}\sqrt{d} = 1$$

$$\mathrm{Var}[\|\mathbf{c}\|] = \frac{1}{d}\mathrm{Var}[Y] = \frac{1}{2d}.$$

Since a $\chi(d)$ distribution approaches a normal distribution for large d, we conclude that $\|\mathbf{c}\| \sim \mathcal{N}\left(1, \frac{1}{2d}\right)$ as $d \to \infty$. □

Lemma A.5. *For* $\mathbf{a}_1, \ldots, \mathbf{a}_m, \mathbf{b}_1, \ldots, \mathbf{b}_m \sim Unif(S^{d-1})$, *the norm of the bundle of* m *bound pairs*

$$\mathbf{r} = \sum_{i=1}^{m} norm(\mathbf{a}_i \circledast \mathbf{b}_i),$$

satisfies that, as $d \to \infty$,

$$\|\mathbf{r}\| \sim \mathcal{N}\left(\sqrt{m}, \frac{m}{2d}\right).$$

Proof. By Lemma A.3, as $d \to \infty$ the components of each non-normalized binding $\mathbf{a}_i \circledast \mathbf{b}_i$ in $\mathbf{r}$ are independent and distributed as $\mathcal{N}(0, 1/d)$. As described in [27], the normalization of such a vector is uniform on the unit sphere S^{d-1}, i.e.

$$\frac{\mathbf{a}_i \circledast \mathbf{b}_i}{\|\mathbf{a}_i \circledast \mathbf{b}_i\|} \sim \mathrm{Unif}(S^{d-1}).$$

Then, since $\mathbf{r}$ is the sum of m independent vectors drawn from the unit sphere, by Lemma A.1 the coordinates of $\mathbf{r}$ are the sum of m independent $\mathcal{N}(0, 1/d)$ variables. Therefore, as $d \to \infty$, the coordinates of $\mathbf{r}$ are independent and distributed as

$$\mathbf{r}[i] \sim \mathcal{N}\left(0, \frac{m}{d}\right).$$

To study the norm for large d, we proceed analogously to the proof of Lemma A.4. Let us consider the standardized variables

$$Z_i = \frac{\mathbf{r}[i] - \mathbb{E}[\mathbf{r}[i]]}{\sqrt{\mathrm{Var}[\mathbf{r}[i]]}} = \frac{\mathbf{r}[i]}{\sqrt{m/d}} \sim \mathcal{N}(0, 1)$$

Then,

$$\|\mathbf{r}\| = \sqrt{\sum_{i=0}^{d-1} \frac{m}{d} Z_i^2} = \sqrt{\frac{m}{d}}\, Y$$

where $Y = \sqrt{\sum_{i=0}^{d-1} Z_i^2} \sim \chi(d)$. By using the expectation and variance of the Chi distribution, we have that

$$\mathbb{E}[\|\mathbf{r}\|] = \sqrt{\frac{m}{d}}\mathbb{E}[Y] = \sqrt{\frac{m}{d}}\sqrt{d} = \sqrt{m}$$

$$\mathrm{Var}[\|\mathbf{r}\|] = \frac{m}{d}\mathrm{Var}[Y] = \frac{m}{2d}.$$

Since a $\chi(d)$ distribution approaches a normal distribution for large d, we conclude that $\|\mathbf{r}\| \sim \mathcal{N}\left(\sqrt{m}, \frac{m}{2d}\right)$ as $d \to \infty$. □

Proposition A.6. *For $1 \le n \le m$, let $\mathbf{a}_1, \ldots, \mathbf{a}_m, \mathbf{b}_1, \ldots, \mathbf{b}_m$ and $\mathbf{v}_1, \ldots, \mathbf{v}_n$ be sampled uniformly from the d-dimensional unit sphere S^{d-1}. For row $r = (a_1 : b_1, \ldots, a_m : b_m)$ and selection equality predicate $q = (a_{k_1} : v_1, \ldots, a_{k_n} : v_n)$ for $\{k_1, \ldots, k_n\} \subseteq \{1, \ldots, m\}$, consider their encoding*

$$\mathbf{r} = norm\left(\sum_{i=1}^{m} norm(\mathbf{a}_i \circledast \mathbf{b}_i)\right), \quad \mathbf{q} = norm\left(\sum_{j=1}^{n} norm(\mathbf{a}_{k_j} \circledast \mathbf{v}_j)\right).$$

