JOURNAL OF CRITICAL REVIEWS

ISSN- 2394-5125 VOL 8, ISSUE 02, 2021

A Survey on the Implementation and Optimization of In- Database Machine Learning Systems

Arko Bhattacharya
Student, School of Computer Science,
Vellore Institute of Technology,
Vellore, India

arko_bhattacharya@hotmail.com

Abstract— **In-database machine learning has been very popular, almost being a cliche. In spite of being so popular, there exists a lack of awareness and following about the latest innovations and trends of utilizing in-database machine learning to overcome the drawbacks of a traditional data science workflow. This survey consolidates a few instances of work done in the latest and greatest research avenues which have thoroughly exploited the possibilities that come forth while using an in-database ML workflow to solve problems that are faced by its traditional counterpart. We will be discussing about the benefits of in-database ML and will also discuss in detail about some of the optimized implementations of it which have successfully utilized itsbenefits.**

Keywords— In-database ML, DBMS, Data Science

I. INTRODUCTION

A database is defined as an organized collection of data, generally stored and accessed electronically from a computer system. The Database Management System (DBMS) is the software that interacts with end users, applications, and the database itself to capture and analyze the data. [7,12] The DBMS software additionally, encompasses the core facilities provided to administer the database. Computer scientists may classify database-management systems according to the database models that they support. Relational Databases Management Systems (RDBMS) became dominant in the 1980s. These model data as rows and columns in a series of tables, and the vast majority use Structured Query Language for writing and querying data. In the 2000s, non-relational databases became popular, referred to as NoSQL because they use different query languages. [7,12] One of the most critical components in machine learning projects is the DBMS. With the help of this system, a large number of data can be sorted and one can gain meaningful insights fromthem.

[13] It is a foundational block within the data science / machine learning workflow as all the data queries and exchanges involve the DBMS. [1, 2, 3, 5, 6, 7] As the data-science application grows to be more expansive and the queries get more elaborate and complex, the process requires frequent movement of data from database engines to analytical engines, which adds an excessive overhead on data scientists and becomes a performance bottleneck for model training.[3] The most obvious solution to this bottlenecking problem isIn-

Database Machine learning, i.e. implementing machine learning natively within the DBMS. [3] In-Database Machine Learning serves to aid the elimination of the data-overhead by applying intelligent data-processing within the database. Technological benefits include:

- **Fast Data Prep** – Analytics databases have things like joining datasets, feature engineering, eliminating outliers, missing value imputation, etc. down pat.[11]
- **More Efficient Processing** – Combining the computation engine and data storage management system eliminates the need to move data between a database and a statistical analysis tool. Moving your data around and transforming it costs CPU, IO, time, and money.[11]
- **Production Ready** – You won't have to redo all the work you did experimenting, training and testing, in some completely different technology to make it production ready.[11]
- **No Need to Down-sample** – Train and test models on the data right where it sits. No need to pull out something you hope is a representative sample to put it in some other technology you hope can handle the volume. [11]

This impact and benefits all the nodes within the ecosystems that utilize any form of database software:

- **For line of business people**, these advantages mean you get your measurable results from machine learning projects faster. Easy prototyping of predictive models means the gap between ideation and practical application is greatly reduced. New ideas can be implemented and put into action, and your business gets incremental ROI quicklyand

ISSN- 2394-5125 VOL 8, ISSUE 02, 2021

continuously. No waiting for months and months to see any return.[10]

- **For data scientists and data analysts**, save overhead and time, and gain ease of use. Most of this work is done through a basic SQL interface that you are already likely to be familiar with. Managing models is easier. Calling, training, testing and other aspects of your workflow are easier to do. In addition, you save the CPU/IO hit and time overhead of moving data from one format to another before you can use it. The biggest advantage is at the end of the process. Moving your model from development to production is vastly simplified by already having it in the same database in development that will use the model in production.[10]

- **For database administrators and architects**, see a wide variety of advantages. Security, and simplicity of architecture are possibly the biggest benefits. Databases have intense levels of built-in security: compaction, encryption, role-based access, etc. In addition, because so much work can be done inside a single location, data management architecture can be simplified. Fewer tools need to interface to make things worksmoothly.[10]

Since we have established the multitude of benefits that accompany in-database machine learning, the next logical thing to know is how to actually implement it in projects. We will look at some of the best and recent applications of in-database ML, that solve a problem that would be prevalent in traditional DBMS workflows.

II. INSTANCES OF SOME TRADITIONAL TECHNIQUES IMPLEMENTED IN ADBMS

Before starting with the actual implementations, let's understand a few basic concepts and techniques implemented in a DBMS to boost its efficacy.

A. Indexing inDBMS

Indexing is a way to optimize the performance of a database by minimizing the number of disk accesses required when a query is processed. It is a data structure technique which is used to quickly locate and access the data in a database. Indexes are created using a few database columns.[32]
The most common traditionally used indexing technique is B-/B+ tress or variations of it.

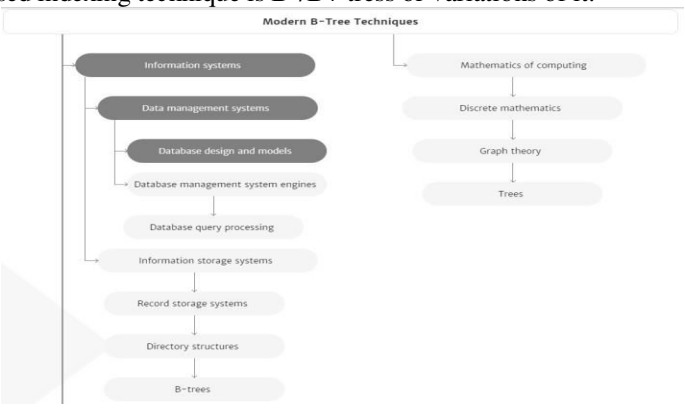


Fig. 1. Categorical representation of B- trees (also applicable for B+ trees) [55]

A **B+ tree** is an m-ary treewith a variable but often large number of children per node. A B+ tree consists of a root, internal nodes and leaves.[54] The root may be either a leaf or a node with two or morechildren.

