TU WIEN Informatics

Enabling Semantic-Aware Query Evaluation in a Traditional Database Framework

MASTERARBEIT

zur Erlangung des akademischen Grades

Master of Science

im Rahmen des Studiums

Data Science

eingereicht von

Nicolas Bschor
Matrikelnummer 12132344

an der Fakultät für Informatik

der Technischen Universität Wien

Betreuung: Univ.Prof. Dr. Emanuel Sallinger
Mitwirkung: Dr. Eleonora Laurenza

Wien, 5. Mai 2025

Nicolas Bschor	Emanuel Sallinger

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

TU WIEN Informatics

Enabling Semantic-Aware Query Evaluation in a Traditional Database Framework

MASTER'S THESIS

submitted in partial fulfillment of the requirements for the degree of

Master of Science

in

Data Science

by

Nicolas Bschor
Registration Number 12132344

to the Faculty of Informatics

at the TU Wien

Advisor: Univ.Prof. Dr. Emanuel Sallinger
Assistance: Dr. Eleonora Laurenza

Vienna, May 5, 2025

Nicolas Bschor | Emanuel Sallinger

Technische Universität Wien
A-1040 Wien ▪ Karlsplatz 13 ▪ Tel. +43-1-58801-0 ▪ www.tuwien.at

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Erklärung zur Verfassung der Arbeit

Nicolas Bschor

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.
Ich erkläre weiters, dass ich mich generativer KI-Tools lediglich als Hilfsmittel bedient habe und in der vorliegenden Arbeit mein gestalterischer Einfluss überwiegt. Im Anhang „Übersicht verwendeter Hilfsmittel“ habe ich alle generativen KI-Tools gelistet, die verwendet wurden, und angegeben, wo und wie sie verwendet wurden. Für Textpassagen, die ohne substantielle Änderungen übernommen wurden, haben ich jeweils die von mir formulierten Eingaben (Prompts) und die verwendete IT- Anwendung mit ihrem Produktnamen und Versionsnummer/Datum angegeben.

Wien, 5. Mai 2025

Nicolas Bschor

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Kurzfassung

Relationale Datenbankverwaltungssysteme (RDBMS) spielen, dank ihrer Fähigkeit Daten effizient zu organisieren, speichern und abzurufen, eine große Rolle in der Datenverarbeitung. Sie beruhen auf strengen Regeln und exakten Übereinstimmungssemantiken. Dies stellt einen Widerspruch zu natürlichen Daten und menschlich generierten Texten dar, da diese oft Unreinheiten, Inkonsistenzen oder semantische Vielfältigkeit aufweisen. Die Anfrage für ein 'Auto' an ein RDBMS würde 'Automobil' nicht finden, obwohl beide Begriffe das selbe beschreiben. Diese semantischen Unstimmigkeiten führen oft zu unvollständigen oder ungenauen Ergebnissen.

Große Sprachmodelle (LLMs) und vortrainierte Transformermodelle (PTMs) haben sich als besonders effektiv in der Verarbeitung verrauschter Daten erwiesen. Daher binden wir LLMs und PTMs direkt in die Evaluation der Abfrage ein, um die Limitierungen traditioneller RDBMS bei verrauschten Daten zu überwinden.

Der Ansatz baut auf dem *Volcano* Modell auf, einem weit verbreiteten Konzept zur Ausführung von Abfragen. Bei *Volcano* werden verschiedene Operatoren zu Abfrageplänen zusammengesetzt. Basierend auf klassischen Operatoren wie `Scan`, `Project (Map)`, `Select`, `Join` und `Aggregate`, entwickeln wir neue Operatoren, die von den strikten Regeln traditioneller RDBMS abweichen und durch die Einbindung von PLMs und LLMs, semantische Schlussfolgerungen ermöglichen. Durch die *Volcano* Schnittstelle können diese neuen Operatoren beliebig in Abfragepläne eingebaut werden. Das erlaubt, zu filtern, zu gruppieren und zu aggregieren basierend auf semantischer statt strikter Gleichheit.

Wir evaluieren unseren Ansatz mit fünf Datensätzen aus verschiedenen Bereichen wie E-Commerce, Musik und Biologie. Dadurch zeigen wir die Generalisierungsfähigkeit und die hohe Effektivität der Operatoren bei Aufgaben wie Entitäten-Abgleich, semantischem Filtern und semantischer Aggregation. Wir analysieren, welchen Einfluss verschiedene Sprachmodelle und Implementierungsstrategien auf die Qualität und den Durchsatz der Ergebnisse haben. Zudem zeigen wir, dass durch die Kombination von PLMs und LLMs in einem Operator, eine höhere Verarbeitungsgeschwindigkeit bei gleichbleibender Ergebnisqualität erzielt werden kann als bei der Nutzung einzelner Strategien.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Abstract

Relational Database Management Systems (RDBMS) play a crucial role in data processing due to their ability to organize, store, and retrieve data efficiently. They rely on strict rules and exact-match semantics which often clash with the noisy, inconsistent, or semantically diverse real-world data, particularly natural language texts. For instance, a traditional RDBMS cannot retrieve a 'automobil' when querying for a 'car' even though both terms describe similar concepts. These semantic mismatches often results in incomplete or inaccurate query results.

Large Language Models (LLMs) and Pretrained Transformer Models (PTMs) have shown high effectiveness on handling the noise occurring in natural language. Therefore, to overcome these limitations of traditional RDBMS, we incorporate LLMs and PTMs directly into the query execution pipeline of a traditional database framework.

Building upon the *Volcano* Model, a widely adopted concept for database query execution where operators are assembled to execution plans, we propose new operators. They are based on traditional operators such as `Scan`, `Project (Map)`, `Select`, `Join` and `Aggregate` and enable semantic inference capabilities by leveraging PLMs and LLMs. Through the *Volcano* interface, they can be integrated at any stage of a relational algebra execution plan, enabling approximate filtering, joining, and grouping based on semantic similarity rather than strict equality.

We evaluate our approach on five real-world datasets from various domains such as music, e-commerce, and biology, demonstrating the operators' generalization capabilities and high effectiveness in tasks such as data integration, semantic filtering, and semantic aggregation. We analyze the influence of different LLMs and implementation strategies on the result quality and the system's throughput. We show that the combination of PTMs and LLMs allows the operators to generate results comparable to those which use PLMs or LLMs alone, while achieving a higher throughput.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Contents

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

CHAPTER 1

Introduction

In today's world, databases play a crucial role in most domains such as healthcare, finance, manufacturing, and artificial intelligence applications [SO24, Bhu25]. Over the last decades, Relational Database Management System (RDBMS) such as PostgreSQL[1] or MySQL[2] were developed to store, manage, and retrieve information efficiently. The underlying query execution framework is optimized for performance, including throughput and scalability [SO24]. Such framework ensures deterministic and explainable results through rigid definitions. This means that they rely on a strict schema and exact-match semantics (e.g. applied in hash-joins [ME92]) for query execution [ZYZD16].

In contrast, real-world data is often heterogeneous or incomplete [LP22]. Particularly unstructured data like natural language text is inherently prone to producing non-standardized content, commonly referred to as noise [ASLS21]. For instance, the term 'car' may be expressed as 'automobile' in English or 'voiture' in French. Consequently, users querying a database for 'car' may fail to retrieve the desired records because traditional databases enforce strict rules to search exclusively for 'car'. This rigidity leads to suboptimal recall and is a substantial limitation of traditional database architectures. Although workarounds such as full-text search (`WHERE name LIKE 'auto%'`) [CRC+19] or query rewriting using synonyms (`WHERE name = 'car'` → `WHERE name = 'car' OR 'automobile'`) [MKK19] exist, they lack flexibility, since they are bound to explicit rules [LM24].

Recent advancements in artificial intelligence, specifically in Pre-Trained Transformer Models (PTMs) and Large Language Models (LLMs), offer solutions to these limitations. Both LLMs and PTMs are trained on large-scale textual corpora to transform and generate texts. PTMs demonstrate advanced capabilities in semantic similarity assessment and

[1] https://www.postgresql.org/
[2] https://www.mysql.com/

contextual representation learning [CLL22], while LLMs show advanced abilities in understanding, generation, and inference over natural language texts [LST24].

Due to these capabilities of LLMs and PTMs, there are a variety of approaches to combine both with a traditional database. These methods extend the RDBMS (e.g. the PostgreSQL extension *pgai*[3]) or operate entirely detached from the database itself (e.g. *Chat-DB* [HFD+23]). In contrast, this thesis aims to directly integrate PTMs and LLMs into a database framework with the objective to relax the rigid constraints typically imposed by traditional RDBMS. This enables advanced semantic filtering, joining, and grouping operations that can handle the naturally occurring noise in real-world data. Resulting in a more intuitive, user-centric interaction with databases, where terminology inconsistencies such as 'car', 'automobile', and 'voiture' do not hinder the quality of the results.

Our proposed solution introduces novel query execution operators that fit into a traditional database framework and directly utilize PTMs for semantic text embeddings and LLMs for inference. We show how established operators have to be altered in order to support such semantic components, while maintaining efficient execution.

1.1 Problem Statement

Traditional database query evaluation relies on strict equality (or inequality) conditions for operations such as **filtering**, **joining**, and **grouping** [SXY+21, ME92, ZJC+23]. However, real-world data often contains incompleteness, inconsistencies, or variations [ASLS21]. This noise can lead to insufficient results. It is particularly problematic in cases where entity names, textual attributes, or categorical values exhibit minor discrepancies due to abbreviations, typographical errors, or semantic differences.

Consider the relational algebra query illustrated in the Figure 1.1. A *Cars* relation lists vehicle models along with their manufacturers and a description, while a *Manufacturer* relation provides names and other non-relevant columns. The query first filters the *Cars* table to retrieve all 'sports cars'. The result of the filtering process is then merged with the *Manufacturer* relation where the car's manufacturer matches the name in the *Manufacturer* relation. The outcome is then grouped by the manufacturer's name and the cars are counted. Hence, the query seeks to count the number of sports cars produced by each manufacturer. Unfortunately, this query faces noise-related challenges:

1. **Filtering**: The selection condition $\sigma_{c.description=\text{'Sports Car'}}$ attempts to retrieve only sports cars from the *Cars* relation. However, textual variations such as 'sport compact car' versus 'sports car' prevent exact matches, leading to potential misclassifications or exclusion of relevant records.

2. **Joining**: The join condition $\bowtie_{\text{p.manufacturer=m.name}}$ needs to associate product manufacturers with the appropriate manufacturer entity. However, discrepancies

[3] https://github.com/timescale/pgai

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

such as 'Toyota Motor' (in *Car*) and 'toyota' (in *Manufacturer*) introduce ambiguity, making traditional joins (e.g. hash-joins) ineffective.

3. **Grouping**: The final aggregation step $\Gamma_{m.name;count(c.name)}$ seeks to count cars per manufacturer. Here, the manufacturer names are not normalized or semantically aligned. For instance, 'toyota' and 'toyota north america' refer to the same company using different textual representations. This leads to fragmented groups, yielding incorrect counts.

These challenges highlight the limitations of traditional databases in handling noisy data. This thesis aims to develop a novel semantic-aware query evaluation system that fits in a traditional database framework. We introduce new operators that accommodate semantic similarity and approximate matching, thereby enabling a more meaningful data retrieval. By incorporating semantic reasoning into database operations such as filtering, joining, and grouping, which traditionally rely on exact constraints, the proposed system aims to improve the completeness of database queries in noisy and semantically ambiguous environments.

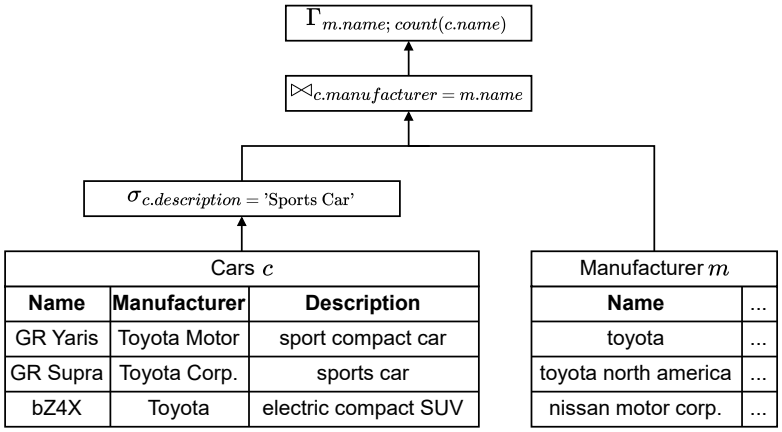

Figure 1.1: Example Query on Noisy Data to Count Sport Cars by all Sport Cars producing Manufacturers

1.2 Research Questions

In this section, we address the research questions discussed throughout this thesis:

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

1. **RQ1**: Which modifications to the traditional database operators `Scan`, `Project (Map)` π, `Select` σ, `Join` $\bowtie$ and `Aggregate` Γ are required to integrate semantic components such as LLMs and PTMs, to enable reasoning over noisy data while maintaining a high throughput?
2. **RQ2**: How do different semantic reasoning strategies (PTM-based, LLM-based, and hybrid approaches) and component selection (different model sizes and hyperparameters) in a database framework influence the precision, recall, and F1 scores when processing noisy data?
3. **RQ3**: How do different semantic reasoning strategies and component choices in a database framework affect the system's throughput for varying data volumes?
4. **RQ4**: What is the relationship between throughput and query accuracy (precision, recall, and F1) in a database framework that utilizes LLMs or PTMs as semantic reasoners?

1.3 Methodology

To address the problem statement in Section 1.1, we begin with a comprehensive literature review of database architectures and state-of-the-art systems for LLM-enhanced reasoning, Information Retrieval (IR), Entity Matching (EM), and clustering. Based on the gathered knowledge, we conceptualize a semantic-aware query evaluation system, offering solutions to the challenges of semantic ambiguity and data inconsistency. According to the developed architecture, we implement the new system and populate it with data, which we use to conduct an extensive evaluation. In the following, we discuss this methodology in detail.

To create such a system, we conceptualize and implement new operators according to the *Volcano* model [Gra94]. The base concepts of the operators `Scan`, `Project (Map)` π, `Select` σ, `Hash-Join` $\bowtie$, and `Hash-Aggregation` Γ [Gra93, KLK+18, ME92, ZJC+23] serve as a foundation for new operators. We integrate PTMs and LLMs directly in the operators' evaluation procedure to enable semantic reasoning over noisy data and create a more user-centered query execution. Furthermore, we design the operators to maintain a reasonable throughput, as performance considerations are crucial for the development of RDBMS [PK01].

We enhance the `Scan` and `Project (Map)` π operator with a standard semantic search functionality using Sentence-BERT embeddings [WA22] to search for relevant relations or columns. The rest of the operator's logic remains unchanged from the traditional procedure. In this way, the user can retrieve desired information without explicitly knowing the entire database schema.

For `Select` σ, we introduce a new criterion, the `Semantic-Equal` $\approx_\tau$, which performs a similarity assessment between two predicates. It first checks if the cosine similarity between the PTM embeddings of both predicates is above a user-defined threshold

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

τ [RG19a]. Then, it validates potential matches by prompting an LLM (zero-shot) [KGR+22], enabling more flexible filtering beyond exact matches.

We create the novel `Similarity-Join` $\bowtie$ based on the `Hash-Join` $\bowtie$ [ME92] procedure. Instead of matching two records using exact equality, the operator matches records using their PTM embedding closeness and an LLM validation. The procedure starts by building a vector index [DGD+24] from the PTM embeddings for the join attributes of the right relation. After indexing all records, the operator iterates over the left relation, embeds the join attributes and performs a range query to find candidate matches. An appended LLM validation confirms true matches using zero-shot strategies.

We alter the `Hash-Aggregation` Γ [ZJC+23] procedure to create the `Semantic-Aggregation` Γ operator. All records are embedded using a PTM and then clustered using algorithms like KMeans [C+21], Density-Based Spatial Clustering of Applications with Noise (DBSCAN) [EKSX96] and Hierarchical Density-Based Spatial Clustering of Applications with Noise (HDBSCAN) [MHA17]. This allows grouping records with semantically similar but textually different keys.

We implement the entire system in the Python[4] language, using PostgreSQL as the underlying database. We utilize the Sentence-BERT model `all-mpnet-base-v2`[5] for embeddings creation and FAISS [DGD+24] for efficient vector search. The evaluation of the system includes three tasks: semantic joins, semantic filtering, and semantic/noise-aware grouping. Each task is evaluated on two out of five different noisy datasets from domains such as music, e-commerce, and biology. We test the system using different strategies for embedding (record-wise vs column-wise) and similarity assessment (threshold-only, LLM-only, or combined). Furthermore, we compare two versions of LLaMA (`Llama-3.2-3B-Instruct`[6] and `Meta-Llama-3-8B-Instruct`[7]) [TLI+23] for semantic validation. To assess the quality of the results we select the metrics precision, recall, F1, and $BLEU_1$ score for semantic joins/ filtering and the Adjusted Rand Score for semantic/ noise-aware grouping. Furthermore, we collect the throughput to determine the system's performance.

1.4 Main Contributions

The main contribution of this work addresses how to relax the rigid exact-match semantics of traditional databases by including LLMs and PTMs into the widely adopted *Volcano* framework. Therefore, we propose strategies for including semantic reasoning approaches directly into operators described by *Volcano*. We show how this methodology enables RDBMS to handle noisy, heterogeneous, and semantically diverse real-world data. Despite the long inference time of LLMs [ZNH+24], we show how to maintain an efficient evaluation

[4] https://www.python.org/
[5] https://huggingface.co/sentence-transformers/all-mpnet-base-v2
[6] https://huggingface.co/meta-llama/Llama-3.2-3B-Instruct
[7] https://huggingface.co/meta-llama/Meta-Llama-3-8B-Instruct

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

through the combination of strategies utilizing PTMs and LLMs. Particularly, our main contributions include:

- **Conceptualization of Semantic-Aware Database Operators**
  As the query evaluation in *Volcano* is carried out by relation algebra operators, we design and implement new operators for `Scan`, `Project (Map)` π, `Select` σ, `Join` $\bowtie$ and `Aggregation` Γ, which utilize PTMs and LLMs to enable semantic reasoning directly within the database.
- **Development of Efficient Semantic Reasoning Strategies in Operators**
  As the inference time of semantic reasoners such as PTMs and LLMs is higher compared to symbolic reasoning strategies of traditional databases, we develop and compare different methods utilizing PTMs, LLMs and combined approaches to reduce inference time while maintaining equally qualitative results. We implement approaches for efficiency improvements using vector-indexes.
- **Extensive Evaluation Across Multiple Tasks and Datasets**
  As such a database does not yet exist, we developed an extensive evaluation framework on common tasks such as data integration, semantic filtering, and semantic grouping using various datasets from different domains. We analyze the trade-offs between accuracy and efficiency depending on the user-defined threshold τ and the selected reasoning strategies.

We provide novel approaches to include semantic components such as LLMs and PTMs directly in the database. We show that the extensible design of *Volcano* allows for novel operators, enabling RDBMS to perform semantically meaningful inference while preserving efficiency. Furthermore, these new operators interact seamlessly with other operators, including symbolic-only ones.

1.5 Overview

In Chapter 2 we provide state-of-the-art methods for LLM-enhanced reasoning, EM, IR and clustering. We also provide the theoretical foundation necessary for building the semantic-aware query evaluation framework. Chapter 3 provides detailed information about the concepts of the new operators, which support PTMs and LLMs for semantic reasoning and similarity assessment. Furthermore, we give insights into the implementation. In Chapter 4, we evaluate the results on tasks such as data integration, semantic filtering and aggregation on real-world datasets, followed by a discussion in Chapter 5. In Chapter 6, we summarize our results and outline directions for future work.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

CHAPTER 2

Theoretical Foundations and Related Work

In this chapter, we establish the theoretical foundations and state-of-the-art methodologies required for the development of a semantic-aware query evaluation system. First, we introduce the *Volcano* (Section 2.1) model, which serves as the underlying framework. Subsequently, we survey recent advancements in LLM-enhanced-reasoning (Section 2.2), EM (Section 2.3), IR (Section 2.4), and clustering techniques (Section 2.5).