Then, as $d \to \infty$, the expectation and variance of their similarity over the random choice of atomic vectors satisfy

$$\mathbb{E}[\langle \mathbf{r}, \mathbf{q} \rangle |\ equality\ match] = \sqrt{\frac{n}{m}}$$
$$Var[\langle \mathbf{r}, \mathbf{q} \rangle |\ equality\ match] \le \frac{8dm - 8d + 8m + 4dn + n - 8}{4md^2}.$$

Proof. By the linearity of the inner product, we can write

$$\begin{aligned}
\langle \mathbf{r}, \mathbf{q} \rangle &= \left\langle \mathrm{norm}(\sum_{i=1}^{m} \mathrm{norm}(\mathbf{a}_i \circledast \mathbf{b}_i)),\ \mathrm{norm}(\sum_{j=1}^{n} \mathrm{norm}(\mathbf{a}_{k_j} \circledast \mathbf{v}_j)) \right\rangle \\
&= \frac{1}{\|\mathbf{r}'\| \cdot \|\mathbf{q}'\|} \sum_{i=1}^{m} \sum_{j=1}^{n} \left\langle \mathrm{norm}(\mathbf{a}_i \circledast \mathbf{b}_i),\ \mathrm{norm}(\mathbf{a}_{k_j} \circledast \mathbf{v}_j) \right\rangle \\
&= \frac{1}{\|\mathbf{r}'\| \cdot \|\mathbf{q}'\|} \sum_{i=1}^{m} \sum_{j=1}^{n} \frac{\left\langle \mathbf{a}_i \circledast \mathbf{b}_i,\ \mathbf{a}_{k_j} \circledast \mathbf{v}_j \right\rangle}{\|\mathbf{a}_i \circledast \mathbf{b}_i\| \cdot \|\mathbf{a}_{k_j} \circledast \mathbf{v}_j\|} \qquad (3)
\end{aligned}$$

where $\mathbf{r}'$ and $\mathbf{q}'$ are the unnormalized bundles of normalized bound pairs for r and q respectively. For simplicity, we will denote them as

$$B_m = \|\mathbf{r}'\| \qquad B_n = \|\mathbf{q}'\|$$

Moreover, since expression (3) contains mn terms, we can consider an arbitrary ordering of these terms and denote each one as

$$\frac{X_i}{D_i}, \quad 1 \le i \le mn$$

where X_i is the dot product of the i-th pair of bound pairs, and D_i is the product of the norms of the bound pairs in that term. With this notation, we can write

$$\langle \mathbf{r}, \mathbf{q} \rangle = \frac{1}{B_m \cdot B_n} \sum_{i=1}^{mn} \frac{X_i}{D_i}. \qquad (4)$$

If row r and predicate q have an equality match, then there are n pairs (from the mn total) that have the form

$$\frac{X_i}{D_i} = \frac{\langle \mathbf{a} \circledast \mathbf{b}, \mathbf{a} \circledast \mathbf{b} \rangle}{\|\mathbf{a} \circledast \mathbf{b}\|^2} = \frac{\|\mathbf{a} \circledast \mathbf{b}\|^2}{\|\mathbf{a} \circledast \mathbf{b}\|^2} = 1$$

thus they are constant.

To obtain the expectation and variances for the terms of non-matching pairs, we have

- From Lemma A.2,

  $$\mathbb{E}[X_i] = 0 \qquad \mathrm{Var}[X_i] \le \frac{2d + 2}{d^2},$$

  where the upper bound comes from the case in which the two bound pairs share a common vector.