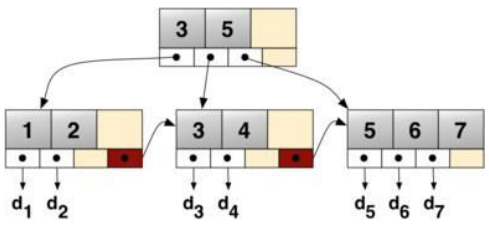


Fig. 2. A simple B+ tree example linking the keys 1–7 to data values d_1-d_7. The linked list (red) allows rapid in-order traversal. This particular tree's branching factor is b=4.[54]

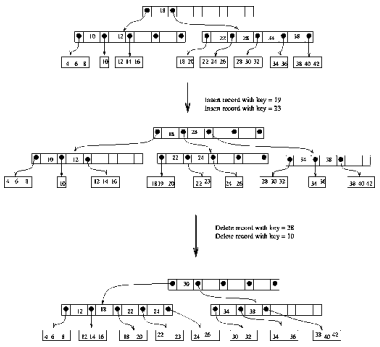


Fig. 3. B+ tree indexing [18]

B. Retrofitting inDMBS

Retrofitting is a graph-based learning technique for using lexical relational resources to obtain higher quality semantic vectors. It is applied as a post-processing step by running belief propagation on a graph constructed from lexicon-derived relational information to update word

vectors. This allows retrofitting to be used on pre-trained word vectors obtained using any vector training model.[16]

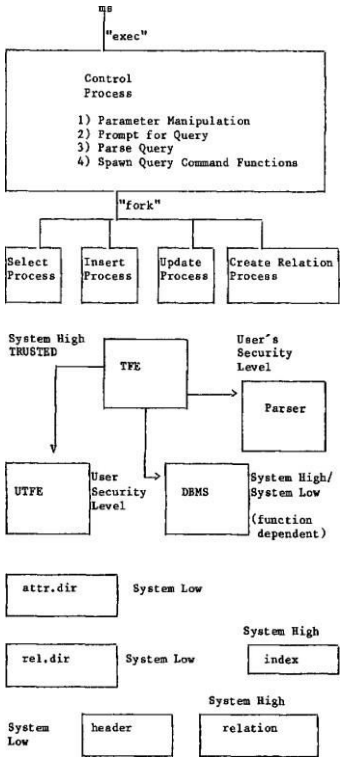


Fig. 4. Flowchart for retrofitting in DBMS [53]

C. Cardinality Estimation in DBMS

Cardinality Estimation model is set on four basic assumptions:

- Independence: Data distributions on different columns are assumed to be independent of each other, unless correlation information is available and usable.

ISSN- 2394-5125 VOL 8, ISSUE 02, 2021

- Uniformity: Distinct values are evenly spaced and that they all have the same frequency. More precisely, within each histogram step, distinct values are evenly spread and each value has same frequency.
- Containment (Simple): Users query for data that exists. For example, for an equality join between two tables, factor in the predicates selectivity1 in each input histogram, before joining histograms to estimate the join selectivity.
- Inclusion: For filter predicates where Column = Constant, the constant is assumed to actually exist for the associated column. If a corresponding histogram step is non-empty, one of the step's distinct values is assumed to match the value from thepredicate.

Sometimes, alternative query formulations or other measures are possible and these are pointed out:

- Queries with predicates that use comparison operators between different columns of the same table.
- Queries with predicates that use operators, and any one of the following aretrue:
- There are no statistics on the columns involved on either side of theoperators.
- The distribution of values in the statistics is not uniform, but the query seeks a highly selective value set. This situation can be especially true if the operator is anything other than the equality (=) operator.
- The predicate uses the not equal to (!=) comparison operator or the NOT logicaloperator.
- Queries that use any of the SQL Server built-in functions or a scalar-valued, user-defined function whose argument is not a constantvalue.
- Queries that involve joining columns through arithmetic or string concatenationoperators.
- Queries that compare variables whose values are not known when the query is compiled and optimized. [52]

D. Performance Regression inDBMS

The practical art of constructing database management systems (DBMSs) involves a morass of trade-offs among query execution speed, query optimization speed, standards compliance, feature parity, modularity, portability, and other goals. It is no surprise that DBMSs, like all complex software systems, contain bugs that can adversely affect their performance. The performance of DBMSs is an important metric as it determines how quickly an application can take in new information and use it to make new decisions. Both developers and users face challenges while dealing with performance regression bugs. First, developers usually find it challenging to manually design test cases to uncover performance regressions since DBMS components tend to have complex interactions. Second, users encountering performance regressions are often unable to report them, as the regression-triggering queries could be complex and database-dependent. Third, developers have to expend a lot of effort on localizing the root cause of the reported bugs, due to the system complexity and software developmentcomplexity.[6]

III. CASE STUDIES:

A. RETRO: Relation Retrofitting For In-Database Machine Learning on Textual Data[1]

This paper proposes a novel approach to learn numerical representations of text values in databases, capturing the best of both worlds, the rich information encoded by word embeddings and the relational information encoded by database tables. There are massive amounts of textual data residing in databases, valuable for many machine learning (ML) tasks. Since ML techniques depend on numerical input representations, word embeddings are increasingly utilized to convert symbolic representations such as text into meaningful numbers. However, a naive one-to-one mappingofeachwordinadatabasetoawordembedding vector is not sufficient and would lead to poor accuracies in ML tasks. Thus, there is a need to additionally incorporatetheinformationgivenbythedatabaseschema into the embedding, e.g. which words appear in the same column or are related to each other. In this paper,relation retrofitting has been considered as a learning problem, whose solution is an efficient algorithm. The impact of various hyperparameters on the learning problem is investigated and good settings for all of them have been devised. The evaluation shows that the proposed embeddings are ready-to-use for many ML tasks such as classification and regression and even outperform state- of-the-art techniques in integration tasks such as null value imputation and linkprediction.[1]

This paper presents a novel approach for generating learned representations for text values in database systems which find application in a wide range of ML tasks. We introduced RETRO, a novel framework which allows generating relational, retrofitted embeddings for an arbitrary database in a fully automated fashion. In this way, one can easily apply machine learning tasks like regression, binary, and multi-class classification tasks to databases. We validated RETRO experimentally by building standard feed-forward ANNs for classification and regression tasks. The experiments exhibit that RETRO embeddings outperform pre-trained, plain embeddingsbutalsoretrofittedembeddingslearnedusing the approach of Faruqui et al. In addition, we showed the applicability of the RETRO-generated embeddings for data integration tasks like missing value imputation and link prediction where it achieves state-of-the-art performance.