In the context of the relational model, the terms *row*, *record*, and *tuple* are often used interchangeably [Cod90]. All three refer to a single entry in a table, representing a collection of related data values. For example, consider the tuple `('GR Yaris', 'Toyota Motor' , 'sport compact car')`. Given a database schema such as `Cars(Name: STRING, Manufacturer: STRING, Description: STRING)`, this tuple can be interpreted as a record by assigning each value to its corresponding attribute, yielding the record `{Name: 'GR Yaris', Manufacturer: 'Toyota Motor', Description: 'sport compact car'}`. For clarity and simplicity, we consistently use the term *record* to refer to such entries throughout this work.

2.1 Volcano Model

The *Volcano* [Gra94] model is a framework for parallel query evaluation in databases, adopted by common RDBMS like PostgreSQL [SBZS17]. In *Volcano*, a query is evaluated by generating an execution plan of relational algebra operators. Every operator creates or receives records from its child operator and passes them to its parent operator or finally the user. This results in a tree-like structure, e.g. Figure 2.1. The operators are implemented as an iterator using a `open-next-close` protocol. The `open()` function initializes the iterator, e.g. loading the relation or initializing a hash table, while `close()`

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

closes and resets all initialized objects. In the `next()` function, the operator performs an instructed operation and returns the record to its parent operator.

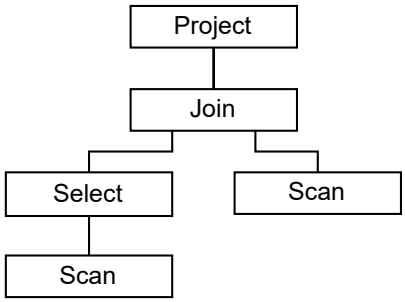

Figure 2.1: Sample *Volcano* Execution Plan

We focus on the five operators `Scan`, `Select` σ, `Project (Map)` π, `Join` $\bowtie$, and `Aggregation` Γ. Due to the extendable nature of *Volcano*, this list is not exhaustive. However, these five operators are essential for query execution and covered by literature [KLK+18]. Note that we will only focus on the functionality of the operator, since implementations vary among methods and systems. Furthermore, for operators located at the leaves of the evaluation plan, we use the term *relation* to refer to the actual data in the database. For subsequent operators, we use the term *relation* to refer to the output generated by its previous operator(s).

- `Scan` R
  The `Scan` operator is located at the leaves of a query evaluation plan and therefore has no child operator. It iterates over a relation R of the database and returns the records in the `next()` function. [Gra93]

- `Select` $\sigma_{condition}(R)$
  The `Select` operator has one child operator and is used to filter data. When the `next()` function is called, it calls its child operator's `next()` function to retrieve a new record until a certain `condition` holds. This record is then returned. [Gra93]

- `Project (Map)` $\pi_{a_1,...a_n}(R)$
  The `Project (Map)` operator, transforms a record by applying mapping functions [KLK+18]. Such functions may rename or remove the record's elements (columns), apply data transformations ($x \leftarrow 2x$), or add new elements (brutto $\leftarrow$ netto + tax) to the record. When `next()` is triggered, it retrieves a record from its child operator, applies the mappings and returns the transformed record.

- `Join` $(L) \bowtie_p (R)$
  There are different types of `Join` operators such as `Semi-Join` $\triangleright$ or `Outer-Join`

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

$\ltimes$ [ME92]. However, due to the scope of this thesis, we focus on the `Inner-Join` $\bowtie$ operator only.

`Inner-Join` $\bowtie$ has two child operators and merges their records if a condition p holds. There are different implementations for the `Join` operator with a varying application. The most basic join operator is `Nested-Loop-Join`. It iterates over both relations in a nested loop. Therefore, it is able to evaluate most join conditions such as range queries (A.x > B.x) or inequality (A.x $\neq$ B.x). Hence, it is highly flexible but very inefficient with a complexity of $\mathcal{O}(n^2)$, due to the nested loop. [ME92]

Higher-performing algorithms such as `Hash-Join` or `Sort-Merge-Join` operate within a lower complexity class, however they have some limitations. Here, we focus on `Hash-Join`, since it provides a foundation for the joining strategy Chapter 3. The `Hash-Join` operator only supports joins on equality and requires a hashable join attribute. Its complexity class, in an ideal scenario with no collisions in the hash table, is $\mathcal{O}(n+m)$, making it highly efficient. First, the operator iterates over the left relation and builds a hash-table on the target attribute. Then, for every element in the right relation, it searches for join partners in the hash table and returns the matches. [Gra93, ME92]

- `Aggregation` $\Gamma_{g_1,...,g_n;f_1(a_1),...,f_m(a_m)}(R)$
  The `Aggregation` operator groups all records obtained from the child operator on n grouping attributes and applies an aggregation function on m other attributes. For instance $\Gamma_{\text{city};count(\text{order\_id})\rightarrow num\_orders,sum(\text{price})\rightarrow sum\_price}(\text{Orders})$ groups all orders from the same cities, counts the 'order_id' and sums all prices together. The keys and aggregated attributes are passed to the parent operator as a new record (e.g. `{city: 'Vienna', num_orders: 15, sum_price: 100€}`).

  As for the `Join` operator, implementations of the `Aggregation` operator separate into hash-based and sort-based approaches [MSL+15] [Gra93]. Again, we focus on the hash-based method. Here, the operator iterates over all records from its child operator and inserts them into a hash table using the grouping attributes $(g_1, ..., g_n)$ as key. Then, it iterates over all keys in the hash table and applies the aggregation functions $f_1, ..., f_m$ on the elements in the hash bucket. The combined record $(g_1, ..., g_n, f_1(...), ..., f_m(...))$ is then returned. The aggregation function f_i may sum all values, find the minimum/ maximum or string-concatenate all values. [ZJC+23]

2.2 LLM-Enhanced-Reasoning

With the emergence of LLMs, several studies have attempted to utilize the semantic understanding of LLMs by including them in the reasoning process [TZL+23]. In the context of logic-based knowledge representation and reasoning, *soft Chase* [BBB+24] leverages LLMs in the classic *Chase* procedure. Regular *Chase* incrementally applies

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

logic-based rules, e.g. from a knowledge graph, to infer new facts until a fixpoint is reached. *Soft Chase* extends this algorithm by identifying and verifying bindings for logic rules from natural language texts using LLMs. Other approaches like *ChatDB* [HFD+23] allow for multi-hop reasoning in classic RDBMS by creating a chain of Structured Query Language (SQL) queries. Thus, *ChatDB* utilizes traditional databases as a symbolic memory for the LLM.

Furthermore, LLMs can be used to enhance the user-to-database interactions. *SQL-PaLM* [SAM+23] offers a framework to create SQL queries from natural language using a LLM. The significance of this task, especially in the presence of noisy data, is demonstrated by the existence of the BIRD benchmark [LHQ+23]. It offers a large collection of text-to-SQL pairs on dirty data. The pairs consist of a natural language text and a working SQL query.

Another field which takes advantage of LLMs is IR which we define in detail in Section 2.4. For IR, documents are ranked according to a user query [WHT+24]. To enhance the results, a task called query expansion can be performed, where new search terms are appended to the original query. Here, LLMs can be employed to generate such new terms [JZQ+23].

The PostgreSQL extension *pgai* allows the user to call LLMs and PTMs directly using SQL. This enables semantic search on texts [BBH16] and provides Retrieval-Augmented Generation (RAG) [LPP+20] functionality directly in the database.

2.3 Entity Matching

The core task of semantic joins and semantic filtering is EM (or Record Linkage). EM determines whether two data representations refer to the same real-world entity [LLS+21]. Such a task may find the same movie based on the descriptions, match company pairs from two different datasets or, in case of the Abt-Buy[1] dataset, match the same product from two different JSON representations. Examples of two matching records are

```
{ "id": 25318, "name": "Terk XM Outdoor Home Antenna - Grey
Finish - XM6", "description": "Terk XM Outdoor ...", "price":
"$80.00" }
```

from Abt and

```
{ "id": 202049393, "name": "Audiovox XM6 Outdoor Home Antenna
- XM-6", "description": null, "manufacturer": "Terk
Technologies", "price": null }
```

from Buy.

[1] https://dbs.uni-leipzig.de/research/projects/benchmark-datasets-for-entity-resolution

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

Since the first statistical models have been developed in 1969 by I. P. Fellegi and A. B. Sunter [FS69], various approaches, utilizing both symbolic and sub-symbolic reasoning strategies, have emerged [IK20, LLS+21].

Some methods such as DITTO [LLS+20] require a training step to fine-tune a PTM using labeled data. However, in a database environment, such training data is not available, because a database request doesn't involve labeling data [HSH+07]. Thus, we focus on methods where an entity match can be inferred directly without any prior training.

Furthermore, considering future implementation in a database context, we identified two different approaches for EM which can be employed: Threshold-Based and Few-/Zero-Shot-Prompting.

2.3.1 Threshold-Based

A deep threshold-based approach is employed by Relation-Aware Entity Matching Using Sentence-BERT: *REMS* [ZHLL22]. Figure 2.2 depicts the *REMS* pipeline for the example Abt-Buy entries. The process begins with serializing the records to a string representation by adding relation information between the attributes. For instance, the Abt record would generate a string such as:

'Terk XM Outdoor Home Antenna - Grey Finish - XM6 [described with] Terk XM Outdoor Home Antenna . . . [costs] $80.00'

The next step in *REMS* is a rule-based blocking phase which filters obvious mismatches to create a set of candidates. These candidate strings are then embedded to fixed-size vectors $\vec{v_a}$ and $\vec{v_b}$ using SBERT-Networks [RG19a] with shared parameters (Siamese SBERT-Networks). The SBERT-Networks internally embed the strings into a list of vectors and apply a mean pooling. Both entities are considered a match if the cosine similarity (Equation 2.1) of the resulting vectors $\vec{v_a}$ and $\vec{v_b}$ exceeds a predefined threshold.

$$cosine\_similarity(\vec{v_a}, \vec{v_b}) = \frac{\vec{v_a} \cdot \vec{v_b}}{\|\vec{v_a}\| \cdot \|\vec{v_b}\|} \tag{2.1}$$

The *REMS* procedure requires labeled training data to fine-tune a pre-trained SBERT model and determine the perfect threshold. However, to overcome the fine-tuning requirement, general-purpose pre-trained SBERT models such as `all-mpnet-base-v2` can be employed. The threshold remains a hyperparameter which must be set by the user.

The Siamese-SBERT architecture is preferred to be employed in a database, because it doesn't require entangling both records. For BERT-based EM frameworks such as DITTO [LLS+20] or EMBA [ZSH24], the transformer's input is structured as '[CLS] Record_a [SEP] Record_b [SEP]'. Hence, to join data, the cross product of all potential candidates has to be constructed. On the contrary, for *REMS* both records are embedded separately. This is beneficial in a database context, as it allows for pre-computation

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

of embedding vectors and therefore, a reduction of runtime by using specialized data structures which are optimized for range queries of vectors [RG19a]. So, when a join is invoked, the computationally expensive embedding task is already completed.

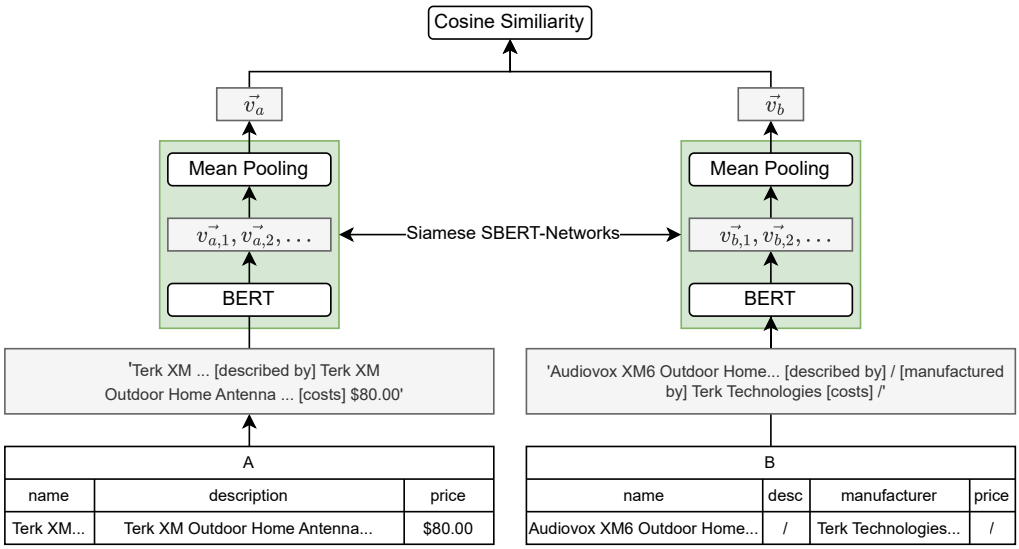

Figure 2.2: *REMS* Pipeline

2.3.2 Few- /Zero-Shot-Prompting

In zero- or few-shot EM, both entities are converted to a string and integrated into a prompt which serves as input for a generative LLM. In a zero-shot-prompting approach, both records are serialized and then inserted into a base prompt. Such base prompt can be general or domain-specific [PSB24]. A general base prompt may look like:

Do the following two records refer to the same real-world entry?
Record A: ...
Record B: ...
Answer with "yes" and "no" only!.

A domain specific prompt contains more context:

Do the following two product entries refer to the same product?
Product A: ...
Product B: ...
Answer with "yes" and "no" only!.

Figure 2.3 demonstrates the zero-shot-procedure for a general prompt.

The few-shot-prompting approach requires some demonstrations, which consist of both positive and negative examples [PSB24]. These demonstrations serve as guidance for the

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

model. Figure 2.4 illustrates the prompt assembly for two demonstrations. The records highlighted in green serve as the positive examples, while the red ones serve as negative examples. The last row represents the test record. These six records, result in the prompt in the gray box. The green sections of the prompt are derived from the positive examples, and the red sections from the negative examples.

The number of demonstrations can vary, however, it requires significantly less training data as for fine-tuning a PTM. Six to ten demonstrations are sufficient to achieve good results. The demonstrations can be obtained through random sampling, searching for related records, or hand picking records with the goal to create a diverse dataset [PSB24].

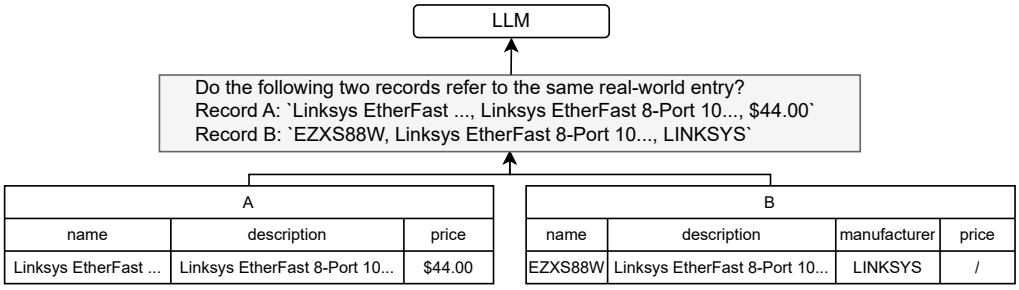

Figure 2.3: Example Zero-Shot-Prompting EM

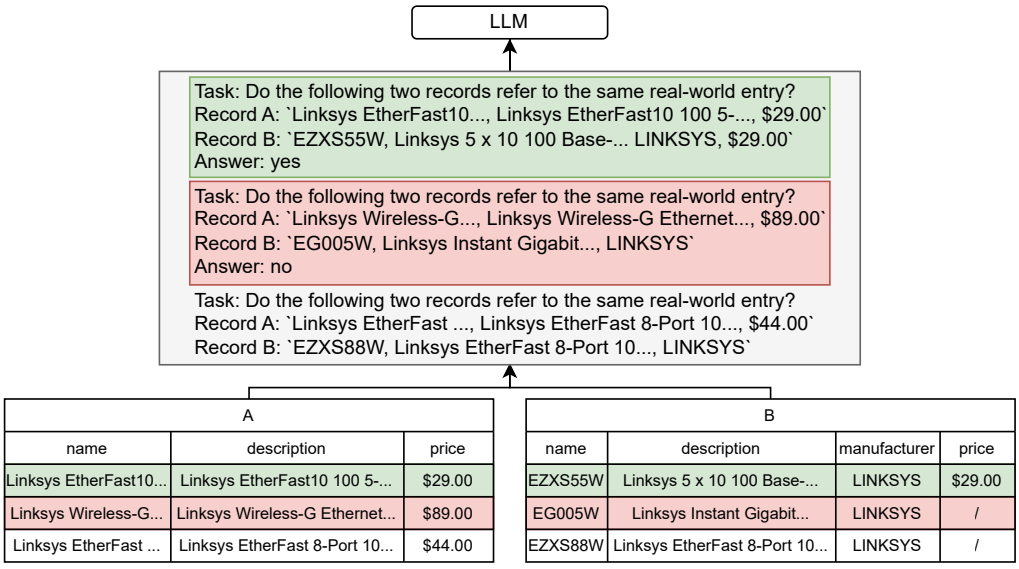

Figure 2.4: Example Few-Shot-Prompting EM

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

2.4 Information Retrieval and Semantic Search

The core task of IR is obtaining relevant information to a user's query from large collections of unstructured data. Therefore, the IR system returns the top-k elements of a sorted ranked list, where all items are assigned with a relevance score. These items, commonly referred to as documents, can have a versatile format. A common example for IR systems are web search engines, where a query is matched to a website [WHT+24].

In the context of a database, a relation's records can be seen as the documents. Therefore, IR tasks are required in two situations. The first one describes the classical IR task of finding certain records based on the unstructured user input ($\sigma_{\text{description}\approx\text{'Car Movie'}}(Movies)$). The second situation involves finding potential join candidates. These records can be directly yielded as the joined records, or be used as potential join pairs in the blocking stage for EM (Section 2.3).

IR is an important and long-existing problem in computer science. It was first mentioned by Calvin N. Mooers in 1951 [Moo51]. Therefore, different approaches have emerged to solve this issue. Traditional approaches such as the vector space model [SWY75] and the probability model [RZ09] take advantage of exact string matches. Unfortunately, these methods are only able to handle ambiguity to a limited extent and do not have a deep semantic understanding. Hence, recent methodologies append *Neural Ranking Models* to traditional approaches, which have shown to outperform symbolic-only methods [WHT+24].

