Can Large Language Models Predict Data Correlations from Column Names?

Immanuel Trummer
Cornell Database Group
Ithaca, NY, USA
itrummer@cornell.edu

ABSTRACT

Recent publications suggest using natural language analysis on database schema elements to guide tuning and profiling efforts. The underlying hypothesis is that state-of-the-art language processing methods, so-called language models, are able to extract information on data properties from schema text.

This paper examines that hypothesis in the context of data correlation analysis: is it possible to find column pairs with correlated data by analyzing their names via language models? First, the paper introduces a novel benchmark for data correlation analysis, created by analyzing thousands of Kaggle data sets (and available for download). Second, it uses that data to study the ability of language models to predict correlation, based on column names. The analysis covers different language models, various correlation metrics, and a multitude of accuracy metrics. It pinpoints factors that contribute to successful predictions, such as the length of column names as well as the ratio of words. Finally, the study analyzes the impact of column types on prediction performance. The results show that schema text can be a useful source of information and inform future research efforts, targeted at NLP-enhanced database tuning and data profiling.

PVLDB Reference Format:
Immanuel Trummer. Can Large Language Models Predict Data Correlations from Column Names?. PVLDB, 16(13): 4310 - 4323, 2023.
doi:10.14778/3625054.3625066

PVLDB Artifact Availability:
The source code, data, and/or other artifacts have been made available at https://github.com/itrummer/DataCorrelationPredictionWithNLP.

1 INTRODUCTION

Consider a table named "cars" with columns named "maker" and "model". Most people would assume, based on column names and commonsense knowledge, that maker and model columns are correlated (i.e., knowing the maker will restrict options for the model). Such reasoning is possible if column names are meaningful. Assigning meaningful column names is good practice, but of course there are rare exceptions which we are not concerned with here. In this paper, we study the question of whether automated tuning tools could apply a similar kind of reasoning, exploiting recent innovations in the domain of natural language analysis (NLP): pre-trained language models [10].

This research question is motivated by my recent work [48, 52], suggesting to use NLP on database schema elements to inform database tuning, in particular, to help prioritizing data profiling operations. The underlying hypothesis behind those suggestions, namely, whether language models are able to infer relevant information with sufficiently high reliability, has not been investigated in detail. This paper closes that gap, focusing on extracting information about data correlations.

Detecting correlations in data has been a topic of significant interest in the database research community [5, 16]. Knowing data correlation is useful in many scenarios. For instance, query optimizers [43] (as well as other tuning tools) often depend on accurate predictions of intermediate result sizes. Classical prediction models assume uncorrelated data, thereby being misled in practice [24]. As pointed out in prior work [16], knowing about correlations can help to correct cardinality estimates. Alternatively, knowing about possible data correlations can help to prune options with correlation-related uncertainty from the search space (e.g., the optimizer can favor join orders where intermediate result sizes do not depend on columns that are likely correlated).

Detecting data correlations requires comparing data in different columns, often making correlation detection more expensive than operations that focus on different columns in separation. This has motivated dedicated research on algorithms that make correlation detection more efficient [5, 16]. Typically, those prior algorithms do not exploit information gained via analysis of the database schema, using language models. However, as suggested in my prior work [48, 52], such analysis could be helpful in order to better allocate and prioritize profiling efforts. For instance, given a limited profiling budget, the analysis scope could be restricted to column subsets that are more likely to be correlated, based on the results of NLP. Within those column subsets, any of the existing algorithms for correlation detection could be used. This assumes, however, that NLP is indeed useful to extract relevant information form the database schema. Whether or not that is actually the case, is the subject of the current study.

The hope of extracting useful information from database schema names alone is filled by recent advances in the field of natural language processing. Primarily, those advances are due to two key developments: a novel neural network architecture, the so-called Transformer [55], as well as new training methods that exploit large amounts of unlabeled training data [40]. Among other advantages, Transformer models enable efficient training of large neural network models with hundreds of millions [10] to hundreds of

Proceedings of the VLDB Endowment, Vol. 16, No. 13 ISSN 2150-8097.
doi:10.14778/3625054.3625066

billions [8, 11] of trainable parameters. Generating task-specific training data at sufficiently large scale is often prohibitively expensive. Fortunately, it is typically possible to reduce the required amount of task-specific training significantly by a pre-training stage that uses large amounts of unlabeled data (e.g., Web text) [13]. This study evaluates pre-trained Transformer models, fine-tuned with a moderate amount of training data that is specific to the task of correlation detection. Whereas large Transformer models with hundreds of billions of parameters are nowadays available, typically hosted remotely by providers such as OpenAI [11], this study focuses on much smaller models (with parameter counts in the hundreds of millions "only") that can be run with moderate overheads on commodity machines. This seems reasonable as overheads due to using large language models may otherwise eclipse data profiling overheads altogether.

This study is based on a newly generated benchmark for data correlation detection. Prior benchmarks of algorithms for correlation detection typically use a small number of data sets [16]. This is reasonable, as long as performance depends on data properties but not on data semantics. When analyzing column names via language models, however, the data domain may have significant impact on prediction performance (e.g., benefiting application domains that appear more frequently in the pre-training data). Hence, to evaluate language models under realistic conditions, this study uses a benchmark generated from around 4,000 tabular data sets, downloaded from the Kaggle platform. For those data sets, the benchmark analyzes correlation between column pairs according to multiple popular correlation metrics, namely Pearson correlation [56], Spearman's correlation coefficient [4], and Theil's U [37]. While this data is useful to test the primary hypothesis evaluated in this paper, i.e. that relevant information can be extracted from schema elements via language models, it can also be used to test NLP-enhanced data profiling approaches. We will see one example of that in Section **??**.

In summary, the original scientific contributions in this experimental paper are the following.

- The paper introduces a new benchmark, useful to test correlation prediction, based on column names, and to evaluate approaches for NLP-enhanced database tuning.
- The paper tests the ability of language models to infer information on data correlation from column names, considering different correlation metrics, scenarios, and models.
- The paper evaluates a simple baseline algorithm for efficient correlation prediction, exploiting information gained via natural language analysis.

The remainder of this paper is organized as follows. Section 2 provides background on the techniques used throughout the paper and discusses related work. Section 3 describes the generation of the benchmark, used to evaluate correlation detection methods. Next, Section 4 analyzes the benchmark data set, in terms of data statistics and correlation properties. Section 5 compares different methods for predicting data correlations from column names, including pre-trained models and simpler baselines. Section 6 studies the impact of several scenario properties, including the amount and quality of training data, to study the impact on prediction performance. Section 7 analyzes prediction performance for different data subsets separately, breaking down, for instance, by column name length among other properties. Section 8 considers different correlation metrics, thereby obtaining insights into how well the prior findings generalize. Finally, Section 9 evaluates the impact of column types on prediction performance.

2 BACKGROUND AND RELATED WORK

This section discusses prior work, related to this study. Section 2.1 discusses prior work on data profiling, a primary application domain for the approaches evaluated in this paper. Section 2.2 discusses, more specifically, prior work on data correlation analysis. Section 2.3 discusses the technology that this study is based upon: pre-trained language models. Finally, Section 2.4 discusses prior work applying such or similar technology in the context of data management.

2.1 Data Profiling

The goal of data profiling is to generate statistics and meta-data about a given data set [31]. Specialized tools have been developed for data profiling, including systems from industry [15, 17] as well as academia [5, 16, 32, 36]. Typically, users specify a target data set for profiling as well as specific types of meta-data to consider. Data profiling is expensive and may have to be repeated periodically as the data changes. Hence, profiling tools often allow users to restrict profiling overheads, e.g. by setting time limits [32, 45].

Profiling methods have been proposed for mining different kinds of meta-data, ranging from statistics over single columns [9] to more expensive operations such as unique column combination discovery [1, 34], detecting inclusion dependencies [33], foreign keys [39], order dependencies [18, 23], or statistical data correlations [5, 16], the focus of this study.

2.2 Detecting Correlations

The fact that data correlations are important has motivated work aimed at finding correlations in data sets [5, 16]. To guide profiling efforts, such tools typically analyze data samples. The sample size is often chosen as a function of total data size. In contrast, the time for predicting correlation based on column names does not depend on the data size. Significant work has been dedicated to the problem of selectivity estimation with correlations [6, 29, 54]. Here, correlations play an important role in estimating aggregate selectivity of predicate groups. More recently, machine learning has been proposed as a method to solve various types of tuning problems in the context of databases [28, 35, 53, 57, 59]. Correlated data is a primary reason to replace more traditional cost models, often based on the independence assumption, via learned models. This stream of work connects to this study as it applies machine learning for predicting correlations. However, this study uses machine learning in the form of NLP-based analysis of database schema elements.

2.3 Language Models

Pre-trained language models, based on the Transformer architecture [55], have recently led to significant advances on a multitude of NLP tasks [58]. Pre-trained language models are based on the idea of "transfer learning". For many specialized NLP tasks, it is difficult to accumulate a large enough body of training data. Also, overheads related to the training of large neural networks from scratch can be significant. This motivates a pre-training step, training a Transformer model on an NLP task for which training data is in ample supply. For instance, this includes the "masked language modelling" task [10]. Here, the goal is to predict masked words in a sentence. Doing so requires many capabilities that are useful for other NLP tasks as well. Furthermore, any written text can be used as training data for the latter task. After pre-training, the resulting network (with associated weights) can be specialized ("fine-tuning") to another task. This refinement step requires only moderate computational resources and few training samples [13], compared to the initial training. Nowadays, pre-trained language models [22, 27, 42] achieve state of the art performance over a wide range of NLP tasks. While the largest models have nowadays hundreds of billions of trainable parameters [11], this study focuses on smaller models that run on today's commodity machines, making them practical to guide data profiling operations with moderate overheads.