- $D_i = C_{i1} \cdot C_{i2}$, for C_{i1}, C_{i2} defined as the norm of a bound pair. From Lemma A.4, we know $C_{i1}, C_{i2} \sim \mathcal{N}(1, \frac{1}{2d})$, so that they highly concentrate around their mean as $d \to \infty$. Therefore, we can use the delta method [23] to obtain a first order approximation for the expectation of D_i is

  $$\mathbb{E}[C_{i1} \cdot C_{i2}] \approx \mathbb{E}[C_{i1}] \cdot \mathbb{E}[C_{i2}] = 1$$

From this, since X_i and D_i are independent are highly concentrated around their mean, we can use the delta method again on X_i/D_i.

$$\begin{aligned}
\mathbb{E}\left[\frac{X_i}{D_i}\right] &\approx \frac{\mathbb{E}[X_i]}{\mathbb{E}[D_i]} = 0 \\
\mathrm{Var}\left[\frac{X_i}{D_i}\right] &\approx \left(\frac{\mathbb{E}[X_i]}{\mathbb{E}[D_i]}\right)^2 \left(\frac{\mathrm{Var}[X_i]}{\mathbb{E}[X_i]^2} + \frac{\mathrm{Var}[D_i]}{\mathbb{E}[D_i]^2}\right) \\
&= \frac{\mathrm{Var}[X_i]}{\mathbb{E}[D_i]^2} + \frac{\mathbb{E}[X_i]^2}{\mathbb{E}[D_i]^4}\mathrm{Var}[D_i] \\
&\le \frac{2d + 2}{d^2}
\end{aligned}$$

Given the mn terms in expression (4), we now have that their sum S satisfies

$$S = \sum_{i=1}^{mn} \frac{X_i}{D_i} = n + \sum_{j=1}^{mn-n} \frac{X_j}{D_j}$$

where the remaining terms represent non-matching pairs. With this, given that variances are $O(\frac{1}{d})$, each term concentrates around its mean, and we have expressions for the expectation and variance of S as

$$\mathbb{E}[S] = n \qquad \mathrm{Var}[S] \le \frac{n(m-1)(2d+2)}{d^2}.$$

For the denominator of expression (4), by Lemma A.5 we have that $B_m \sim \mathcal{N}(\sqrt{m}, \frac{m}{2d})$ and $B_n \sim \mathcal{N}(\sqrt{n}, \frac{n}{2d})$. Then,

$$\begin{aligned}
\mathbb{E}[B_m B_n] &= \mathbb{E}[B_m] \cdot \mathbb{E}[B_n] \\
&= \sqrt{mn} \\
\mathrm{Var}[B_m B_n] &= \mathrm{Var}[B_m]\mathbb{E}[B_n]^2 + \mathrm{Var}[B_n]\mathbb{E}[B_m]^2 \\
&\quad + \mathrm{Var}[B_m]\mathrm{Var}[B_n] \\
&= \frac{mn(4d+1)}{4d^2}
\end{aligned}$$

Finally, we apply the delta method for the desired variable

$$\langle \mathbf{r}, \mathbf{q} \rangle = \frac{S}{B_m \cdot B_n}.$$

We obtain,

$$\begin{aligned}
\mathbb{E}\left[\frac{S}{B_m B_n}\right] &\approx \frac{\mathbb{E}[S]}{\mathbb{E}[B_m B_n]} = \frac{n}{\sqrt{mn}} = \sqrt{\frac{n}{m}} \\
\mathrm{Var}\left[\frac{S}{B_m B_n}\right] &\approx \left(\frac{\mathbb{E}[S]}{\mathbb{E}[B_m B_n]}\right)^2 \left(\frac{\mathrm{Var}[S]}{\mathbb{E}[S]^2} + \frac{\mathrm{Var}[B_m B_n]}{\mathbb{E}[B_m B_n]^2}\right) \\
&\le \left(\frac{n}{\sqrt{nm}}\right)^2 \left(\frac{n(m-1)(2d+2)}{d^2}\frac{1}{n^2} + \frac{mn(4d+1)}{4d^2}\frac{1}{mn}\right) \\
&= \frac{n}{m}\left(\frac{(m-1)(2d+2)}{nd^2} + \frac{(4d+1)}{4d^2}\right) \\
&= \frac{8dm - 8d + 8m + 4dn + n - 8}{4md^2}
\end{aligned}$$