IR systems utilizing *Neural Ranking Models* start by pre-processing the user's query e.g. through query expansion, where related search terms are appended to the query. The altered query is then used to find candidate documents from a set of indexed documents, using unsupervised ranking methods such as BM25 or TF-IDF. Both the processed query and the previously identified documents are passed into a *Neural Ranking Component*, which uses deep neural networks, such as BERT or GPT to generate a relevance score for the document according to the user's query. A list, sorted by the relevance score is then returned by the IR system. The unsupervised ranking step is used to reduce the amount of candidates that have to be passed into the *Neural Ranking Component*, which is computationally quite expensive. [TCDH21]

Implementations of the *Neural Ranking Component* divide into two strategies: *representation-focused models* and *interaction-focused models*. For *representation-focused models*, both the document D and the query Q are embedded using neural networks NN_D and NN_Q, where NN_D is used to create the document embedding $\vec{d}$ and NN_Q for the query embedding $\vec{q}$. Both embeddings are then passed into a scoring function $M(\vec{u}, \vec{v})$ to create a relevance score. If NN_D and NN_Q are siamese networks, e.g. SBERT models, a vector similarity function such as the cosine similarity can be used as scoring function M. [TCDH21]

Interaction-focused models exploit the dependencies of the query and the document. They create an interaction output from a query-document-pair $F(d, q)$. This interaction

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

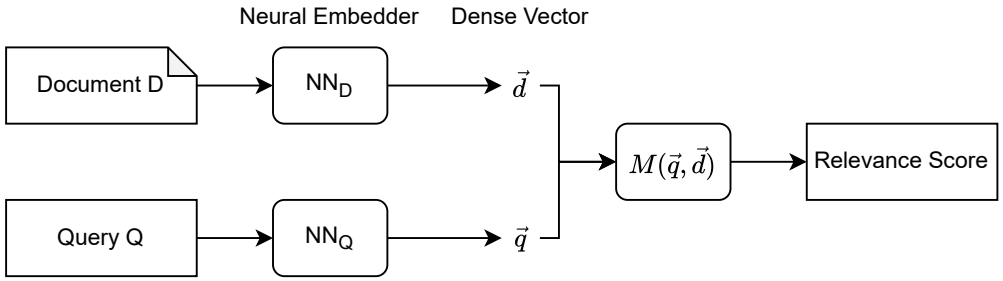

Figure 2.5: Representation Focused Model Architecture

output is then passed into a scoring function M. Such interaction output can be obtained through various techniques, one being a cosine similarity matrix from embeddings of all query and document tokens.[TCDH21]

2.5 Clustering

As a common database task is the aggregation of data based on the equality of their attributes (`Aggregation` Γ), we discuss methods to group data based on the similarity of their attributes. Clustering algorithms are unsupervised machine learning techniques and play a crucial role in data mining [RM05]. Common clustering algorithms include KMeans [C+21], DBSCAN [EKSX96], and HDBSCAN [MHA17].

KMeans [C+21] requires the hyperparameter k, which determines the number of clusters. The algorithm starts by creating k random cluster centroids. Every data point is then assigned to the cluster with the closest centroid. After all data points are assigned, the new centroid is calculated as the mean of the cluster points. This process is repeated until convergence is achieved, i.e., the clusters no longer change or a maximum of iterations is reached.

DBSCAN [EKSX96] detects clusters of varying shapes and sizes by considering the density of points with regard to the distance ϵ and the minimum number of points for a dense region $minPts$. To do so, it starts by selecting an arbitrary data point and retrieves all data points within a distance of ϵ. If the number of data points is greater than $minPts$, the algorithm assigns these points to the same cluster. For all new data points, all unclustered points in range ϵ are fetched and marked. This process is repeated until all points in recursive range are found. Then, the algorithm continues with the next unclustered data point and repeats these steps. If a point does not meet the density requirement of $minPts$, it is marked as noise.

HDBSCAN [MHA17] is designed to reduce DBSCAN's sensitivity to parameter tuning. Therefore, it runs DBSCAN for varying ranges of ϵ, building a cluster hierarchy and combines the results to identify clusters with the highest stability. This approach enables HDBSCAN to detect clusters with varying densities.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

Since a low minimum cluster size requirement $minPts$ usually leads to an over-fragmentation in dense areas, a combination of HDBSCAN and DBSCAN can be utilized. This approach also requires the distance parameter ϵ [MB20].

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

CHAPTER 3

System Design and Architecture

As stated in Chapter 2, there are various methods to combine PTMs and LLMs with a database. Most architectures operate independently of the database itself. For instance, *ChatDB* [HFD+23] modifies the database using `INSERT`, `UPDATE` and `DELETE` operations generated through LLMs [HFD+23]. These decoupled methods work well for generalization among different RDBMS, because SQL is mostly standardized. Furthermore, it can provide the user with a reasoning chain, e.g. a list of `INSERT`.

However, this separation of the semantic reasoner and the RDBMS reduces the efficiency, because data has to be exchanged between the two systems. Including LLMs and PTMs directly in the operator execution reduces this overhead. Furthermore, inclusion enables the possibility for further efficiency improvements by creating index structures over semantic embeddings directly in the database.

To embed LLMs and PTMs directly in the database framework, we create semantic-aware counterparts of the *Volcano* operators `Scan`, `Project (map)` π, `Select` σ, `Inner-Join` $\bowtie$, `Aggregation` Γ. In Section 3.1, we conceptualize the operators and examine their functionality. Section 3.2 provides detailed information about the employed implementation strategies.

3.1 System Design

In this section, we present the conceptual design of semantic-aware database operators. We illustrate how the functionality of traditional *Volcano* operators is altered to create new operators that incorporate semantic reasoning by integrating LLMs and PTMs. This enables semantic, noise-tolerant query evaluation, while preserving compatibility with other operators through adherence to the `open-next-close` interface.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

3.1.1 Scan

As discussed in Chapter 2, the `Scan` operator returns the records of a database relation. Consequently, it does not inherently benefit from incorporating LLMs or PTMs during the iteration over the relation. However, the users may not always have the full overview over all relations, hence some guidance is beneficial. For instance, the relation may be called 'Automobiles' while the user wants to retrieve all 'Cars'. Therefore, to enhance the usability of the database system, we support semantic search functionality for identifying the target relation.

To find the desired relation, the operator generates embeddings for all available relations and ranks them based on cosine similarity to the embedding of the user-provided relation name. The names of the top-ranked relations are then validated by an LLM according to the user input. The first relation that passes this validation is selected for scanning.

Algorithm 3.1 depicts the `open-next-close` procedure for the `Scan` operator. Both `next()` and `close()` remain unchanged from the original *Volcano* implementation [Gra93]. The `open` function implements the default semantic search procedure using cosine similarity assessment with SBERT embeddings [WA22] to search for the desired database relation. The operator first embeds the relation name defined by the user. Then, it embeds all available relations and calculates the cosine similarity. After sorting the relations according to similarity, it iterates over the list, beginning with the most similar relation. If the LLM validation passes, the operator opens the corresponding relation.

3.1.2 Project (Map) π

The `Project (Map)` π operator performs a mapping of records. Some common use cases are the reduction to columns of interest, renaming columns, or merging columns using techniques such as summation ($c \leftarrow a + b$). Here, the user has to define the desired columns in the mapping functions. As for `Select`, we enhance this process through a semantic search of the columns.

3.1.3 Select σ

The `Select` σ operator requires a child operator and a criterion p. Algorithm 3.2 describes the general functionality as described by *Volcano* [Gra93]. In the `open()` and `close()` functions, it executes the respective functions from its input operator. The `next()` function iterates over the operator's child operator until the current record passes the criteria or the input is exhausted. The inclusion of a semantic component (LLM/ PTM) is implemented within the criteria.

The classic criterion is a conjunction or disjunction of predicates. These predicates usually follow an `attribute-operator-value` structure, where the `operator` describes a mathematical operator such as $=$, $>$, $<$, ...and `attribute/value` are literals or columns. For instance, the relational algebra expression in Equation 3.1 searches for

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

Algorithm 3.1: Scan

```
Require: PTM, LLM, DB, name
Function Open():
    nameEmbedding ← PTM.embed(name)
    similarityScores ← []
    for i ∈ 1, 2, ..., |DB.relations| do
        R ← DB.relations_i
        relationEmbedding ← PTM.embed(R.name)
        sim ← cosine_similarity(nameEmbedding, relationEmbedding)
        similarityScores_i ← (sim , R)
    end
    similarityScores ← sortBySimilarity(similarityScores)
    for similarity, R ∈ similarityScores do
        if LLM.validate(R.name, name) then
            Scan.R ← R
            Scan.R.open()
            return
        end
    end
Function Next():
    return Scan.R.next()
Function Close():
    Scan.R.close()
```

products with a rating of 5.0 and a price lower than 50 or lower than average price within its category.

$$\sigma_{\text{rating=5.0}\wedge(\text{price<50}\vee\text{price<category\_average\_price})}(\text{Products}) \tag{3.1}$$

To extend this operator with semantic understanding, we introduce the novel criteria `Semantic-Equal` $x \approx_\tau y$, described by Algorithm 3.3. The input x and y are defined as single values or records. If the inputs are records or non-string values, they have to be serialized into strings. Then, both inputs are embedded into two vectors using a PTM. If the cosine similarity is below the threshold τ, the algorithm considers both records to be unequal. Then a zero-shot validation step is applied that utilizes a LLM to determine if the records are semantically equal. The result of this validation is returned in the final step. Through the consecutive steps, this method avoids an expensive validation through an LLM if the PTM similarity assessment already fails.

To include criteria if two elements are semantically unequal, the `Semantic-Equal` criteria can be negated $\neg(x \approx_\tau y)$. For semantic range queries ($<$ or $>$), the usage of PTM embeddings can't be employed, because the cosine similarity can only measure

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

the angular similarity. Therefore, only the zero-shot validation can be used, where the operator prompts the LLM if one element is greater (or less) than the other.

Algorithm 3.2: Select

Require: Operator Op, criteria p

```
1  Function Open():
2  |  Op.open()
3  Function Next():
4  |  t ← Op.next()
5  |  while ¬p(t) do
6  |  |  t ← Op.next()
7  |  end
8  |  return t
9  Function Close():
10 |  Op.close()
```

Algorithm 3.3: Semantic-Equal

Require: PTM, LLM

Input: input left x, input right y, threshold τ

Output: $x \approx_\tau y$

```
1  x ← serialize(x)
2  y ← serialize(y)
3  x ← PTM.embed(x)
4  y ← PTM.embed(y)
5  if cosineSimilarity(x, y) < τ then
6  |  return False
7  end
8  if ¬ LLM.validate(x, y) then
9  |  return False
10 end
11 return True
```

3.1.4 Similar-Join ⋈

As covered in section 2.1, there are many different types and implementations of joins. The simplest implementation of joins is a nested-loop join, defined as $\sigma_p(R \times L)$ [NLK17]. Therefore, a naive `Similar-Join` can be implemented by building the cartesian product followed by a `Select`, utilizing the previously defined `Semantic-Equal` on the result: $\sigma_{x \approx_\tau y}(L \times R)$. However, the high computational complexity associated with the cartesian product ($\mathcal{O}(|L| \times |R|)$) should be avoided to save computational resources and increase the throughput.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

To tackle the efficiency issue, we propose an alternative approach, the `Similar-Join` $\bowtie_{\approx_\tau}$. This operator joins all records from the left relation with all records from the right relation where the embeddings cosine similarities are above a threshold τ and they pass a validation through a LLM. It derives from `Hash-Join` and takes advantage of the recent research in efficient similarity search through index structures [DGD+24], by replacing the exact matches of the hashed join attributes (point search) with a range query on index structures filled with the embeddings of a PTM.

To execute the `Similar-Join` $\bowtie_{\approx_\tau}$, the operator inserts the entire right relation into a vector index and for every element in the left relation, a range query with threshold τ is performed to determine all candidates. The operator then validates all candidates, leveraging zero-shot strategies using an LLM. If the validation succeeds, the merged record is then returned.

Figure 3.1 illustrates the process in which all elements from relation R are already embedded as dense vectors and inserted into the vector index (circles in the vector index). The figure depicts the state in which a range query is performed using the embedding of the first record in the relation L. The red circle portrays the range query for the threshold τ with the embedding as the circle center. This results in three records (green, orange and blue circle) from the right relation as potential join candidates. As one can observe, the embeddings of the records 'Toyota Motor in Toyota, Japan' and 'nissan motor corp. in yokohama, kanagawa, japan' are close, however they do not refer to the same company. Therefore, a validation step is applied to filter these records.

R

Name	Address
toyota	torrance, california, united states
hp	palo alto
nissan motor corp.	yokohama, kanagawa, japan
hewlett-packard	palo alto, CA, US
toyota north america	plano, texas, united states
ford motor company	dearborn, michigan, US

L

Name	Address
Toyota Motor	Toyota, Japan
HP	Palo Alto- California

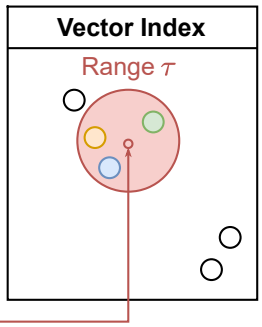

Figure 3.1: Demonstration of the `Similar-Join` $\bowtie_{\approx_\tau}$

Algorithm 3.4 demonstrates the integration of this operator into the *Volcano* framework. The `open()` function iterates over the entire right relation and extracts the join values from the records. The key values are then serialized. A possible serialization for the

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

record *{"Name": "Toyota Motor", "Address": "Toyota, Japan", "industry": "Automotive"}* with the key 'Name' & 'Address' is *'Name: Toyota Motor, Address: Toyota, Japan'*. Then, the operator embeds the serialized key and inserts the embedding into a vector index together with the corresponding record.

The `next()` function searches for the next join pair, as long as the left relation is not exhausted. To achieve this, the operator iterates over the left relation and performs the same serialization and embedding steps as for the right relation. Then, it performs a range query on the vector index using the threshold τ. The result includes all potential join candidates from the right relation, which are stored by the operator for a later *next()* call. The operator then starts with the second validation phase by fetching the next potential join candidate. If the candidate passes a validation through a LLM, the operator merges both records and returns it to its parent operator. When no candidate pair passes the validation or the candidate list is exhausted, the operator fetches a next record from the left relation and repeats all previous steps.

For a later `next()` call, the operator first checks if the list of potential join candidates is exhausted. If not, the operator continues by fetching the next right-relation record from the candidate list, otherwise it continues with the next record from the left relation.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Algorithm 3.4: Similar-Join

Require: LLM, PTM, serialization, vectorIndex
Input: Left Relation L, Right Relation R, join_attributes j_r, j_l, threshold τ

```
1  Function Open():
2      R.open()
3      vectorIndex.init()
4      while r ← R.next() do
5          rightKeyValues ← r.get(j_r)
6          rightSerialized ← serialization(rightKeyValues)
7          rightEmbed ← PTM.embed(rightSerialized)
8          vectorIndex.insert(rightEmbed, r)
9      end
10     R.close()
11     L.open()
12     Join.currentLeftRecord ← null
13     Join.joinCandidates ← []
14 Function Next():
15     while true do
16         if ¬ Join.currentLeftRecord then
17             Join.currentLeftRecord ← L.next()
18             if ¬Join.currentLeftRecord then
19                 return null
20             end
21             lKeyValues ← l.get(j_l)
22             lSerialized ← serialization(lKeyValues)
23             emb ← PTM.embed(lSerialized)
24             Join.joinCandidates ← vectorIndex.rangeQuery(emb, τ)
25         end
26         while currentRightRecord ← Join.joinCandidates.next() do
27             if LLM.validate(Join.currentLeftRecord, currentRightRecord)
                then
28                 return Join.currentLeftRecord + currentRightRecord
29             end
30         end
31         Join.currentLeftRecord ← null
32         Join.joinCandidates ← []
33     end
34 Function Close():
35     L.close()
```

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

3.1.5 Semantic-Aggregation Γ

To create a `Semantic-Aggregation` $\Gamma_{\approx_\tau}$ operator, the same issues have to be solved as for the `Similar-Join` operator. Both approaches, sort-based and hash-based, produce undesired results for semantically equal but lexically unequal data. If unequal key attributes are aggregated in a hash table or sorted, the aggregation on semantical equality will fail, because of the lexical differences. For instance, the terms 'hp' and 'hewlett-packard' describe the same company, however, they would appear in different positions on a sorted list or in different buckets of the hash-table.

However, as for `Similar-Join`, this problem can be solved by replacing the point search with an approximate search for the embeddings of PTMs. Unfortunately, there are no reference embeddings to apply a range query, as in `Similar-Join`. Hence, the `Semantic-Aggregation` $\Gamma_{\approx_\tau}$ operator solves this issue by clustering the embeddings.

To perform the semantic aggregation, the operator embeds all key attributes of the relation's records and performs a clustering step. When this finishes, the operator iterates over the clusters and applies the aggregation functions (`SUM`, `COUNT`, ...) on all the records in the cluster.

To illustrate this method, we use the same relation as in Section 3.1.4. In this example, the serialized key ('Name' + 'Address') is embedded to a dense vector (Figure 3.2). This vector is used as input for a clustering algorithm. Here, we use K-Means, however, other clustering algorithms like DBSCAN or HDBSCAN are suitable. For a $K = 2$, the algorithm finds two clusters representing car brands and IT companies. By tuning the hyperparameter k, or selecting another clustering algorithm, the results differ. Clustering the embeddings using DBSCAN results in Figure 3.3, where the clusters relate to actual companies.

In the traditional `Hash-Aggregation`, all elements for the exact same key are collected, hence one unique key exists and the values can be set as columns in the record. In case of the `Semantic-Aggregation` $\Gamma_{\approx_\tau}$, this is not possible anymore, because multiple values for a key are possible. Instead of grouping the exact same values, the operator groups similar ones. For instance, in Figure 3.3, the data is grouped by 'name' and 'address' as key. A sample cluster contains the two elements: `{'name': 'hp', 'address': 'palo alto'}` and `{'name': 'hewlett-packard', 'address': 'palo alto, CA, US'}`. Hence, two different key values exist for the same group: 'hp, palo alto' and 'hewlett-packard, palo alto, CA, US'. So, if the operator needs to yield a key value, it requires a dedicated aggregation function. Such aggregation function may sample one specific value, return the comma-separated values as a string, or utilize an LLM to create a summary from all elements.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

Name	Address
toyota	torrance, california, united states
hp	palo alto
nissan motor corp.	yokohama, kanagawa, japan
hewlett-packard	palo alto, CA, US
toyota north america	plano, texas, united states
ford motor company	dearborn, michigan, US

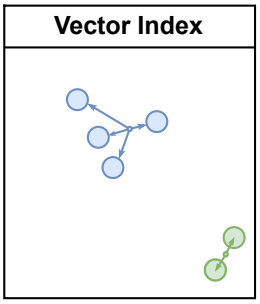

Figure 3.2: `Semantic-Aggregation` $\Gamma_{\approx_\tau}$ using K-Means

Name	Address
toyota	torrance, california, united states
hp	palo alto
nissan motor corp.	yokohama, kanagawa, japan
hewlett-packard	palo alto, CA, US
toyota north america	plano, texas, united states
ford motor company	dearborn, michigan, US

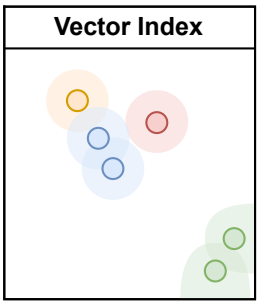

Figure 3.3: `Semantic-Aggregation` $\Gamma_{\approx_\tau}$ using DBSCAN

We demonstrate how the `Semantic-Aggregation` $\Gamma_{\approx_\tau}$ aligns with the *Volcano* model in Algorithm 3.5. In the `open()` function, the operator iterates over the entire input relation. For each record, the operator serializes the key values. The serialized keys are then embedded using a PTM. After the entire relation is processed, the operator performs a clustering process with the embeddings as input. When the clustering algorithm terminates, the operator retrieves all groups (clusters).