2.4 NLP for Databases

There have been significant efforts to leverage NLP techniques for database systems [51]. Typically, the focus of those efforts is the query interface. Here, a long standing goal in database research is to enable natural language query interfaces [2, 19, 20, 25, 26, 41, 44]. Sequence-to-sequence models have been successfully used for this task over the past years [14, 60, 62]. Recently, pre-trained language models, based on the Transformer architecture [55], have achieved excellent results on text-to-SQL benchmarks such as Spider [61] or WikiSQL [62]. They form the basis for this study as well. Other applications of language models in the context of databases include data discovery and integration [21, 30] as well as data preparation tasks [46].

This study connects to prior work exploiting language models to support the database backend. For instance, this includes work leveraging such models to write code for data processing or process data directly [3, 47, 49] or to parse technical documentation to support automated database tuning [50]. More specifically, this study is motivated by my prior work suggesting the use of language models on database schema elements to support database tuning and data profiling [48, 52].

3 BENCHMARK

This section describes a benchmark for correlation prediction and NLP-enhanced database tuning, created specifically for the purpose of this study. This benchmark covers a wide variety of data sets from different domains. This ensures that the results on prediction performance are representative.

3.1 Benchmark Data

The benchmark uses data sets from the Kaggle Web site[1] (using a corresponding API[2]). The choice of Kaggle data is motivated by the large number and diversity of data sets, available on that platform. At the same time, Kaggle data is used for analysis by many data scientists[3]. Any approach that works well on Kaggle data is likely to benefit a large number of users. As a potential drawback, since Kaggle data is often discussed on the Web, it is possible that data used for pre-training language models contains references to Kaggle data. However, determining data correlation (the prediction target of this benchmark) requires additional data analysis, beyond mere access to data. To the best of my knowledge, no large-scale correlation analysis with similar correlation metrics has been conducted on Kaggle data, prior to the time period during which the models used for experiments were pre-trained. This makes it unlikely that pre-training data contains relevant information on Kaggle data correlation.

Data sets are obtained by querying the Kaggle API for data sets with the following filters. First, data sets are filtered based on their format, retrieving data sets in ".csv" format (i.e., tabular data). Second, to enable retrieval and analysis of a large number of data sets, covering various domains, a size limit of one megabyte was used. The benchmark integrates several thousand data sets, taken from the result of this retrieval query (in the order in which they are returned by the Kaggle API). For those data sets, the benchmark contains various correlation metrics for column pairs within the same table (obtained by analyzing the corresponding data). Prior work reports that considering correlation between column pairs, as opposed to correlation between more than two columns, "can remove most of the correlation-induced selectivity estimation errors" [16]. At the same time, the number of possible correlations grows exponentially in the number of correlated columns, making it more expensive to search for multi-column correlation. Hence, the benchmark focuses on discovering correlation between column pairs. For each table, only up to 100 column pairs are analyzed. More precisely, for tables with more than ten columns, the first ten columns are selected for the benchmark (thereby enabling 100 pairs). The benchmark contains each column pair only once. More precisely, for any given columns c_1 and c_2, the benchmark contains only one of $\langle c_1, c_2 \rangle$ or $\langle c_2, c_1 \rangle$ (but not both). While some of the correlation metrics we consider are not symmetric, this avoids using training samples that are too similar (thereby, potentially, leading to overly optimistic prediction performance results). For each column pair, the benchmark contains column names and different correlation metrics. The correlation analysis was executed using Python 3 and SciPy's stats package[4].

The benchmark measures correlation according to different metrics. First, it contains results for the Pearson correlation coefficient [56]. This coefficient is a measure of linear correlation between two data sets. The coefficient itself, denoted as R, is contained in the interval $[-1, 1]$. It comes with a p-value, indicating the probability of obtaining a specific R value by chance. The following experiments define correlation via different thresholds for $|R|$ while typically using a threshold of 5% for the p-Value.

Beyond Pearson correlation, measuring linear dependencies, the benchmark also considers Spearman's correlation coefficient [12]. This coefficient, typically denoted as ρ, measures how well the relationship between two data sets can be characterized by a monotonic (but not necessarily linear) function. Again, the coefficient takes values from the interval $[-1, 1]$. It comes with a p-Value, indicating the probability of observing given correlations for uncorrelated data. The following experiments consider different thresholds on

[1] www.kaggle.com
[2] https://github.com/Kaggle/kaggle-api
[3] https://www.kaggle.com/code/carlmcbrideellis/kaggle-in-numbers
[4] https://docs.scipy.org/doc/scipy/reference/stats.html

ρ while typically requiring a p-value of 5% or less (to qualify as statistically significant correlation).

The two aforementioned coefficients, Pearson's and Spearman's coefficient, apply to numerical columns. For categorical columns (in addition to numerical ones), the benchmark measures the entropy coefficient [38], also called Theil's U, instead. This coefficient is a normalized version of the mutual information between two variables. Intuitively, it measures how many bits we can predict for one column, given the value in the other one. The following experiments vary the threshold on Theil's U, starting from which we consider two columns correlated.

For all coefficients, the benchmark models correlation prediction as a binary classification problem (classifying column pairs as correlated or uncorrelated, based on the column names). E.g., as an alternative, it is also possible to formulate correlation prediction as a regression problem, aiming to predict correlation metrics such as p-Values or raw coefficient values. Arguably, this is a more challenging problem, as a perfect predictor for the regression variant yields a perfect predictor for the classification version but not vice-versa. At the same time, the classification variant has practical applications, e.g., to select or prioritize column pairs for profiling. Section **??** of the extended technical reports results for a proof-of-concept system, focusing on that use case. For those reasons, this paper focuses on the classification variant and leaves the regression version for future work.

The result of data preparation is a benchmark, containing the names of columns pairs, meta-data such as the column data type, as well as correlation results according to different correlation metrics.

3.2 Benchmark Metrics

The following experiments use the benchmark data for two types of experiments. First, the experiments evaluate the ability of language models to predict data correlation from column names. Second, the experiments evaluate a simple algorithm for NLP-enhanced data profiling, exploiting predictions on data correlation to prioritize data profiling steps.

To measure the ability of language models to predict data correlation, the experiments measure prediction quality according to multiple metrics. More precisely, the experiments consider five metrics of prediction quality: recall, precision, and the F1 score (which combines recall and precision). Here, recall is the percentage of correlated column pairs that were accurately identified. Precision is the ratio of actual column pairs among the ones predicted to be correlated. The F1 score is defined as $2 \cdot p \cdot r/(p+r)$ (where p and r are precision and recall, respectively). The aforementioned three metrics are typically used in scenarios where a relatively sparse class of elements should be identified. Strongly correlated column pairs qualify as they tend to be relatively sparse (as shown in the next section).

The experiments also measure Matthew's Correlation Coefficient (MCC) [7] and simple prediction accuracy (considering the two classes "correlated" and "uncorrelated" for each column pair). All of the aforementioned quality metrics yield values from the interval $[0, 1]$ and higher values represent better quality. The following plots report all five metrics. When verifying hypotheses about prediction quality, we consider a hypotheses as validated, if it is validated according to all of those five metrics.

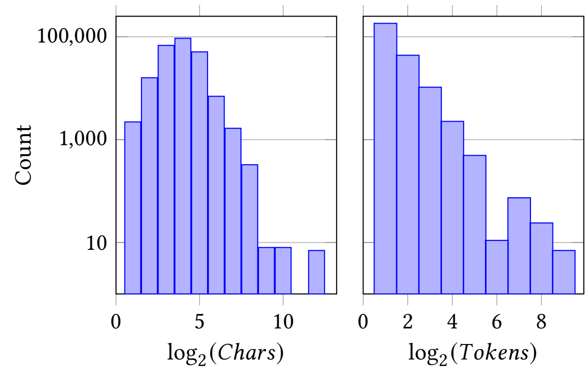


Figure 1: Distribution of column name length, measured as the number of characters (left) and number of tokens (right).

The performance of NLP-enhanced data profiling tools is evaluated by the number of correlated column pairs, verified within a budget on computation overheads. This budget can be measured, for instance, by the number of column pairs analyzed (the benchmark contains time measurements for correlation analysis as well, enabling experiments with budgets on computation time). Section **??** contains more details on this scenario.

4 BENCHMARK ANALYSIS

This section analyzes the benchmark, introduced in the previous section. Table 1 shows an extract from this data set. The upper part of the table shows five highly correlated columns, measured via the Pearson coefficient, the lower half shows five column pairs with low correlation. At the same time, it shows column names and the names of the associated data sets.

For the examples in the table, it seems often possible, using commonsense knowledge, to identify likely candidates for correlations. For instance, the number of points for a team in a table with sports statistics often correlates with the number of wins (more points often lead to more wins). Indeed, the corresponding column pair shows relatively high correlation that is statistically significant (using the common thresholds of 5% to separate statistically significant from non-significant p-Values). On the other side, there is no obvious indication of any correlation between two columns named "Williamsburg Bridge" and "Unnamed: 0". Indeed, the corresponding column pair shows only weak correlation. This paper studies the question whether language models are able to simulate such reasoning.

Table 2 summarizes size-related statistics, describing the benchmark data. Altogether, the benchmark contains correlations from about 4,000 data sets. Those data sets derive from various sources and cover various topics (the examples in Table 1 give a first impression of their diversity). For those data sets, the benchmark contains results about 120K column pairs, about half of them of numerical (or integer) type. In average, the source data sets contain over 100K rows and each column contains over 6K unique values.