□

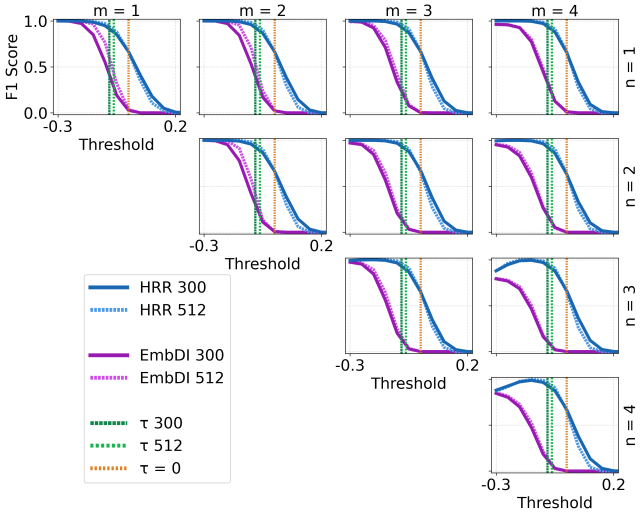


Figure 5: F1-score vs similarity threshold for row retrieval, for all table sizes m and predicate lengths n for the DBLP dataset on non-equality predicates.

Proposition A.7. *For $1 \le n \le m$, let $\mathbf{a}_1, \dots, \mathbf{a}_m, \mathbf{b}_1, \dots, \mathbf{b}_m$ and $\mathbf{v}_1, \dots, \mathbf{v}_n$ be sampled uniformly from the d-dimensional unit sphere S^{d-1}. For row $r = (a_1 : b_1, \dots, a_m : b_m)$ and selection non-equality predicate $q = (a_{k_1} : v_1, \dots, a_{k_n} : v_n)$ for $\{k_1, \dots, k_n\} \subseteq \{1, \dots, m\}$, consider their encoding*

$$\mathbf{r} = norm\left(\sum_{i=1}^{m} norm(\mathbf{a}_i \circledast \mathbf{b}_i)\right), \quad \mathbf{q} = norm\left(\sum_{j=1}^{n} norm(\mathbf{a}_{k_j} \circledast -\mathbf{v}_j)\right),$$

as $d \to \infty$, the expectation and variance of their similarity over the random choice of atomic vectors satisfy

$$\mathbb{E}[\langle \mathbf{r}, \mathbf{q} \rangle | \text{ non-equality match}] = 0$$

$$Var[\langle \mathbf{r}, \mathbf{q} \rangle | \text{ non-equality match}] \le \frac{2d+2}{d^2}.$$

Proof. The sign change in the encoding does not modify the distributions used in the proof of Proposition A.6.

If r and q have a non-equality match, then for every j there is an i such that $a_i = a_{k_j}$ while $b_i \neq v_i$. Since $\mathbf{b}_i, \mathbf{v}_j$ are independent and sampled from Unif(S^{d-1}) and the uniform unit sphere is isotropic (Proposition 3.3.8. in [27]), $\mathbf{b}_i$ and $-\mathbf{v}_j$ are also independent with the same distribution. Therefore, the hypotheses of Lemma A.2 apply. Then, expression 4 consists only of non-matching pairs. Defining

$$S = \sum_{i=1}^{mn} \frac{X_i}{D_i},$$

using the same procedure as in the proof of Proposition A.6, we obtain the following results for the sum of mn non-matching pairs

$$\mathbb{E}[S] = 0 \qquad \mathrm{Var}[S] \le \frac{mn(2d+2)}{d^2}.$$

Finally, we obtain

$$\begin{aligned}
\mathbb{E}\left[\frac{S}{B_m B_n}\right] &\approx \frac{\mathbb{E}[S]}{\mathbb{E}[B_m B_n]} = 0 \\
\mathrm{Var}\left[\frac{S}{B_m B_n}\right] &\approx \left(\frac{\mathbb{E}[S]}{\mathbb{E}[B_m B_n]}\right)^2 \left(\frac{\mathrm{Var}[S]}{\mathbb{E}[S]^2} + \frac{\mathrm{Var}[B_m B_n]}{\mathbb{E}[B_m B_n]^2}\right) \\
&= \frac{\mathrm{Var}[S]}{\mathbb{E}[B_m B_n]^2} + \frac{\mathbb{E}[S]^2}{\mathbb{E}[B_m B_n]^4} \mathrm{Var}[B_m B_n] \\
&\le \frac{2d+2}{d^2}\frac{1}{mn} + \frac{0}{m^2 n^2}\frac{mn(4d+1)}{4d^2} \\
&= \frac{2d+2}{d^2}
\end{aligned}$$