The `next()` function iterates over all clusters and applies the aggregation functions for all elements within the group. Then, the operator merges the results of the functions to a new record and passes them to its parent operator.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Algorithm 3.5: Semantic-Aggregation

Require: PTM, serialization, clustering
Input: Relation R, Group By Attributes g, Aggregation Functions f

Function `Open()`**:**
  $R.open()$
  $c \leftarrow clustering.init()$
  while $r \leftarrow R.next()$ **do**
    $groupByKeys \leftarrow r.get(g)$
    $serializedGroupByKeys \leftarrow serialization(groupByKeys)$
    $emb \leftarrow PTM.embed(serializedGroupByKeys)$
    $c.insert(emb, r)$
  end
  $R.close()$
  $Aggregation.clusters \leftarrow c.getCluster()$
Function `Next()`**:**
  $cluster \leftarrow Aggregation.clusters.next()$
  $result \leftarrow \{\}$
  for $f_i \in f$ **do**
    $result_{f_i.name} \leftarrow f_i(cluster.elements)$
  end
  return $result$
Function `Close()`**:**
  $Aggregation.clusters.close()$

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

3.2 Implementation

We implemented the system using the Python language. The PostgreSQL database serves as the base. All operators are implemented as Iterator Classes[1]. To fit the *Volcano* model, all operators implement the `open()` and `close()` functions. The `next()` function is implemented in the built-in `__next__()` function of the Python Iterator.

To create embeddings, we utilize the `all-mpnet-base-v2` pre-trained Sentence Transformers from the SBERT project[2] [RG19b, RG20]. To store and query the embeddings, we employ the FAISS library [DGD+24]. For text generation and semantic validation we use `Llama-3.2-3B-Instruct` and `Meta-Llama-3-8B-Instruct` [TLI+23].

In the following sections, we examine the specific implementation details for `Select` σ (Section 3.2.1), `Similar-Join` $\bowtie$ (Section 3.2.2), and `Semantic-Aggregation` $\Gamma_{\approx_\tau}$ (Section 3.2.3). We provide information regarding the prompts used for zero-shot validation using LLMs, as well as the employed embedding and similarity assessment strategies.

3.2.1 Select σ

As we expect the inference time of LLMs to be higher than the creation of embeddings using a PTM, the operator first checks whether the cosine similarity of the embeddings is above the threshold τ. If it succeeds, it continues with the LLM validation. Thus, the computationally expensive LLM operation is not executed in every case.

We select a serialization function that removes the key attributes from the records and concatenates the values using comma-separation to create more natural embeddings and inputs for the zero-shot validation. A query $\sigma_{name,address \approx \text{'Car Brand'}}$ would serialize the input for the record `{"Name": "toyota", "Address": "torrance, california, united states"}` to *'toyota, torrance, california, united states'*.

For the zero-shot input, we chose *'You are a validator. Respond with "no" and "yes" only!'* as the system prompt and *'Does "<left>" describes "<right>"?'* as the template for both inputs. Some previous tests showed, that only system prompting yields reliable responses that exclusively use 'yes' and 'no' for both LLaMA models. So, the input for the LLM, using the previous example, is *'Does "toyota, torrance, california, united states" describes "Car Brand"?'*.

3.2.2 Similar-Join $\bowtie$

For the implementation of the Similar-Join $\bowtie_{\approx_\tau}$, we focus on two aspects: the embedding methods and how the operator performs the zero-shot entity matching.

[1] https://docs.python.org/3/glossary.html#term-iterator
[2] https://sbert.net/

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

We have implemented three different methods for this operator. The first one only utilizes the embeddings, the second method creates the cartesian product and utilizes only LLMs and the third version employs the described combined approach.

Embedding Methods

To compare the embeddings, we have implemented two different methods: record- and column-wise. For a record-wise comparison, the operator compares one embedding for the entire record. This embedding can be obtained through serializing and embedding the complete record or mean pooling the embeddings for all join values. We chose the same serialization strategy as for `Semantic-Equal` in Section 3.2.1, where the record's elements are converted to strings merged using comma separation. So, the record `{'name': 'hp', 'address': 'palo alto'}` can be embedded as $emb('\text{hp, palo alto}')$ or $MEAN(emb('\text{hp}'), emb('\text{palo alto}'))$, both resulting in an equal sized vector. This comparison method pools the record and therefore only allows to compare $R \bowtie_{(r_1,r_2,\ldots r_x)\approx(l_1,l_2,\ldots l_y)} L$. An advantage of this method is that it doesn't require to compare the exact same amount of columns.

However, queries like $L \bowtie_{(l_1\approx r_1)\wedge(l_2\approx r_2)\vee(l_3\approx r_3)} R$ require a column-level comparison. Therefore, we also included column-wise comparison. In this method, the embeddings of the columns are directly compared and if the average cosine similarity $MEAN(cos(r_1, r_1), cos(r_2, r_2), \ldots)$ is below the threshold, both records are considered as a match. To avoid the overlapping of attributes and to keep the vector indexes as small as possible, the operator creates a separate vector index for all join attributes. So, in the `open()` function, all key columns are embedded separately and inserted in their respective vector index. The entire procedure to build the vector index is illustrated in Algorithm 3.6. In the `next()` function, to find possible candidates for a fixed left record from the left relation, the operator performs a range query on the vector indexes using the column embeddings of the record. If any column embedding of a right-record t is located within the range τ, the record is collected in a set. Then, the operator uses the average cosine similarity across all column embeddings for the join assessment. We depict the procedure in Algorithm 3.7.

Algorithm 3.6: Column-Wise Embedding Comparison for Vector Indexes Creation

Require: PTM
Input: Record r, Vector Indexes v, Join Attributes j_l, j_r
for $i \in \{1, 2, \ldots |JoinAttributes|\}$ **do**
  $embeddings_i \leftarrow PTM.embed(r.get(j_{r,i}))$
  $v_i.add(embeddings_i, r)$
end

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

Algorithm 3.7: Column-Wise Embedding Comparison to retrieve Join Candidates

Require: PTM
Input: Record l, Vector Indexes v, Join Attributes j_l, j_r

$embeddings \leftarrow []$
$candidates \leftarrow \{\}$
for $i \in \{1, 2, \ldots |JoinAttributes|\}$ **do**
  $embeddings_i \leftarrow PTM.embed(l.get(j_{l,i}))$
  $candidates.put(v_i.rangeQuery(embeddings_i, \tau))$
end
for $r \in candidates$ **do**
  $cosine\_similarities \leftarrow []$
  for $i \in \{1, 2, \ldots\}$ **do**
    $cosine\_similarities_i \leftarrow$
     $cosine\_similarity(embeddigs_i, PTM.embed(r.j_{r,i}))$
  end
  if $average(cosine\_similarities) < \tau$ **then**
    $candidates.remove(r)$
  end
end

Zero-Shot Entity Matching

As stated for zero-shot EM in Section 2.3.2, the operator instructs an LLM to decide whether two records are semantically equal. Hence, the operator must create a prompt that is passed to a LLM. The LLM's answer must be a boolean value that the operator can use for further assessment. Therefore, the response must be understood by a symbolic reasoner. We utilize the same system prompt *'You are an object-matcher. Check if the two tuples A and B refer to the same real world entity. If so, answer with "yes", if not, answer with "no" only!'*

To generate the actual prompt, the operator fills the template: *'A is <a>\nB is <b>'*.
A sample prompt for the records
`{"Name": "toyota", "Address": "torrance, california, united states", "industry": "automotive"}`
and
`{"Name": "Toyota Motor", "Address": "Toyota, Japan", "industry": "Automotive"}`
would be:
A is toyota, torrance, california, united states, automotive
B is Toyota Motor, Toyota, Japan, Automotive

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

3.2.3 Semantic Aggregation Γ

For the implementation of the `Semantic-Aggregation` $\Gamma_{\approx_\tau}$ operator, we focus on the feature creation and the clustering algorithms.

Features Creation

As for the `Similar-Join` $\bowtie_{\approx_\tau}$, we have implemented two methods to create input features for the clustering algorithm: record- and column-wise embeddings. The record-wise embeddings create input features by pooling the entire record, either through embedding the serialized record, or averaging all the column embeddings. The creation of column-wise embeddings differs, because there are no reference records to compare it with. To create column-wise embeddings, the operator concatenates the embeddings for all key column values $([emb(g_1), emb(g_2), \ldots, emb(g_n)])$ to a single vector of shape $sizeEmbedding \cdot numberColumns$.

Both embedding methods produce vectors of high dimensionality. The `all-mpnet-base-v2` SBERT model embeds text to a vector of 768 dimensions. Thus, the input for the clustering algorithm is at least the size of a single embedding. Usually, search and cluster algorithms, which work well for low dimensions, but become less effective when the dimensionality increases. This is due to the fact that the distances in high dimensions become almost equal, a phenomenon known as 'The curse of dimensionality' [KE11]. Therefore, we added an optional dimensionality reduction step utilizing the UMAP library[3] [MHM20].

Clustering

To group the data, we employ the clustering implementations provided by the *scikit-learn* library[4] [PVG+11]. The library supports numerous clustering algorithms, with a unified interface. Hence, with minor tweaks, all of those clustering algorithms can be included in the noise-aware query evaluation framework. For comparison, we selected the three methods KMeans, DBSCAN and HDBSCAN, because KMeans and DBSCAN are two of the most popular clustering algorithms [CZL+24], while HDBSCAN has demonstrated strong performance in text clustering, particularly in combination with UMAP [AM21].

[3] https://umap-learn.readthedocs.io/en/latest/
[4] https://scikit-learn.org/stable/

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

CHAPTER 4

Evaluation

To evaluate the system, we created different scenarios, in which such a semantic-aware system can be employed: **data integration** (Section 4.1), **semantic filtering** (Section 4.2) and **semantic/ noise-aware grouping** (Section 4.3). For every task, we describe the datasets, the employed metrics, and present the results.

For the data integration task, we join two noisy datasets. Hence, this scenario evaluates how effectively the `Similar-Join` $\bowtie_{\approx_\tau}$ operator can find matching records. The semantic filtering task involves filtering noisy records based on semantics. We use this task to evaluate the `Semantic-Equal` $\approx_\tau$ of the `Select` σ operator. Lastly, we evaluate the `Semantic-Aggregation` $\Gamma_{\approx_\tau}$ operator by solving the semantic/ noise-aware grouping task. In this task, the system has to group noisy data to consolidate all duplicates or records belonging to the same category.

Since databases are usually applied in various domains, we evaluate the generalization of the system using two different datasets for every task. We create an execution plan for each dataset that solves the respective task, which we execute in the system. We measure the quality of the results based on the ground truth provided by the dataset. Furthermore, we measure the runtime and number of processed records to determine the throughput (Equation 4.1). For comparison, we vary parameters such as the models used, thresholds, embedding-, serialization- and clustering-methods.

$$\text{Throughput} = \frac{|\text{Input Records}|}{\text{Execution Time}} \tag{4.1}$$

All experiments and evaluations were conducted on Google Colab[1], utilizing Python 3.11.11 as the primary development environment. The hardware configuration included $53GB$ of RAM and a L4 GPU with $22.5GB$ of VRAM, providing hardware acceleration for computational tasks.

[1] https://colab.research.google.com/

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

4.1 Data Integration

This task is about matching the entries of two different relations with different representations. We select the *iTunes-Amazon* and *Abt-Buy* Deep-Matching Datasets [MLR+18]. Both datasets contain two separate relations and a matching table. We compute how close the result of the `Similar-Join` $\bowtie_{\approx_\tau}$ operator is to the matching table.

The *iTunes-Amazon* dataset contains songs collected from the *iTunes* and *Amazon* stores. The full dataset contains 6908 songs from *iTunes* and 55922 songs from *Amazon*. A labeled dataset contains 539 candidates. We reduced the input to the 132 true matches resulting in 128 *iTunes* and 132 *Amazon* songs. This step was necessary because the original dataset contains matching songs, that are not declared as matches. Hence, `Similar-Join` $\bowtie_{\approx_\tau}$ would correctly yield these matches, however, since they are not present in the matching table, they are counted as false positives.

A sample record from the *iTunes* table is
`{'sno': 6105, 'album_name': 'born to die - the paradise edition', 'artist_name': 'lana del rey', 'released': '13-Nov-12', 'song_name': 'yayo' time: '5:21'}`.
The correct match from the *Amazon* dataset is the record
`{'sno': 7984, 'album_name': 'born to die - the paradise edition [explicit]', 'name': 'lana del rey', 'released': 'November 13, 2012', 'song_name': 'yayo', 'time': '3:57'}`.
As semantically relevant columns, we select 'album_name', 'artist_name', 'released', 'song_name' and 'time'. Consequently, we define the data integration execution plan as:

$$(\text{iTunes } l) \bowtie_{\text{album\_name,artist\_name,released,song\_name,time}} (\text{Amazon } r)$$

For the *Abt-Buy* dataset, the task is to match products obtained from the online retailers *Abt.com* and *Buy.com*. The dataset contains 1081 (*Abt*) and 1092 (*Buy*) entries with 1097 matches, which we reduced to a 100×100 subset with 100 matches. This reduction is necessary due to computational issues, which we discuss in Chapter 5.

Two matching sample records are
`{'idAbt': 38477, 'name': 'Linksys EtherFast 8-Port ...', 'description': 'Linksys EtherFast 8-Port ...', 'price': '$44.00'}`
and
`{'idBuy': 10011646, 'name': 'Linksys EtherFast EZXS88W Ethernet Switch ...', 'description': 'Linksys EtherFast 8-Port...', 'manufacturer': 'LINKSYS', 'price': null}`.
So, we join both relations on 'name', 'description' and 'price'. Leading to the execution plan:

$$(\text{Abt } l) \bowtie_{\text{name,description,price}} (\text{Buy } r)$$

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

For comparison, we employ seven different strategies:

1. `Zero-Shot`: Considering the LLM answer only
2. `Thresh (Col)`: Considering the cosine similarities of PTM embeddings only, where each join column pair is compared separately
3. `Thresh (Rec, Full)`: Considering the cosine similarities of PTM embeddings only, where the record is pooled by serializing the entire record to a string
4. `Thresh (Rec, Field)`: Considering the cosine similarities of PTM embeddings only, where the record is pooled by averaging the embeddings of all fields
5. `Combi (Col)`: Considering both the cosine similarities of PTM embeddings and the LLM answer, by blocking potential pairs generated using `Thresh (Col)` and subsequently filtering using `Zero-Shot`
6. `Combi (Rec, Full)`: Considering both the cosine similarities of PTM embeddings and the LLM answer, by blocking potential pairs generated using `Thresh (Rec, Full)` and subsequently filtering using `Zero-Shot`
7. `Combi (Rec, Field)`: Considering both the cosine similarities of PTM embeddings and the LLM answer, by blocking potential pairs generated using `Thresh (Rec, Field)` and subsequently filtering using `Zero-Shot`

We investigate how the choice of LLMs and threshold values influences the effectiveness and efficiency of the data integration task. Therefore, we test the strategies using the two different LLaMA-3 (3B & 8B) LLMs and thresholds ranging between $\tau = 0.1$ and $\tau = 0.9$. A $\tau = 0$ would yield the cartesian product, while a $\tau = 1$ would perform an inefficient `Hash-Join`.

4.1.1 Metrics

The query result is converted to a result set R, which we compare with the ground truth set G provided by the matching table. We can determine the true positives $TP = R \cap G$, the false positives $FPs = R \setminus G$ and the false negatives $FNs = G \setminus R$. Using these sets, we can calculate the following metrics:

1. Recall $\frac{|TP|}{|TP|+|FN|}$: How many of the expected matches are present in the result set?
2. Precision $\frac{|TP|}{|TP|+|FP|}$: How many of the resulting records are actual matches?
3. F1 score $\frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision}+\text{Recall}}$: Harmonic mean of precision and recall

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Furthermore, we calculate the $BLEU_1$ score. It assesses how well the words and phrases in the evaluated text match the reference text. In our scenario, the $BLEU_1$ score measures how well the integrated dataset represents the ground truth dataset. We calculate the $BLEU_1$ scores by tokenizing both the serialized ground truth records G and the serialized result records R. Afterward, we compute the $BLEU_1$ score between each entity in the G and each entity in R and collect the maximum for every $g \in G$. The overall $BLEU_1$ score is defined as the average over G (Equation 4.2) [PBGF22].

$$BLEU_1(G, R) = \frac{1}{|G|} \sum_{g \in G} \max_{r \in R} BLEU_1(g, r) \tag{4.2}$$

Note that the $BLEU_1$ metric is coupled to the recall. If the recall is 1, there is always an identical record present in the result set, resulting in the existence of $BLEU_1(x, x) = 1$ for any element in the ground truth. So, if the recall is 1, then $BLEU_1(G, R) = 1$.

4.1.2 Results

First, we investigate which methods work best for the *iTunes-Amazon* dataset. For all thresholds, we compare the results with the highest F1 scores for LLaMA3.3 3B (Figure 4.1) and LLaMA3 8B (Figure 4.2). Since the threshold-based approaches are not influenced by the choice of the LLM, they resulted in identical scores for both models. Figure 4.1 shows that for the smaller LLaMA model, threshold-based approaches perform better (in terms of F1) compared to zero-shot-based approaches. Furthermore, they have a much higher throughput and therefore a significantly lower inference time. On our machine, the `Zero-Shot` approach takes $774s$, while the Thresh (Col) takes $2.6s$. The best performing combination approaches are able to achieve the same values for F1 and $BLEU_1$ as `Zero-Shot` while also being faster (e.g. `Combi (Col)` takes $12s$).

The larger LLaMA model (Figure 4.2) outperforms the smaller model in terms of F1 and $BLEU_1$. It achieves a $BLEU_1$ of 0.98 for `Zero-Shot` and `Combi (Rec, Field)`. All methods utilizing LLMs outperform `Thresh (Rec, Full)` and `Thresh (Rec, Field)` in both F1 and $BLEU_1$. `Combi (Rec, Full)` as the combination of `Zero-Shot` and `Thresh (Rec, Full)`, even surpasses the performance of both methods it is composed of. However, one can observe that F1 performance of `Thresh (Col)` exceeds all other methods. So, in this scenario, the threshold-based approach is both faster and has a higher F1 score compared to the LLM based approaches, which makes it preferable in this case.

It is also worth noting that the optimal threshold differs for all serialization and embedding strategies within one class of methods. The optimal threshold for `Thresh (Rec, Field)` is $\tau = 0.3$ and $\tau = 0.8$ for `Thresh (Col)`. It also differs between the same serialization and embedding strategies across the two classes of methods. For `Thresh (Rec, Full)` the optimal τ is 0.9 while it is 0.8 for `Combi (Rec, Full)`. Furthermore, both Figure 4.1 and 4.2 show that the record-wise comparison with field

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

serialization has a much lower optimal threshold compared to the other strategies for both threshold-only and the combined-approach. This indicates that the mean pooling dilutes the embeddings, which also results in lower $BLEU_1$ and F1 scores.