Table 1: Examples for strongly correlated (top) and less correlated (bottom) columns according to Pearson's coefficient.

Data Set	Column 1	Column 2	R value	p value
epl1920leaguetable.csv	Points	Wins	99%	0%
emission data.csv	1751	1752	100%	0%
india-districts-census-2011.csv	Female_Literate	Male_Literate	98%	0%
Google_Stock_Price_Test.csv	High	Open	96%	0%
housing.csv	total_bedrooms	total_rooms	93%	0%
time_series_2019-ncov-Confirmed.csv	1/24/20 0:00	Lat	-6%	83%
diabetes_merged_date-time-sorted-includes-patient-id.csv	code	patient_id	2%	0%
Heart.csv	RestECG	Sex	2%	71%
2020.12.09/2020.12.09.csv	num_pkts_out	dest_ip	-5%	0%
nyc-east-river-bicycle-counts.csv	Williamsburg Bridge	Unnamed: 0	10%	16%

Table 2: Statistics on benchmark data sets.

Property	Value
Number of data sets	3,952
Number of column pairs	119,384
Number of numerical column pairs	59,449
Number of rows (Avg.)	103,126
Number of distinct values per column (Avg.)	6,200

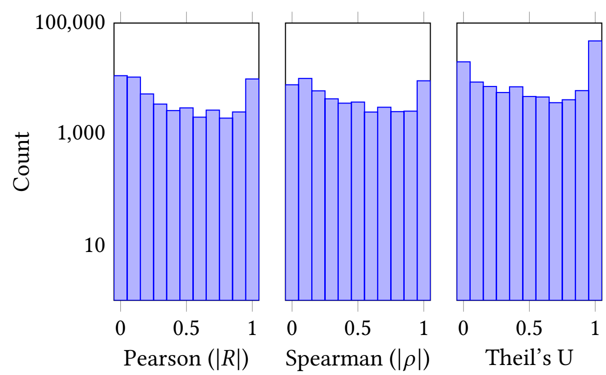


Figure 2: Distribution of correlation coefficient values according to different metrics.

The ability to predict likely correlation from column names may depend on features such as the column name length. Intuitively, having longer column names should be more informative. Potentially, this makes correlation prediction easier. Figure 1 shows histograms summarizing the distribution of column name length, measured according to different metrics. The left plot shows the distribution over character length. The right plot measures the number of tokens (i.e., text snippets separated by spaces or underscores) in column pairs. Note the logarithmic x-axis. The average column name length is around 16 characters. This is sufficient for few, short words. The number of tokens is typically limited to two (i.e., one word in each of the columns).

The following experiments vary the threshold, starting from which columns are considered correlated. This makes it interesting to analyze how correlation is distributed over column pairs. Figure 2 shows histograms, characterizing the corresponding distributions. From left to right, it shows correlation according to Pearson's coefficient, Spearman's coefficient, and according to Theil's U. In particular for Pearson's and Spearman's coefficients, low values are more likely than higher ones. For all correlation metrics, we see a slightly bimodal distribution with increased probability for maxima and minima.

5 COMPARING PREDICTION METHODS

This section compares different prediction methods in terms of their training time (if any) and output quality. Most of them use pre-trained language models, based on the recently proposed Transformer architecture [55]. Such models achieve state of the art performance in a variety of NLP tasks [13].

5.1 Description of Methods

The experiments consider three pre-trained models, small enough to be used locally on today's commodity machines. All of them are encoder models, pre-trained to associate input tokens with high-dimensional vectors. For classification problems, a thin layer is added that maps vectors to scores for the relevant classes. Roberta [27] (short for "Robustly Optimized BERT approach") expands BERT [10], another pre-trained language model that has achieved widespread popularity. Compared to BERT, Roberta is pre-trained using more data and for a longer period of time. The resulting model outperforms the original BERT model on various benchmarks.

Albert (short for "A lite BERT architecture") [22] reduces the number of parameters, compared to BERT and Roberta, significantly. It uses two parameter reduction techniques. First, it decomposes the vocabulary embedding matrix into two smaller matrices. Second, it shares parameters across different layers (thereby reducing parameter growth as a function of network depth). This model achieves significant speedups without affecting result quality significantly.

Distilbert [42], a "distilled" version of BERT, uses knowledge distillation to reduce parameters and training time. As suggested by the name, it uses knowledge distillation to reduce the network size, compared to BERT. Here, BERT serves as a "teacher" that trains a smaller network, Distilbert. The authors of Distilbert show that the resulting model realizes attractive tradeoffs between training time and result quality.

Beyond pre-trained models, the evaluation considers a simple baseline. This baseline decomposes names of compared columns into tokens. Then, it computes the Jaccard similarity on the two token sets, associated with the compared columns. It predicts a correlation if the Jaccard similarity is at least 0.5. Hence, this baseline considers columns as correlated if their names are sufficiently similar.

Note that the evaluation focuses uniquely on methods that work with column names alone, as opposed to methods that exploit the actual data for correlation analysis. As demonstrated in Section **??**, methods of the former category can be used to guide application of methods that belong to the latter category.

5.2 Experimental Setup

The experiments in this and the following sections use an EC2 instance of type p3.2xlarge, recommended for machine learning workloads. It features a Tesla V100 GPU with 5,120 CUDA cores, 8 vCPUs, and 488 GB of RAM. Prediction methods are implemented using the simpletransformers Python library[5], using the default parameter settings of that library, unless noted otherwise. The simpletransformers library is internally based on the Huggingface library[6] which supports a wide range of pre-trained language models.

5.3 Comparison Results

The following experiments use the Pearson correlation coefficient. Two columns are considered correlated if $|R| \geq 0.9$ with a p-value of at most 5%. For training, 80% of numerical column pairs are used (around 47K pairs) while reporting results for the remaining 20%. Prediction quality is measured according to all metrics introduced in Section 3.2.

Hypothesis 1. *Pre-trained language models predict correlations better than simpler baselines.*

Figure 3 compares prediction quality across the four prediction methods. The simple baseline performs quite well (even though not as good as the other methods) for precision and accuracy. Here, the baseline benefits as it predicts no correlation in most cases. As correlated column pairs are rare, this simple strategy can achieve a relatively high accuracy (which, among other things, motivates the use of multiple quality metrics). The baseline predicts a correlation for columns with very similar names. It seems that such column pairs tend to be correlated indeed, explaining the reasonably high precision values. However, the simple baseline achieves only poor results for recall, F1 score, and the MCC metric. For instance, its recall is around 1% only. This demonstrates the need for a more sophisticated approach (thereby validating Hypothesis 1).

[5]https://simpletransformers.ai/
[6]https://huggingface.co/transformers/

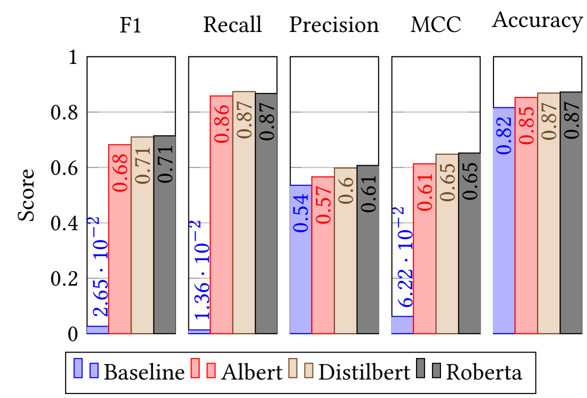


Figure 3: Comparison of correlation prediction methods.

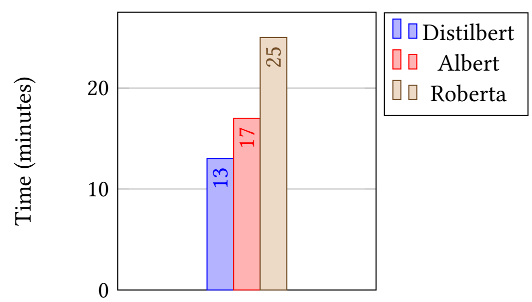


Figure 4: Training time of different transformer variants.

Hypothesis 2. *Larger models predict correlations more reliably than smaller models.*

The performance of all three pre-trained models is quite similar. For the F1 score, precision, MCC, and recall, Roberta performs best, even though only by a small margin. This is expected as Roberta is the largest of the three models. For recall, Distilbert has a slight advantage. In general, Distilbert performs slightly better than Albert in the experiments. MCC is the metric with the largest gap between the three models. Here, Roberta gains a performance advantage of 4% over Albert. Altogether, the three models realize however comparable prediction performance (providing only weak evidence for Hypothesis 2).

Hypothesis 3. *Varying the model size enables different tradeoffs between computational overheads and accuracy.*

Figure 4 reports training time (in minutes) for the three pre-trained models. More precisely, it reports the time for fine-tuning the three models to the problem of correlation prediction (i.e., it does not report time for pre-training which is significantly more expensive).

Albert and Distilbert have been designed with the goal of reducing overheads, compared to larger models such as Roberta. The results indicate that this approach pays off for the correlation prediction task as well. For instance, training time is more than two times smaller for Distilbert, compared to Roberta. Given the slightly better performance of Distilbert, compared to Albert, this model seems like the best alternative to reduce training overheads. In the following, due to slightly higher precision, Roberta is used as default method for correlation prediction. Overall, the results support Hypothesis 3.

6 SCENARIO VARIANTS

The following experiments analyze how scenario properties influence prediction performance. This section considers variations in the amount of training data as well as in the relationship between training and test data. Also, it compares performance for different definitions of correlation.