ISSN- 2394-5125 VOL 8, ISSUE 02, 2021

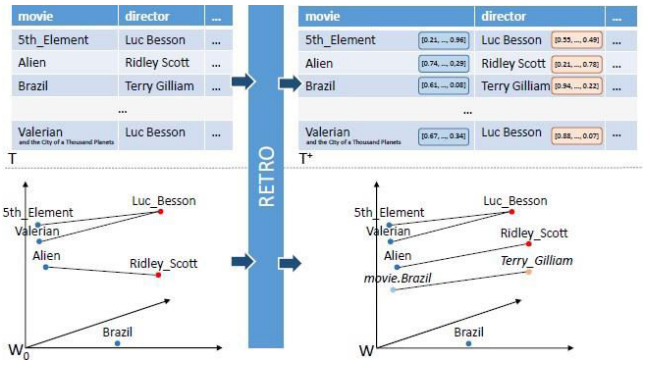


Fig. 5. Relational Retrofitting: original word embedding W0 and relation T (left), retrofitted word embedding W and augmented relation T + (right) [1]

The advantages of RETRO are manifold:

- RETRO exploits the knowledge encoded by pre- trained word embeddings to find meaningful vectors for all terms in adatabase.
- RETRO is able to reflect relational meanings of words that cannot be captured by pure word co- occurrences [56] and modify word vectors to specialize in a specifictask.
- RETRO does not rely on re-training, which allows us to incrementally maintain the word vectors whenever the data in the database changes. Hence, the learned representations are incrementally updateable which is an important requirement for in-database ML systems.
- RETRO is applicable to different types of word vectors, either pre-trained or specifically trained for a givendomain.

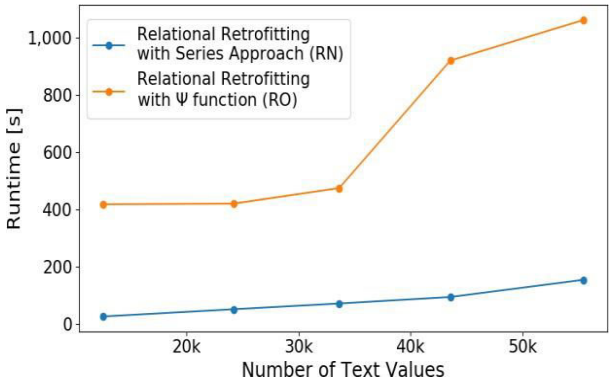


Fig. 6. Runtime of Relational Retrofitting [1]

B. Active Learning for ML Enhanced Database Systems[2]

In many production deployments, the ML models' performance degrades significantly when the test data diverges from the data used to train these models. This paper, addresses this performance degradation by using B- instances to collect additional data during deployment. The paper proposes an active data collection platform, ADCP, that employs active learning (AL) to gather relevant data cost effectively. In this, the researchers have developed a novel AL technique, Holistic Active Learner (HAL), that robustly combines multiple noisy signals for data gathering in the context of database applications. HAL applies to various ML tasks, budget sizes, cost types, and budgeting interfaces for database applications. ADCP is evaluated on both industry-standard benchmarks and real customer workloads. The evaluation shows that, compared with other baselines, the technique improves ML models' prediction performance by up to 2 times with the same cost budget. In particular, on production workloads, this technique reduces the prediction error of ML models by 75% using about 100 additionally collectedqueries.

The training-test data mismatch can significantly limit the applicability of ML techniques in databases, especially in production deployments. We propose an active data collection platform (ADCP) to address this issue by collecting more labeled data with extra resources. We formulate an AL problem to collect labels for specific target test data, and design a simple yet

ISSN- 2394-5125 VOL 8, ISSUE 02, 2021

effective AL strategy, HAL, that is robust to unreliable signals, cost-sensitive, and batch-friendly. Empirically, ADCP greatly improves the ML model's performance with small resource budget, and HAL significantly outperforms state-of-the-art AL baselines in a wide range of scenarios. We contend that this is a crucial step towards building a complete ML pipeline to enhance databasesystems.

ADCP:

An active data collection platform (ADCP) to collect labels to adapt the ML model to the production workload.
The ADCP uses additional resources to collect the labels during production workloads. As shown in Figure 7, an ADCP connects to the ML applications in the production database and the labeling handlers in the B-instances / replicas. The ADCP can acquire the target test data and the ML model from the specific ML application in the database, e.g., the optimizer's plan search space for the production workload

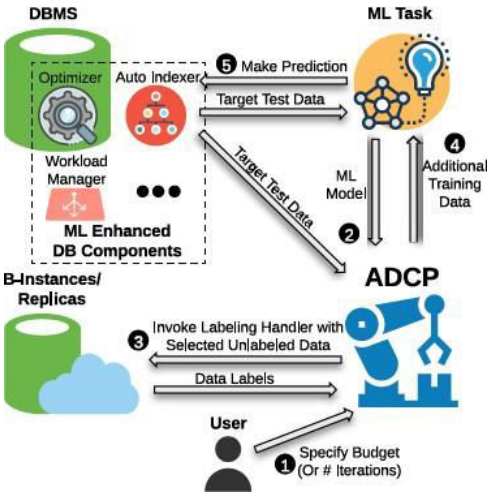


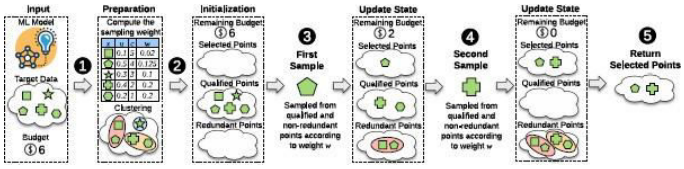
Fig. 7. Active Data Collection Platform (ADCP) [2]

The ADCP actively collects training data with additional resources and returns new labels to retrain the ML model. and the ECP (PRP) model used by the optimizer. The ADCP can also invoke the labeling handler to execute a query plan and acquire its label, e.g., the plan's execution cost, for model retraining.

The users shall adopt the ADCP in the following steps:

1. The users specify the ML application whose model needs improvement and the resource budget for collectingdata.
2. The ADCP acquires the target (unlabeled) data generated by the production database and the MLmodel's predictions on the targetdata.
3. The ADCP selects a batch of points from the target data under the resource budget, and invokes labeling handlers to get the actual labels for thesepoints.
4. The ML model is retrained with the newly collected data.
5. The new model is installed back to the ML application to make predictions for theML enhanced components in the productiondatabase.

The users may optionally specify an iteration number for the data collection and model retraining before installingthe model back to the application. In this case, the ADCP will repeat 3 and 4 for the specified number of iterations while ensuring that the resources consumed stay underbudget.



ISSN- 2394-5125 VOL 8, ISSUE 02, 2021

Fig. 8. HAL Execution Example – HAL selects two points to label under the budget [2]

ISSN- 2394-5125 VOL 8, ISSUE 02, 2021

HAL:

The paper presents a novel AL strategy, Holistic Active Learner (HAL), that addresses the ADCP's AL challenges.

Steps to implement HAL:

We now illustrate the execution of HAL using a running example,

1. Prepare the necessary information by deriving the sampling weight and perform clustering on the target data.

2. Initialize the algorithm's relevant states, including theselectedpoints(None),thequalifiedpoints(all 5 points), and the redundant points (None) as defined in Algorithm. HAL maintains the set of redundant points by tracking the IDs of the clusters that reach the rejection threshold.