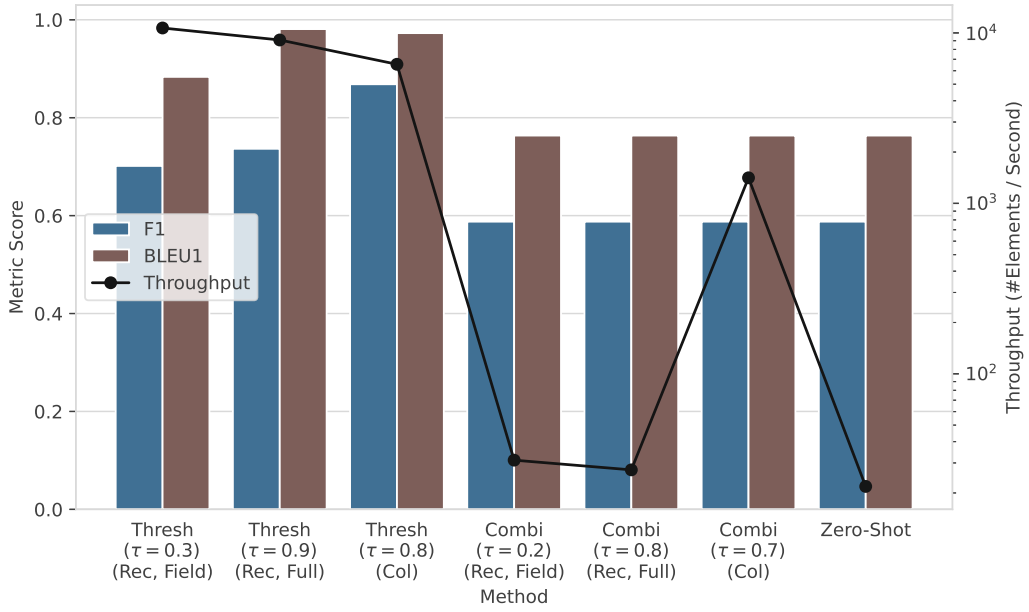

Figure 4.1: Overall Evaluation Results *iTunesAmazon* Dataset using LLaMA 3B

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

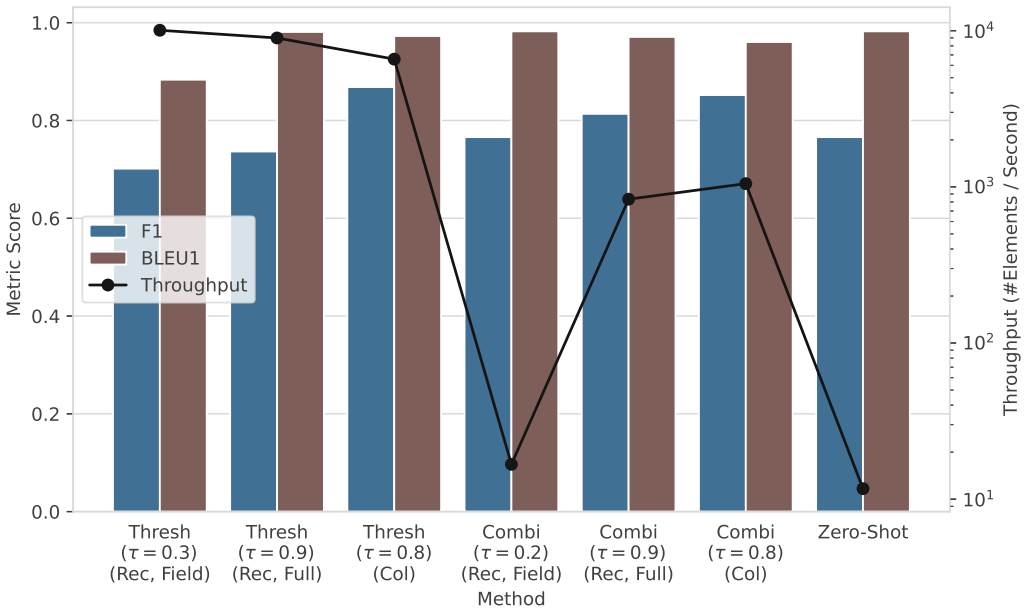

Figure 4.2: Overall Evaluation Results *iTunesAmazon* Dataset using LLaMA 8B

As `Thresh (Col)` achieves the best performance on F1, it is worth investigating the relationship between the threshold and the precision, recall and F1 scores. Figure 4.3 depicts the metric results and the number of true positives, false positives and false negatives for another test run where we employ the `Thresh (Col)` method using 100 evenly distributed values for the threshold τ between 0.5 and 1.0. One can observe that for $\tau \in [0.1, 0.74]$, the recall is 1.0, meaning that the number of true positives is at its maximum. However, for these thresholds the system yields a high number of false positives, resulting in low precision and, therefore, a low F1 score. For $\tau > 0.92$ the false positive rate drops to 0.0, which results in a precision of 1.0. This comes at the cost of the number of true positives which steadily declines for higher τ falling to 0 for $\tau > 0.96$. Hence, the selection of τ strongly influences the outcome of the execution plan. A lower threshold increases the number of records in the result set and therefore increases the recall score. A threshold of 0.0 will always yield the cartesian product and force a recall of 1.0 and a precision of 0.0. A higher threshold will optimize precision. For $\tau = 1$ only exact string matches will occur in the result set (the same result as a `Hash-Join`). Because no equal records exist in this dataset, the operator cannot infer any matches, so the precision drops to 0.0.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

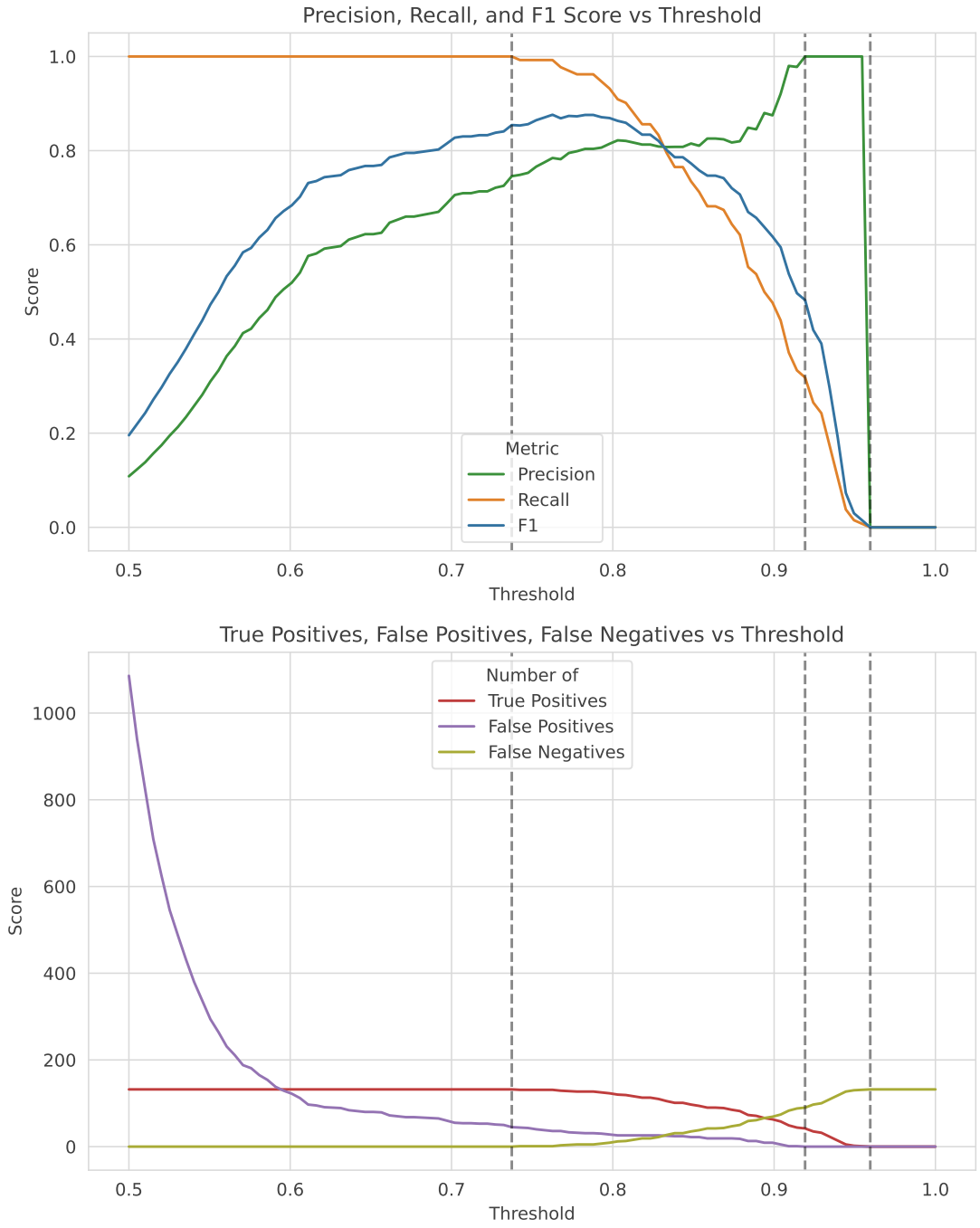

Figure 4.3: Relationship between Precision, Recall, F1 and the Threshold Hyperparameter for `Thresh (Col)` on the *iTunes-Amazon* dataset

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Another important aspect is the relationship between zero-shot-, threshold-based and combination approaches with regard to the threshold τ. Since `Combi (Rec, Full)` achieved a higher F1 than `Zero-Shot` and `Thresh (Rec, Full)`, we investigate the causes of this phenomenon in Figure 4.4. The bars depict the F1 scores for `Combi (Rec, Full)` with $\tau = 0.8, 0.9, 1.0$ and `Zero-Shot` ($\tau = 0$). The orange and green line depict the recall and precision of `Thresh (Rec, Full)` for the respective threshold. One can observe, for a recall of 1.0, the combination approach yields the exact same F1 score as `Zero-Shot`. This is because when the recall of `Thresh (Rec, Full)` is 1.0, the ground truth G is a subset of the result set R ($G \subseteq R$). Since the combination approach filters the outcome of the threshold approach, which contains all records in G in this case, the result set of the combined approach contains the same true positives as the `Zero-Shot` approach, resulting in an equal recall value. The F1 score may vary if the threshold-based approach filters records which are false positives in `Zero-Shot`. In Figure 4.4 for $\tau = 0.8$, the threshold-based approach does not filter out more false positives than `Zero-Shot`, therefore the scores are identical. Nonetheless, the threshold-based approach serves as a blocking stage, which removes clear mismatches. As indicated with the red line, the amount of elements that are yielded by the threshold-based approach is reduced for higher τ. This reduces the number of comparisons that have to be computed by the LLM in the combined approach, resulting in a slightly better runtime compared to `Zero-Shot`. For $\tau = 0.9$, `Thresh (Rec, Full)`, the operator filters some of the true positives, therefore the recall is below 1. However, it also reduces false positives which are inferred by `Zero-Shot`. This results in an increase of the F1 score and a much higher throughput.

The results for the AbtBuy datasets differ for LLaMA 3B (Figure 4.5) and LLaMA 8B (Figure 4.6). As for the *iTunes-Amazon* dataset, a threshold-based approach performs best compared to the results of LLaMA 3B. In contrast, the LLaMA 8B model outperforms all threshold-based approaches with a maximum F1 of 0.93 and a $BLEU_1$ of 0.95. While `Combi (Rec, Field)` and `Combi (Col, Field)` require a threshold of $\tau = 0.2$ to achieve the same performance of `Zero-Shot`, `Combi (Rec, Full)` requires $\tau = 0.7$, this applies for both models. As the relationship between the threshold-based and the combined approach in Figure 4.7 shows, higher thresholds lead to fewer input for the LLM in the combined approach. So, a $\tau = 0.7$ significantly reduces LLM calls and, therefore a much higher throughput.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

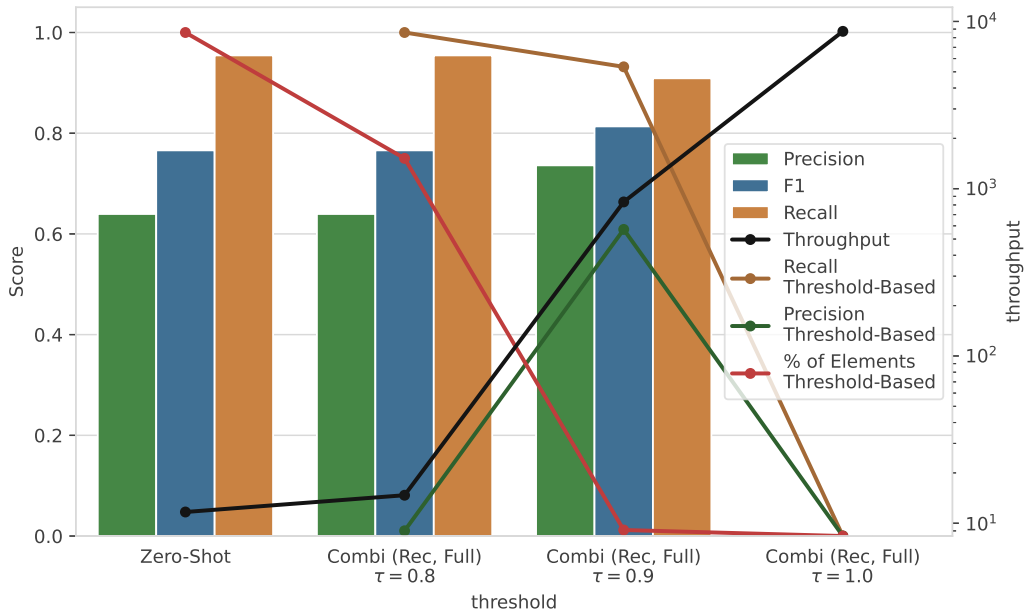

Figure 4.4: Relationship of the threshold-based `Thresh (Rec, Full)` & zero-shot-approach (`Zero-Shot`) and the combination `Combi (Rec, Full)` for *iTunes-Amazon* Dataset using LLaMA 8B

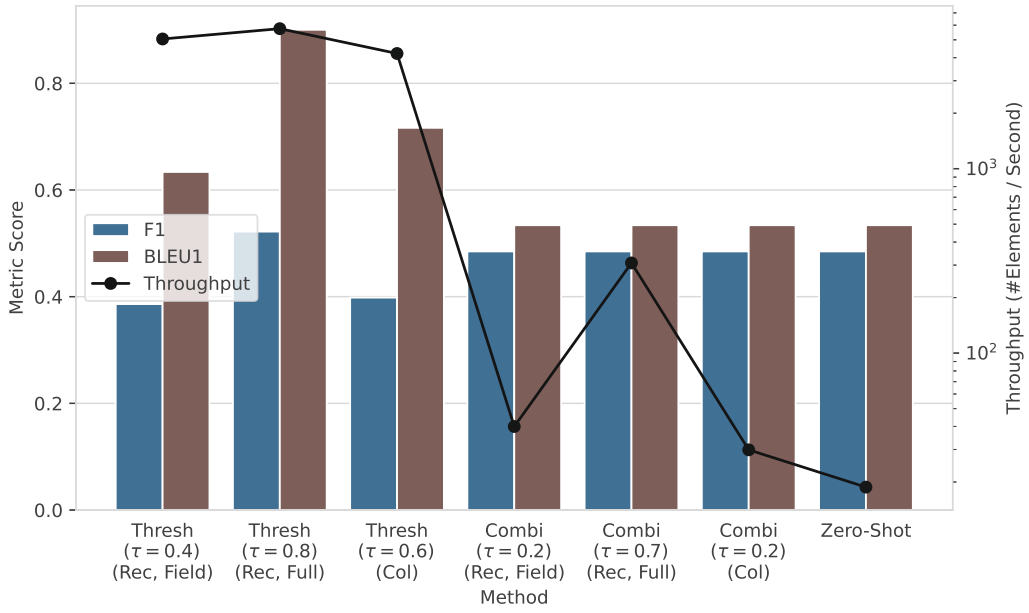

Figure 4.5: Overall Evaluation Results *AbtBuy* Dataset for LLaMA 3B

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

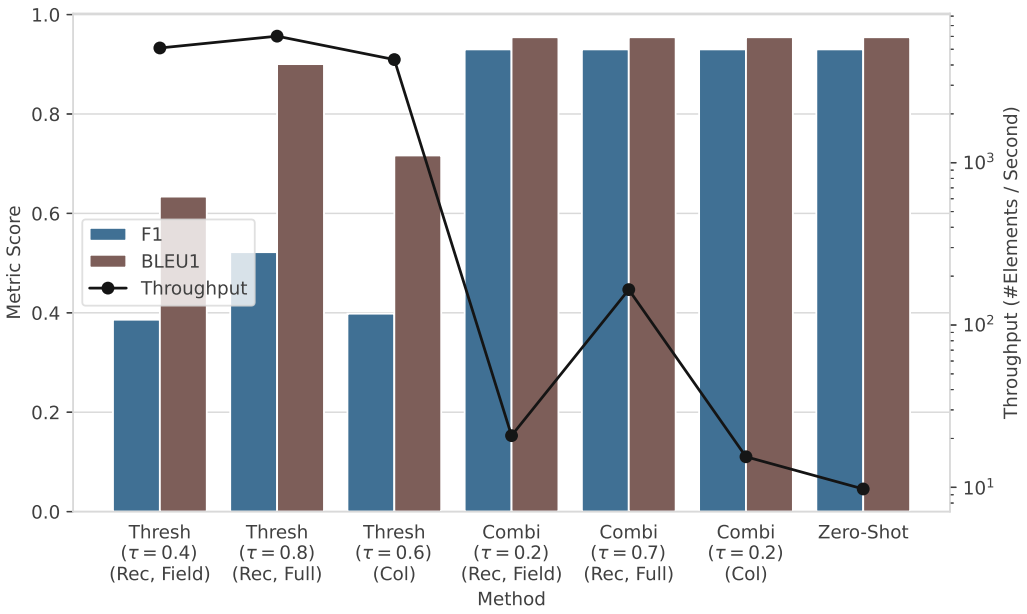

Figure 4.6: Overall Evaluation Results *AbtBuy* Dataset for LLaMA 8B

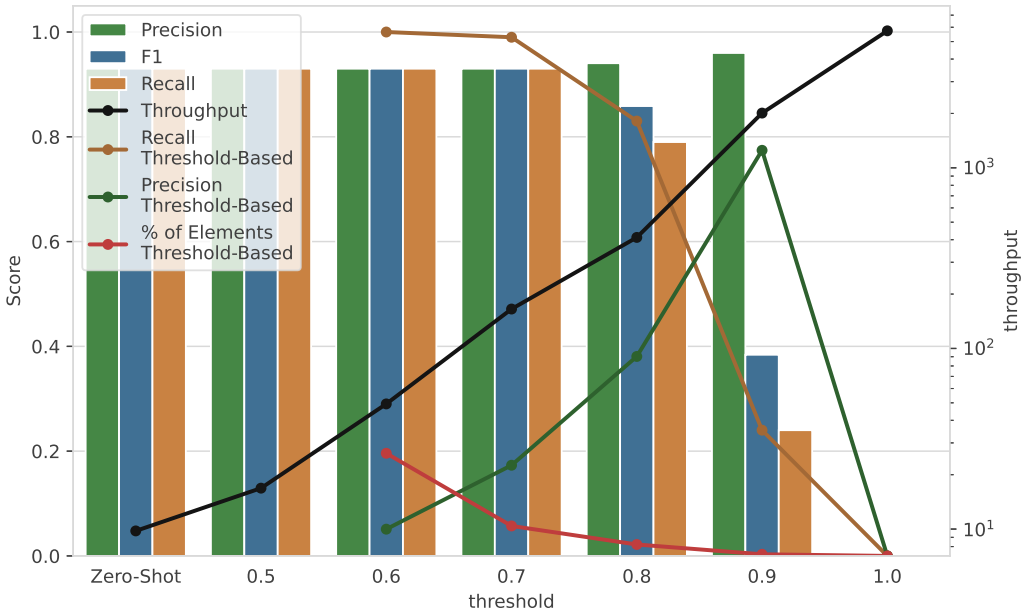

Figure 4.7: Results of the Combination Approach for *AbtBuy* Dataset using LLaMA 8B in Comparison with the Recall of the Threshold-Only Approach and the Throughput

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

4.2 Semantic Filtering

In this task, we test how efficiently and effectively the system can find records based on a user's search query. We use the *Product Classification and Clustering* [Akr20] and the *Zoo* [For90] dataset. *Product Classification and Clustering* contains 35311 product titles, cluster, and category labels. A sample record is
`{'product_title': 'samsung galaxy s9 lilac purple 6.2 128gb 4g unlocked sim free', 'cluster_label': 'Samsung Galaxy S9+ 128GB', 'category_label': 'Mobile Phones'}`.
The category labels describe the broad category, while the cluster label is more fine-grained. The *Zoo* dataset consists of animals with a name, indicator attributes such as 'hair', 'feathers', 'eggs', 'milk' ... and the classification variable, containing values such as 'Mammal', 'Bird' and 'Reptile'. We transform the indicator attributes to a string representation. For instance, we transform the boolean values in the 'hair' column to 'has hair' and 'has no hair'. A sample record would be
`{'name': 'bear', 'hair': 'has hair', 'feathers': 'has no feathers', ..., 'class': 'Mammal' }`.