The following experiments use the Roberta model and define correlation via the Pearson coefficient, using a maximal p-Value of 5%. The threshold on the absolute value of R varies in the following, using a default of $|R| \geq 0.9$. The quantity of training data varies as well, using 80% of numerical column pairs as default. The remaining data is used for testing.

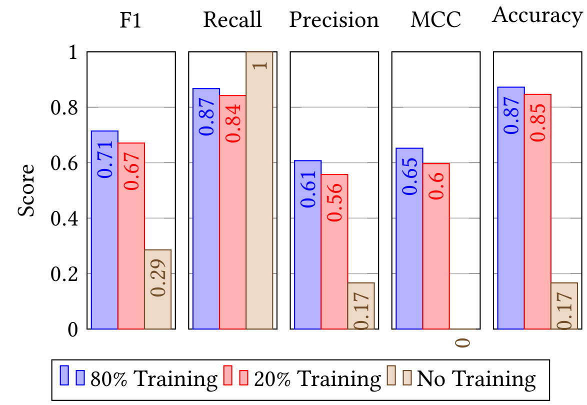
Hypothesis 4. *Correlation predictions become more accurate when training on more column pairs.*

Figure 5a reports results related to Hypothesis 4. It compares prediction performance as a function of the training (and test) ratio. It compares performance in two scenarios. The first scenario uses 80% of column pairs as training data (i.e., around 47K column pairs) and the rest for testing. The second scenario uses 20% of column pairs as training data while using the rest for testing. The third scenario uses no training data whatsoever (i.e., zero-shot setting). For all five metrics of prediction quality, as expected, having more training data helps. This validates Hypothesis 4.

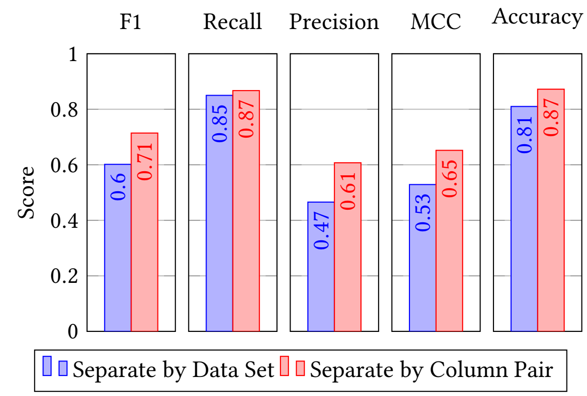
However, given the significant difference in the amount of training data (the amount of training data differs by a factor of four across the two scenarios), the differences in prediction performance seem moderate. The maximal difference across all five performance metrics is six percent. This is consistent with prior results for other tasks from the NLP domain, showing that pre-trained language models achieve reasonable performance, already with modest amounts of training data [13].

Hypothesis 5. *Predicting correlations for new column pairs becomes easier after observing correlations from other column pairs in the same data set.*

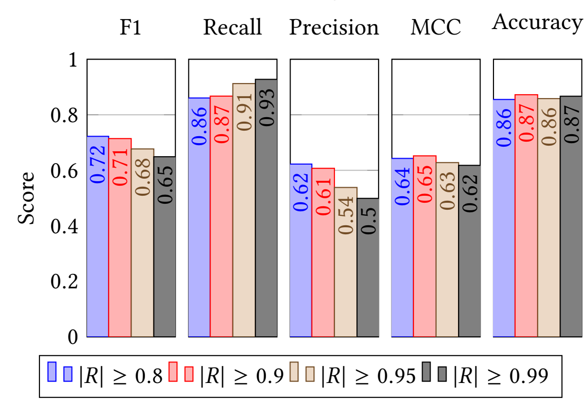
The following analysis focus on relationship between training and test data on prediction performance. It considers two scenarios. The first scenario separates training and test data at the granularity of column pairs. This means that training and test data may contain column pairs from the same data set. Of course, each column pair is considered only once. The second scenario ensures that training and test data are derived from different data sets. This means separating data sets (the ones used for correlation analysis) into training and test data sets, then deriving column pairs for training and testing only from the corresponding data sets. With this method, training and test samples derive from entirely different data sets. Of course, training and test data sets may still have similarities. For instance, the same column names may appear in different data sets. However, even if column names are identical, the associated data (and therefore the correlation results) may still differ. Overall, since similarities between data sets are common beyond Kaggle,



(a) Impact of training data quantity.



(b) Impact of training data quality.



(c) Impact of correlation strength.

Figure 5: Prediction quality for Pearson correlation coefficient in different scenarios.

the results for the second scenario are of high practical relevance. They illustrate the performance obtained by training predictors on a representative collection of data sets, then applying them to new data sets (while benefiting from naturally occurring similarities between training data and new data).

Figure 5b shows corresponding results. It varies the quality (i.e., how closely it relates to the test examples) but not the quantity of training data. For all five quality metrics, allowing column pairs from the same data set in training and testing improves performance. Depending on the metric, this increase is moderate (recall increase by only 2%) or more significant (precision increases by 14%). In any case, the results support Hypothesis 5.

Hypothesis 6. *Predicting strong correlations is easier than predicting weak correlations.*

Hypothesis 6 connects the criterion for defining correlation to prediction performance. The following experiment uses a p-Value threshold of 5% but varies the threshold on $|R|$ between 0.8 and 0.99. Figure 5c shows corresponding results. The results do not show clear tendencies. Recall generally increases while precision and F1 scores decrease, as the requirements for correlation become tighter. Accuracy and MCC do not show clear tendencies. Altogether, the experimental results do not provide strong evidence for Hypothesis 6.

7 RESULT BREAKDOWNS

This section explores the question of which properties of test cases contribute to making correlations more or less difficult to predict, breaking down results based on properties of column names.

The following experiments use the Roberta model and the Pearson correlation coefficient. They consider columns correlated for an absolute R-value of at least 0.9 and a p-Value of at most 5%.

Hypothesis 7. *Longer column names yield more information and make prediction more accurate.*

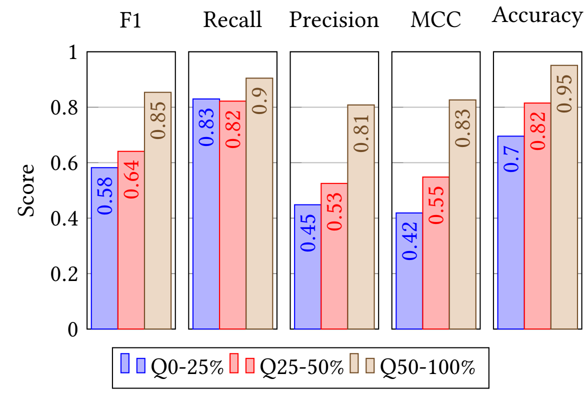
The following experiments focus on Hypothesis 7 and consider two metrics of column name length: the number of characters and the number of tokens (i.e., the number of text snippets, separated by spaces and underscores). Figure 6a reports results for the number of characters and Figure 6b results for the number of tokens. Both figures report results for three length ranges separately. Those ranges refer to the quantiles (e.g., the range "Q50-100%" includes pairs of columns whose length is at or above average length).

The differences are significant. For both column name metrics and all five prediction quality metrics, having longer column names improves performance. Those differences are more pronounced when measuring length as the number of characters (compared to the number of words). For instance, when measuring length as the number of characters, the MCC score improves from 42% to 83% when going from short to long names. Precision improves from 45% to 81%. Those results provide strong experimental evidence validating Hypothesis 7. Shorter column names may indicate a higher ratio of "placeholder" names (e.g., based on column numbers) or correlate with less detailed explanations of column semantics. The next hypothesis relates to the nature of column names.

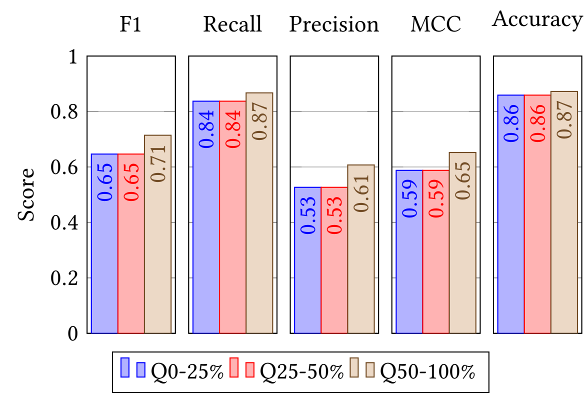
Hypothesis 8. *Column names with a higher ratio of English words (as opposed to abbreviations or other symbols) can be more easily interpreted and make predictions more accurate.*

Figure 6c reports results for subsets of column pairs, characterized by the ratio of English words in column names. It measures that ratio as follows. First, both column names are divided into tokens, using common separators. Then, each token is compared to an English dictionary. The ratio of words, contained in the dictionary, to the number of all tokens is the word ratio. For Figure 8 separates column pairs into three groups, associated with different ranges for that ratio (e.g., "Q50-100%" includes column pairs whose ratio of English words is at or above the average).

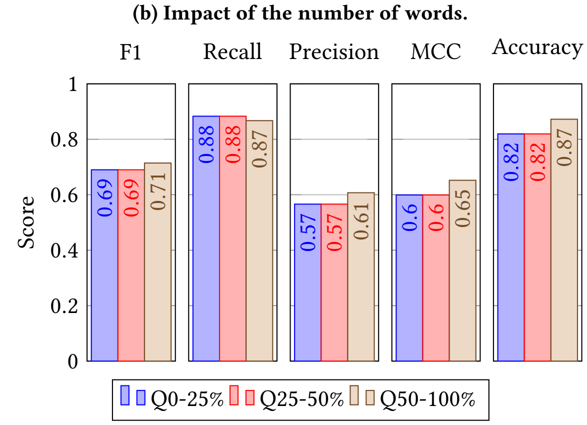
Here, the absolute differences between low and high word ratios are relatively small. For instance, MCC scores increase only from



(a) Impact of the number of characters.