3. Sample one point with a cost 4 according to weight. HAL updates the remaining budget to 2. It calculates the qualified points (2 points) under the new budget, and the clusters (1 cluster) that reach the rejection threshold.

4. Sample a second point with a cost 2. After this step, there are no qualified points since the budget is 0. There are also two redundant clusters given the selected points.

5. Return the selected points and terminate, since the budget has beenexhausted.

There are several interesting future directions for this work. For example, ADCP may jointly learn a labeling cost estimation model with the newly labeled points. There is also the opportunity to combine the additionally collected labels for the individual production workload into a global model with better generalization. The early investigations also show that adjusting the training weight after collecting more labels, like covariate shift correction, can further benefit the model error reduction on the target data. It is suggested to view this work as the first step towards tackling the holistic AL problem that arises from applying ML to complex databasesystems.

C. Machine Learning-based Cardinality Estimation in DBMS on Pre-AggregatedData:[5]

For cardinality estimation, a lot of example queries have to be executed during the model training phase to learn a data-dependent ML model leading to a very time-consuming training phase. Many of those example queries use the same base data, have the same query structure, and only differ in their predicates. Thus, index structures appear to be an ideal optimization technique at first glance. However, their benefit is limited. To speed up this model training phase, the core idea is to determine a predicate-independent pre- aggregation of the base data and to execute the example queries over this pre-aggregated data. Based on this idea, we present a specific aggregate-enabled training phase for ML- based cardinality estimation approaches in this paper. As it goes to show with different workloads in our evaluation, we are able to achieve an average speed up of 63 with our aggregate-enabled trainingphase.

The paper proposes the case for cardinality estimation as a candidate for database support of machine learning for DBMS. It detailed an approach for pre-aggregating count information for cardinality estimation workloads. It uses grouping sets, a well-known database data structure, to reduce the data to be scanned by example queries for cardinality estimation with machine learning models. This reduces the execution time of a given workload even though the researchers spend extra time to construct the intermediate datastructures.

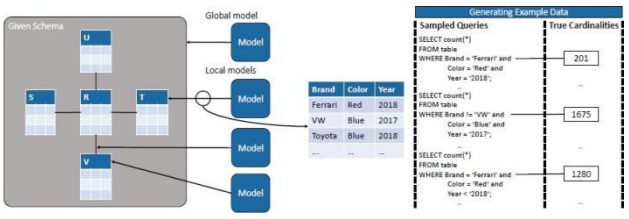


Fig. 9. Overview of ML-based cardinality estimation approaches [5]

For the experiments, they used the original workloads for the local and global ML model approaches on the IMDB dataset. The IMDB contains a snowflake database schema with several millions of tuples in both the fact and the dimension tables. As already presented in Section 2.3, the global model workload contains 100,000 queries. The local model workload consists of three consecutive workloads where each workload has 5,000 queries more than the previous one. This corresponds to an increase of one local model per workload. That means, they have four workloads for our experiments: one for a global

ISSN- 2394-5125 VOL 8, ISSUE 02, 2021

model and three workloads for an increasing number of local models. Moreover, all experiments are conducted on an AMD A10- 7870K system with 32GB main-memory with PostgreSQL 10 as the underlying databasesystem.

In the experiments, the researchers measured the workload execution times, whereby we distinguish three different execution modes:

- Base Data w/oIndexes:

  ML workload is executed on the IMDB base data without any index on the base data.

- Base Data w/Indexes:

  ML workload is executed on the IMDB base data with indices on all columns in use.

- Grouping Set(GS):

  ML workload is executed on pre-aggregated data as determined by our approach.

The first two execution modes are our baselines because both a currently used in the presented ML model approaches for cardinality estimation. Moreover, the workloads are executed for all three different execution modes using the Python-PostgreSQL bridge.

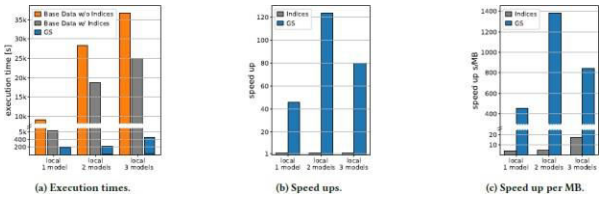


Fig. 10. Localal model evaluation results. [5]

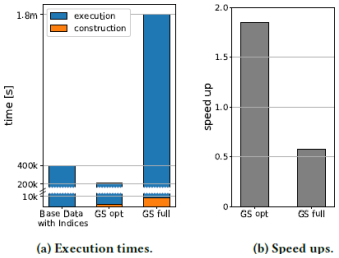
For both types of ML models for cardinality estimation, our aggregation-enabled training phase offers a unique database-centric way to reduce execution time.

Table 1 summarizes our evaluation results. The overhead introduced by the construction of a grouping set is much smaller than the savings in execution time. So, grouping sets reduce the workload execution times and amortize their own construction time. Overall, the benefits of grouping sets as a aggregation-enabled training phase for machine learning-based cardinality estimation can be shown. This is advantageous, especially for the local model approach. The simpler structure of the local model workloads is better supported by grouping sets because they contain fewer combinations of columns and fewer distinct values. These are exactly two of the assets for grouping sets identified in This leads to a significantly higher performance speed up for local model workloads than for global model workloads. Thus, we can afford a larger number of local models to reach the schema coverage of a global model. Even if these models request more queries than the global model, their benefits from the use of grouping sets outweigh the higher number of queries. This makes up for the major disadvantage of the local approach to have numerous models to be competitive to a globalmodel.

This case has a strong potential to be applied to the other similar machine learning problems, like plan cost modeling or indexing. It likens the potential to machine learning workloads for any of these machine learning problems in DBMS where information about the data in the DB is aggregated. These parallels make grouping sets and therefore DB support beneficial for ML for DBMS in general.

model	Base Data w/o Index	Base Data w/ Index	Construction GS	Execution GS	Total	Coverage GS
local 1	2*h* 29*m* 57*s*	1*h* 44*m* 01*s*	6.17*s*	191.34*s*	197.51*s*	5 000/5 000 = 100%
local 2	7*h* 52*m* 52*s*	5*h* 10*m* 27*s*	23.70*s*	205.91*s*	229.61*s*	10 000/10 000 = 100%
local 3	10*h* 12*m* 24*s*	6*h* 55*m* 55*s*	29.10*s*	430.36*s*	459.46*s*	15 000/15 000 = 100%
global full	–	4*d* 14*h* 11*m*	2*h* 22*m* 20*s*	20*d* 20*h* 02*m*	20*d* 22*h* 24*m*	100 000/100 000 = 100%
global opt	–	4*d* 14*h* 11*m*	34*m* 29*s*	2*d* 11*h* 03*m*	2*d* 11*h* 38*m*	54 722/100 000 = 55%

ISSN- 2394-5125 VOL 8, ISSUE 02, 2021



(a) Execution times. (b) Speed ups.