□

Appendix B EXTENDED RESULTS

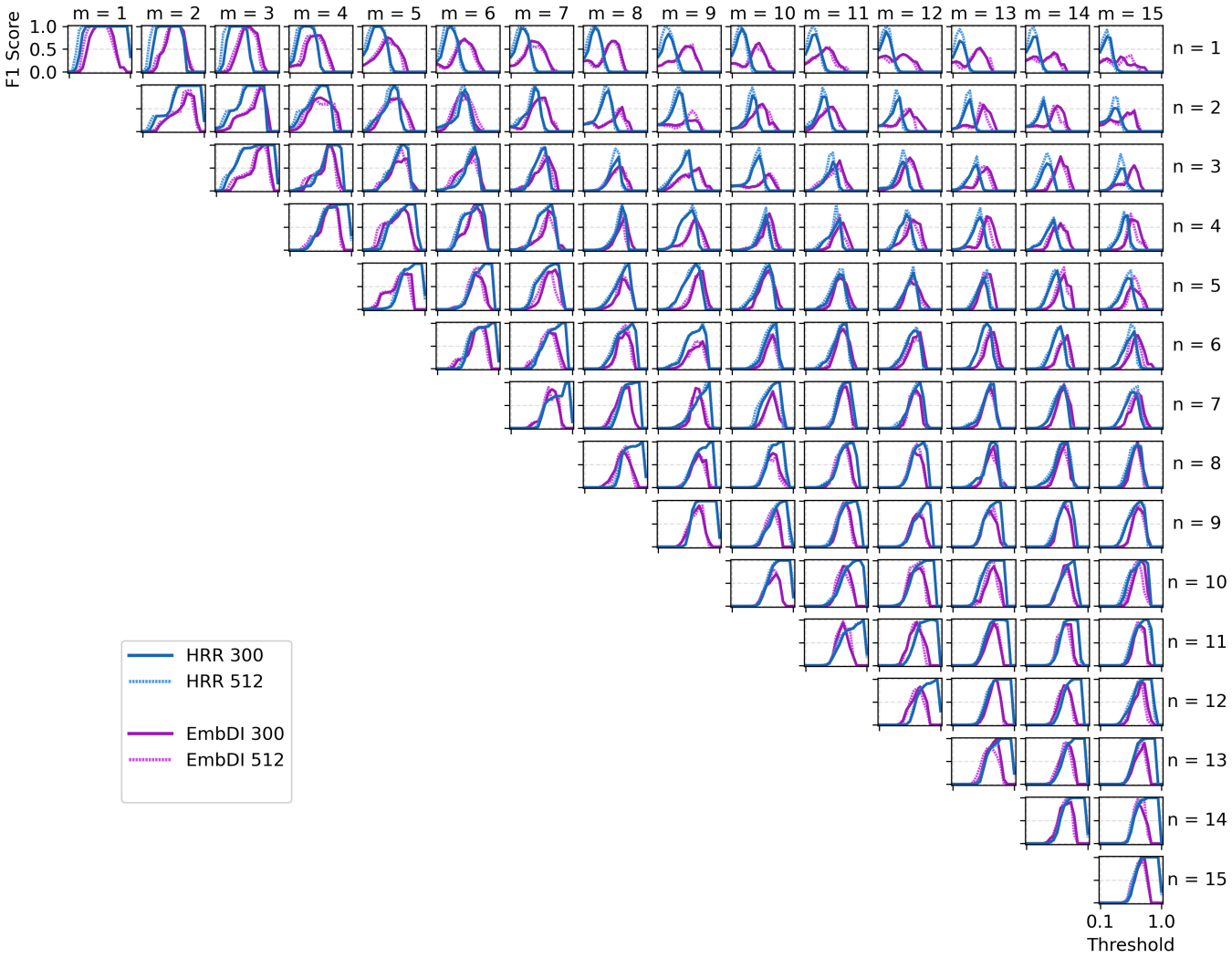


Figure 6: F1-score vs similarity threshold for row retrieval, for all table sizes m and predicate lengths n for the Movie dataset on equality predicates. All subplots share the same axes and scale.