(b) Impact of the number of words.



(c) Impact of ratio of English words in column names.

Figure 6: Breakdown of prediction quality by test case properties for the Pearson correlation coefficient.

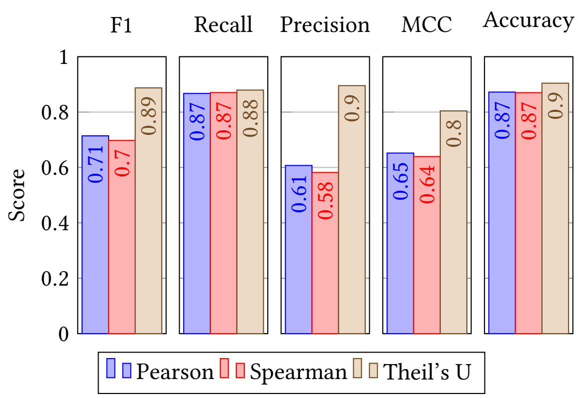
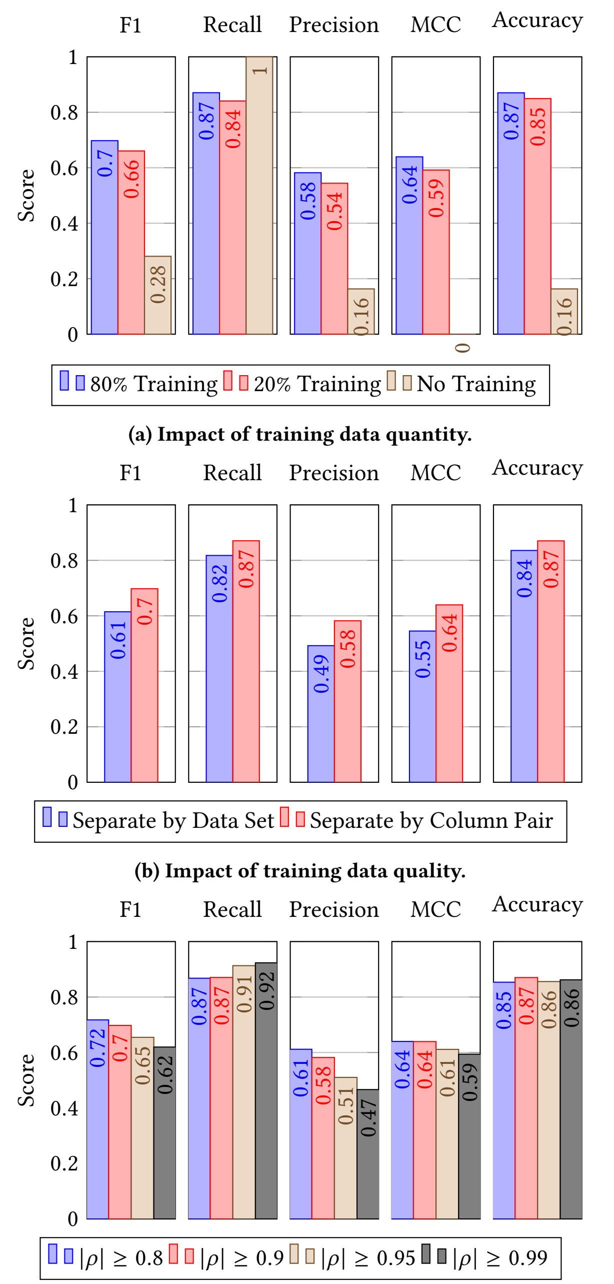


Figure 7: Comparison of different correlation measures.



(a) Impact of training data quantity.

(b) Impact of training data quality.

(c) Impact of degree of correlation.

Figure 8: Prediction quality for Spearman's coefficient in different prediction scenarios.

60% to 65%, moving from low to high ratios. Recall, for instance, decreases slightly from 88% to 87%. Altogether, the experimental evidence for Hypothesis 8 is weak.

The results in this section also show that it is possible to assess the confidence of correlation predictions, based on properties of column names (specifically: the length). This may be useful for systems exploiting correlation prediction as a component, as discussed in more detail in Section **??** of the extended technical report.

8 OTHER CORRELATION METRICS

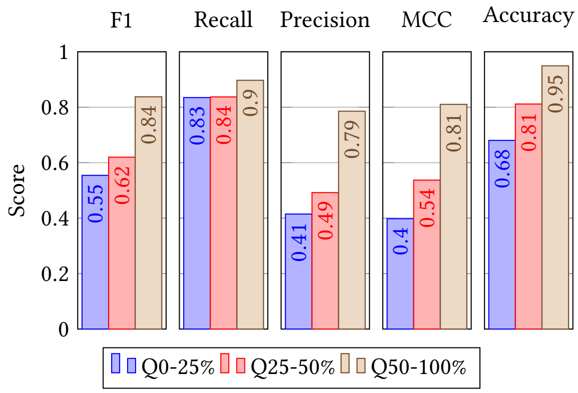
This section expands the experimental scope from Pearson correlation to other correlation metrics, thereby verifying whether prior findings generalize. The following experiments consider Spearman's coefficient and Theil's U (discussed in more detail in Section 3). Unless noted otherwise, they consider columns correlated, according to Spearman's coefficient, if the absolute coefficient value is at or above 0.9 ($|\rho| \geq 0.9$) with a p-value of at most 5%. For Theil's U, it uses a threshold of 0.9 as well. By default, it uses again 80% of column pairs as training data and separate training from test data at the granularity of column pairs. Note that Theil's U applies to all column data types, thereby increasing the number of eligible column pairs (for training and testing) to around 119,000.

Figure 7 compares prediction performance for all three definitions of correlation. While prediction performance is close for Pearson and Spearman coefficients, prediction performance increases for Theil's U according to all quality metrics. A first hypothesis is that the higher amount of training data (columns of all types as opposed to numerical columns only) contributes to that performance.

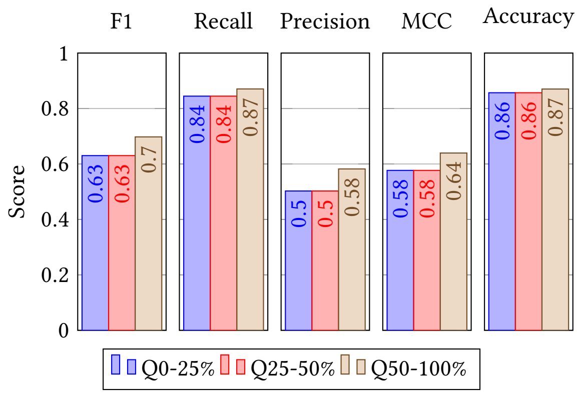
Figure 8 re-tests the hypotheses from Section 6 for Spearman's coefficient. Clearly, increasing the amount of training data also increases prediction performance (Hypothesis 4), as well as sharing data sets among training and test cases (Hypothesis 5). The tendencies are less clear for the threshold on ρ (Hypothesis 6).

Figure 9 validates the hypotheses from Section 7 for Spearman's coefficient. It considers different data subsets and compare prediction performance. Again, the most important parameter influencing prediction performance seems to be the column name length (Hypothesis 7).

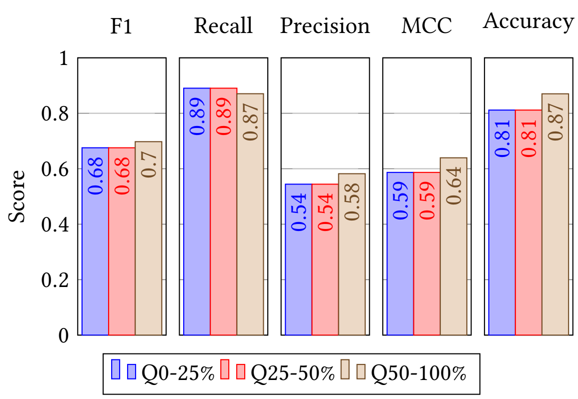
Figure 10 compares different prediction scenarios for Theil's U. Here, the relative tendencies are similar to prior experiments while the absolute values are significantly better. It is interesting that prediction quality for Theil's U, when using 20% training data (i.e., around 24K training samples), is still better than prediction performance for the other coefficients when using 80% of training data (i.e., around 47K training samples). This shows that, beyond



(a) Impact of column name length.

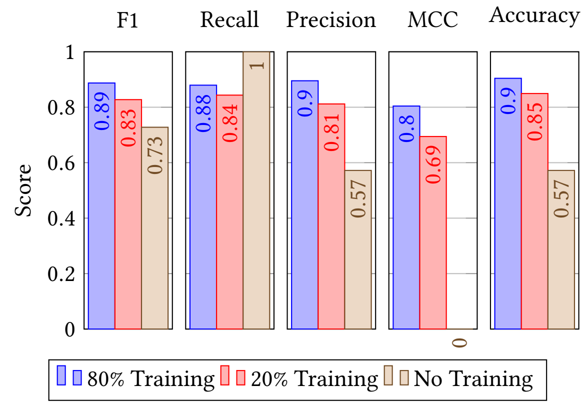


(b) Impact of number of words in column names.