To simulate the user queries, we iterate over the unique cluster classes and formulate an execution plan that filters for the fixed cluster class. For the *Zoo* dataset, such execution plan may be $\sigma_{(\text{name,hair,feathers,...,catsize}) \approx' Mammal'}(\text{Zoo})$. Since the *Product Classification and Clustering* dataset contains a 'cluster_label' and a 'category_label', we test both levels of abstraction. Therefore, two sample queries are:

$$\sigma_{\text{product} \approx' \text{Samsung Galaxy S9+ 128GB}'}(\text{Products})$$

and

$$\sigma_{\text{product} \approx' \text{Mobile Phones}'}(\text{Products})$$

4.2.1 Metrics

To create a ground truth, we add the ID of every record and the respective class to the set G. For the *Zoo* dataset an element of G would be `{'name': 'bear', 'class': 'Mammal'}`.

We compare the ground truth G with the result set R. To create R, we add every record from the query result to a set, together with the selected class. So, if bear appears in the result of $\sigma_{(\text{name,hair,feathers,...catsize}) \approx' Mammal'}(\text{Zoo})$, `{'name': 'bear', 'class': 'Mammal'}` would be added to R.

As in Section 4.1, we calculate precision, recall, F1 and the throughput for a threshold-based, a zero-shot and a combined approach. Furthermore, we investigate the influence of the two different LLaMA models on all metrics. Finally, we explore how the level of abstraction influences the effectiveness of the operator.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

4.2.2 Results

Figure 4.8 shows the evaluation results for the *Zoo* dataset. The threshold-based approach performs best in terms of throughput. This method yields a high recall, however the precision is the worst across all methods. Therefore, this method achieves the worst F1 score. The zero-shot and combined approaches for both LLaMA models achieve similar scores. However, the larger LLaMA model achieves a higher F1 score in general, because both precision and recall are high, while the smaller model yields lower recall.

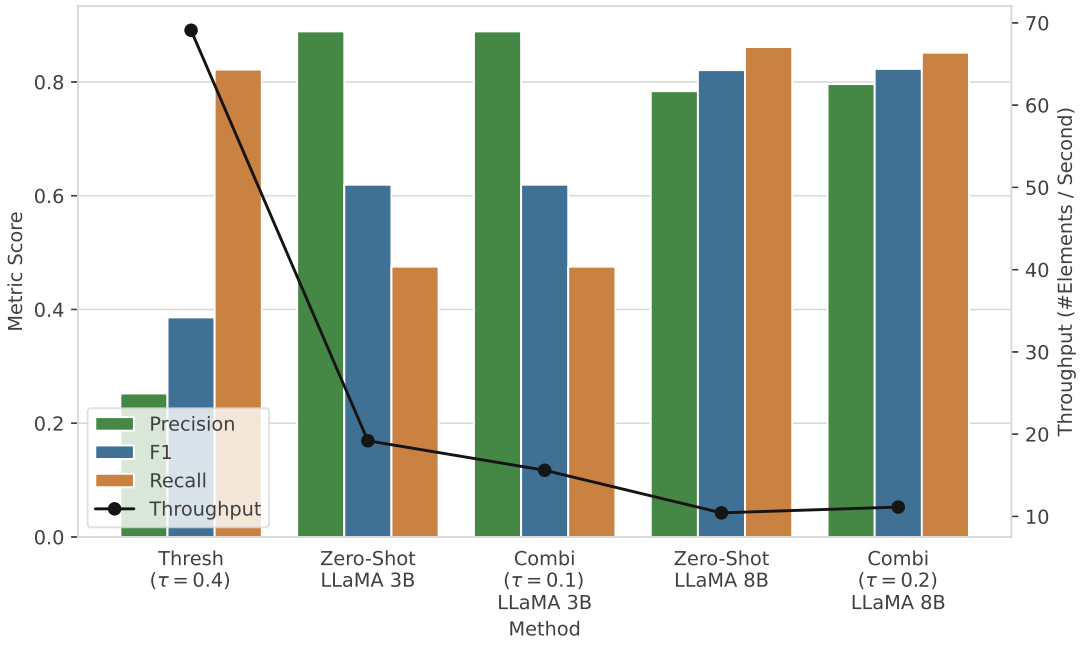

Figure 4.8: Comparison of Precision, Recall and F1 for the Highest F1 Scores across all Methods and Models for the *Zoo* Dataset

One can observe that the throughput for `Zero-Shot LLaMA 3B` of 20 records per second is higher compared to the combined approach `Combi LLaMA 3B` with 17 records per second. We further investigate this phenomenon in Figure 4.9. The brown line represents the number of records (TPs and FPs) that are yielded by the threshold-approach. The gray line depicts the throughput for the zero-shot and the combined approach for the respective threshold. One can observe that zero-shot achieves a higher throughput than the combined approach for thresholds $\tau \leq 0.4$, while for higher τ the combined approach is faster. Since the LLM inference is more expensive than creating and comparing embeddings, first filtering the records using embeddings and afterwards applying an LLM, improves performance only if a sufficient amount of records are filtered. If most of the records are passed to the LLM, as for $\tau \leq 0.4$ in Figure 4.9, the runtime costs of creating embeddings and the LLM inference time just adds together. This results

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

in a higher runtime compared to the zero-shot-only approach.

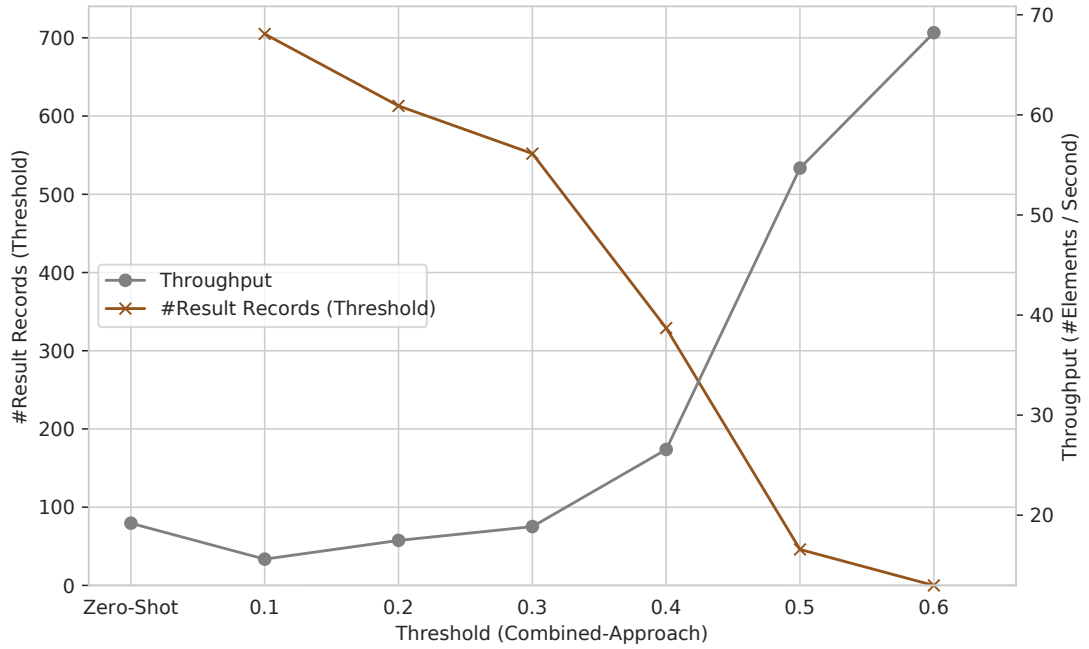

Figure 4.9: Comparison of the Throughput of the Combined Approach with the number of records (TP+FP) in the Threshold-Only Approach for the *Zoo* Dataset

Figure 4.10 shows the evaluation results for the broad categories of the *Product Classification and Clustering* dataset. The zero-shot and combined approaches produce equal F1 scores for both models. The smaller model also produces equal values for precision and recall, while for the larger model, there is a higher variance between the values. As for the *Zoo* dataset in Figure 4.8, the smaller LLaMA model produces higher precision and lower recall scores compared to the larger one. Again, this results in higher F1 scores for the larger LLaMA model, with a maximum F1 of 0.61 for LLaMA 3B and 0.73 for LLaMA 8B. The threshold method performs in between both models regarding the F1 score. One can observe that the combined approaches have higher throughput compared to the zero-shot ones.

In Figure 4.10, we compare the results for the fine-grained classes. As for the broad category, LLaMA 3B shows the least effectiveness on the F1 score (0.64), while the PTM-only approach scores 0.8. Here, the differences between both LLaMA models is very high, where both `Zero-Shot (8B)` and `Combi` are able to score F1 scores of 0.9. In this case, the combined-approach increases the throughput by leveraging the two stage filtering step. Also there is a slight increase of precision from 0.98 to 1.0 using the combined methods.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

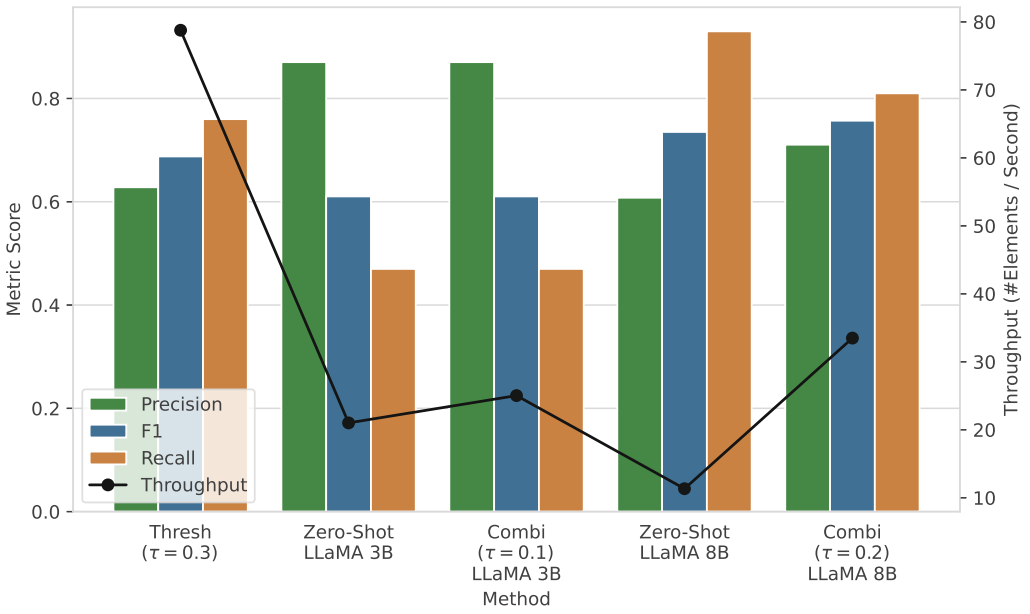

Figure 4.10: Highest F1 Scores for the *Product Classification and Clustering* Dataset (Broad Category)

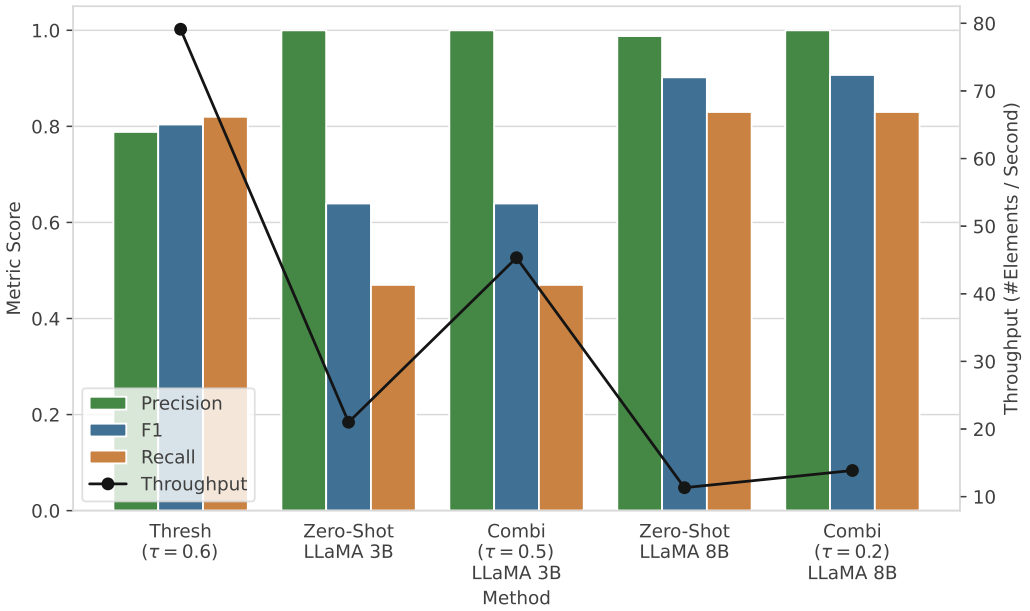

Figure 4.11: Highest F1 Scores for the *Product Classification and Clustering* Dataset (Fine-Grained Category)

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

We compare the F1 scores and threshold for the fine-grained and the broad categories in Figure 4.12. One can obtain two main observations. First, the usage of fine-grained categories will increase the result's F1 score and secondly, it requires lower thresholds τ for the broader category. This phenomenon can be explained by examining examples from the broad and fine-grained categories. For the query term 'Samsung Galaxy S9+ 128GB', there is a higher textual overlap to 'samsung galaxy s9 lilac purple 6.2 128gb 4g unlocked sim free', compared to the only conceptual overlap with 'Mobile Phones'. Hence, the embeddings' cosine similarity is higher for the fine-grained category and thus, higher threshold with higher precision can be used.

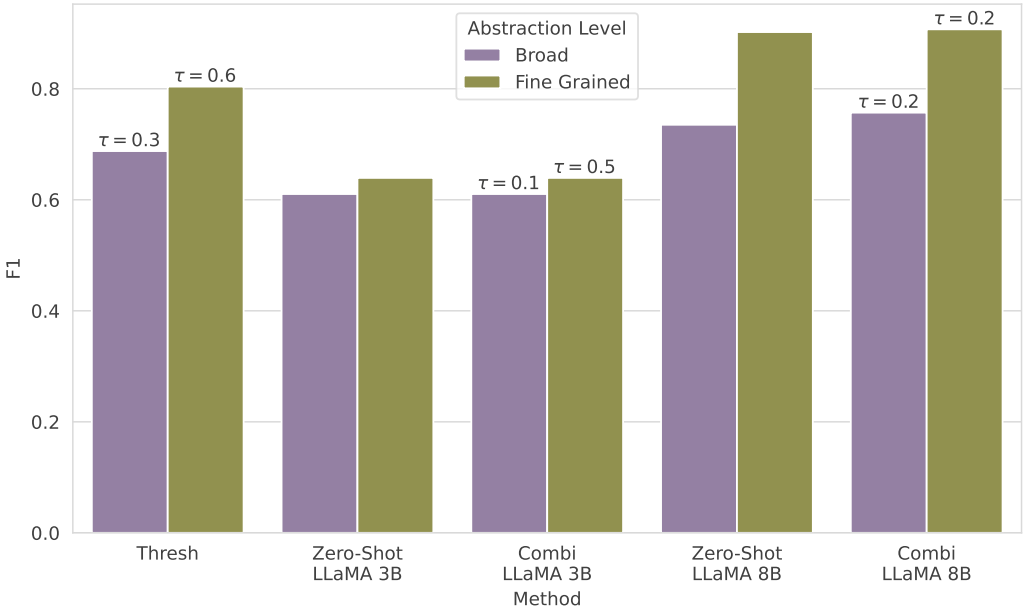

Figure 4.12: Comparison of highest F1 Scores for Product Filtering using Broad and Fine Grained Categories for the *Product Classification and Clustering* Dataset

4.3 Noise-Aware / Semantic-Grouping

We evaluate the noise-aware grouping task using the *MusicBrainz20K* dataset. This dataset contains 10,000 unique songs from the *MusicBrainz* database. The songs are modified using the *DaPo* data generator, which duplicates the data and adds noise. This results in a total of 19,375 noisy entries [HPWR17]. The task is to group duplicate songs. We reduced the entries to relevant columns. The final columns include 'TID', the unique identifier, 'CID', which indicates the cluster affiliation, 'CTID' as a unique identifier within the cluster and data columns such as the 'title', 'artist' or 'length'. Two sample song entries that represent the same song are:

```
{'TID': 1748, 'CID': 3592, 'CTID': 3, 'number':  'b', 'title':
'Brothers and Sisters', 'length':  289000, 'artist':
```


Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

'Coldplay', 'album': 'Trouble', 'year': 1990, 'language': 'English'}
and
{'TID': 6905, 'CID': 3592, 'CTID': 1, 'number': 'B', 'title': 'Brothers and Sisters - Trouble', 'length': 4.817, 'artist': 'Coldplay', 'album': null, 'year': '54617017MB-01', 'language': 'English' }.

We define the execution plan as:

$$\Gamma_{\text{number, title, length, artist, album, year, language; SET\_AGG(TID)}}(\text{MusicBrainz})$$

This will return records, where every record represents one group, containing sets with the IDs of the records belonging to that group. For the previous example, the set is {1748, 6905}.

Next to the noise-aware grouping, we also evaluate semantic grouping using the *Product Classification and Clustering* dataset, which we defined in Section 4.2. We use the following execution plan to group products based on their broad or fine-grained category:

$$\Gamma_{\text{product; SET\_AGG(product)}}(\text{Products})$$

We investigate how the selection of clustering algorithms and dimensionality reduction impacts the throughput and the effectiveness. We select DBSCAN, HDBSCAN, and KMeans. For KMeans, we chose k, the number of final clusters, to be exactly the number of classes, so the algorithm has a chance to find the correct clusters. We evaluate how the selection of ϵ, the minimal required similarity to other data points, affects the results of DBSCAN and HDBSCAN. For HDBSCAN, we test both the standalone version with $\epsilon = 0$ and the combined implementation for $\epsilon > 0$.

Note that in a real-world application, the number of classes is usually unknown, hence such selection of k for KMeans cannot be applied.

4.3.1 Metric

To evaluate both grouping tasks, we assign unique cluster labels for all buckets. Thus, if a result record from the *MusicBrainz20K* query contains the IDs 1748 and 6905, they are assigned a unique label. Using the ground truth, we calculate the Adjusted Rand Score (ARS) [CR23] to determine how accurately the system grouped the data into the desired buckets. The Rand Score (RS) measures the similarity between two clusterings by comparing all sample pairs. So, as the name indicates, the ARS adjusts the RS with an expected RS $RS_{Expected}$, where both clusterings are entirely random (Equation 4.3). If the ARS is close to one indicates a good clustering, while an ARS close to zero indicates a random group assignment.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

$$ARS = \frac{RS - RS_{Expected}}{max(RS) - RS_{Expected}} \tag{4.3}$$

For *MusicBrainz20K* an ARS = 1 means that all duplicate songs end up in the same group. For the *Product Classification and Clustering*, such a score can be achieved when e.g., all 'Mobile Phones' or 'Samsung Galaxies' (depending on the level of abstraction) are grouped together.