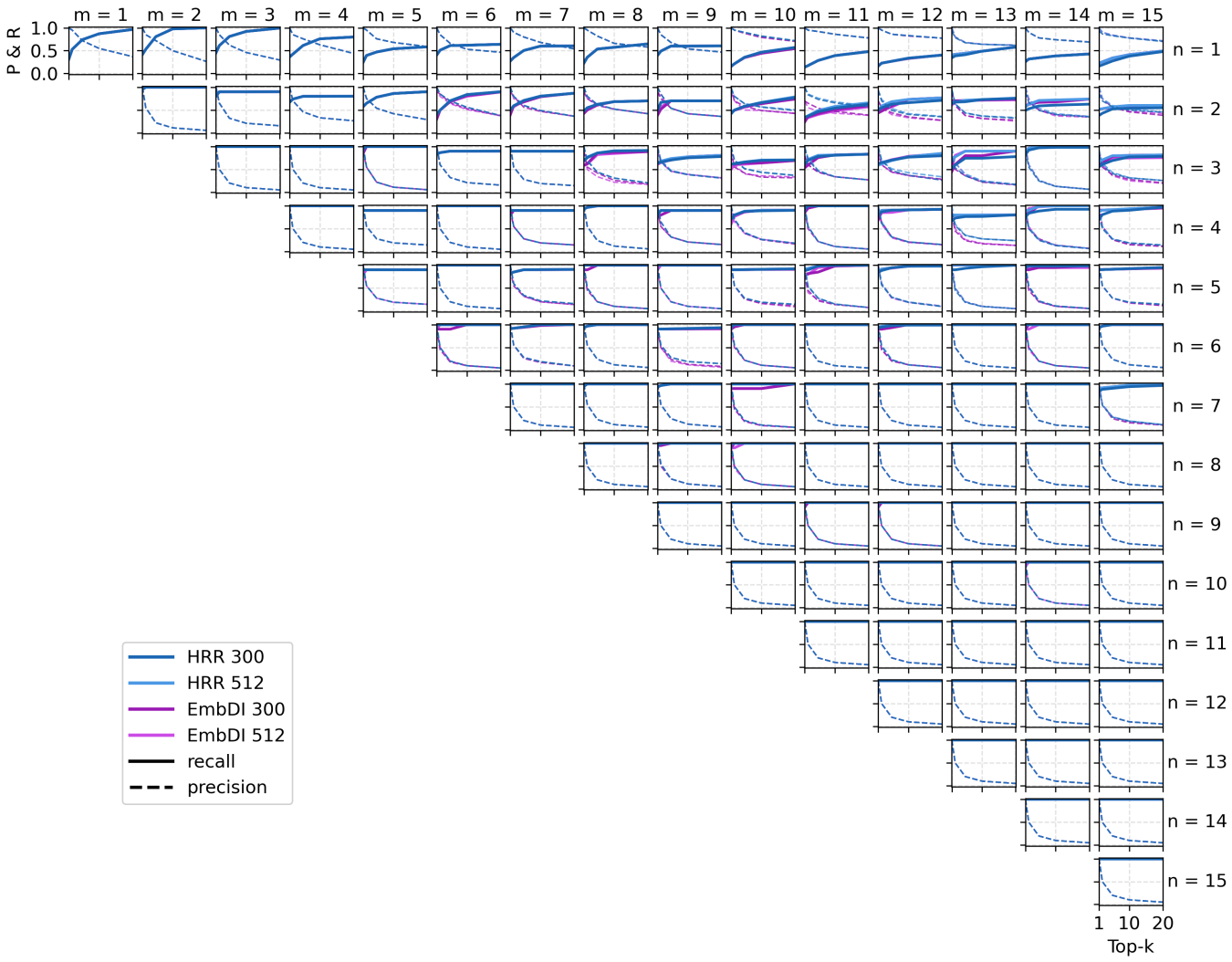


Figure 7: Precision and recall vs top-k for row retrieval, for all table sizes m and predicate lengths n for the Movie dataset on equality predicates. The possibles values are $k \in \{1, 2, 5, 10, 20\}$. All subplots share the same axes and scale.