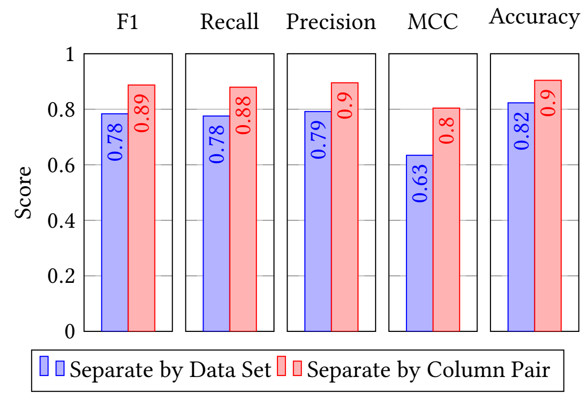


(c) Impact of ratio of English words in column names.

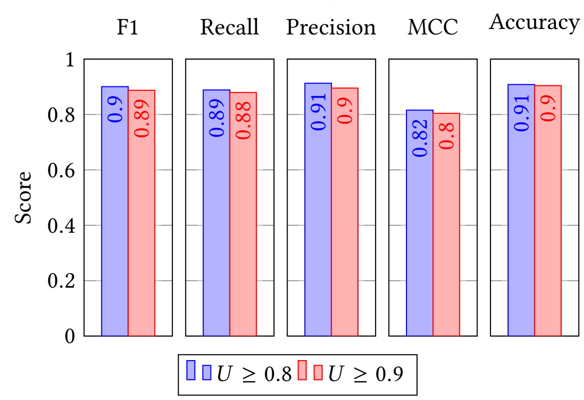
Figure 9: Breakdown of prediction quality by test case properties for Spearman's coefficient.



(a) Impact of training data quantity.



(b) Impact of training data quality.



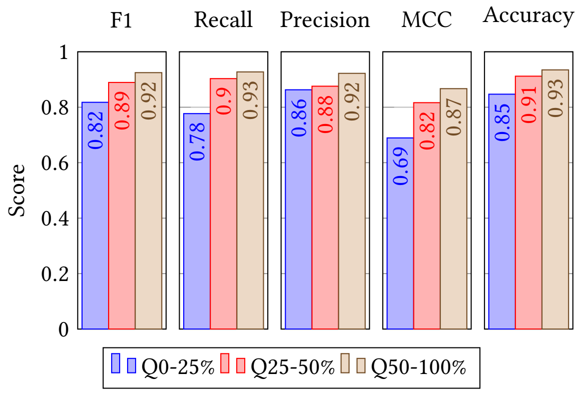
(c) Impact of degree of correlation.

Figure 10: Prediction quality for Theil's U in different prediction scenarios.

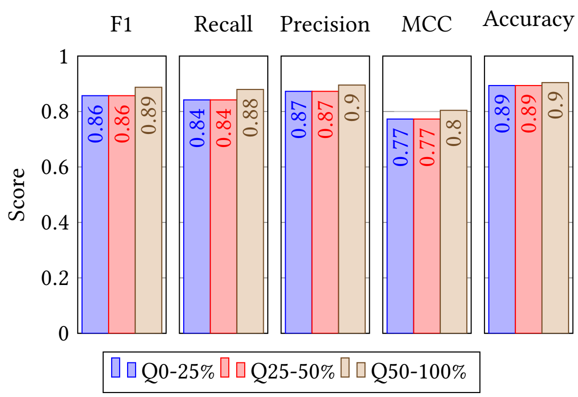
the amount of training data, other factors must contribute to the improved performance.

Figures 11a to 11c study prediction performance for Theil's U and different data subsets. While column name length remains the most important factor, a higher ratio of English words in column names relates to better prediction accuracy as well (except for the precision metric). Despite of that, the absolute differences remain relatively small.

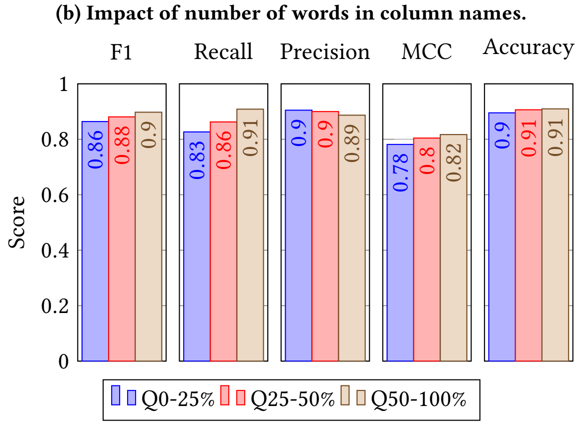
Altogether, the primary outcomes of prior experiments generalize to other definitions of correlation.



(a) Impact of column name length.



(b) Impact of number of words in column names.



(c) Impact of ratio of English words in column names.

Figure 11: Breakdown of prediction quality by test case properties for Theil's U.

9 COLUMN TYPES

This section focuses on column types and their impact on prediction quality.

HYPOTHESIS 9. *Prediction quality varies as a function of column types.*

Tables 3 to 5 break down prediction results according to the data types of the involved columns. Column types are inferred automatically by the pandas framework, considering types bool (B), float64 (F), int64 (I), and object (O). The first column of each table contains the types of column pairs (e.g., "I-F" indicates a column pair where the first column is of type int64 whereas the second column is of type float64). Table 3 breaks down results for Pearson correlation, Table 4 reports results for the Spearman correlation coefficient, and Table 5 reports results for Theil's U.

Clearly, column types have significant impact on prediction accuracy. E.g., for Pearson correlation, MCC scores vary by 30% across different column type combinations. The order of types matters (e.g., comparing F-I versus I-F in Table 3). This can be explained by the fact that the column position correlates with the likelihood of correlations. For instance, key columns tend to appear first in tables and are often of type integer. Hence, the probability of correlation for an integer column, followed by a float column, is different

Table 3: Impact of types on Pearson correlation prediction.

Types	F1	Pre	Rec	Acc	MCC
F-F	0.88	0.84	0.93	0.93	0.84
F-I	0.67	0.58	0.78	0.98	0.66
I-F	0.85	0.83	0.88	0.99	0.84
I-I	0.62	0.49	0.85	0.81	0.54

Table 4: Impact of types on Spearman correlation prediction.

Types	F1	Pre	Rec	Acc	MCC
F-F	0.87	0.82	0.93	0.93	0.83
F-I	0.68	0.60	0.78	0.97	0.67
I-F	0.71	0.63	0.81	0.97	0.70
I-I	0.56	0.41	0.90	0.76	0.49

Table 5: Impact of types on Theil's U correlation prediction.

Types	F1	Pre	Rec	Acc	MCC
O-O	0.92	0.92	0.92	0.90	0.80
O-F	0.93	0.92	0.93	0.90	0.79
O-I	0.94	0.95	0.93	0.92	0.83
O-B	0.50	0.33	1.00	0.86	0.53
F-O	0.95	0.95	0.94	0.93	0.85
F-F	0.95	0.96	0.95	0.93	0.80
F-I	0.94	0.94	0.94	0.93	0.86
F-B	0.00	0.00	0.00	0.80	0.00
I-O	0.93	0.92	0.93	0.92	0.85
I-F	0.94	0.96	0.93	0.93	0.85
I-I	0.80	0.89	0.73	0.85	0.69
I-B	0.00	0.00	0.00	1.00	0.00
B-O	0.89	0.84	0.93	0.85	0.69
B-F	0.80	0.67	1.00	0.80	0.67
B-I	0.96	1.00	0.92	0.95	0.90
B-B	0.18	0.11	0.50	0.67	0.10

than for the complementary order. This can influence accuracy and other performance metrics for prediction. At the same time, it motivates extensions of the approach that exploit the column position in addition to column names.

Overall, the results clearly support Hypothesis 9 and could be exploited, e.g., to assess the confidence in a prediction, based on column types. At the same time, they raise the question whether column types may be useful as additional input for the language model itself.

Hypothesis 10. *Integrating column types as additional feature increases prediction accuracy.*

Prior experiments have focused on predicting data correlation from column names alone. The following experiment considers column types as an additional feature. Figure 12 reports corresponding results. Different from before, the input to the language model now contains column names, followed by column types (separated by a single space). E.g., for two columns "car" and "maker" of types "object", the input consists of the pair "car object" and "maker object".

Figure 12 compares results with and without types (using 80% of data for training). Considering types leads to moderate benefits for most correlation metrics and in most scenarios. E.g., using types leads to improvements of four percentage points in precision when predicting correlation according to Theil's U. On the other hand, it leads to slight losses when predicting correlation according to Spearman's correlation coefficient. Overall, the results provide weak support for Hypothesis 10.

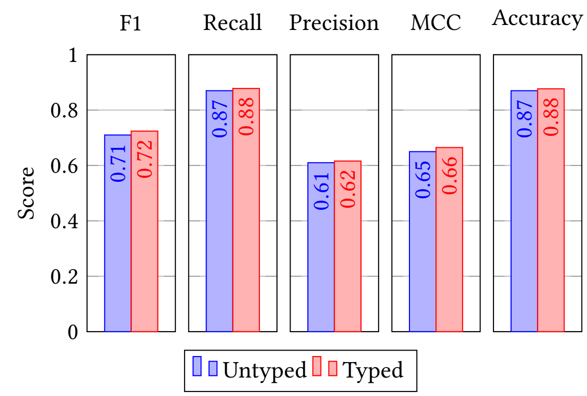
10 CONCLUSION

In recent publications, I suggest using advanced natural language analysis on text associated with database schema elements. This is a cheap source of information as the cost depends only on the schema, but not on the data size. Ideally, natural language analysis yields insights on likely data properties, that are helpful to guide automated tuning or data profiling efforts.