Fig. 11. Global model evaluation results. [5]

Table. 1: Execution times ML workloads (gs: grouping sets). [5]

D. APOLLO: Automatic Detection and Diagnosis of Performance Regressions in Database Systems[6]

The practical art of constructing database management systems (DBMSs) involves a morass of trade-offs among query execution speed, query optimization speed, standards compliance, feature parity, modularity, portability, and other goals. It is no surprise that DBMSs, like all complex software systems, contain bugs that can adversely affect their performance. The performance of DBMSs is an important metric as it determines how quickly an application can take in new information and use it to make new decisions. Both developers and users face challenges while dealing with performance regression bugs. First, developers usually find it challenging to manually design test cases to uncover performance regressions since DBMS components tend to have complex interactions. Second, users encountering performance regressions are often unable to report them, as the regression-triggering queries could be complex and database-dependent. Third, developers have to expend a lot of effort on localizing the root cause of the reported bugs, due to the system complexity and software development complexity. Given these challenges, this paper presents the design of APOLLO, a toolchain for automatically detecting, reporting, and diagnosing performance regressions in DBMSs. We demonstrate that APOLLO automates the generation of regression-triggering queries, simplifies the bug reporting process for users, and enables developers to quickly pinpoint the root cause of performance regressions. By automating the detection and diagnosis of performance regressions, APOLLO reduces the labor cost of developing efficientDBMSs.

We implement APOLLO with 3,054 lines of Python code and 156 lines of C++ code. We develop SQLFUZZ based on SQLSmith and Random Query Generator (RQG) We leverage DynamoRIO to collect execution traces in SQLDEBUG. During the fuzzing, we use the TPC-C benchmark in our fuzzing and corresponding evaluations. In particular, we use the queries as a corpus for bootstrapping SQLFUZZ and execute the queries on the tables contained in the benchmark. We configure the benchmark's scale factor to be one and 50 for fuzzing and validation, respectively.
Our evaluation aims to answer the following questions:

- Regression Detection: Is APOLLO effective at finding performance regressions in real-world DBMSs? How effective is SQLFUZZ at removing falsepositives?
- Query Reduction: Can SQLMIN outperform RAGS on reducing discovered queries? How effective are differentstrategies?
- RegressionDiagnosis:CanSQLDEBUGlocalizethe root cause of detected performanceregressions?

	Configuration	Default
Query Generator	Non_Deterministic	exclude
	Max_Query_Size	< 4000 bytes, < 4 joins
	Allow_DB_Update	do not modify DB
	Timeout	0.2 second per query
Query Executor	Threshold	150%
	DBMS	PostgreSQL and SQLite
	DB_Configuration	WORK_MEM(128MB), SHARED_MEM(128MB)
	Execute_Analyze	run every 1,000 Execs.
	Timeout	5 seconds per query
Bug Validator	Non_Executed	remove
	LIMIT_Clause	include when query

Table. 2: APOLLO Configuration [6]

ISSN- 2394-5125 VOL 8, ISSUE 02, 2021

- Query Patterns: Can the feedback improve the performance of fuzzing? Are there certain query patterns that are strongly correlated with performanceregressions?

Experimental Setup:
Evaluating APOLLO on two DBMSs:
SQLite (v3.23 and v3.27.2) in the client-server mode, and PostgreSQL (v9.5.0 and v11.1) in the embedded mode. We evaluate APOLLO on a server with Intel(R) Xeon(R) Gold 6140 CPU (32 processors) and 384 GB of RAM. We ran APOLLO on these systems for two months using the configuration shown in Table 2..

Fuzzing Performance:
On average, SQLFUZZ effectively produces and executes
5.8 queries per second. Although the query mutation is efficient, about 68% of the generated queries are discarded due to syntax or semantic errors. To improve the performance, we utilize multi-threaded fuzzing. Also, SQLMIN and SQLDEBUG can process one regression- triggering within 30 minutes.

	Feedback-driven	Random Testing
Regression queries	441 / 347,616 (0.126%)	243 / 340,416 (0.071%)
Regression rate (Avg.)	10.8×	10.2×

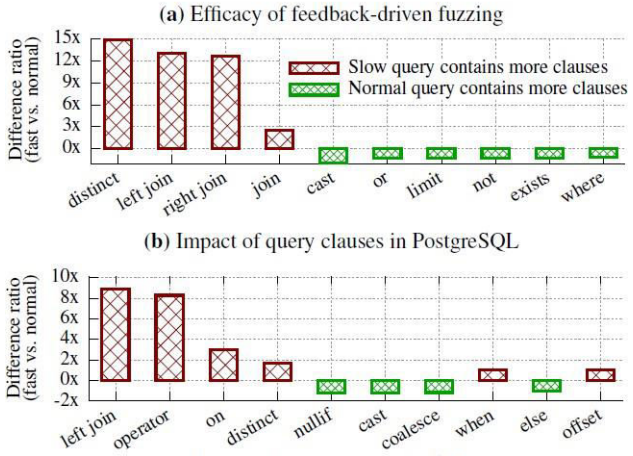


Fig. 12. Importance of feedback and query clauses. Table (a) shows impact of feedback-driven fuzzing. Figure (b) and (c) show importance of query clauses. Red and green-colored bars indicate that regression-triggering queries contain the associated clauses more and less frequently compared to normal queries, respectively. [6]

E. Intelligent Indexing—Boosting Performance in Database Applications by Recognizing Index Patterns [7]

An issue that most databases face is the static and manual character of indexing operations. This old-fashioned way of indexing database objects is proven to affect the database performance to some degree, creating downtime and a possible impact in the performance that is usually solved by manually running index rebuild or defrag operations. Many data mining algorithms can speed up by using appropriate index structures. Choosing the proper index largely depends on the type of query that the algorithm performs against the database. The statistical analyzers embedded in the Database Management System are neither always accurate enough to automatically determine when to use an index nor to change its inner structure. This paper provides an algorithm that targets those indexes that are causing performance issues on the databases and then performs an automatic operation (defrag, recreation, or modification) that can boost the overall performance of the Database System. The effectiveness of proposed algorithm has been evaluated with several experiments developed and show that this approach consistently leads to a better resulting index configuration. The downtime of having a damaged, fragmented, or inefficient index is reduced by increasing the chances for the optimizer to be using the proper indexstructure.