4.3.2 Results

The *Product Clustering Dataset* contains two levels of abstraction: broad and fine-grained categories. First, we investigate the influence of the minimal cosine similarity ϵ on the ARS for DBSCAN and HDBSCAN on both levels. Figure 4.13 depicts the ARS (top) and the amount of predicted classes (bottom) for values of ϵ between 0.0 and 0.4. We depict the results for HDBSCAN on the left and for DBSCAN on the right side of the figure. The orange lines illustrate the results for the broad category and the blue line for the fine-grained. The figure shows that for higher ϵ, the number of clusters declines, because higher ϵ allow for lower-density areas to be considered as a cluster. As a result, for a certain ϵ all data points are clustered into the same group. This results in an ARS score close to 0. Furthermore, there are two distinct peaks where the ARS reaches its maximum for both levels of abstraction. For both HDBSCAN and DBSCAN, an optimal ϵ value is at approximately 0.09 for the fine-grained and approximately 0.15 for the broad category. As for semantic filtering, the fine-grained category was able to produce results of a higher quality, especially for DBSCAN.

Using the optimal ϵ from the hyperparameter estimation in Figure 4.13, we test how the different clustering algorithms perform for n samples of the data. We illustrate the results in Figure 4.14 for the broad category. KMeans overall performs best resulting in an ARS of 0.63. Furthermore, the usage of KMeans results in consistent ARS above 0.53 on the full dataset. HDBSCAN achieves the second best ARS of 0.52. For $n > 10.000$, HDBSCAN produces stable results, while DBSCAN's performances steadily decreases for higher n. On the entire dataset ($n = 35311$) DBSCAN achieves an ARS of 0.14. That means that the assigned groups are close to a random assignment. One can observe that the throughput of KMeans overall increases for larger data volume, while the throughput of DBSCAN and HDBSCAN reduces for higher n.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

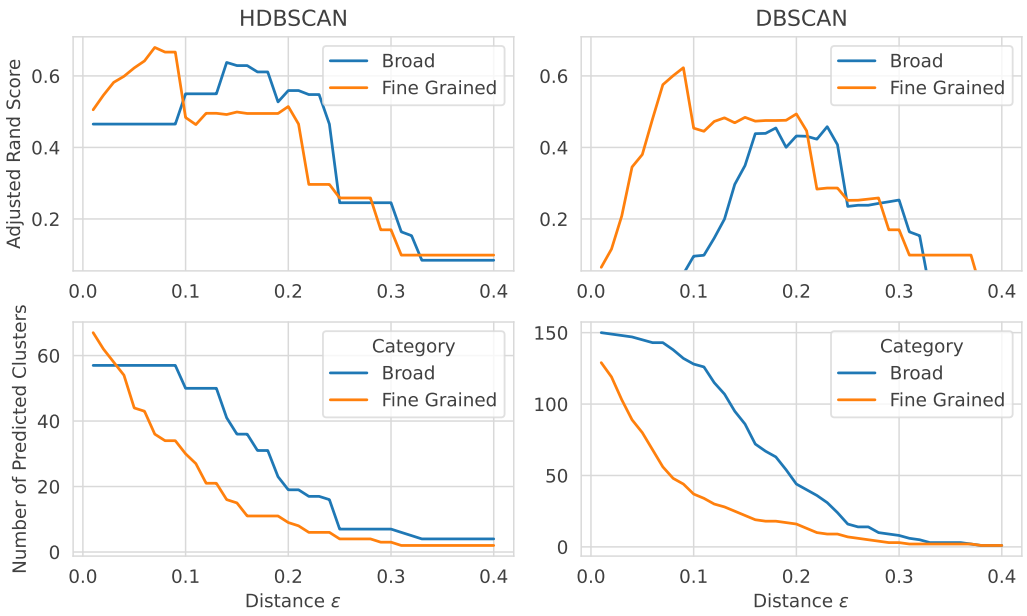

Figure 4.13: Influence of ϵ on the ARS and the Number of Clusters for the Broad and the Fine-Grained Category of the *Product Classification and Clustering* Dataset

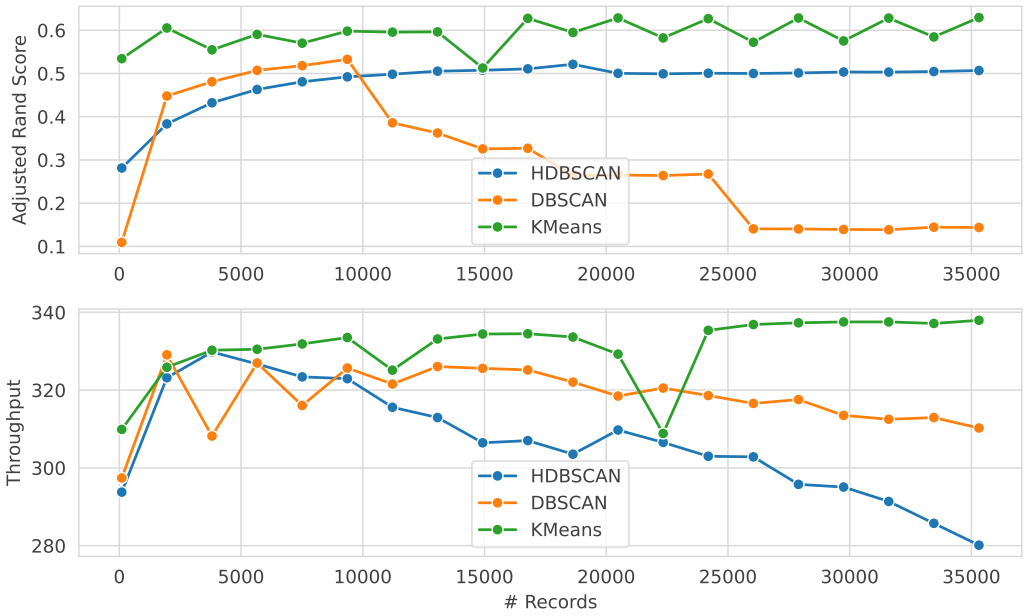

Figure 4.14: Influence of the Data Volume on the ARS and Throughput for the *Product Classification and Clustering* Dataset

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

In Figure 4.15, we investigate why DBSCAN fails for higher n. The figure depicts a 2D projection of the embeddings. The colors indicate the predicted class affiliation. One can observe that for the small subset (left sub-figure), the classes are easily distinguishable making it ideal for a density based cluster algorithm. However, a higher number of records (right sub-figure), the data points are distributed in a big data cloud. The extreme density hinders the algorithm to distinguish clusters. Hence, DBSCAN assigns the same class for most of the records.

We discovered a similar behavior for both DBSCAN and HDBSCAN on the fine-grained dataset. Due to the highly dense areas and numerous clusters (35311 different classes), the algorithms are not able to provide meaningful grouping for any ϵ.

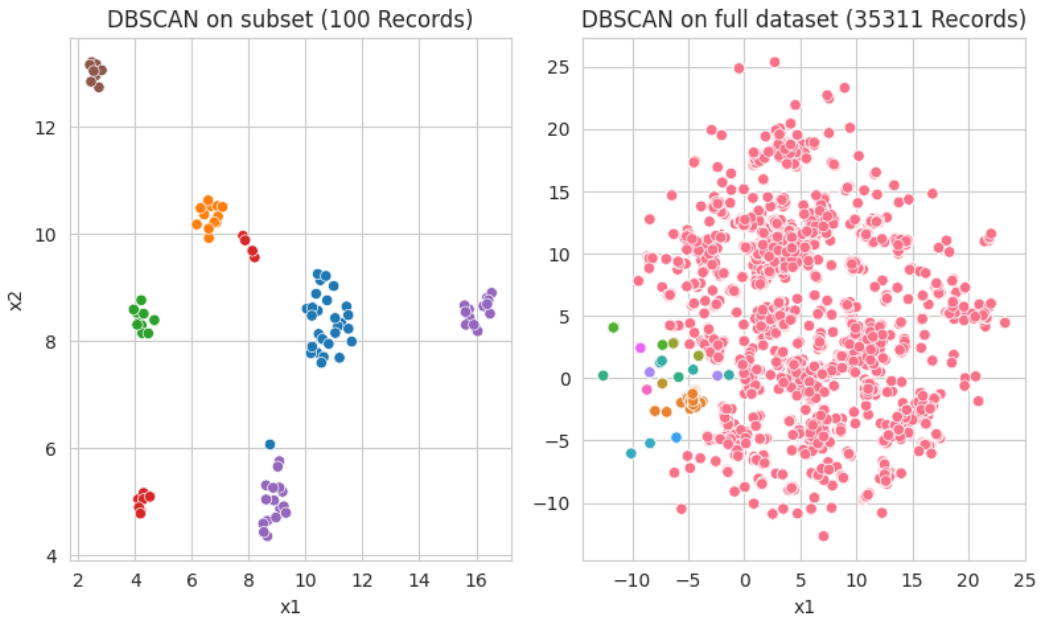

Figure 4.15: Visualization of the Data Distributions for the Embeddings from the *Product Classification and Clustering* Dataset using a subset of 100 records and the full dataset

Continuing with the *MusicBrainz20K* dataset, which we use to evaluate what influence the serialization method and the dimensionality reduction have on the ARS and the throughput. We performed the same hyperparameter estimation as for the product dataset, resulting in KMeans($k = |\text{UniqueSongs}|$), DBSCAN($\epsilon = 0.1$) and HDBSCAN($\epsilon = 0.0$).

As covered in Section 3.2.3, we differentiate between full- and field serialization. For the full serialization, the entire record is converted to a string and then passed to the PTM, while for field serialization, every column is converted to a string and the resulting vectors are merged to a large vector. As the SBERT model embeds strings into vectors

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

of size 768, and we select 5 columns for grouping, this full serialization method creates an input vector of size 768 and $768 \cdot 5 = 3840$ for field serialization.

In Figure 4.16, we plot the ARS (top sub-figures) and the throughput (bottom sub-figures) for different dimensions (x-axis). Thus, a dimension of 2 shows the results for a high dimensionality reduction, while for a dimension of 768 at the full serialization, no reduction is involved. The left sub-figures depict the results for the full serialization, while the right sub-figures show the results for the field serialization. The colors indicate the algorithm used.

HDBSCAN and KMeans overall achieve the highest ARS for no dimensionality reduction on the full serialization. DBSCAN has the lowest metric values for all dimensions and serialization methods. HDBSCAN achieves a maximum ARS score of 0.87 and 0.82 for KMeans. The figure shows that except for a dimensionality of two, the full serialization achieves higher scores for ARS compared to the field serialization. For all dimensionality reductions, the ARS is close to zero for all clustering algorithms when field serialization is applied. Regarding the throughput shown in the bottom sub-figures, one can observe that with higher dimensions, the throughput decreases for HDBSCAN and KMeans for both serializations. HDBSCAN achieves a throughput of approximately 80 records per second for 2 dimensions and field serialization, while for the non-reduced embeddings (dimensionality of 768), it results in a throughput of 20. The throughput of DBSCAN remains comparably stable at approximately 80 records per second for full serialization and 74 for field serialization. When comparing the throughputs for both serialization methods, it shows that the values for the field serialization are generally lower for the respective clustering method and dimension. Overall, for the *MusicBrainz20K* dataset, full serialization is generally preferable for both effectiveness and efficiency. DBSCAN is the least effective algorithm for this dataset, while KMeans and HDBSCAN achieve comparable results.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

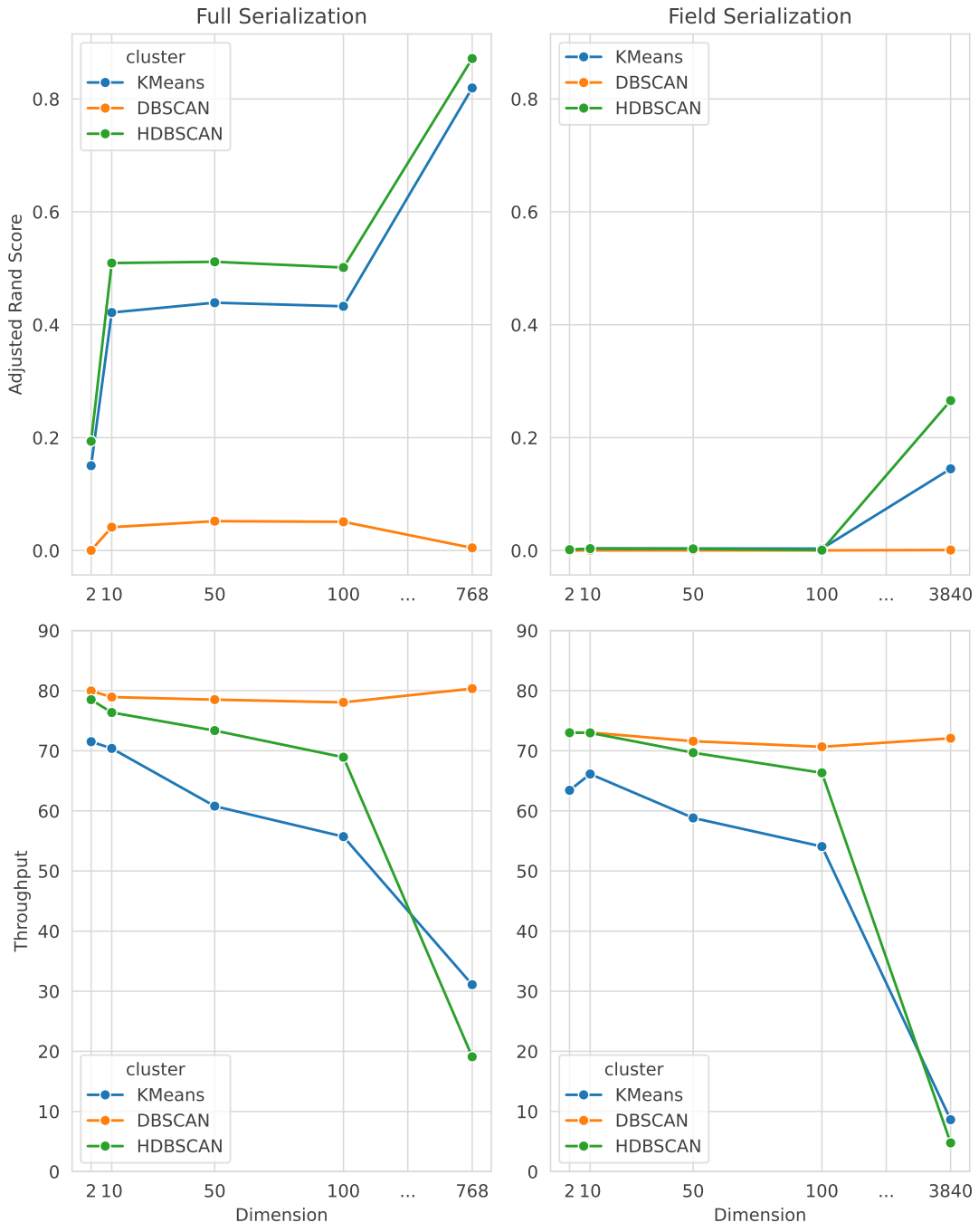

Figure 4.16: Comparison of Cluster Algorithms and Serialization Methods to Group Duplicate Songs from the *MusicBrainz20K* Dataset

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

CHAPTER 5

Discussion

Employing PTMs and LLMs in a database framework can have a huge benefit for tasks that require semantic understanding of text. Traditional techniques where the metric relies on lexical similarity may work for some cases like blocking in entity matching (SPARKLY, utilizing TF-IDF vectors is able to score a recall of 0.992 on the *AbtBuy* dataset [PGD23]). However, such methods fail when semantic understanding is required. In tasks such as semantic filtering, these traditional methods are not able to find records such as 'Samsung Galaxy S8+ 64GB' when querying for 'Mobile Phones', because there is no explicit textual overlap. Our system, utilizing semantic reasoners, is able to find the desired records for such queries. However, we show that textual overlap also increases the quality of the results for the semantic-aware system. Unfortunately, these semantic capabilities come at the cost of runtime and add additional effort to users by requiring them to tune a parameter.

The evaluation demonstrates that both the quality of results and system efficiency are highly dependent on the selected methods. For semantic-filtering and data integration, we employ threshold-based, zero-shot-based, and a combined approach. While zero-shot approaches using LLMs yield superior F1 scores due to increased precision, they come at a significant computational cost, especially when the amount of parameters increases. Threshold-based approaches, on the other hand, offer a highly efficient alternative with tunable precision and recall through setting the threshold. The combined approach emerges as the best trade-off between effectiveness and efficiency. If the threshold τ is chosen in a way that the pure threshold-based method will score a precision of 1.0, meaning that all correct records are found, the combined approach will always produce as good results as the zero-shot approach while drastically reducing execution time. In some cases, the combined approach is able to achieve an even better F1 score compared to the zero-shot-approach, because some of the LLM-inferred false positives are filtered by the threshold-based-approach, increasing the precision and therefore the F1.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

However, there is no universal parameter setting that guarantees optimal results across all datasets, query types and methods. The ideal parameters vary depending on factors such as method, data, query objectives, and semantic granularity. For instance, the *iTunesAmazon* achieves best results with a column wise comparison of the embeddings, while for *AbtBuy*, the usage of full-record-wise embeddings performs better. This freedom of parameters gives users better control over the results, but makes the generation of results less intuitive.

The parameter τ emerges as the most significant hyperparameter. Generally, a lower threshold τ for semantic-filtering and data integration increases the recall by allowing more candidate matches, whereas a higher τ favors precision by filtering out potentially noisy results. For instance, in the data integration task for the *AbtBuy* dataset, the combined approach using full, record-wise embeddings requires a threshold of $\tau = 0.7$ for optimal results, while the threshold-only approach with the same embedding methods requires $\tau = 0.8$. The semantic-filtering and semantic-grouping tasks on the *roduct Classification and Clustering* dataset, exemplifies which influence semantic granularity has on the optimal thresholds. Filtering and grouping for the fine-grained category requires higher thresholds compared to the broad category. This behavior is attributed to the fact that narrow categories have closer embeddings, whereas broader categories exhibit more variance.

As τ has a huge influence on the outcome of `Similar-Join` $\bowtie_{\approx_\tau}$ and `Semantic-Equal` $\approx_\tau$, the clustering algorithm and the hyperparameter k and ϵ are equally important for `Semantic-Aggregation` $\Gamma_{\approx_\tau}$. The evaluation demonstrates that DBSCAN doesn't outperform a random assignment for larger datasets. HDBSCAN and KMeans reliably achieve better results, however again at the cost of a higher runtime. We also show that dimensionality reduction has no positive effect on the effectiveness, as the ARS always worsens with greater reduction. KMeans requires the parameter k, the expected number of groups, which is usually unknown for generic database requests. Therefore, HDBSCAN is more applicable in most situations, since it automatically determines patterns and doesn't necessarily require any hyperparameter.

Another quite important factor is the choice of the generative language models and the amount of parameters. Despite testing with relatively small LLaMA models (3B and 8B parameters), the system demonstrates reasonably strong effectiveness. We show, that for higher parameters the quality of results also increases, suggesting that even larger models, such as LLaMA 70B or 405B, could further enhance results. However, the computational demands of LLM-based methods remain a major limitation. Even with an 8B parameter model, executing a zero-shot join on the full *AbtBuy* dataset of 1081×1092 entries would take $1081 \cdot 1092 \cdot 0.08s = 94436s$ which is approximately one day and two hours, making it impractical for real-world database operations. This underscores the necessity of the combined approach, where a threshold-based blocking stage significantly reduces the number of required LLM operations. Furthermore, it highlights the necessity of vector index structures which further reduces the overall execution time.