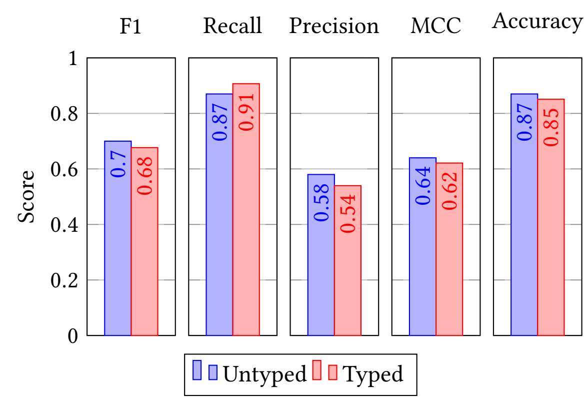
This suggestion is based on the assumption that pre-trained language models are indeed able to extract useful insights from schema text. For the first time, this study evaluates that hypothesis in detail, focusing on the problem of correlation detection. Correlation detection is an expensive process that has received significant attention in the database community, due to its various use cases in database optimization. Hence, obtaining additional information to guide corresponding profiling efforts is practically useful.

The experiments yields the following insights (among others):

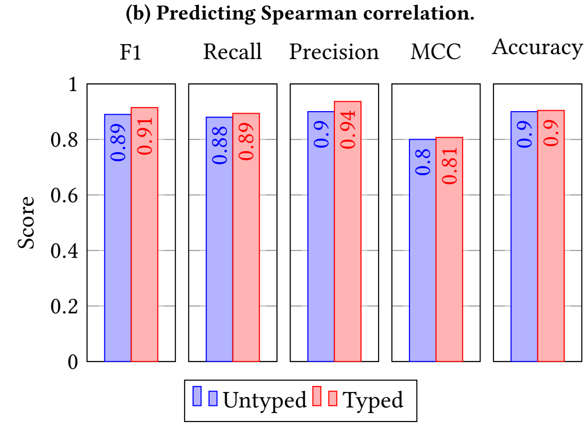
- In many, even though not all, cases, pre-trained language models are able to infer useful information on data correlation from column names alone.
- This is already possible with relatively small models, e.g. Distilbert, with parameter counts in the tens of millions, enabling their use on commodity machines.
- Those findings hold for a variety of popular data correlation metrics, including Pearson correlation, Spearman correlation, and Theil's U.
- Training models for correlation prediction on data sets that are similar to test data increases performance, motivating domain specialization.



(a) Predicting Pearson correlation.



(b) Predicting Spearman correlation.



(c) Predicting Theil's U correlation.

Figure 12: Exploiting types of columns for classification (in addition to column names).

- Surprisingly, prediction accuracy is only marginally affected by the degree of data correlation.
- On the other hand, predictions become more accurate if more text is available, i.e. if column names are longer.

The experimental results inform future research aimed at NLP-enhanced database tuning and data profiling. They provide evidence supporting assumptions underlying that nascent research direction.

REFERENCES

[1] Ziawasch Abedjan, Jorge Arnulfo Quiané-Ruiz, and Felix Naumann. 2014. Detecting unique column combinations on dynamic data. In *ICDE*. 1036–1047. https://doi.org/10.1109/ICDE.2014.6816721

[2] I. Androutsopoulos, G. D. Ritchie, and P. Thanisch. 1995. Natural language interfaces to databases - an introduction. *Journal of Natural Language Engineering* 1, 1 (1995), 29–81. https://doi.org/10.1017/S0269888900005476 arXiv:9503016 [cmp-lg]

[3] Simran Arora, Brandon Yang, Sabri Eyuboglu, Avanika Narayan, Andrew Hojel, Immanuel Trummer, and Christopher Ré. 2023. Language Models Enable Simple Systems for Generating Structured Views of Heterogeneous Data Lakes. *CoRR* abs/2304.0 (2023), 1–30. https://doi.org/10.48550/arXiv.2304.09433 arXiv:2304.09433

[4] R. Artusi, P. Verderio, and E. Marubini. 2002. Bravais-Pearson and Spearman correlation coefficients: Meaning, test of hypothesis and confidence interval. *International Journal of Biological Markers* 17, 2 (2002), 148–151. https://doi.org/10.5301/JBM.2008.2127

[5] PG Brown and PJ Hass. 2003. BHUNT: Automatic discovery of fuzzy algebraic constraints in relational data. In *VLDB*. 668–679. http://dl.acm.org/citation.cfm?id=1315509

[6] N Bruno, N Bruno, S Chaudhuri, and S Chaudhuri. 2004. Conditional selectivity for statistics on query expressions. In *SIGMOD*. https://doi.org/10.1145/1007568.1007604

[7] Davide Chicco and Giuseppe Jurman. 2020. The advantages of the Matthews correlation coefficient (MCC) over F1 score and accuracy in binary classification evaluation. *BMC Genomics* 21, 1 (2020), 1–13. https://doi.org/10.1186/s12864-019-6413-7

[8] Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, Parker Schuh, Kensen Shi, Sasha Tsvyashchenko, Joshua Maynez, Abhishek Rao, Parker Barnes, Yi Tay, Noam Shazeer, Vinodkumar Prabhakaran, Emily Reif, Nan Du, Ben Hutchinson, Reiner Pope, James Bradbury, Jacob Austin, Michael Isard, Guy Gur-Ari, Pengcheng Yin, Toju Duke, Anselm Levskaya, Sanjay Ghemawat, Sunipa Dev, Henryk Michalewski, Xavier Garcia, Vedant Misra, Kevin Robinson, Liam Fedus, Denny Zhou, Daphne Ippolito, David Luan, Hyeontaek Lim, Barret Zoph, Alexander Spiridonov, Ryan Sepassi, David Dohan, Shivani Agrawal, Mark Omernick, Andrew M. Dai, Thanumalayan Sankaranarayana Pillai, Marie Pellat, Aitor Lewkowycz, Erica Moreira, Rewon Child, Oleksandr Polozov, Katherine Lee, Zongwei Zhou, Xuezhi Wang, Brennan Saeta, Mark Diaz, Orhan Firat, Michele Catasta, Jason Wei, Kathy Meier-Hellstern, Douglas Eck, Jeff Dean, Slav Petrov, and Noah Fiedel. 2022. PaLM: Scaling Language Modeling with Pathways. *CoRR* abs/2204.0 (2022), 1–87. arXiv:2204.02311 http://arxiv.org/abs/2204.02311

[9] Graham Cormode, Minos Garofalakis, Peter J. Haas, and Chris Jermaine. 2011. Synopses for massive data: Samples, histograms, wavelets, sketches. *Foundations and Trends in Databases* 4, 1-3 (2011), 1–294. https://doi.org/10.1561/1900000004

[10] Jacob Devlin, Ming Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *NAACL*. 4171–4186. arXiv:1810.04805

[11] Luciano Floridi and Massimo Chiriatti. 2020. GPT-3: Its Nature, Scope, Limits, and Consequences. *Minds and Machines* 30, 4 (2020), 681–694. https://doi.org/10.1007/s11023-020-09548-1

[12] Wassily Hoeffding and M. G. Kendall. 1957. Rank Correlation Methods. *Econometrica* 25, 1 (1957), 181. https://doi.org/10.2307/1907752

[13] Jeremy Howard and Sebastian Ruder. 2018. Universal Language Model Fine-tuning for Text Classification. In *ACL*. 328–339. https://doi.org/10.3760/cma.j.issn.04124081.2010.02.006

[14] Wonseok Hwang, Jingyeung Yim, Seunghyun Park, and Minjoon Seo. 2019. (SQLNova) A Comprehensive Exploration on WikiSQL with Table-Aware Word Contextualization.

[15] IBM. 2021. IBM Infosphere Information Analyzer.

[16] I F Ilyas, V Markl, P Haas, P Brown, and Ashraf Aboulnaga. 2004. CORDS: Automatic discovery of correlations and soft functional dependencies. In *SIGMOD*. 647–658. https://doi.org/10.1145/1007568.1007641 arXiv:ISBN 0-89791-128-8

[17] Informatica. 2021. Informatica Data Profiling Solutions.

[18] Yifeng Jin, Zijing Tan, Weijun Zeng, and Shuai Ma. 2021. Approximate order dependency discovery. In *ICDE*. IEEE, 25–36. https://doi.org/10.1109/ICDE51399.2021.00010

[19] Georgios Karagiannis, Mohammed Saeed, Paolo Papotti, and Immanuel Trummer. 2020. Scrutinizer: A Mixed-Initiative Approach to Large-Scale, Data-Driven Claim Verification. *PVLDB* 13, 12 (2020), 2508–2521. https://doi.org/10.14778/3407790.3407841

[20] Georgios Karagiannis, Mohammed Saeed, Paolo Papotti, and Immanuel Trummer. 2020. Scrutinizer: Fact Checking Statistical Claims. *PVLDB* 13, 12 (2020), 2965–2968. https://doi.org/10.14778/3415478.3415520

[21] Moe Kayali, Anton Lykov, Ilias Fountalis, Nikolaos Vasiloglou, Dan Olteanu, and Dan Suciu. 2023. CHORUS: Foundation Models for Unified Data Discovery and Exploration. *CoRR* (2023). arXiv:2306.09610 http://arxiv.org/abs/2306.09610

[22] Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2019. ALBERT: A Lite BERT for Self-supervised Learning of Language Representations. (2019), 1–17. arXiv:1909.11942 http://arxiv.org/abs/1909.11942

[23] Philipp Langer and Felix Naumann. 2016. Efficient order dependency detection. *VLDB Journal* 25, 2 (2016), 223–241. https://doi.org/10.1007/s00778-015-0412-3

[24] Viktor Leis, Andrey Gubichev, Peter Boncz, Alfons Kemper, and Thomas Neumann. 2015. How good are query optimizers, really? *PVLDB* 9, 3 (2015), 204–215.