ANNs are, by definition, a non-linear classifier, which suits the nature of the distribution of our data. They have also shown the lowest testing errors when comparing with other algorithms such as K-NN, described in the testing section, that is the main reason why the ANN model is included in the proposed algorithms. The algorithm is applied to a simulated environment and compared with a real DBMS. The results have been extracted by randomizing six distributed data sets of hits to a given table. The study compares the use of this algorithm with the merging module versus the traditional algorithm embedded in most DBMS when advising the use of the creation or removal of a given index. The overall latency determines the efficiency of any data that is currently being accessed, which provides a straight forward insight. It compares the latency of accessing random searches in a table with the simulated environment where the same data is obtained by using our hybrid algorithm. Based on the dynamic

ISSN- 2394-5125 VOL 8, ISSUE 02, 2021

character of the index generation, the latency of the reads improves. However, the cost of having a learning algorithm embedded in the DBMS exists, and it is part of the system's overhead. The DBMS implementing this technique checks the structure of their indexes and modifying them if necessary, regularly. The simulation has been done by storing the information of the latencies and indexes linked to a set of queries in the standard DBMS combining all possible indexes. Then we have applied the outcome of the ANN model, forcing the DBMS to choose the indexes the ANN recommends for the same queries. We have ignored the index recommendations that are the same as the ones that the standard DBMS uses. Table 4 shows the different ones. The number of queries used in our study has been increasing withastepm=10%foreverynewrandomdistribution

(new row). In the initial distribution (row 1), 10,000 queries have been run. The queries are randomly accessing fragment (rows) of the tables of the case of study. The main literals
of the queries have been randomly chosen. The simulation automatically generates the ranges of last names and towns. That means that the sets of values of the tables are being randomly elected, offering guarantees of a more robust system.

DBMS Standard	DBMS Learning Algorithm (Simulation)
5.1121	4.2321
78.0523	56.9832
10.1232	17.1031
11.008	4.7751
13.1242	13.111
10.9877	5.2113
5.4332	6.2323

Table. 3: Comparative costs at different intervals of reads access [7]

Table 3 shows the average calculation of the overall costs obtained by the same queries in different batches. The table compares a standard DBMS index optimizer with our intelligent merging based technique. When running this comparative in the target databases, the database buffer pool and cache queries were disabled.

	TP	FP	TPR	FN	TN	TNR	AUCROC
ANN-1	30256	5467	0.899458	3382	32371	0.855515	0.932716
ANN-2	30125	5598	0.898690	3396	32357	0.852509	0.923334
K-NN	27512	8241	0.853770	4712	31041	0.790209	0.867185

Table. 4: Comparative sensitivity and specificity of the three models. Comparative costs at different intervals of reads access [7]

Table 4 compares the sensitivity, also called the true positive rate (TPR) and the specificity (true negative rate— TNR) of the three approaches. It displays the true positives (TP), the true negatives (TN), the false positive (FP), the false negative (FN) and the rates (TPR and TPN) along with the AUCROC, which represents represents TPR vs. False positive rate (FPR). In ML, performance measurement is an essential task. So when it comes to a classification problem, an Area Under The Curve Receiver Operating Characteristics (AUCROC) curve can be used to check or visualize the performance of the multi-class classification problem. It is one of the most important evaluation metrics for checking any classification model'sperformance. Figure 13, depicts the comparative results in terms of random scatter searches and by comparative the resulting trends (continuous comparative).

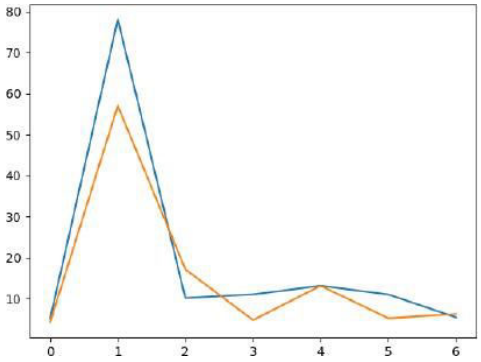


Fig. 13. Results in the target database using proposed model. Random search comparative trends.. [5]

ISSN- 2394-5125 VOL 8, ISSUE 02, 2021

F. Database Schema Matching Using MachineLearning with Feature Selection[9]

Schema matching, the problem of finding mappings between the attributes of two semantically related database schemas, is an important aspect of many database applications such as schema integration, data warehousing, and electronic commerce. Unfortunately, schema matching remains largely a manual, labor-intensive process. Furthermore, the effort required is typically linear in the number of schemas to be matched; the next pair of schemas to match is not any easier than the previous pair. In this paper we describe a system, called Automatch, that uses machine learning techniques to automate schema matching. Based primarily on Bayesian learning, the system acquires probabilistic knowledge from examples that have been provided by domain experts. This knowledge is stored in a knowledge base called the attribute dictionary. When presented with a pair of new schemas that need to be matched (and their corresponding database instances), Automatch uses the attribute dictionary to find an optimal matching. The paper also reports initial results from the Automatch project.

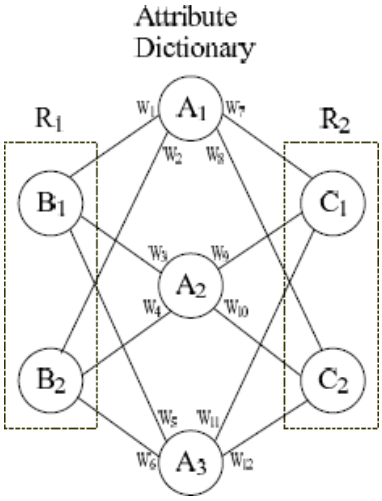


Fig. 14. Weighted tripartite graph for representing individual attribute scores.[9]

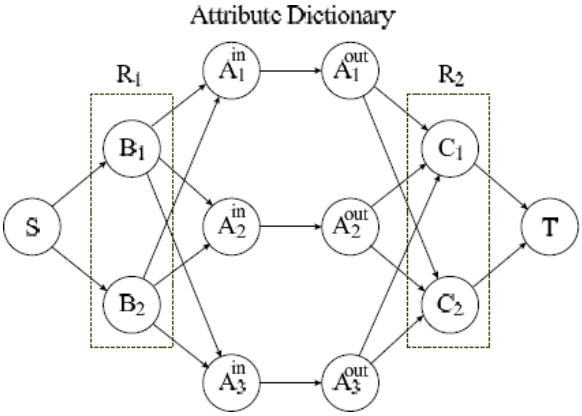


Fig. 15. Minimum-Cost-Maximum-Flow graph for finding optimal schema matching [9]