Furthermore, hardware requirements limit the accessibility of LLM-based methods.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

Unlike lightweight databases such as SQLite[1], which operate efficiently on low-end hardware, LLM-enhanced query execution requires GPUs for reasonable throughput [Sag24]. This dependency creates barriers for users with limited computing resources. However, the grouping and threshold-based approaches leverage PTMs only, which requires significantly less powerful hardware to achieve comparably good results, making them a good compromise.

Overall, the system can be employed in versatile situations. Due to the generalization powers of the PTMs and LLMs the system has no domain in which the performance significantly varies. We show, that such a system can have a positive effect on tasks involving semantic text understanding, mostly at the cost of runtime.

[1] https://www.sqlite.org/

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

CHAPTER 6

Conclusion and Future Work

This chapter summarizes the results of the thesis and discusses potential directions for future improvements. The complete source code of the developed semantic-aware query evaluation system is available at https://github.com/HackerBschor/SemanticQueryEvaluation.

6.1 Conclusion

This thesis presented a novel system that integrates semantic reasoning via LLMs and PTMs directly into query execution operators of a traditional database framework, addressing fundamental limitations of traditional RDBMS when confronted with noisy or semantic ambiguous data. These novel operators allow users to filter, select and group data based on semantic equivalence rather than exact matches. The main contributions include:

- **Design and Implementation of Semantic-Aware Database Operators**
  We provide detailed design instructions for developing new semantic-aware operators within the *Volcano* framework and also demonstrate their practical feasibility through implementation. The proposed operators for `Scan`, `Project (Map)` π, `Select` σ, `Join` $\bowtie$, and `Aggregation` Γ enable semantic reasoning through utilization of SBERT models for semantic embeddings and LLaMA models for zero-shot validation.
- **Efficient Integration Strategies for Semantic Reasoning**
  We have designed and implemented multiple embedding and comparing strategies for database records including field or full serialization and record-wise or column-wise comparison. Each method has different advantages depending on the operator and the data. This offers flexible trade-offs between the effectiveness and the computational efficiency of the query.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

The proposed two-stage validation pipeline combines the fast cosine similarity assessment via index structures of SBERT embeddings with the precise LLaMA-based zero-shot validation. This demonstrates high quality results, while minimizing inference costs.

- **Extensive Evaluation Across Multiple Tasks and Datasets**
  The selected tasks: semantic joins, semantic filtering, and semantic/noise-aware grouping, effectively evaluate the operators' performance on queries over noisy data. Our system performs well with a maximal F1 score of ≥ 0.8 on the joining and filtering tasks for all datasets under certain hyperparameter settings. For the semantic/noise-aware grouping task, the result quality varies with a maximal ARS of 0.6 for the *Product Classification and Clustering* dataset and 0.9 for *MusicBrainz20K*.

Through the dedicated operator design, the system provides users with high quality results in a reasonable amount of time. Moreover, we demonstrate how these strategies can be tailored to specific user needs or hardware constraints. As the current development in natural language processing makes progress rapidly, we expect that such a system will become more efficient and effective in the future. Hence, the reasoning will require less powerful hardware for higher quality results.

As the novel operators fit perfectly into existing database frameworks, current state-of-the-art RDBMS can simply extend the new operators, enabling semantic reasoning if needed. Therefore, in the future, querying both structured and unstructured data is not limited by rigid syntax but becomes more human-centric, context-aware, and semantically meaningful.

6.2 Future Work

Since scalability in data volume is one of the biggest limitations of the system, the next step is to increase performance through parallel execution and the implementation of modern databases methods. As our system already implements the Volcano model, an inherently parallel model, our next goal is to parallelize the operators. Further improvements may include data-centric code generation or vectorization as they are the primary branches of modern databases [KLK+18]. Since batch processing is commonly used in most deep learning applications, we identify vectorization as the more promising approach, as it directly aligns with this paradigm.

Despite the fact that LLMs are the primary bottleneck for efficiency, we must not neglect the runtime of the embeddings generation. One major advantage of using Sentence Transformers is that they allow independent generation of embeddings for comparison. While for the zero-shot inference in the data integration task, two records must be serialized and merged into one string, methods utilizing Sentence Transformers can create the embeddings separately. Both embeddings are only required during the comparison step.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

Hence, the system can invoke the embedding generation at record insertion, improving the performance later for query evaluation. This would also enable the pre-computation of index structures, further increasing the performance. Extensions such as *pgai* already implemented a similar behavior for PostgreSQL.

Another way to improve efficiency is the usage of other, more traditional approaches to solve the discussed tasks which don't require deep neural models. Such methods include TF-IDF Vectorization [MR23] or Fuzzy-String-Matching using the Levenshtein distance [PA21]. Hence, the next step is to create stand-alone operators utilizing traditional methods and using these methods in the blocking stage of the `Similar-Join` $\bowtie$ operator.

The current threshold-based approaches require the user to set different parameters. As mentioned before, this reduces reduces intuition, which works contrary to the idea to enable a more human-centered interaction with databases. Therefore, a further step is to use LLMs for hyperparameter tuning, where an LLM evaluates results on a small subset for pre-defined parameters. So users can just execute the query without any human parameter-tuning involved. However, to give users more control over the outcome, we thought of allowing them to tune an intuitive parameter, which then influences the hyperparameter τ / ϵ of the operator. Such a parameter may be an option such as 'prioritize precision' or 'prioritize recall', depending on the user's needs.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Overview of Generative AI Tools Used

We've used **ChatGPT-4o** during the conceptualizing phase to suggest ideas. Furthermore, we've used the model for rephrasing and typo correction purposes.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

List of Figures

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

List of Tables

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

List of Algorithms

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Bibliography

[Akr20] Leonidas Akritidis. Product Classification and Clustering. UCI Machine Learning Repository, 2020. DOI: https://doi.org/10.24432/C5M91Z.

[AM21] Muhammad Sidik Asyaky and Rila Mandala. Improving the performance of hdbscan on short text clustering by using word embedding and umap. In *2021 8th International Conference on Advanced Informatics: Concepts, Theory and Applications (ICAICTA)*, pages 1–6, 2021.

[ASLS21] Khetam Al Sharou, Zhenhao Li, and Lucia Specia. Towards a better understanding of noise in natural language processing. In Ruslan Mitkov and Galia Angelova, editors, *Proceedings of the International Conference on Recent Advances in Natural Language Processing (RANLP 2021)*, pages 53–62, Held Online, September 2021. INCOMA Ltd.

[BBB+24] Teodoro Baldazzi, Davide Benedetto, Luigi Bellomarini, Emanuel Sallinger, and Adriano Vlad. Softening ontological reasoning with large language models, 2024.

[BBH16] Hannah Bast, Björn Buchhold, and Elmar Haussmann. Semantic search on text and knowledge bases. *Foundations and Trends® in Information Retrieval*, 10:119–271, 01 2016.

[Bhu25] Santosh Bhupathi. Role of databases in genai applications, 2025.

[C+21] Bao Chong et al. K-means clustering algorithm: a brief review. *Academic Journal of Computing & Information Science*, 4(5):37–40, 2021.

[CLL22] Silvia Casola, Ivano Lauriola, and Alberto Lavelli. Pre-trained transformers: an empirical comparison. *Machine Learning with Applications*, 9:100334, 2022.

[Cod90] E. F. Codd. *The relational model for database management: version 2.* Addison-Wesley Longman Publishing Co., Inc., USA, 1990.

[CR23] José E. Chacón and Ana I. Rastrojo. Minimum adjusted rand index for two clusterings of a given size. *Advances in Data Analysis and Classification*, 17(1):125–133, Mar 2023.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

[CRC+19] B. Sri Sai Krishna Chaitanya, D. Ajay Kumar Reddy, B. Pavan Sai Eshwar Chandra, A. Bala Krishna, and Remya R. K. Menon. Full-text search using database index. In *2019 5th International Conference On Computing, Communication, Control And Automation (ICCUBEA)*, pages 1–5, 2019.

[CZL+24] Dongdong Cheng, Cheng Zhang, Ya Li, Shuyin Xia, Guoyin Wang, Jinlong Huang, Sulan Zhang, and Jiang Xie. Gb-dbscan: A fast granular-ball based dbscan clustering algorithm. *Information Sciences*, 674:120731, 2024.

[DGD+24] Matthijs Douze, Alexandr Guzhva, Chengqi Deng, Jeff Johnson, Gergely Szilvasy, Pierre-Emmanuel Mazaré, Maria Lomeli, Lucas Hosseini, and Hervé Jégou. The faiss library. *arXiv preprint arXiv:2401.08281*, 2024.

[EKSX96] Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, KDD'96, page 226–231. AAAI Press, 1996.

[For90] Richard Forsyth. Zoo. UCI Machine Learning Repository, 1990. DOI: https://doi.org/10.24432/C5R59V.

[FS69] Ivan P Fellegi and Alan B Sunter. A theory for record linkage. *Journal of the American statistical association*, 64(328):1183–1210, 1969.

[Gra93] Goetz Graefe. Query evaluation techniques for large databases. *ACM Comput. Surv.*, 25(2):73–169, June 1993.

[Gra94] G. Graefe. Volcano - an extensible and parallel query evaluation system. *IEEE Trans. on Knowl. and Data Eng.*, 6(1):120–135, February 1994.

[HFD+23] Chenxu Hu, Jie Fu, Chenzhuang Du, Simian Luo, Junbo Zhao, and Hang Zhao. Chatdb: Augmenting llms with databases as their symbolic memory, 2023.

[HPWR17] Kai Hildebrandt, Fabian Panse, Niklas Wilcke, and Norbert Ritter. Large-scale data pollution with apache spark. *IEEE Transactions on Big Data*, 6(2):396–411, 2017.

[HSH+07] Joseph M Hellerstein, Michael Stonebraker, James Hamilton, et al. Architecture of a database system. *Foundations and Trends® in Databases*, 1(2):141–259, 2007.

[IK20] Eleni Ilkou and Maria Koutraki. Symbolic vs sub-symbolic ai methods: Friends or enemies? In *CIKM (Workshops)*, volume 2699, 2020.

[JZQ+23] Rolf Jagerman, Honglei Zhuang, Zhen Qin, Xuanhui Wang, and Michael Bendersky. Query expansion by prompting large language models. *arXiv preprint arXiv:2305.03653*, 2023.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

[KE11] Nikolaos Kouiroukidis and Georgios Evangelidis. The effects of dimensionality curse in high dimensional knn search. In *2011 15th Panhellenic Conference on Informatics*, pages 41–45, 2011.

[KGR+22] Takeshi Kojima, Shixiang (Shane) Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. Large language models are zero-shot reasoners. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems*, volume 35, pages 22199–22213. Curran Associates, Inc., 2022.

[KLK+18] Timo Kersten, Viktor Leis, Alfons Kemper, Thomas Neumann, Andrew Pavlo, and Peter Boncz. Everything you always wanted to know about compiled and vectorized queries but were afraid to ask. *Proc. VLDB Endow.*, 11(13):2209–2222, September 2018.

[LHQ+23] Jinyang Li, Binyuan Hui, Ge Qu, Jiaxi Yang, Binhua Li, Bowen Li, Bailin Wang, Bowen Qin, Ruiying Geng, Nan Huo, et al. Can llm already serve as a database interface? a big bench for large-scale database grounded text-to-sqls. *Advances in Neural Information Processing Systems*, 36:42330–42357, 2023.

[LLS+20] Yuliang Li, Jinfeng Li, Yoshihiko Suhara, AnHai Doan, and Wang-Chiew Tan. Deep entity matching with pre-trained language models. *arXiv preprint arXiv:2004.00584*, 2020.

[LLS+21] Yuliang Li, Jinfeng Li, Yoshihiko Suhara, Jin Wang, Wataru Hirota, and Wang-Chiew Tan. Deep entity matching: Challenges and opportunities. *J. Data and Information Quality*, 13(1), January 2021.

[LM24] Jie Liu and Barzan Mozafari. Query rewriting via large language models, 2024.

[LP22] Fang Liu and Demosthenes Panagiotakos. Real-world data: a brief review of the methods, applications, challenges and opportunities. *BMC Medical Research Methodology*, 22(1):287, Nov 2022.

[LPP+20] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in neural information processing systems*, 33:9459–9474, 2020.

[LST24] Michael Li, Jianping Sun, and Xianming Tan. Evaluating the effectiveness of large language models in abstract screening: a comparative analysis. *Systematic reviews*, 13(1):219, 2024.

[MB20] Claudia Malzer and Marcus Baum. A hybrid approach to hierarchical density-based cluster selection. In *2020 IEEE international conference on multisensor*

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

fusion and integration for intelligent systems (MFI), pages 223–228. IEEE, 2020.

[ME92] Priti Mishra and Margaret H Eich. Join processing in relational databases. *ACM Computing Surveys (CSUR)*, 24(1):63–113, 1992.

[MHA17] Leland McInnes, John Healy, and Steve Astels. hdbscan: Hierarchical density based clustering. *Journal of Open Source Software*, 2(11):205, 2017.

[MHM20] Leland McInnes, John Healy, and James Melville. Umap: Uniform manifold approximation and projection for dimension reduction, 2020.

[MKK19] Aritra Mandal, Ishita K Khan, and Prathyusha Senthil Kumar. Query rewriting using automatic synonym extraction for e-commerce search. In *eCOM@ SIGIR*, 2019.

[MLR+18] Sidharth Mudgal, Han Li, Theodoros Rekatsinas, AnHai Doan, Youngchoon Park, Ganesh Krishnan, Rohit Deep, Esteban Arcaute, and Vijay Raghavendra. Deep learning for entity matching: A design space exploration. In *Proceedings of the 2018 international conference on management of data*, pages 19–34, 2018.

[Moo51] C.N. Mooers. *Making Information Retrieval Pay*. Zator technical bulletin. Zator Company, 1951.

[MR23] Mohannad T Mohammed and Omar Fitian Rashid. Document retrieval using term term frequency inverse sentence frequency weighting scheme. *Indonesian Journal of Electrical Engineering and Computer Science*, 31(3):1478–1485, 2023.

[MSL+15] Ingo Müller, Peter Sanders, Arnaud Lacurie, Wolfgang Lehner, and Franz Färber. Cache-efficient aggregation: Hashing is sorting. In *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*, SIGMOD '15, page 1123–1136, New York, NY, USA, 2015. Association for Computing Machinery.

[NLK17] Thomas Neumann, Viktor Leis, and Alfons Kemper. The complete story of joins (inhyper). In *Datenbanksysteme für Business, Technologie und Web (BTW 2017)*, pages 31–50. Gesellschaft für Informatik, Bonn, 2017.

[PA21] Malgorzata Pikies and Junade Ali. Analysis and safety engineering of fuzzy string matching algorithms. *ISA transactions*, 113:1–8, 2021.

[PBGF22] Matteo Paganelli, Francesco Del Buono, Francesco Guerra, and Nicola Ferro. Evaluating the integration of datasets. In *Proceedings of the 37th ACM/SIGAPP Symposium on Applied Computing*, pages 347–356, 2022.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

[PGD23] Derek Paulsen, Yash Govind, and AnHai Doan. Sparkly: A simple yet surprisingly strong tf/idf blocker for entity matching. *Proceedings of the VLDB Endowment*, 16(6):1507–1519, 2023.

[PK01] Gerald Post and Albert Kagan. Database management systems: design considerations and attribute facilities. *Journal of Systems and Software*, 56(2):183–193, 2001.

[PSB24] Ralph Peeters, Aaron Steiner, and Christian Bizer. Entity matching using large language models, 2024.

[PVG+11] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[RG19a] Nils Reimers and Iryna Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 11 2019.

[RG19b] Nils Reimers and Iryna Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 11 2019.

[RG20] Nils Reimers and Iryna Gurevych. Making monolingual sentence embeddings multilingual using knowledge distillation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 11 2020.

[RM05] Lior Rokach and Oded Maimon. Clustering methods. *Data mining and knowledge discovery handbook*, pages 321–352, 2005.

[RZ09] Stephen Robertson and Hugo Zaragoza. The probabilistic relevance framework: Bm25 and beyond. *Found. Trends Inf. Retr.*, 3(4):333–389, April 2009.

[Sag24] Sriramaraju Sagi. Advancing ai: Enhancing large language model performance through gpu optimization techniques. *International Journal of Science and Research (IJSR)*, 13, 03 2024.

[SAM+23] Ruoxi Sun, Sercan Ö Arik, Alex Muzio, Lesly Miculicich, Satya Gundabathula, Pengcheng Yin, Hanjun Dai, Hootan Nakhost, Rajarishi Sinha, Zifeng Wang, et al. Sql-palm: Improved large language model adaptation for text-to-sql (extended). *arXiv preprint arXiv:2306.00739*, 2023.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

[SBZS17] E Yu Sharygin, RA Buchatskiy, RA Zhuykov, and AR Sher. Query compilation in postgresql by specialization of the dbms source code. *Programming and Computer Software*, 43:353–365, 2017.

[SO24] Sanket Salunke and Abdelkader Ouda. A performance benchmark for the postgresql and mysql databases. *Future Internet*, 16:382, 10 2024.

[SWY75] G. Salton, A. Wong, and C. S. Yang. A vector space model for automatic indexing. *Commun. ACM*, 18(11):613–620, November 1975.

[SXY+21] Xuan Sun, Chun Jason Xue, Jinghuan Yu, Tei-Wei Kuo, and Xue Liu. Accelerating data filtering for database using fpga. *Journal of Systems Architecture*, 114:101908, 2021.

[TCDH21] Mohamed Trabelsi, Zhiyu Chen, Brian D. Davison, and Jeff Heflin. Neural ranking models for document retrieval. *Information Retrieval Journal*, 24(6):400–444, October 2021.

[TLI+23] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. Llama: Open and efficient foundation language models, 2023.

[TZL+23] Xiaojuan Tang, Zilong Zheng, Jiaqi Li, Fanxu Meng, Song-Chun Zhu, Yitao Liang, and Muhan Zhang. Large language models are in-context semantic reasoners rather than symbolic reasoners, 2023.

[WA22] Hannah S Walsh and Sequoia R Andrade. Semantic search with sentence-bert for design information retrieval. In *International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, volume 86212, page V002T02A066. American Society of Mechanical Engineers, 2022.

[WHT+24] Jiajia Wang, Jimmy X. Huang, Xinhui Tu, Junmei Wang, Angela J. Huang, Md Tahmid Rahman Laskar, and Amran Bhuiyan. Utilizing bert for information retrieval: Survey, applications, resources, and challenges, 2024.

[ZHLL22] Huchen Zhou, Wenfeng Huang, Mohan Li, and Yulin Lai. Relation-aware entity matching using sentence-bert. *Computers, Materials & Continua*, 71(1), 2022.

[ZJC+23] Hui Zhang, Dexing Jia, Lei Chen, Xiongru Wang, Shuai Wang, and Rui Li. Acceleration and implementation of database aggregation query based on fpga. In *2023 China Automation Congress (CAC)*, pages 817–822, 2023.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

[ZNH+24] Zixuan Zhou, Xuefei Ning, Ke Hong, Tianyu Fu, Jiaming Xu, Shiyao Li, Yuming Lou, Luning Wang, Zhihang Yuan, Xiuhong Li, Shengen Yan, Guohao Dai, Xiao-Ping Zhang, Yuhan Dong, and Yu Wang. A survey on efficient inference for large language models, 2024.

[ZSH24] Jing Zhang, Huan Sun, and Joyce C Ho. Emba: Entity matching using multi-task learning of bert with attention-over-attention. In *EDBT*, pages 281–293, 2024.

[ZYZD16] Rashid Zafar, Eiad Yafi, Megat F. Zuhairi, and Hassan Dao. Big data: The nosql and rdbms review. In *2016 International Conference on Information and Communication Technology (ICICTM)*, pages 120–126, 2016.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.