[25] Fei Li and HV Jagadish. 2014. NaLIR: an interactive natural language interface for querying relational databases. In *SIGMOD*. 709–712.

[26] Fei Li and HV Jagadish. 2016. Understanding natural language queries over relational databases. *SIGMOD Record* 45, 1 (2016), 6–13.

[27] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. RoBERTa: A robustly optimized BERT pretraining approach. *CoRR* abs/1907.1, 1 (2019), 1–13. arXiv:1907.11692 https://arxiv.org/abs/1907.11692

[28] Ryan Marcus, Parimarjan Negi, Hongzi Mao, Chi Zhang, Mohammad Alizadeh, Tim Kraska, Olga Papaemmanouil, and Nesime Tatbul. 2018. Neo: A Learned query optimizer. *PVLDB* 12, 11 (2018), 1705–1718. https://doi.org/10.14778/3342263.3342644 arXiv:1904.03711

[29] V. Markl, P. J. Haas, M. Kutsch, N. Megiddo, U. Srivastava, and T. M. Tran. 2007. Consistent selectivity estimation via maximum entropy. *VLDB Journal* 16, 1 (2007), 55–76. https://doi.org/10.1007/s00778-006-0030-1

[30] Avanika Narayan, Ines Chami, Laurel Orr, and Christopher Ré. 2022. Can Foundation Models Wrangle Your Data? *PVLDB* 16, 4 (2022), 738–746. arXiv:2205.09911 http://arxiv.org/abs/2205.09911

[31] Felix Naumann. 2013. Data profiling revisited. *SIGMOD Record* 42, 4 (2013), 40–49. https://doi.org/10.1145/2590989.2590995

[32] Thorsten Papenbrock, Tanja Bergmann, Moritz Finke, Jakob Zwiener, and Felix Naumann. 2015. Data profiling with metanome. *Proceedings of the VLDB Endowment* 8, 12 (2015), 1860–1863. https://doi.org/10.14778/2824032.2824086

[33] Thorsten Papenbrock, Sebastian Kruse, Jorge Arnulfo Quiané-Ruiz, and Felix Naumann. 2015. Divide and conquer-based inclusion dependency discovery. *Proceedings of the VLDB Endowment* 8, 7 (2015), 774–785. https://doi.org/10.14778/2752939.2752946

[34] Thorsten Papenbrock and Felix Naumann. 2017. A hybrid approach for efficient unique column combination discovery. In *BTW*. 195–204.

[35] Yongjoo Park, Shucheng Zhong, and Barzan Mozafari. 2020. QuickSel: Quick Selectivity Learning with Mixture Models. In *SIGMOD*. 1017–1033. https://doi.org/10.1145/3318464.3389727 arXiv:1812.10568

[36] Jinglin Peng, Weiyuan Wu, Brandon Lockhart, Song Bian, Jing Nathan Yan, Linghao Xu, Zhixuan Chi, Jeffrey M. Rzeszotarski, and Jiannan Wang. 2021. DataPrep.EDA: Task-Centric Exploratory Data Analysis for Statistical Modeling in Python. *Proceedings of the ACM SIGMOD International Conference on Management of Data* (2021), 2271–2280. https://doi.org/10.1145/3448016.3457330 arXiv:2104.00841

[37] Sage Publications. 2018. Theil ' s Forecast Accuracy Coefficient : Clarification. 10, 4 (2018), 444–446.

[38] Sage Publications. 2018. Theil ' s Forecast Accuracy Coefficient : Clarification. 10, 4 (2018), 444–446.

[39] Alexandra Rostin, Oliver Albrecht, Jana Bauckmann, Felix Naumann, and Ulf Leser. 2009. A machine learning approach to foreign key discovery. *12th International Workshop on the Web and Databases (WebDB), Providence, Rhode Island* WebDB (2009), 1–6. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.150.2150&rep=rep1&type=pdf

[40] Sebastian Ruder, Matthew E Peters, Swabha Swayamdipta, and Thomas Wolf. 2019. Transfer Learning in Natural Language Processing. In *ACL: Tutorials*. 15–18.

[41] Diptikalyan Saha, Avrilia Floratou, Karthik Sankaranarayanan, Umar Farooq Minhas, Ashish R Mittal, and Fatma Ozcan. 2016. ATHENA: An ontology-driven system for natural language querying over relational data stores. *VLDB* 9, 12 (2016), 1209–1220.

[42] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. (2019), 2–6. arXiv:1910.01108 http://arxiv.org/abs/1910.01108

[43] PG G Selinger, MM M Astrahan, D D Chamberlin, R A Lorie, and T G Price. 1979. Access path selection in a relational database management system. In *SIGMOD*. 23–34. http://dl.acm.org/citation.cfm?id=582095.582099

[44] Jaydeep Sen, Chuan Lei, Abdul Quamar, Fatma Özcan, Vasilis Efthymiou, Ayushi Dalmia, Greg Stager, Ashish Mittal, Diptikalyan Saha, and Karthik Sankaranarayanan. 2020. ATHENA++: natural language querying for complex nested SQL queries. *Proceedings of the VLDB Endowment* 13, 12 (2020), 2747–2759. https://doi.org/10.14778/3407790.3407858

[45] Talend. 2021. Talend Data Explorer.

[46] Nan Tang, Ju Fan, Fangyi Li, Jianhong Tu, Xiaoyong Du, Guoliang Li, Sam Madden, and Mourad Ouzzani. 2021. Rpt: Relational pre-trained transformer is almost all you need towards democratizing data preparation. *PVLDB* 14, 8 (2021),

1254–1261. https://doi.org/10.14778/3457390.3457391 arXiv:2012.02469

[47] James Thorne, Majid Yazdani, Marzieh Saeidi, Fabrizio Silvestri, Sebastian Riedel, and Alon Halevy. 2021. From natural language processing to neural databases. *Proceedings of the VLDB Endowment* 14, 6 (2021), 1033–1039. https://doi.org/10.14778/3447689.3447706

[48] Immanuel Trummer. 2021. The Case for NLP-Enhanced Database Tuning: Towards Tuning Tools that "Read the Manual". *PVLDB* 14, 7 (2021), 1159–1165. https://doi.org/10.14778/3450980.3450984

[49] Immanuel Trummer. 2022. CodexDB: Synthesizing Code for Query Processing from Natural Language Instructions using GPT-3 Codex. *PVLDB* 15, 11 (2022), 2921 – 2928. https://doi.org/10.14778/3551793.3551841

[50] Immanuel Trummer. 2022. DB-BERT: a Database Tuning Tool that "Reads the Manual". In *SIGMOD*. 190–203. https://doi.org/10.1145/3514221.3517843

[51] Immanuel Trummer. 2022. From BERT to GPT-3 Codex: Harnessing the Potential of Very Large Language Models for Data Management. *PVLDB* 15, 12 (2022), 3770 – 3773. https://doi.org/10.14778/3554821.3554896

[52] Immanuel Trummer. 2022. Towards NLP-Enhanced Data Profiling Tools. In *CIDR*. 1–1. https://www.cidrdb.org/cidr2022/papers/a55-trummer.pdf

[53] Immanuel Trummer, Junxiong Wang, Ziyun Wei, Deepak Maram, Samuel Moseley, Saehan Jo, Joseph Antonakakis, and Ankush Rayabhari. 2021. SkinnerDB: Regret-bounded Query Evaluation via Reinforcement Learning. *ACM Transactions on Database Systems* 46, 3 (2021), 1–45. https://doi.org/10.1145/3464389

[54] Kostas Tzoumas, Amol Dehspande, and Christian S Jensen. 2011. Lightweight graphical models for selectivity estimation without independence assumptions. In *VLDB*. 852–863.

[55] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*. 5999–6009. arXiv:1706.03762

[56] Jiguang Wang. 2013. Pearson Correlation Coefficient. *Encyclopedia of Systems Biology* 2 (2013), 1671–1671. https://doi.org/10.1007/978-1-4419-9863-7_372

[57] Junxiong Wang, Immanuel Trummer, and Debabrota Basu. 2021. UDO: Universal Database Optimization using Reinforcement Learning. *PVLDB* 14, 13 (2021), 3402–3414. arXiv:2104.01744 https://doi.org/10.14778/3484224.3484236

[58] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. 2020. Transformers: State-of-the-Art Natural Language Processing. In *EMNLP*. 38–45. https://doi.org/10.18653/v1/2020.emnlp-demos.6 arXiv:arXiv:1910.03771v5

[59] Lucas Woltmann, Claudio Hartmann, Maik Thiele, and Dirk Habich. 2019. Cardinality estimation with local deep learning models. In *aiDM*. 1–8.

[60] Xiaojun Xu, Chang Liu, and Dawn Song. 2017. SQLNet: generating structured queries from natural language without reinforcement Learning. 1–13. arXiv:1711.04436 http://arxiv.org/abs/1711.04436

[61] Tao Yu, Rui Zhang, Kai Yang, Michihiro Yasunaga, Dongxu Wang, Zifan Li, James Ma, Irene Li, Qingning Yao, Shanelle Roman, Zilin Zhang, and Dragomir R. Radev. 2020. Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-SQL task. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, EMNLP 2018*. 3911–3921. https://doi.org/10.18653/v1/d18-1425 arXiv:1809.08887

[62] Victor Zhong, Caiming Xiong, and Richard Socher. 2017. Seq2SQL: Generating Structured Queries from Natural Language using Reinforcement Learning. *CoRR* abs/1709.0, 1 (2017), 1–12. arXiv:1709.00103 http://arxiv.org/abs/1709.00103