First, we measured the performance of Automatch without any attempt at optimizing the dictionary through feature selection; that is, we use the Bayesian approach to score matches and the flow graph approach to optimize matches. Using cross validation, we achieved a performance of 66% (measured as the harmonic mean of soundness and completeness). In a separate experiment, we used random guessing to match the same schemas and achieved a performance of 10%. Next, we compared the three feature selection strategies and assessed their impact on schema matching. Figure 16 shows the performance for schema matching for each of the feature selection strategies. The x- axis is the percentage of low-scoring features that have been discarded, and the y-axis is the performance, measured as the harmonic mean of soundness and completeness. The leftmost point in the graph corresponds to our first experiment with no feature selection. Initially, with 5% feature reduction, all the feature selection strategies improve performance by at least 6%. The strategies then perform comparably up to 60% reduction. At levels of reduction over 80%, IG and LR continue to produce improved matching performance (relative to no feature selection) while MI falls below performance with no feature selection. All three feature selection strategies improve performance when compared to the initial performance with no feature selection (though the level to which they sustain this improvement varies). This observation indicates that all of these approaches are acceptable for

ISSN- 2394-5125 VOL 8, ISSUE 02, 2021

reducing the feature space. Furthermore, if we are seeking the most ambitious reduction in the feature space, LR is preferable to IG which is preferable toMI.

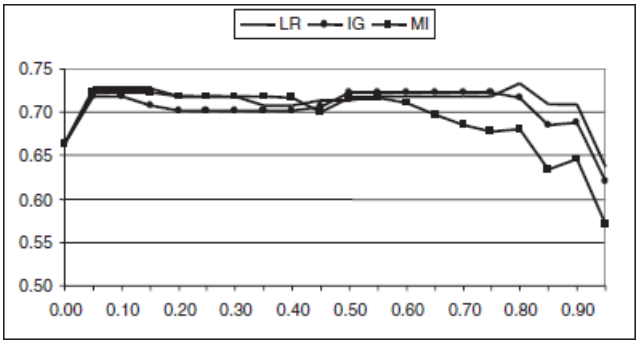


Fig. 16. Harmonic mean of soundness and completeness (y-axis) as the feature set is reduced in increments of 5 percent (x-axis) [9]

Significant findings and contributions of this paper were:

The Automatch system is a new and viable approach to eliminate the schema-matching bottleneck present in modern database applications. The results are encouraging as they show performance that exceeds 70%(measured as the harmonic mean of the soundness and the completeness of the attribute matchingprocess).

Statistical feature selection can be used to improve the performance of Automatch.

The improvement is in three areas:

1. in the storage requirements for the auxiliary knowledgebase,
2. in the computational costs of the matching algorithm
3. in the quality (soundness and completeness) of the results.

It is estimated that statistical feature selection can be used to improve the performance of other automated schema- matching approaches that must deal with high-dimensional feature spaces.

Statistical feature selection incurs little overhead in Automatch since we are using a probabilistic learning approach. Learning after feature selection consists simply of normalizing the probabilities of the remaining features.
In contrast, other machine learning approaches (e.g. neural networks, rule learners, etc.) must execute their respective learning algorithms after feature selection is completed.

IV. CONCLUSION

Afterthethoroughdiscussionoftheselectedcasestudies, we can conclude that in-database ML can be implemented in various aspects to either conceive a solution to a pre-existing problem posed by the traditional data science workflow or to gainasignificantboostinperformance.Thispaperservesthe purpose of bringing the readers up-to date regarding the various advancements made in this field. This in turn shall motivatethemtoimplementthesameintheirworkflows.

V. REFERENCES

[1] Günther, M., Thiele, M., & Lehner, W. (2019). RETRO: Relation Retrofitting For In-Database Machine Learning on Textual Data. arXiv preprintarXiv:1911.12674.

[2] Ma, L., Ding, B., Das, S., & Swaminathan, A. (2020, June). Active Learning for ML Enhanced Database Systems. In Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data (pp.175-191).

[3] Sandha, S. S., Cabrera, W., Al-Kateb, M., Nair, S., & Srivastava, M. (2019). In-database distributed machine learning: demonstration using Teradata SQL engine. Proceedings of the VLDB Endowment, 12(12), 1854-1857.

[4] Kara, K. (2020). Specialized Hardware Solutions for In-Database Analytics and Machine Learning (Doctoral dissertation, ETHZurich).

[5] Woltmann, L., Hartmann, C., Habich, D., & Lehner, W. (2020). Machine Learning-based Cardinality Estimation in DBMS on Pre- Aggregated Data. arXiv preprintarXiv:2005.09367.

[6] Jung, J., Hu, H., Arulraj, J., Kim, T., & Kang, W. (2019). APOLLO: automatic detection and diagnosis of performance regressions in database systems. Proceedings of the VLDB Endowment, 13(1), 57- 70.

[7] Arteta Albert, A., Gómez Blas, N., & Mingo López, L. F. D. (2020). Intelligent Indexing—Boosting Performance in Database Applications by Recognizing Index Patterns. Electronics, 9(9),1348.

[8] AboKhamis,M.,Ngo,H.Q.,Nguyen,X.,Olteanu,D.,&Schleich, M. (2018, May). In-database learning with sparse tensors. In Proceedings of the 37th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems (pp.325-340).

[9] Berlin, J., &Motro, A. (2002, May). Database schema matching using machine learning with feature selection. In International Conference on Advanced Information Systems Engineering (pp. 452- 466). Springer, Berlin,Heidelberg.

[10] https://www.vertica.com/blog/doing-in-database-machine-learning-1-why-would-you-do-that/

[11] https://www.datanami.com/2019/04/01/you-cant-do-machine-learning-inside-a-database-can-you/

[12] https://en.wikipedia.org/wiki/Database

[13] https://analyticsindiamag.com/top-databases-used-in-machine- learning-projects/

[14] Alghunaim, A. (2015). A vector space approach for aspect-based sentiment analysis (Doctoral dissertation, Massachusetts Institute of Technology).

[15] Zhou, X., Wan, X., & Xiao, J. (2015, February). Representation learning for aspect category detection in online reviews. In Twenty- Ninth AAAI Conference on ArtificialIntelligence.

[16] Faruqui, M., Dodge, J., Jauhar, S. K., Dyer, C., Hovy, E., & Smith,N. A. (2014). Retrofitting word vectors to semantic lexicons. arXiv preprintarXiv:1411.4166.

[17] Rodeh, O. (2008). B-trees, shadowing, and clones. ACM Transactions on Storage (TOS), 3(4),1-27.

[18] http://lcm.csa.iisc.ernet.in/dsa/node122.html

[19] https://www.tutorialcup.com/dbms/shadow-paging.htm#:~:text=This%20is%20the%20method%20where,be%20r eflected%20in%20the%20database.

[20] https://www.sciencedirect.com/topics/computer-science/fuzzing

[21] Lu, H.; Ng, Y.Y.; Tian, Z. T-tree or B-tree: main memory database index structure revisited. In Proceedings of the 11th Australasian Database Conference. ADC 2000 (Cat. No.PR00528), Canberra, Australia, 31 January–3 February 2000; pp.65–73.