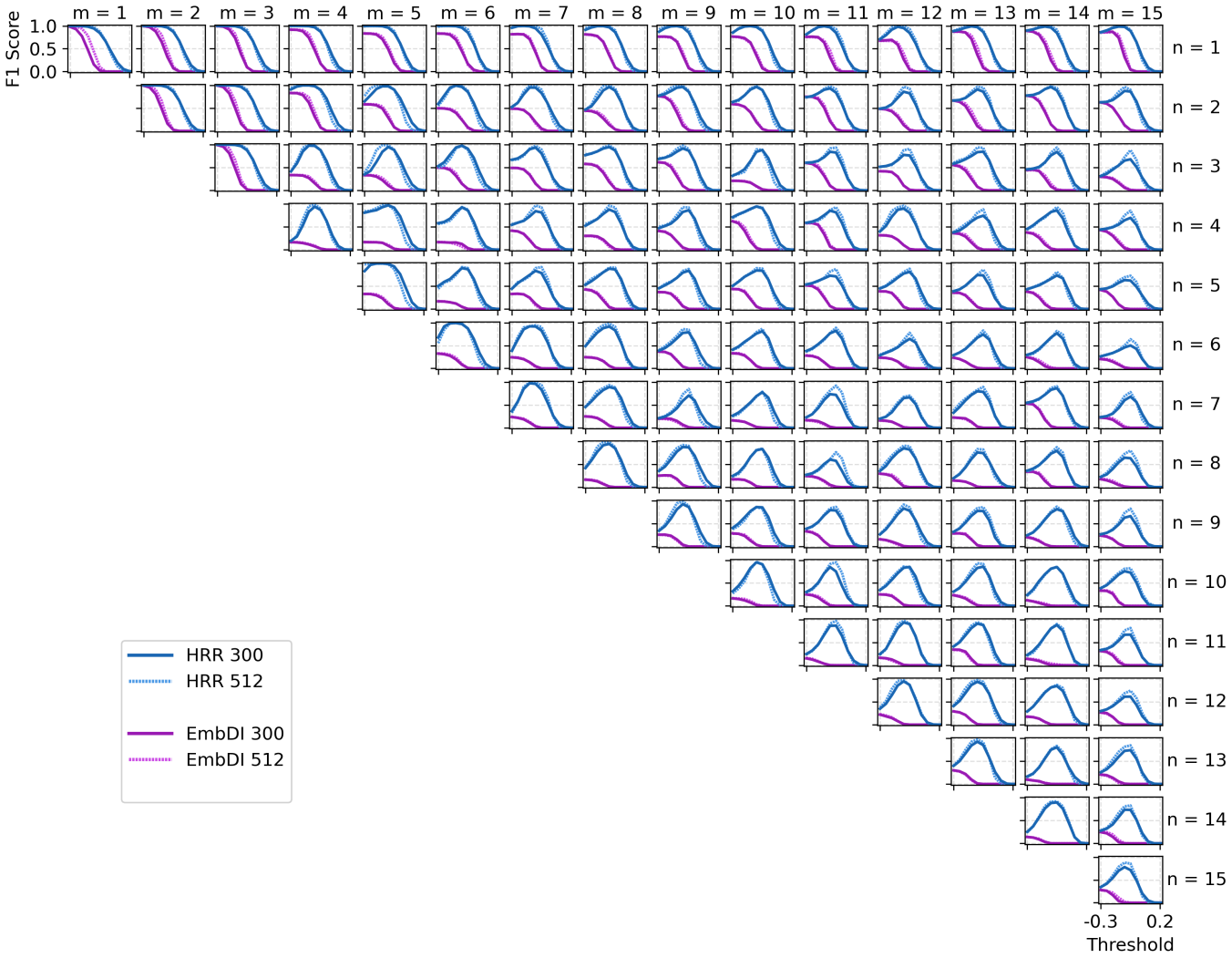


Figure 8: F1-score vs similarity threshold for row retrieval, for all table sizes m and predicate lengths n for the Movie dataset on non-equality predicates. All subplots share the same axes and scale.