[22] Xie, Z.; Cai, Q.; Chen, G.; Mao, R.; Zhang, M. A Comprehensive Performance Evaluation of Modern In-Memory Indices. In Proceedings of the 2018 IEEE 34th International Conference on Data Engineering (ICDE), Paris, France, 16–19 April 2018; pp.641–652.

[23] Kiranyaz, S.; Gabbouj, M. Hierarchical Cellular Tree: An Efficient Indexing Scheme for Content-Based Retrieval on Multimedia Databases. IEEE Trans. Multimed. 2007, 9,102–119.

[24] Kvet, M.; Matiasko, K. Impact of Index Structures on Temporal Database Performance. In 2016 European Modelling Symposium (EMS); IEEE Computer Society: Los Alamitos, CA, USA, 2016; pp. 3–9.

[25] OSS-Fuzz: Continuous Fuzzing for Open Source Software, 2016. https://github.com/google/oss-fuzz.

[26] C. Holler, K. Herzig, and A. Zeller. Fuzzing with Code Fragments. In SECURITY, pages 445–458,2012.

[27] J. Wang, P. Zhang, L. Zhang, H. Zhu, and X. Ye. A Model-based Fuzzing Approach for DBMS. In CHINACOM, pages 426–431. IEEE, 2013.

[28] I. Yun, S. Lee, M. Xu, Y. Jang, and T. Kim. QSYM: A Practical Concolic Execution Engine Tailored for Hybrid Fuzzing. In Proceedings of the 27th USENIX Security Symposium (Security), Baltimore, MD, Aug.2018.

[29] Christopher Bishop. 2006. Pattern Recognition and Machine Learning. Pattern Recognition and Machine Learning(2006).

[30] Thiago NC Cardoso, Rodrigo M Silva, SérgioCanuto, Mirella M Moro, and Marcos A Gonçalves. 2017. Ranked batch-mode active learning. Information Sciences 379 (2017),313–337.

[31] Bailu Ding, Sudipto Das, Ryan Marcus,WentaoWu, Surajit Chaudhuri, and Vivek R Narasayya. 2019. Ai meets ai: Leveraging query executions to improve index recommendations. In Proceedings of the 2019 International Conference on Management of Data. 1241– 1258.

[32] https://www.geeksforgeeks.org/indexing-in-databases-set-1/#:~:text=Indexing%20is%20a%20way%20to,using%20a%20few% 20database%20columns.

[33] Thorsten Joachims. 1999. Transductive Inference for Text ClassificationusingSupportVectorMachines.InProceedingsofthe Sixteenth International Conference on Machine Learning. Morgan Kaufmann Publishers Inc., 200–209.

[34] F. Biessmann, D. Salinas, S. Schelter, P. Schmidt, and D. Lange. Deep Learning for Missing Value Imputation in Tables with Non- Numerical Data. In Proceedings of the 27th ACMInternational Conference on Information and Knowledge Management, pages 2017–2025. ACM,2018.

[35] T. Iwata and N. Ueda. Unsupervised Object Matching for Relational Data. arXiv preprint arXiv:1810.03770,2018.

[36] S. Jastrzebski, D. Lesniak, andW. M. Czarnecki. How to evaluate word embeddings? On importance of data efficiency and simple supervised tasks. CoRR,abs/1702.02170,2017.

[37] K. Kara, K. Eguro, C. Zhang, and G. Alonso. ColumnML: Column- Store Machine Learning with On-The-Fly Data Transformation. Proceedings of the VLDB Endowment, 12:348–361, 122018.

[38] D. Kiela, F. Hill, and S. Clark. Specializing Word Embeddings for Similarity or Relatedness. In Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 2044– 2048,2015.

[39] T. Kilias, A. Löser, F. A. Gers, R. Koopmanschap, Y. Zhang, and M. Kersten. IDEL: In-Database Entity Linking with Neural Embeddings. arXiv preprint arXiv:1803.04884,2018.

[40] T. Kraska, A. Talwalkar, and J. Duchi. MLbase: A Distributed Machine-learning System. In In CIDR,2013.

[41] A. Lenci. Distributional Models of Word Meaning. Annual review of Linguistics, 4:151–171,2018.

[42] B. Lengerich, A. Maas, and C. Potts. Retrofitting Distributional Embeddings to Knowledge Graphs with Functional Relations. In Proceedings of the 27th International Conference on Computational Linguistics, pages 2423–2436, Santa Fe, New Mexico, USA, Aug. 2018. Association for ComputationalLinguistics.

[43] D. Liben-Nowell and J. Kleinberg. The Link-Prediction Problem for Social Networks. Journal of the American society for information science and

technology, 58(7):1019–1031,2007.

[44] T. Kraska, A. Talwalkar, and J. Duchi. MLbase: A Distributed Machine-learning System. In In CIDR,2013.

[45] Jayant Madhavan, Philip A. Bernstein, and Erhard Rahm. Generic schema matching with cupid. In Proceedings of the 27th International Conferences on Very Large Databases, pages 49–58, 2001.453

[46] Ren´ee Miller, Laura Haas, and Mauricio Hern´andez. Schema mapping as query discovery. In Proceedings of the 26th International Conferences on Very Large Databases, pages 77–88, 2000.452

[47] Tom Mitchell. Machine Learning. McGraw-Hill, 1997. 456,460

[48] Judea Pearl. Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference. Morgan Kaufmann, 1988.461

[49] Erhard Rahm and Philip Bernstein. On matching schemas automatically. Technical Report MSR-TR-2001-17,Microsoft, Redmond, WA, February 2001. 452, 453

[50] Mehran Sahami, Susan Dumais, David Heckerman, and Eric Horvitz. A Bayesian approach to filtering junk e-mail. AAAI-98 Workshop on Learning for Text Categorization, 1998.460

[51] Ian H. Witten and Eibe Frank. Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations. Morgan Kaufmann, 2000.462

[52] https://docs.microsoft.com/en-us/sql/relational- databases/performance/cardinality-estimation-sql-server?view=sql- server-ver15

[53] Graubart, R., & Duffy, K.J. (1985). Design Overview for Retrofitting Integrity-Lock Architecture onto a Commercial DBMS. 1985 IEEE Symposium on Security and Privacy,147-147.

[54] https://en.wikipedia.org/wiki/B%2B_tree

[55] Faerber, F., Kemper, A., Larson, P. Å., Levandoski, J., Neumann, T., &Pavlo, A. (2017). Main memory database systems. NowPublishers.

[56] A. Lenci. Distributional Models of Word Meaning. Annual review of Linguistics,4:151–171,